Implementation of a Surfactant Model in MRST With Basis in Schlumberger's Eclipse

Kristian Jørgensen

Petroleum Geoscience and Engineering
Submission date: June 2013
Supervisor: Ole Torsæter, IPT

Norwegian University of Science and Technology
Department of Petroleum Engineering and Applied Geophysics

Acknowledgement

I would like to thank prof. Ole Torsæter at NTNU, Jan Åge Stensen at SINTEF and Knut-Andreas Lie at SINTEF Applied Mathematics in Oslo for providing me the opportunity to write this thesis. I would also like to thank Sindre K. Hilden for taking time during his studies to assist me and answer my questions.

Most importantly, I would like to thank Halvor M. Nilsen at SINTEF Applied Mathematics for providing invaluable feedback, input, constructive criticism and great support during the course of this thesis.

Abstract

In conjunction with the Open Porous Media (OPM) Initiative, SINTEF Applied Mathematics in Oslo have developed the Matlab Reservoir Simulation Toolbox (MRST) with the purpose to function as a an efficient testing platform for implementation of new solution and discretisation methods in reservoir simulations applications. MRST has been released as an open-source program under the GNU General Public License (GPL[1]), and in this thesis, the author intends to modify the existing source code of MRST release 2012b[2] to implement a surfactant model as an extension to a black-oil framework of equations. The governing equations are evaluated using the finite-volume method and the system of equations is solved fully-implicitly using the Newton-Raphson method. The model created in this thesis is based on the surfactant model in Eclipse® which is described in the Eclipse® Technical Description.

A central part of MRST is the use of automatic differentiation. This is a way to compute function derivatives of coded equations without the tedious, explicit coding of derivative with respect to each separate variable. This is useful when implementing new physics to the already existing framework of equations, as the Jacobian matrix in the Newton-Raphson method can be obtained automatically for newly implemented equations, simply by associating them with a specific Matlab class. A feature of MRST is that it supports industry standard input formatting, so the surfactant model specific keywords in Eclipse® have also been implemented in MRST to allow for an easy transition when comparing the results from the two simulation programs.

Based on observations made, the authors concludes that the work of implementing a surfactant model to an existing black-oil formulation for general grids in MRST has been successful. The model compares well with Eclipse® and predicts largely the same behaviour in terms of surfactant distribution, saturation and pressure

[1]http://www.gnu.org/licenses/gpl.html
[2]Source code, related article, tutorials etc. can be found at www.sintef.no/MRST

profile along with production data. However, it is not able to fully replicate the results from Eclipse®. The reason for this is largely undetermined, but the author points to two possible explanations : *1)* The observed smearing of the results in the surfactant profile is indicative of differences in the numerical solution between Eclipse® and the implemented surfactant model. This difference results in a higher degree of numerical dispersion in the implemented surfactant model than for Eclipse ®. *2)* The implementation of fluid transport properties, perhaps relative permeability especially, is also pointed out as a reason for the observed differences.

The author means that the model is a good foundation for further research and development in surfactant injection modelling.

Sammendrag

I forbindelse med Open Porous Media (OPM) Initiative har SINTEF Anvendt Matematikk i Oslo utviklet Matlab® Reservoir Simulation Toolbox (MRST) som har som hensikt å fungere som en effektiv test-plattform for implementering av nye løsnings- og diskretiseringsmetoder i reservoar-simulering. MRST er lansert som et åpen kildekode-program gjennom GNU General Public License (GPL), og i denne avhandlingen skal forfatteren implementere en surfaktant-modell som en utvidelse av rammeverk av black-oil ligninger. De gjeldende partielle differensialligningene løses ved å ta i bruk en finite-volume metode og løser det resulterende ligningssystemet fullt-implisitt ved hjelp av Newton-Raphson metoden. Modellen utviklet i denne avhandlingen er basert på surfaktant-modellen inkludert i Eclipse® og en beskrivelse finnes i Eclipse® Technical Description.

En sentral del av MRST, er bruken av automatisk derivasjon. Dette er en måte å beregne deriverte av kodede matematiske funksjoner, uten eksplisitt å implementere ny kode for å beregne den deriverte med hensyn på hver funksjonsvariabel. Dette er nyttig for å inkludere ny fysikk til et allerede eksisterende rammeverk av ligninger, da Jakobi-matrisen i Newton-Raphson metoden kan beregnes automatisk for nye ligninger ved å assosiere hver variabel med en spesifikk Matlab-klasse. En av MRSTs funksjonaliterer er at det støtter industri-standard input-formatering. For å sørge for en enkel overgang mellom MRST og Eclipse®, ble kodeordene spesifikke til surfaktant-modellen i Eclipse® implementert i modellen i denne avhandlingen. Dette for å lettere kunne sammenligne resultatene fra de to modellene.

Basert på observasjoner, konkluderer forfatteren med at arbeidet med å implementere en surfaktant-modell i den allerede eksisterende black-oil-modellen for generelle grid i MRST, har vært vellykket. Modellen samsvarer godt med Eclipse®, og predikerer stort sett den samme adferden med tanke på surfaktantfordeling i reservoaret, metning- og trykk-profiler og produksjonsdata. Den implementerte modellen klarer likevel ikke å replikere resultatene fra Eclipse® ned

til minste detalj. Årsaken til dette er i stor grad ubestemt, men forfatteren peker på to mulige forklaringer: *1)* Den observerte utsmøringen av resultater er en indikasjon på at det er forskjeller i den numeriske løsningsmetoden implementert i surfatant modellen i avhandlingen. Denne forskjellen ser ut til å gi en større numerisk dispersjon enn hva Eclipse® gir. *2)* Implementasjonen av fluidtransport-egenskaper, kanskje spesielt relativ permeabilitet, pekes også på som en mulig årsak til forskjellene.

Forfatteren mener modellen er et godt grunnlag for videre utvikling og forskning innen modellering av surfaktant-injeksjon i EOR.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Since Phillips Petroleum initiated production from Ekofisk in 1971, kick-starting Norwegian petroleum industry, oil and gas has been one of the largest sources of revenue for the Norwegian state, cumulative adding more than NOK 9000 billion to the national GDP over the past 40 years. The Norwegian Petroleum Directorate stated that 23 % of the total state revenue in 2012 came from the oil and gas sector (Norwegian Petroleum Directorate), constituting a total of NOK 270 billion. The revenues from oil and gas are transferred to the Norwegian Government Pension Fund – Global which, by the end of 2012, was valued to NOK 3 816 billion. The petroleum production originates from 76 fields on the Norwegian Continental Shelf totalling in 225.14 million Sm^3 of produced oil equivalents in 2012, ranking Norway as the worlds number 7th largest oil exporter and the 14th largest oil producer in the world.

As the oil and gas sector is one of the highest grossing industry in Norway, being able to maintain production rates from the existing petroleum reservoirs is a sought after goal. With the giant, "easy to access" fields discovered and producing, attention is being directed at enhancing production from existing oil fields to sustain a declining revenue (Njå, 1994). Enhanced oil recovery (EOR) is a collective term for measures aimed at increasing the cumulative amount of hydrocarbons extracted from reservoirs, by utilizing various mechanisms applied in a tertiary recovery scenario. There are several categories of EOR processes, exploiting different physical effects to improve both *sweep efficiency* and *microscopic sweep*. Among these methods are thermal, miscible and chemical methods.

The chemical processes involve injection of various chemicals such as polymers, surfactants and alkaline. In a surfactant injection scheme, surfactants are added to the injected brine to decrease the interfacial tension (IFT) between the brine and the in situ oil to ultra-low levels. This lowered IFT will result in a significant decrease in capillary pressure and thus reduce capillary trapping of residual oil, manifesting it-self as incremental oil production in form of an oil-bank (Sulaiman and Lee, 2012 and Pope, 1980). Another important feature is the surfactant's ability to promote flow properties, improving phase relative permeability in the ultra-low IFT zones.

Surfactant EOR operations are often expensive and a well developed planning stage prior to making a costly investment decision is invaluable in terms of NPV calculations. Being able to accurately predict how the reservoir fluid flow and surface production rates will be affected by injecting surfactants is paramount for maximising revenue. Because of the industry's need for such tools to predict reservoir recovery, not purely for chemical EOR processes, several operator and service companies have created their own simulation tools. Among these are SINTEF Applied Mathematics in Oslo, which have developed an open source reservoir simulation program. The Matlab Reservoir Simulation Toolbox (MRST), has been developed in conjunction with the Open Porous Media (OPM) initiative to function as an efficient testing platform for new discretisation and solution methods in reservoir simulation applications (Lie et al., 2012).

The objective in this thesis is to modify the MRST source to include a module for simulating chemical EOR processes using surfactant. In order to do this, the physical effects that will result in improved flow properties and an increased oil recovery have to be implemented. The model is based on the surfactant model which is included in Schlumberger's Eclipse®, with which the simulation results will be verified against. Also, a feature of MRST is the industry standard input formatting support (Lie et al., 2012) and for an easy transition from MRST to Eclipse®, the surfactant model specific keywords implemented in Eclipse® were also implemented in the model.

Chapter 2

Fundamentals of Flow in Porous Medium

In this chapter, fundamentals of fluid flow in porous medium are presented to better give understand of various effects and governing equations used later in this thesis.

2.1 Relative and Absolute Permeability

When fluids are produced from a reservoir, they move through the interconnected pore space towards the well. The ability of the porous medium to transmit fluid throughout the interconnected pore system is characterized by its permeability (Zolotukhin and Ursin, 2000). Permeability is clearly a dependent on reservoir porosity, but also of pressure, though this dependency is relaxed in most reservoir engineering applications. Absolute permeability refers to single-phase flow in the porous media. The rock permeability is a proportionality coefficient, relating fluid flow rate through a porous medium to viscous pressure drop. This relationship was proposed by Henry Darcy in 1856.

Due to the nature of porous rocks, the permeability is often not isotropic, meaning that fluids are more easily conducted through the pores in some directions than others. Because of this, permeability is often expressed as a symmetric and positive definite tensor in reservoir simulation applications. The principal directions are often assumed to fall along the axis in a Cartesian coordinate system, which results in a diagonal tensor. The most common orientation is that permeability in Z direction is perpendicular and X and Y are parallel to the sediment layers.

$$\mathbf{K} = \begin{bmatrix} k_{xx} & k_{xy} & k_{xz} \\ k_{yx} & k_{yy} & k_{yz} \\ k_{zx} & k_{zy} & k_{zz} \end{bmatrix}$$

When a porous medium is saturated with two or more phases, as is the case in most reservoir engineering applications, its ability to conduct fluids is altered, reducing each fluid's *effective* permeability. A fluid's *relative* permeability is a dimensionless ratio of effective permeability to absolute permeability.

$$k_{r,i} = \frac{k_{e,i}}{\mathbf{K}}$$

Where:

- $k_{r,i}$ is the relative permeability of phase i
- $k_{e,i}$ is the effective permeability of phase i

Phase relative permeability is strongly correlated with phase saturation in the porous medium. It is important to note that relative permeability is a semi-empirical parameter, which does not describe the physical effects occurring on pore scale. Rather it describes how these effects are correlated with the phase saturation in a certain volume. Behrenbruch and Goda (2006) presents the Corey type equations which can be used to correlate relative permeability of oil and water to water saturation in a porous rock.

$$k_{rw}(S_w) = k_{rw}^0 S_{wn}(S_w)^n \tag{2.1}$$

$$k_{ro}(S_w) = k_{ro}^0 (1 - S_{wn}(S_w))^n \tag{2.2}$$

Where:

- S_{wn} is the normalized water saturation
- $k_{r,i}^0$ is the end-point relative permeability of phase i
- n is the Corey curve exponent

2.2 Darcy's Law

Darcy's law relates fluid velocity and viscosity to pressure drop over a porous medium. It is a vectorial equations, expressing fluid flow as a vector.

$$\vec{v_i} = -\frac{\mathbf{K}}{\mu_i}\nabla\Phi_i \tag{2.3}$$

Where:

- $\vec{v}$ is the velocity vector of fluid i
- $\mathbf{K}$ is the permeability tensor from section. 2.1.
- μ is the viscosity of fluid i
- Φ is the potential of fluid i which is equal to the sum of pressure and hydrostatic head, $p + \rho_i g\nabla z$, where ρ_i is the density, $\vec{z}$ is the depth vector, p is the pressure and $\vec{g}$ is the gravity acceleration vector.

Darcy's law is important for understanding how fluids flow through porous medium, and is used to derive flow equations used in commercial reservoir simulation programs. Refer to chapter 6 for details on use.

2.3 Capillary Pressure

In a porous medium, there most often exist two or more immiscible fluids at the same time. At the interface between these fluids, there exist a finite pressure difference because of interfacial tension, referred to as *capillary pressure*. The capillary pressure may be defined as the pressure difference between two immiscible fluids over a curved interface such as a sphere or a droplet. By convention, it is defined as oil pressure minus water pressure ($p_o - p_w$), but if any other fluids than oil and water are used it is defined as

$$p_c \equiv p_{nw} - p_w \tag{2.4}$$

Where:

- p_c is the capillary pressure
- p_{nw} is the non-wetting phase pressure
- p_w is the wetting phase pressure

Capillary pressure is related to interfacial tension (IFT) by Young-Laplace equation and presented by Adamson and Gast (2010).

$$p_c = \sigma_{i,j} \left(\frac{1}{r_1} + \frac{1}{r_2} \right) \tag{2.5}$$

Where:

- $\sigma_{i,j}$ is the IFT between the two phases i and j
- $r_{1,2}$ is the principal radii of curvature of an interface

Eq. 2.5 can be applied to calculate capillary pressure between oil and water in pore systems, where r_1 and r_2 are the principal radii of the pore. Although this equation is not extensively used in reservoir engineering applications, it is useful for seeing the effect of increasing/decreasing pore size and understanding the capillary pressure/water saturation relationship during drainage and imbibition. Various geometrical variations for the capillary pressure/pore size relationship are used in pore-network simulation softwares to describe the physics occurring at the pore-scale during a displacement process (Nilsen et al., 1996).

Chapter 3

Reservoir Recovery

A thorough evaluation of recovery methods is a vital part of the field development work essential for a successful production. Understanding how different recovery methods will affect the reservoir and what the expected recovery potential is, is crucial. In order for reservoir engineers to make the best decisions for a cost effective production, understanding the nature of oil and gas recovery is important. Dake (1985) distinguishes between two types of oil recovery: *1) Primary* and *2) supplementary recovery.* A perhaps more common distinction is *1) primary*, *2) secondary* and *3) tertiary recovery.* Tertiary recovery is what is commonly referred to as EOR methods.

1. Primary recovery is the first production from a reservoir, extracting the 'easy oil' by the means of natural driving mechanisms. It utilizes the energy available in the reservoir to produce hydrocarbons, exploiting fluid and rock compressibility to expel oil and gas from the pore space by decreasing the pressure. The recoverable potential is often very limited and it is not a strategy maintained for long periods of time.

2. Where primary recovery aims at producing hydrocarbons by lowering the pressure, secondary recovery aims at maintaining production rates by raising or maintaining an elevated reservoir pressure. This is commonly done by injecting either water or gas in large quantities into the reservoir. The injected water or gas also displaces the hydrocarbons in place, "sweeping" the reservoir.

3. Tertiary recovery is recovery processes initiated after the onset of secondary recovery processes. It involves a number of different methods.

3.1 Secondary Recovery by Water Injection

The water injected into the reservoir during a secondary recovery process serves two purposes: *1)* acting as pressure support to maintain reservoir pressure and *2)* functioning as a way to displace the oil in the pore system by pushing it ahead of the injected water front.

Buckley and Leverett (1941) points out that the displacing fluid does not act as a 'piston' pushing *all* the oil in front of the injected water, but rather flows simultaneously through the same pores. This implies that the relative permeability of the displaced fluid in presence of the displacing fluid is affecting how both fluids flow and then the overall efficiency of the flooding.

To better understand the effects of a water injection, it is beneficial to study how two immiscible fluids flow together through the porous medium. By applying Darcy's equation for the two flowing fluids in a one dimensional reservoir with an angle α, the *fractional flow equation* can be derived.

$$q_o = -\frac{k_{xx}k_{ro}A}{\mu_o}\left(\frac{\partial p_o}{\partial x} + \rho_o g \sin\alpha\right) \tag{3.1}$$

$$q_w = -\frac{k_{xx}k_{rw}A}{\mu_w}\left(\frac{\partial p_w}{\partial x} + \rho_w g \sin\alpha\right) \tag{3.2}$$

Where:

- q_i is the volumetric flow rate of phase i
- A is the area available to fluid flow
- α is the dip angle of the flow direction

By recalling the definition of capillary pressure in eq. 2.4 and introducing a new parameter f_w, the expression can be solved for f_w.

$$f_w = \frac{1 + \frac{k_{xx}k_{ro}A}{q_t\mu_o}\left(\frac{\partial P_c}{\partial x} - \Delta\rho g \sin\alpha\right)}{1 + \frac{k_{ro}}{\mu_o}\frac{\mu_w}{k_{rw}}} \tag{3.3}$$

Where $f_w = q_w/q_t$ and expresses what fraction of the total flow rate water accounts for and q_t is the total flow rate. In the case of horizontal flow and no capillary pressure, the expression can be reduced to:

$$f_w = \frac{1}{1 + \frac{k_{ro}}{\mu_o}\frac{\mu_w}{k_{rw}}} \tag{3.4}$$

In eqs. (3.3) and (3.4), k_{ro}, k_{rw} and P_c are all functions of water saturation, so f_w is also a function of water saturation, $f_w(S_w)$. This means that during an oil-water immiscible displacement, the flow rate fraction which is attributed water depends mainly on its own saturation. Among the topics significant for future discussion in this thesis, are the effect of relative permeability on the fractional flow. Generic relative permeability curves can be constructed by assuming the Corey type equations are valid (eqs. 2.1 and 2.2). The flow fraction of water changes as the relative permeability curves change (fig. 3.1). The first set of relative permeability curves predict a higher flow fraction of water at lower water saturations ($S_w \approx .2 \to .5$).

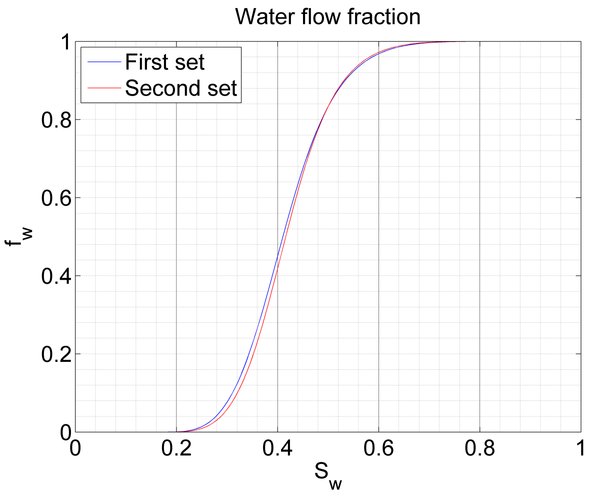


Figure 3.1: *Water flow fraction plotted against water saturation for two sets of relative permeability curves. Corey equation parameters for the two sets are 1)* $S_{wc} = S_{or} = 0.18, k^0_{ro} = k^0_{rw} = 0.8$, $n = 2.8$ *and 2)* $S_{wc} = S_{or} = 0.2, k^0_{ro} = k^0_{rw} = 0.7$, $n = 3$.

3.1.1 Buckley-Leverett Frontal Advance

As the fractional flow relationship for an oil-water immiscible displacement has been introduced, how the displacing and displaced fluids propagate through the reservoir during a water flood can be investigated. The equation below is derived by applying a mass balance for water over a control volume and is presented by Dake (1985). The following equation is known as the Buckley-Leverett equation, first derived by S. E. Buckley and M. C. Leverett in 1941. It states that the velocity of a plane of constant water saturation is proportional to the rate of change in the flow fraction of water at that particullar saturation.

$$\left.\frac{dx}{dt}\right|_{S_w} = \frac{q}{A\phi}\left.\frac{df_w}{dS_w}\right|_{S_w} \tag{3.5}$$

Where:

- ϕ is the porosity
- S_w is the water saturation

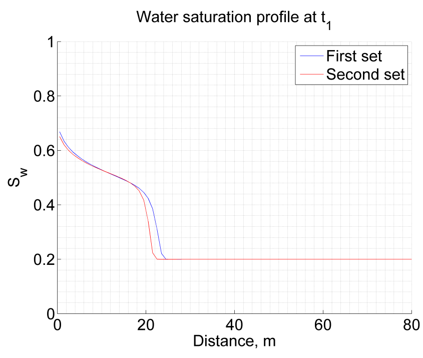
Considering eq. 3.5 and noticing the curving nature of the flow fraction curve in fig. 3.1, it is evident there exist a saturation value for which the velocity reaches a maximum value. The result of this, as explained by Pope, is that some saturation planes upstream travels faster than downstream planes. The faster saturation planes will eventually catch up to the slower moving saturation planes, and this will create a shock front where the saturation discontinuously drops from S_{wf} to S_{wi}. Upstream from the front, the saturation increase and the velocity decrease. To calculate the magnitude of the shock and to visualize the saturation distribution, a material balance is applied over the front.

$$v_{\Delta S_w} = \frac{q}{A\phi}\frac{f_{wf} - f_{wi}}{S_{wf} - S_{wi}} \tag{3.6}$$

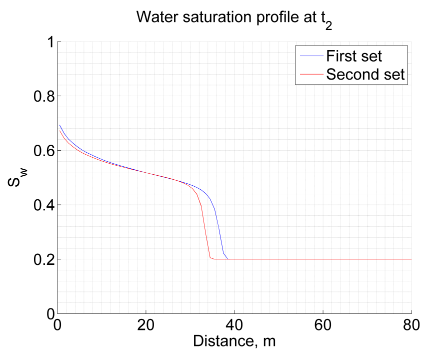
Where S_{wf} is the shock front water saturation.

Pope (1980) notes that at the contact between the saturation shock and the continuous saturation distribution, the velocities must be equal. The left hand side of eq. 3.7 represents the velocity of the shock front given by eq. 3.6. The right hand side is the velocity of a plane of saturation S_{wf} which travels at the same velocity. This can be solved for S_{wf} to obtain the shock front saturation.

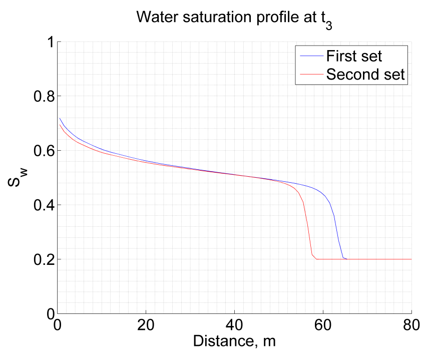
$$\frac{f_{wf} - f_{wi}}{S_{wf} - S_{wi}} = \left.\frac{\partial f_w}{\partial S_w}\right|_{S_w=S_{wf}} \tag{3.7}$$



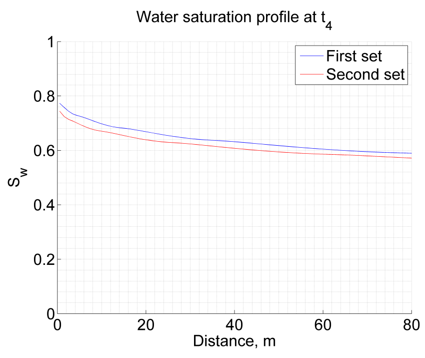
(a) *S_w profile at t_1*



(b) *S_w profile at t_2*



(c) *S_w profile at t_3*



(d) *S_w profile at t_4*

Figure 3.2: *Saturation profile for two different sets of relative permeability curves, visualising the Buckley-Leverett frontal advance. The first set uses $S_{wc} = S_{or} = 0.18, k_{ro}^0 = k_{rw}^0 = 0.8$, $n = 2.8$ and 2) $S_{wc} = S_{or} = 0.2, k_{ro}^0 = k_{rw}^0 = 0.7$, $n = 3$. Results from MRST reservoir simulation program*

3.1.2 Recovery Potential During an Immiscible Water Flood

Figure 3.2 shows the saturation profile at different times during a constant bottom hole pressure water flood in a homogeneous one-dimensional reservoir. The front propagating through the reservoir as water is continuously injected is spotted. Downstream of the front, the water saturation is at S_{wi}, and upstream of the front it is increase towards an upper value. As time goes on from fig. 3.2a on to 3.2d, a general trend is forming: The water saturation approaches a uniform distribution equal to an upper target value. This is the largest possible water saturation during an immiscible flood, and injecting additional water will not displace any of the remaining oil.

The two different saturation profiles correspond to the fractional flow curve plotted in fig. 3.1, with the same Corey equation parameters. As the relative permeability curves are straightened out (decreasing the exponent, n), end-points move outwards and end-point relative permeability increased, the water flood changes characteristics (fig. 3.2): At the given times plotted in fig. 3.2, the water flooding with the more straight curves and largest end-points is more effective, displacing more oil efficiently.

The remaining oil saturation can be determined in connection with the relative permeability curves. As the oil saturation decreases, the relative permeability of oil decreases. When it reaches zero, the remaining oil can not move and thus the end point on the relative permeability curve determine residual saturations.

Considering this fact and the displacement fronts in fig. 3.2, it is evident that shifting the end point saturation and relative permeabilities as well as straightening the curves will result in increased oil recovery.

Chapter 4

Enhanced Oil Recovery

EOR refers to the process of further improving the recovery of the reservoir beyond the limitations imposed by primary and secondary recovery. The incremental oil production as a result of an EOR process, can be expressed as the product of two entities (eq. 4.1), *1)* sweep efficiency and *2)* microscopic sweep. Sweep efficiency is how well the reservoir rock is exposed to the fluids injected during the EOR process, and subsequently displaces the in place volumes. Microscopic sweep is how well the EOR process is able to displace remaining hydrocarbons on pore scale. These two terms are fundamentally different, and have EOR processes designated to enhancing each one separately.

$$N_p = E_s E_m \tag{4.1}$$

Where:

- N_p is the dimensionless incremental oil
- E_m is the microscopic sweep
- E_s is the sweep efficiency

From an academic perspective, an increase in oil recovery is always interesting. From the industry perspective however, the EOR process has to be considered in relation to cost and revenue. EOR processes are expensive, so the expected returns have to be thoroughly calculated. Even though there might be a potential for increased oil recovery, the cost of the correct EOR measures may not be justified by the expected returns.

4.1 EOR Methods

There are a variety of methods constituting the EOR terminology, involving methods such as injection of gases under miscible conditions, thermal recovery methods with steam injection and the injection of various chemicals. The general idea is to create more favourable flow conditions to allow for better production. Some EOR methods are better suited than others, and the correct EOR method should be carefully chosen to best suit the reservoir conditions under which they are applied. The main classifications of EOR methods are listed below.

4.1.1 Thermal Methods

Thermal recovery methods are most commonly applied to improve oil recovery from heavy oil reservoirs, where the high crude oil viscosity makes flow conditions unfavourable. The idea is to inject hot fluids, such as steam or hot water, to reduce crude oil viscosity and promote flow. A common method of thermal recovery is *steam assisted gravity drainage* (SAGD). As explained by Mobeen and Kharrat (2011), this methods employs two horizontal wells placed in a vertical plane. The steam is continuously injected from the upper most well, condensing and heating up the oil in the area around it. As the oil is heated up, the viscosity drops and the oil flows more easily towards the producing well below.

4.1.2 Miscible Methods

When two fluids are completely miscible, the IFT between the two fluids is zero. As will become apparent in the next section, the absence of capillary forces will increase the recovery potential significantly. Miscible methods involves injection of a fluid that completely mix with the in situ hydrocarbons. For a gas field, ordinary gas injection will behave in a miscible way since gases mix. Injecting gas in an oil field will not have the same effect and will result in immiscible flow conditions, as described in section 3.1. For a certain combination of oil and gas (in terms of composition), there exist a lower pressure limit over which miscible conditions will arise. This is known as the *minimum miscibility pressure* (MMP) and is significant in EOR applications. Exploiting this mechanism, increasing the pressure of the injected gas will form miscible displacement which is applied to improve microscopic sweep.

4.1.3 Chemical Methods

In chemical recovery processes, a variety of chemicals are utilized to improve oil recovery, involving the injection of polymers, surfactants, alkaline solution and/or various combinations of these. Polymers are added to the injected water to increase its viscosity and create more favourable mobility ratios, thus improving sweep efficiency. Surfactant injection utilizes the surfactant's ability to reduce IFT between the oil and water, similar to miscible methods. This will result in less trapping of remaining oil and increased recovery. Alkaline injection exploits the chemical properties in the crude oil to produce surfactants in situ. Surfactant and alkaline injection both aim to increase microscopic sweep. In field applications of chemical EOR processes, polymer slugs are often used in combination with a surfactant slug for proper mobility control.

4.2 Increasing Microscopic Sweep

As pointed out in section 3.1.2, the maximum recovery potential during an immiscible water flooding is determined by the end-point saturation of oil. Additional water flooding at these conditions will only displace the already injected water, leaving trapped oil behind.

Batycky and McCaffrey (1978) expresses the microscopic sweep in eq. 4.1 as:

$$E_m = \frac{1 - S_{wi} - S_{or}}{1 - S_{wi}} \tag{4.2}$$

Where:

- S_{wi} is the irreducible water saturation
- S_{or} is the residual oil saturation

From eq. 4.2 it is evident that decreasing the residual oil saturation will result in an increase in microscopic sweep.

The reason why the remaining oil is immobile, is because the forces holding the oil drops in place exceeds the exerting forces trying to mobilize it. This phenomenon is known as capillary trapping.

Imagine a drop of oil, trapped in a pore with water on both sides. To more easily understand the mechanisms behind capillary trapping, a force balance is derived. Forces trying to mobilize the oil drop is exerted from the injected water as a viscous pressure drop, equal to $L \times \partial P_w / \partial x$, where L is the length of the oil drop. The forces trying to keep the oil drop in place is cause by the pressure difference across the oil/water interface due to surface tension, which can be expressed as $2\sigma/r$. This is a special case of eq. 2.5 assuming $r_1 = r_2$. The pressure drop is derived from Darcy's law in eq. 2.3. In order to move the trapped oil drop, the viscous forces have to locally exceed the capillary forces.

$$L \frac{\partial p_w}{\partial x} \geq \frac{2\sigma}{r} \tag{4.3}$$

The dimensionless quantity, N_c, is introduced as the ratio of viscous to capillary forces. It describes the in situ conditions for mobilization of trapped oil.

$$N_c = \frac{F_v}{F_c} = \frac{v\mu}{\sigma} \tag{4.4}$$

Where:

- N_c is the capillary number
- F_v are the viscous forces
- F_c are the capillary forces
- v is the flow velocity of the *displacing* fluid

According to Sheng (2011), the capillary number in eq. 4.4 should be considered a semi-empirical parameter and the author presents several other forms of N_c. These variations include wettability effects, Darcy velocity, permeability and buoyancy effects. Regardless, they all express the ratio of viscous forces to capillary.

It is important to note that this trapping mechanism is represented on the pore scale. Heterogeneity effects are very difficult to express in a way such that they are correctly accounted for. It is therefore important to be able to sufficiently describe how viscous and capillary forces relate to mobilized oil in a way that can be used in engineering applications. As pointed out by Bashiri and Kasiri (2011), when performing a simulation on a reservoir model, local heterogeneities and the intricacy of trapping mechanisms on pore level are not easily accounted for and are often expressed as empirical correlations. The *capillary de-saturation curve* (CDC) is such a correlation and links residual oil saturation to the capillary number, representative for each type of reservoir rock under different wetting conditions (fig. 4.1).

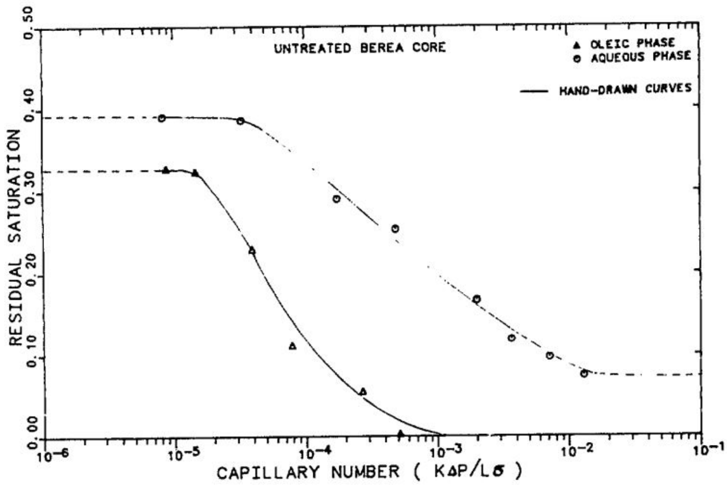


Figure 4.1: *Capillary de-saturation curve for an untreated Berea core, clearly showing the effect of increased capillary number. Above a certain critical value, N_c^*, the residual oil saturation decreases - taken from Delshad et al. (1986)*

From fig. 4.1 a region of constant residual oil saturation can be seen at low values for N_c. Displacement with a capillary number in this range is often referred to as "capillary dominated" flow and is commonly assumed to be the case for water injections. Above a critical capillary number, N_c^*, the trapped oil starts to mobilize and the residual oil saturations drops. Note that in fig. 4.1, the capillary number is defined as $K\Delta P/L\sigma$, which differs from the definition in eq. 4.4.

From equation 4.4, the capillary number can be increased to exceed the critical value by either increasing the mobilizing viscous forces, F_v or decreasing the restraining capillary forces, F_c.

Increasing the viscous forces is be done by applying a greater pressure drop, i.e. increasing injection pressure, decrease production pressure or increasing the viscosity of the injected fluid. The effect of this is quite limited and will not result in an significant increase in N_c. The alternative is to decrease the capillary forces and this the primary objective in a surfactant injection EOR.

Chapter 5

Surfactants

In this chapter, an introduction to surfactants in general is given. Chemical structure, phase behaviour, types of surfactants employed in EOR practice and the effect these has on flow conditions are among the topics discussed.

5.1 Chemical Structure

Surfactants are a class of molecules with a distinct chemical structure. They are composed of two parts: a non-polar lipophilic tail and a polar hydrophilic head. Lipophilic and hydrophilic refers to the two parts' affinity to oil and water respectively, where lipophilic means a high affinity to oil ("oil loving") and hydrophilic means a high affinity to water ("water loving") [1].

This chemical structure ensures that the surfactant molecules have good solubility in both oleic and aqueous phase[2]. Rosen and Kunjappu (2012) explains that when the surfactant is added to the oleic phase in a oil/water two-phase system, the hydrophilic group may distort the structure of the oil which in turn will increase the free energy of the system. The system will then respond by minimizing the contact with the hydrophilic group in order to reduce the free energy. Equivalently, when the surfactant is added to the aqueous phase, lipophilic group will

[1]Lipophilic and hydrophilic are equivalent to hydrophobic and lipophobic respectively.

[2]It is distinguished between water and aqueous phase. Aqueous refers to the phase and water refers to the water *component* in the aqueous phase. The same logic applies to oil and oleic phase. This distinction is made due to the possibility of dissolving other components in both phases, though the difference is not emphasized in this study.

distort the structure of the water and the same reaction is expected. Because of these reactions, the surfactant molecules are adsorbed at the oil/water interface and replace the initial water and oil molecules.

5.1.1 Reduction of IFT

Fluid molecules close to or at a fluid-fluid interface have less kinetic energy than molecules in the bulk of the fluid. This is because of the intermolecular surface forces between fluids holding them in place. As a result, molecules close to or at the interface are in a state of higher potential energy than molecules in the bulk. It is therefore required a finite amount of energy to move the molecules from the bulk to the interface, i.e. to increase the surface area of the fluid. The potential energy for a fluid molecule at the interface is larger than for a molecule in the bulk by an amount equal to *1)* the interaction energy with the same molecule in the bulk minus *2)* the interaction energy with a non-similar molecule across the interface. Following the convention applied by Rosen and Kunjappu (2012), the potential energy for a fluid molecule a at the interface which is interacting with a similar molecule in the bulk and a non-similar molecule b across the interface, may be expressed as $A_{aa} - A_{ab}$. The potential energy for a molecule b may be expressed equivalently as $A_{bb} - A_{ab}$. Here A_{aa} and A_{bb} is the interaction with a similar molecule in the bulk and A_{ab} is the interaction with a non-similar molecule across the interface. The potential energy of all fluid molecules at the interface is larger than for those in the bulk equal to the sum $A_{aa} - A_{ab} + A_{bb} - A_{ba}$, which adds up to $A_{aa} + A_{bb} - 2A_{ab}$. The surface energy per unit area, which is the IFT, is then:

$$\sigma_l = \sigma_{aa} + \sigma_{bb} - 2\sigma_{ab} \tag{5.1}$$

Where σ_{aa} and σ_{bb} are the surface free energies per unit area and σ_{ab} is the interaction energy over the interface. The interaction energy σ_{ab} depends on the similarity of the two molecules, so when the molecules are different, as for two pure fluids, the interaction energy across the interface can be neglected. However, when surfactant molecules are introduced to fluid-fluid interface, as explained in the previous section, the interaction energy between the two, now similar, molecules, is significant. This way, the surfactants reduce the IFT between the two fluids.

Increasing the surfactant concentration in the system will thus cause the IFT to drop. However, as more and more surfactant molecules are introduced to the fluid-fluid interface, it will gradually "fill up", and at some critical concentration, the additional surfactant molecules will start aggregating into micelles while a decrease in IFT is *not* observed. The formation of micelles is also a way for the system to minimize the contact with the non-soluble groups in the molecules. This concentration is referred to as the *critical micelle concentration* (CMC), and any further increase in surfactant concentration will only cause more micelles to form. The CMC is typically very small, so for nearly all surfactant flooding applications, the surfactants are in micelle form, either as *1)* water external and oil internal micelles or a *2)* water internal and oil external micelles, depending on in which phase the micelles partition out in.

5.2 Microemulsion Phase Behaviour

The surfactant's ability to dissolve in either water or oil, is strongly dependent on the salinity of the brine. For low salinities, the surfactants exhibit good solubility in aqueous phase, but poor solubility in oleic phase. The surfactant molecules then tend to partition out in the aqueous phase and micelles are formed here. We then have a two-phase system: *1)* excess oleic phase which is free of surfactants and *2)* a water external - oil internal micellar solution, hereby referred to as a *microemulsion*. This microemulsion phase contain water, surfactant and oil. Since the microemulsion is in the aqueous phase, which is denser than the oil, it is referred to as a lower phase microemulsion.

For high salinities, the surfactant displays the opposite behaviour. It exerts good solubility in oleic phase and a poor solubility in aqueous phase. By the same logic, the surfactant molecules now partition in the oleic phase and micelles are formed. Again, we have a two-phase system, only now with *1)* excess aqueous phase which is free of surfactants and a *2)* oil external - water internal microemulsion phase which contains oil, surfactants and water. Since the microemulsion is in the oleic phase which is lighter than the water, it is referred to as a upper phase microemulsion.

The effects of salinity on microemulsion phase behaviour is best investigated with the use of a ternary diagram, using the pseudo components *1)* water, *2)* oil and *3)* surfactant. Microemulsion phase behaviour for different salinity values is plotted in fig. 5.1 with increasing salinity from left to right.

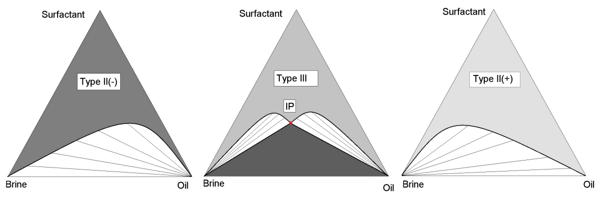


Figure 5.1: *Ternary diagrams showing microemulsion phase behaviour at increasing salinity levels. At some intermediate salinity values, the formation of a three-phase region is possible.*

From fig. 5.1, the two situations described above can be seen at each of the salinity values. The areas beneath the curves are the two-phase regions for type II(-) and type II(+) microemulsions. The shaded regions above these curves denote the one-phase region. At some salinity range between type II(-) and type II(+), there exist a *type III* microemulsion. At these salinity values, up to three phases may be observed. The light shaded area is the one-phase region in this regime. The dark shaded region is the three-phase region where we have excess water, excess oil and a microemulsion solution with both types of micelles (oil internal and oil external). The microemulsion composition is given at the invariant point, denoted *IP*. The blank areas between the one- and three-phase regions are the two-phase regions.

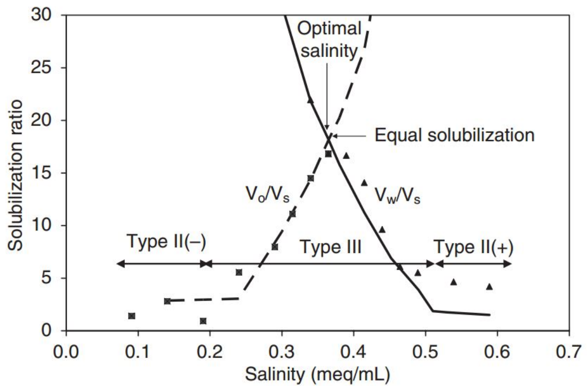


Figure 5.2: *Solubilization ratio for oil and water for different salinity values in a salinity scan. At some intermediate salinity value, oil and water solubilization ratios are equal. This salinity is the optimum salinity. Taken from Shen et al. (2006)*

The salinity regions for which a type II(-), type III and type II(+) is observed are identified through a *salinity scan*, during which brine/surfactant solutions at varying values of salinity are added to the investigated reservoir oil. After phases have properly settled and microemulsions formed, the ratio of solubilized water and oil volume to surfactant volumes are calculated. This is known as the solubilization ratio, and is plotted against salinity value (fig. 5.2). The "optimum salinity" is where the surfactants ability to solubilize water and oil is equal and is where the IFT for the entire system reaches its minimum. This is the target salinity value used in surfactant flooding.

5.3 Types of Surfactants

Sheng (2011) lists four main categories of surfactants, distinguished by different properties:

- Anionic surfactants
- Cationic surfactants
- Non-ionic surfactants
- Zwitterionic surfactants

The anionic surfactants have a negative charge and are thus adsorb very little onto the negatively charged sandstone surface. Anionic surfactants are also able to effectively reduce IFT, which makes them the most used type of surfactant in EOR applications.

As opposed to anionic surfactants, the cationic surfactants have a positive charge which promotes adsorption onto the rock surface and are therefore generally not used for EOR in sandstone reservoirs.

Non-ionic surfactants does not have an electric charge. They are more tolerant to high salinities than the anionic surfactants, but their ability to reduce IFT between water and oil is not as good. Non-ionic surfactants are often applied as co-solvents in a surfactant injection scheme.

Zwitterionic surfactants contain two active groups and can be different combinations of anionic-, cationic- and non-ionic groups. Lake (1989) refers to this type as amphoteric surfactants.

5.4 Use of Surfactants in EOR

Section 4.2 presented the capillary de-saturation curve and its significance in correlating viscous and capillary forces to residual oil saturation. Eq. 4.4 and fig. 4.1 show that decreasing the surface tension between the two immiscible fluids will help mobilize trapped oil.

Oil/water IFT typically lies around 20 - 30 mN/m. A good surfactant will have the ability to reduce the IFT to the range of 1E-4 - 1E-3 mN/m, an increase of three to four orders of magnitude in N_c, which will put it above the critical value N_c^*. From fig. 4.1, this will significantly lower the residual oil saturation. It is the ability to strongly reduce IFT that make surfactants such a viable option in EOR.

5.4.1 Low IFT Oil-Water Relative Permeability Curves

Not only does lowering the IFT have an impact on the residual oil saturation, it also alters the two-phase flow properties. As capillary number increase and residual saturation decrease, the end-points on the relative permeability curve are obviously shifted. A lower IFT will cause changes to the oil-water relative permeability which is not limited only to this shifting, but also changing the shape of the curves. Harbert (1983) presented the results of a series of normal range IFT and ultra-low IFT flooding experiments. The general trend was that the curves were found to shift upwards to more straight lines, which suggest a less degree of interference between the flowing fluids. Shen et al. (2006) presented a study where they correlated oil-water relative permeability to IFT. What they found was that there exist a critical value for IFT, σ_c, for which the relative permeability curves behaved differently above and below. For normal range IFT, $\sigma_c < \sigma$, the relative permeabilities for oil and water were unaltered. As the IFT goes below σ_c, an increase in relative permeability was experienced. The experimental observed and associated fitted data are presented in fig. 5.3 at water saturation of 0.5.

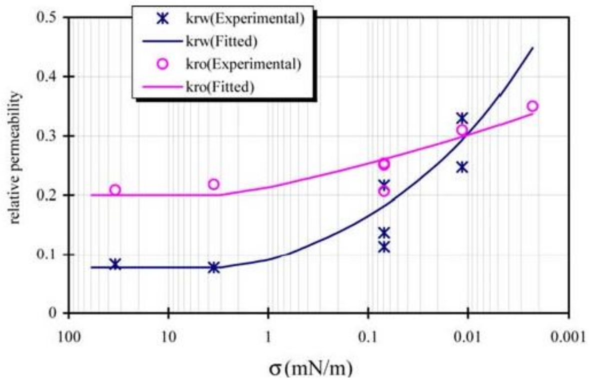


Figure 5.3: *Oil and water relative permeability plotted against IFT for $S_w = 0.5$. At some critical value for IFT, σ_c, an increase in relative permeability is observed. Laboratory data is plotted along with fitted data. Presented by Shen et al. (2006)*

Shen et al. (2006) normalized and plotted relative permeability values against normalized saturation values to see the overall trend.

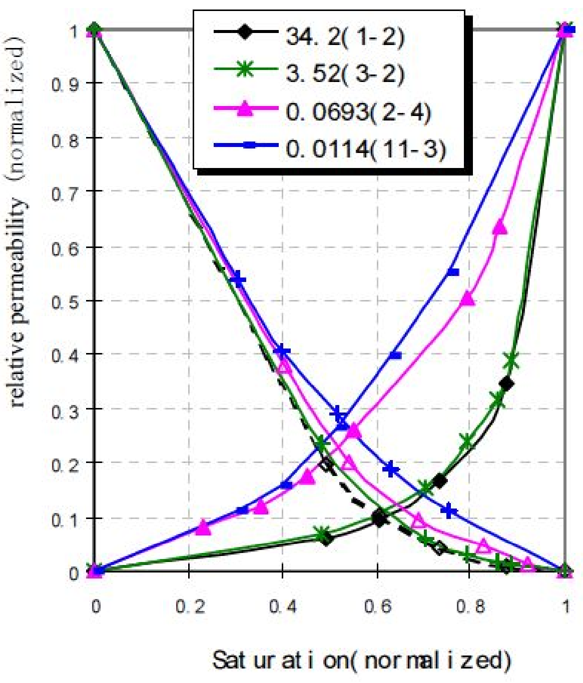


Figure 5.4: *Normalized relative permeability plotted againts normalized water saturation. The overall trend as IFT decreases is that the relative permeability curves straighten out. Presented by Shen et al. (2006)*

From the discussion presented by Shen et al. (2006), it is not explicitly noted whether a type II(-) or type II(+) system was investigated. Since the microemulsion phase flow behaviour is affected by the micellar structure (oil internal / oil external), this would be an important aspect to relate to the presented results. Also the presence of a third phase, type III system, would affect the results.

5.5 Oil Production During Surfactant Injection

When the injected surfactant solution comes into contact with the residual oil, the IFT between the oil and the injected brine is reduced in the exposed regions. This causes the contacted trapped oil to be mobilized and start flowing. The region contacted by surfactant-brine solution will have (according to discussion in section 5.4.1) a higher relative permeability than in the regions down stream of the injected surfactant, in the region in which IFT has yet to be reduced.

From Darcy's law, since the velocity is proportional to relative permeability, oil velocity will increase in these regions. By applying similar logic as in the discussion of shock front in Buckley-Leverett displacement, the faster moving oil will now "catch up" to slower moving oil in unaffected regions. Because of this, a new shock will form. This shock and its corresponding saturation is referred to as the *oil bank*. Pope (1980) presented the same results, derived from a fractional flow perspective.

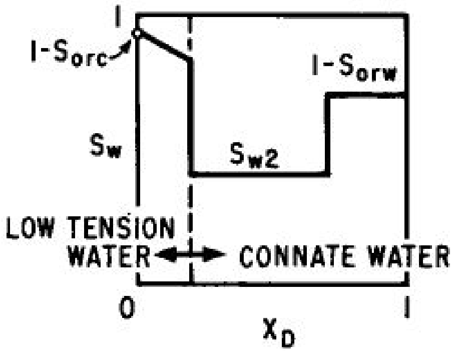


Figure 5.5: *Saturation profiles for tertiary recovery surfactant injection. The oil bank saturation can be seen at S_{w2} and the initial water saturation after the water flood is $1 - S_{orw}$. The saturation region left of the oil bank is the region with ultra-low IFT. The new residual oil saturation is S_{orc}. Figure taken from Pope (1980)*

When a surfactant flood is applied in a tertiary recovery scheme, the water is at maximum saturation, $1 - S_{orw}$. Injecting surfactants at these conditions will result in the formation of the oil bank, resulting in a region of lower water saturation (higher oil saturation) between the injection front and the areas of $1 - S_{orw}$ (fig. 5.5). The oil bank saturation is S_{w2}. This is also observed in field applications of surfactant flooding: when surfactant injection is applied as tertiary recovery, the incremental oil production is observed as a "slug" of oil.

Chapter 6

Numerical Simulation of Two-Phase Flow

Equations describing the physics in a porous medium during fluid flow are well known and can be expressed with a set of partial differential equations (PDE). As these mathematical models become increasingly more complex, there does not exist an analytical solution. Because of this, the equations are spatially and temporally approximated to make them applicable in a computer solution scheme, where they are *discretized* and solution values calculated at computational *nodes* throughout a simulation grid, rather than as continuous functions. The grid in which the computational nodes are placed, is a discrete representation of the problem domain, i.e. the reservoir. The idea is to divide the geological model into a number of *cells*, in which the solution variables such as pressure and saturation, are held constant. The placement of computational nodes in the grid and discretisation methods of the governing PDEs are done in accordance with the specific numerical solution method applied in the simulation software.

This chapter will focus on the mathematical aspect of the reservoir simulation program used in this thesis. Derivation of governing partial differential equations, how these equations are discretized and applied in a grid model and how they are used to assemble a system of equations is thoroughly discussed. This is followed by a detailed walk-through of the how the governing equations are used in the Newton-Raphson method to obtain a solution. A short presentations of alternative solution methods for PDEs are also given.

6.1 Matlab Reservoir Simulation Toolbox

The Matlab Reservoir Simulation Toolbox (MRST) is an open-source program developed in conjunction with the Open Porous Media initiative by SINTEF Applied Mathematics in Oslo. The purpose of MRST is to function as an efficient testing platform for new implementations of discretisation and solution methods in reservoir simulation applications. It contains modules for generating structured and unstructured grids, industry standard input format support, a complete set of functions and routines for handling fluid parameters and wells- and boundary conditions, post-processing and visualization tools for 2D and 3D scalar cell and face data as well as a number of modules for numerical solution methods and optimization.

The objective in this thesis is to modify the source code of MRST to be able to investigate the effects of a surfactant injection. The goal is to mimic the surfactant model included in Schlumberger's Eclipse ®, with which the model will be verified against in chapter 8. This is done by incorporating numerous physical effects resulting form the presence of surfactant in the system, along side with several function calls, Eclipse format keyword implementations and various post-processing functions.

6.1.1 Model Description

The Eclipse surfactant model assumes a black-oil fluid representation. Here, the surfactant is assumed to only be dissolved in the aqueous phase[1] and is added to the injected water as a mass per volume concentration (kg/Sm^3). As the surfactant concentration propagates throughout the reservoir, it will result in a number of effects altering the flow properties. These effects will be covered in chapter 7.

As pointed out in the Eclipse Technical Description, Schlumberger (2011a), the surfactant concentrations in the system is updated fully-implicitly at the end of each time step, after water, oil and gas flows have been computed. The same methodology is adopted by MRST, although limiting the flowing fluids to oil and water only.

A thorough description of Eclipse's surfactant model and the use of model specific keywords is included in the Eclipse Technical Description and Eclipse Reference Manual, Schlumberger (2011b), and the implementation of these effects is given

[1]Oil dissolved in aqueous phase and water dissolved in oleic phase has been neglected. The terms are hereby used interchangeably

in chapter 7. In this thesis, the MRST 2012b release was used, but the code will be backwards compatible with previous versions of MRST.

6.2 Governing Flow Equations

The governing partial differential equations used to describe fluid flow in a porous medium can be derived from the mass-balance principle for each phase present. The general conservation equation is used to derive the mass balance equation.

$$\{Mass\ in\} - \{Mass\ out\} \text{ +/- } \{Generation/Consumption\} = \{Accumulation\} \tag{6.1}$$

White (2008) introduces the The Reynolds transport theorem which is applied for mass conservation over a control. Applying the balance equation over an arbitrarily shaped control volume, Ω, with the confining boundary, $\partial\Omega$, yields:

$$\frac{\partial}{\partial t}\int_{\Omega} (c_{\beta i} S_i \rho_i \phi)\ d\nu + \int_{\partial\Omega} (c_{\beta i} \vec{v_i} \rho_i) \cdot \vec{n}\ dS = \int_{\Omega} c_{\beta i} Q_i\ d\nu \tag{6.2}$$

Where $c_{\beta i}$ is the concentration of *component* β in phase i. The first term on the left hand side is the rate of change in saturation of phase i in control volume Ω. The second term is the flux term, which accounts for mass transfer of phase i across the control volume boundaries. The term on the right hand side is the sink and source term. The governing units are mass per volume time and the source term, Q_i, is the average generation of mass of phase i per volume time.

$$Q_i = \frac{\rho_i q_i}{V} \tag{6.3}$$

Where V is the volume of the control volume in which the source is placed.

The densities in eq. 6.2 can be expressed at surface conditions by using the formation volume factors: $\rho_{i,sc} = \rho_i B_i$, where B_i has the units $\mathrm{m}^3/\mathrm{Sm}^3$. Re-writing eq. 6.2 with surface densities allows us to cancel them out, and further re-write eq. 6.2 as:

$$\frac{\partial}{\partial t}\int_{\Omega} \left(\frac{c_{\beta i} S_i \phi}{B_i}\right)\ d\nu + \int_{\partial\Omega} \left(\frac{c_{\beta i} \vec{v_i}}{B_i}\right) \cdot \vec{n}\ dS = \int_{\Omega} \frac{c_{\beta i} q_i}{V B_i}\ d\nu \tag{6.4}$$

Recalling Gauss' theorem from calculus, the surface integral of $\vec{v_i} \cdot \vec{n}$ can be written as the volume integral of $\nabla \cdot \vec{v_i}$.

$$\int_{\Omega} \nabla \cdot \vec{v_i} \; d\Omega = \int_{S_\Omega} \vec{v_i} \cdot \vec{n} \; dS \tag{6.5}$$

Inserting the relationship in eq. (6.4) to produce:

$$\frac{\partial}{\partial t} \int_{\Omega} \left(\frac{c_{\beta i} S_i \phi}{B_i} \right) \; d\nu + \int_{\Omega} \nabla \cdot \left(\frac{c_{\beta i} \vec{v_i}}{B_i} \right) \; d\nu = \int_{\Omega} \frac{c_{\beta i} q_i}{V B_i} \; d\nu \tag{6.6}$$

Dropping the integrals and terming the fraction $q_i / V B_i \;\; = \tilde{q}_i$

$$\frac{\partial}{\partial t} \left(\frac{c_{\beta i} S_i \phi}{B_i} \right) + \nabla \cdot \left(\frac{c_{\beta i} \vec{v_i}}{B_i} \right) \;\; = c_{\beta i} \tilde{q}_i \tag{6.7}$$

Where $\tilde{q}_i$ is the volumetric flow rate of phase i per volume in surface conditions.

This is the vectorial form of the general continuity equation used to describe the fluid flow problems in this thesis. The governing PDEs for the two-phase surfactant model are then the continuity equations for the two phases and surfactant. Assuming that $c_{oil/oleic}$ and $c_{water/aqueous}$ (concentration of oil and water in the oleic and aqueous phase respectively) is 1. The governing PDEs are then:

Oil:

$$\frac{\partial}{\partial t} \left(\frac{S_o \phi}{B_o} \right) + \nabla \cdot \left(\frac{\vec{v_o}}{B_o} \right) \;\; = \tilde{q}_o \tag{6.8}$$

Water:

$$\frac{\partial}{\partial t} \left(\frac{S_w \phi}{B_w} \right) + \nabla \cdot \left(\frac{\vec{v_w}}{B_w} \right) \;\; = \tilde{q}_w \tag{6.9}$$

Surfactant:

$$\frac{\partial}{\partial t} \left(\frac{S_w \phi c_s}{B_w} \right) + \nabla \cdot \left(\frac{\vec{v_w} c_s}{B_w} \right) \;\; = \tilde{q}_w c_s \tag{6.10}$$

6.3 Solving the Flow Equations in Discrete Form

The governing equations derived in section 6.2 are used to describe fluid flow in a porous medium in the reservoir simulation program. As mentioned in the beginning of this chapter, to solve sets of partial differential equations (without an apparent analytical solution), the equations have to be discretized and evaluated at computational nodes in a discrete problem domain. The derivation of discrete equations from the continuous PDEs is shown in this section.

6.3.1 The Finite-Volume Method

Schafer (2006) points out that an important property of the finite-volume method (FVM) is the balance principle which makes it well suited for solving conservation problems. This is one of the reasons why the finite-volume method is very applicable in reservoir simulation, and is used in this study. It is also the industry standard in reservoir simulation softwares.

In the finite-volume method, the discrete problem domain is composed of a set of control volumes, either as a structured or unstructured grid representation. A structured grid is way to discretize the problem domain by using cells with fixed shapes, as opposed to unstructured grids which may consist of grid cells with varying shapes. This way, the finite-volume is superior to the finite-difference method as it is able to represent the curved confining boundaries of the problem domain by using differently shaped grid cells as opposed to purely rectangular cells. Grid cells are often constructed as corner-point grids, where the confining region of each grid cell is determined by its corner points.

The solution values in a finite-volume method approach is assumed to be cell-constant quantities and the nodes where the solution values are computed are either placed in a *1)* cell-centred (CC) or *2)* vertex-centred (VC) scheme. In a CC configuration, the computational nodes are located in the cell centres and in a VC configuration, they are located at the cell vertices making up the control volume.

The general idea of the finite-volume method is to integrate the governing PDEs over the control volumes, Ω_i, with the confining boundary, $\partial\Omega_i$, which is a subset of the entire problem domain, $\Omega_i \in \Omega$. Consider the continuity equation in eq. 6.7. In order to derive the discrete form of this equation, a spatial integral over the control volume and a temporal integral from t to $t + \Delta t$ is applied.

$$\int_{t}^{t+\Delta t}\int_{\Omega}\frac{\partial}{\partial t}\left(\frac{c_{\beta i}S_i\phi}{B_i}\right)\,d\nu dt+\int_{t}^{t+\Delta t}\int_{\Omega}\nabla\cdot\left(\frac{c_{\beta i}\vec{v_i}}{B_i}\right)\,d\nu dt=\int_{t}^{t+\Delta t}\int_{\Omega}c_{\beta i}\tilde{q}_i\;d\nu dt \tag{6.11}$$

Again, invoking Gauss' theorem for divergence to express the second term on the left hand side as a surface integral of the flux (essentially going the opposite way from eq. 6.4 to 6.5):

$$\int_{t}^{t+\Delta t}\int_{\Omega}\frac{\partial}{\partial t}\left(\frac{c_{\beta i}S_i\phi}{B_i}\right)\,d\nu dt+\int_{t}^{t+\Delta t}\int_{\partial\Omega}\left(\frac{c_{\beta i}\vec{v_i}}{B_i}\right)\cdot\vec{n}\;dS\;dt=\int_{t}^{t+\Delta t}\int_{\Omega}c_{\beta i}\tilde{q}_i\;d\nu\;dt \tag{6.12}$$

The saturation terms are expanded and the equation is discretized in time and space to produce:

$$V\left[\left(\frac{c_{\beta i}S_i\phi}{B_i}\right)^{l+1}-\left(\frac{c_{\beta i}S_i\phi}{B_i}\right)^{l}\right]+\Delta t\left(\frac{f_i}{B_i}\right)=c_{\beta i}\tilde{q}_iV\Delta t \tag{6.13}$$

Where:

- l and $l{+}1$ denotes discrete times at which the quantities are evaluated
- Δt is the discrete step size between time l and $l{+}1$
- f_i is the flux of phase i across the grid cell boundaries

Dividing through by Δt to obtain the final form shown below. This equation is applied for oil, water and surfactant in all the grid cells constituting the solution domain and results in a system of equations, as will be shown.

$$\frac{V}{\Delta t}\left[\left(\frac{c_{\beta i}S_i\phi}{B_i}\right)^{l+1}-\left(\frac{c_{\beta i}S_i\phi}{B_i}\right)^{l}\right]+\left(\frac{f_i}{B_i}\right)=\left(\frac{c_{\beta i}q_i}{B_i}\right) \tag{6.14}$$

This is the discrete form of eq. 6.7 used in the two-phase model in this thesis.

Determining the flux term

The step from eqs. 6.12 to 6.13 defines the flux, f_i, as $\int_{\partial\Omega} \vec{v_i} \cdot \vec{n}\, dS$ and from Darcy's law, we have that the fluid velocity in a porous medium is governed by mobility, λ, and fluid potential as:

$$\vec{v_i} = -\lambda_i \nabla \Phi \tag{6.15}$$

Where:

- λ_i is the fluid mobility, defined as: $\frac{\mathbf{K} k_{r,i}}{\mu_i}$

Plugging eq. 6.15 into the flux term will result in:

$$f_i = -\int_{\partial\Omega} c_{\beta i} \lambda_i \nabla p \; dS \tag{6.16}$$

In the finite-volume evaluation of the governing two-phase fluid flow PDEs, a structured hexahedral[2] grid representation with sides aligned to the principal axes is used. A generic grid cell is shown in the figure below. This means that each control volume, $\Omega_n \in \Omega$, is made up of six planar faces, each face connecting control volume n to a neighbouring control volume across an interface, $\gamma_{n,m} = \partial\Omega_n \cap \partial\Omega_m$. With this control volume representation, the flux term for control volume n is the sum of the flux across all six interfaces.

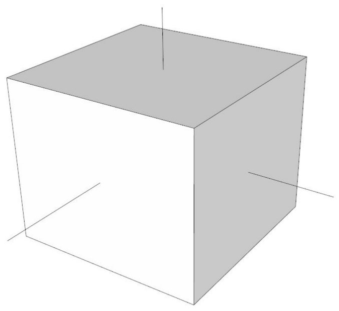


Figure 6.1: *Hexahedral control volume with planar faces and corresponding perpendicular vectors*

Across each planar face on the control volume in fig. 6.1 (assuming it does constitute any reservoir boundaries), the flux is calculated using the fluid potential

[2]A hexahedral is any polyhedron with six faces

in the centre of the grid cell and its neighbouring grid cells, i.e. in a cell centred configuration. This method is known as the *two-point flux approximation* (TPFA), and utilizes two points in the flux calculations. This is most often used for structured grids, as opposed to for unstructured grids where a *multi-point flux approximation* (MPFA), which employs several points, is used. Using the TPFA method, the integral in eq. 6.16 is evaluated for each face on block n.

First consider the flux across the interface between block n and m with normal vector in positive x-direction, $(1, 0, 0)^T$. The pressure gradient across this face $\gamma_{n,m}$ is then given below, ignoring gravity effects.

$$\delta p_{m,n} = \frac{2(p_m - p_n)}{\Delta x_m + \Delta x_n} \tag{6.17}$$

Where:

- Δx_m and Δx_n are the distance from the cell-centred computational node to the interface $\gamma_{n,m}$ in cell m and n respectively.
- $\delta P_{m,n}$ is the discrete pressure drop from cell centred computational nodes in cell m to n.

Inserting this relationship into the eq. 6.16, the result is:

$$f_{i,m\to n} = -\frac{2(p_m - p_n)}{\Delta x_m + \Delta x_n} \int_{\gamma_{m,n}} c_{\beta i} \lambda_i dS \tag{6.18}$$

The absolute permeability, **K**, is in most cases spatially varying throughout the reservoir and have different values associated with each principal direction in different grid cells. This, and the fact that permeability is not defined *at* the grid cell interfaces, complicates the calculation of the flux terms. This means that as the fluid travels from the centre of block m to the centre of block n, the mobility changes along the flow path. To account for this, a distance weighted harmonic average of the fluid mobilities in the two blocks is used (Lunde, 2007). Notation is adopted from this author.

$$\lambda_{i,m\to n} = \frac{(\Delta x_n + \Delta x_m)}{\left(\frac{\Delta x_n}{\lambda_{i,n}} + \frac{\Delta x_m}{\lambda_{i,m}}\right)} \tag{6.19}$$

Inserting eq. 6.19 in eq. 6.18 results in:

$$f_{i,m\to n} = -2(p_m - p_n)\int_{\gamma_{m,n}} c_{\beta i}\left(\frac{\Delta x_n}{\lambda_{i,n}} + \frac{\Delta x_m}{\lambda_{i,m}}\right)^{-1} dS \tag{6.20}$$

Because all quantities are held constant in a control volume, there are no variations in mobility within each grid block. This means that the integral over the surface of the interface will produce:

$$f_{i,m\to n} = -2c_{\beta i}A_{\gamma_{m,n}}(p_m - p_n)\left(\frac{\Delta x_n}{\lambda_{i,n}} + \frac{\Delta x_m}{\lambda_{i,m}}\right)^{-1} dS \tag{6.21}$$

Where $A_{\gamma_{m,n}}$ is the area of the interface. Transmissibility is defined such that the flux between grid block m and n across one face is calculated with the equation below. This term is summed up for all faces connecting two grid blocks.

$$f_{i,m\to n} = c_{\beta i}T_{i,m\to n}(p_m - p_n) \tag{6.22}$$

Where T is the transmissibility. The total transmissibility is the distance weighted harmonic average of grid cell *half-transmissibilities*, which is the transmissibility in one of the two blocks in which the fluid travels to get from m to n.

The total flux of phase i for a control volume is the sum of eq. 6.22 over all interfaces connecting it to a neighbouring cell.

$$f_i = \sum c_{\beta i}T_i\Delta p \tag{6.23}$$

Where Δp is the pressure difference between the grid cell centre and the centre of the neighbouring grid cell.

Inserting the transmissibility relationship into the discrete continuity equation, it is applied for oil, water and surfactant. Again assuming that the concentration of water and oil in the aqueous and oleic phase respectively, is 1. The discrete form of eqs. 6.8, 6.9 and 6.10 can then be written as eqs. 6.24, 6.25 and 6.26. Notice that pressures are evaluated at *l+i*, consistent with an implicit solution.

Oil:

$$\frac{V}{\Delta t}\left[\left(\frac{S_o\phi}{B_o}\right)^{l+1}-\left(\frac{S_o\phi}{B_o}\right)^{l}\right]+\sum T_o\frac{\Delta p}{B_o}^{l+1}=\left(\frac{q_o}{B_o}\right) \tag{6.24}$$

Water:

$$\frac{V}{\Delta t}\left[\left(\frac{S_w\phi}{B_w}\right)^{l+1}-\left(\frac{S_w\phi}{B_w}\right)^{l}\right]+\sum T_w\frac{\Delta p}{B_w}^{l+1}=\left(\frac{q_w}{B_w}\right) \tag{6.25}$$

Surfactant:

$$\frac{V}{\Delta t}\left[\left(\frac{S_wc_s\phi}{B_w}\right)^{l+1}-\left(\frac{S_wc_s\phi}{B_w}\right)^{l}\right]+\sum c_sT_w\frac{\Delta p}{B_w}^{l+1}=\left(\frac{q_wc_s}{B_w}\right) \tag{6.26}$$

The observant reader might have noticed that the phase in which the pressure is evaluated in, is not specified. In MRST, the pressure is evaluated in the oil phase, and the water pressure is calculated using the capillary pressure relationship. The Matlab code where the discrete conservation equations are implemented is included in appendix C.

Upwinding
An important concept when calculating fluxes in numerical applications, is *upwinding*. This means that low parameters are evaluated in the grid cell where the fluid flow *from*, i.e. where the potential is highest. This is a way to ensure a physically realistic solution as it e.g. limits the saturation to stay above the irreducible saturation values. The equations above may be extended to using fluid potential.

6.4 Other Solution Methods

There are also other ways to evaluate PDEs in a discrete environment. The two other used methods are the *1)* finite-difference method and the *2)* finite-element method. Though these are not implemented in MRST, alternative solution methods should be mentioned.

6.4.1 Finite-Difference Method

In the finite difference method (FDM), the solution variables are computed at computational nodes in the discrete grid, which are most commonly cell centre placed. The continuous governing PDEs are replaced by an approximation obtained using Taylor polynomials, to create a discrete function in time and space which is evaluated at these points (Peaceman, 1977). Due to the nature of the finite-difference method, it is difficult to implement for unstructured grids. This limits the usage of the finite-difference method as it is unable to accurately represent the curved confining regions of a reservoir.

6.4.2 Finite-Element Method

Similar to the finite-difference and finite-volume method, the computational domain is divided into discrete parts, now called *finite elements*. These elements may be assign a various shapes and are therefore excellent for constricting the discrete domain to the continuous problem domain boundary. The finite-element method is mostly applied in solid mechanics engineering application, and is not wide spread for continuum mechanics. Note though, that even though it is not a common approach, it does occur.

The goal of the finite-element method (FEM) is to approximate the continuous solution function as a set of N functions defined over sub-domains of the total solution domain. An *ansatz* function[3] to the solution function is defined, constituting a set of N linearly independent element equations. These functions fulfils the restrictions given in the boundary conditions and the ansatz function is inserted into the governing PDE. Also, introducing a set of *shape functions*, the ansatz function can be expressed as variable values at computational nodes in each element, E_i.

[3]An ansatz function is an approximation to the real solution function, e.g. how Taylor polynomials describe a continuous function.

The shape functions can be thought of as an on/off switch, similar to *Heaviside step function*, which is 0 for all computational nodes except for the ones defined in the input $\mathbf{x_i}$ vector for element E_i. All the nodes in the problem domain is enumerated and each sub-domain or element, E_i holds a sub-set of these nodes. This way, the ansatz functions can be defined in the global domain, assembled and solved for the solution variable. Similar to FDM and FVM, this method constitutes a system of equations which can be solved numerically by applying the appropriate solution algorithm.

6.4.3 Remarks

The different discretisation schemes all have advantages and disadvantages. The finite-element method more accurately represent the problem domain in terms of curved boundaries, but again is more slow in terms of computation time. The finite-difference method assumed squared or rectangular grid cells and thus poorly represent the curves boundaries of the geological reservoir model. Due to the simplicity of this approach, it is significantly faster than the finite-element approach and also provide good accuracy. The finite-volume is a sort of middle approach, with the ability to more accurately represent domain boundaries, although inferior to the finite-element method and also computationally faster. It is well suited for conservation problems such as the fluid flow presented above, and is the industry standard in reservoir simulation.

6.5 Solving the System of Equations

The equations derived in section 6.3.1 are evaluated for each grid block in the discrete domain, meaning that for each grid cell, $i = 1, 2, 3, \ldots, N$, there are three calculated values associated with it. By computing the flux terms, values for pressure, water saturation and surfactant concentration in one cell will be related to values in adjacent cells, resulting in a system where the change in one cell is a function of the changes in neighbouring cells. This can then be written as a system of equations that can be solved to obtain the pressure, water saturation and concentration values in every grid cell. Note that in the eqs. 6.24, 6.25 and 6.26, the source term is only evaluated for cells that contain either a production or injection well. In most of the grid cells, the q_i term is thus zero and vanishes. The *Newton-Raphson method* is a numerical solution method for solving system of equations, and is implemented in MRST in the 2012b release.

6.5.1 The Newton-Raphson Method

Given a mathematical function, the Newton-Raphson method is a numerical method to calculate an increasingly accurate approximation to the real solution through an unknown number of iterations. This approximation implies that there exist an error which is the difference between the approximated solution and the real solution. This error is most often not possible to compute, but there exist different criteria for terminating the iterative algorithm when a "sufficiently" good approximation to the real solution is obtained.

The Newton-Raphson method can be derived by approximating the function with Taylor series and ignoring higher order terms:

$$f(x^*) = f(x_n) + f'(x_n)(x^* - x_n) + \mathrm{R} \qquad (6.27)$$

The purpose of the Newton-Raphson is to find the solution, x^*, such that the function $f(x^*) = 0$. Inserting this and solving x^*:

$$x^* = x_n - \frac{f(x_n)}{f'(x_n)} - \mathrm{R} \qquad (6.28)$$

An approximation to the solution x^* is obtained by ignoring the remaining terms (R) and iterating until the approximation is sufficiently accurate.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{6.29}$$

Where:

- x_n is the approximation to the real solution after n iterations
- x_{n+1} is the approximation to the real solution after *n+1* iterations

For a single-variant function, the Newton-Raphson method can be visualized with the figure below.

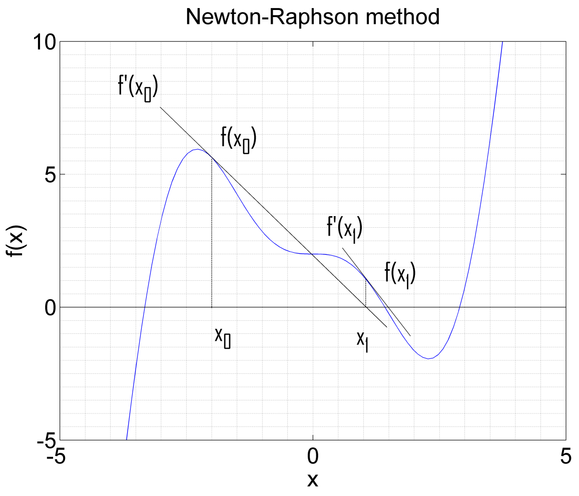


Figure 6.2: *Step-wise visualization of the Newton-Raphson algorithm for a single-variant system. The calculated slopes $f(x)$ brings the variable x closer to the solution where $f(x) = 0$.*

Given an initial "guess" to the real solution, x_0, the slope of the function at x_0, $f'(x_0)$, is calculated. Following the slope until $y = 0$ is reached at x_1 and then computing $f'(x_1)$. These steps are repeated and a new and better approximations to the real solution is calculated for every iteration. This method will eventually make the x-values close enough to the real solution to terminate the loop.

Now consider a system of n equations with n unknowns

$$\begin{aligned} f_1(x_1, x_2, x_3, \cdots, x_n) &= 0 \\ f_2(x_1, x_2, x_3, \cdots, x_n) &= 0 \\ &\vdots \\ f_n(x_1, x_2, x_3, \cdots, x_n) &= 0 \end{aligned} \tag{6.30}$$

By approximating the functions with the multi-variant Taylor series and applying the same logic, the result is a system of equations, though the form differs slightly from what previously shown.

$$\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \cdots \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \cdots \frac{\partial f_2}{\partial x_n} \\ \vdots & \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} \cdots \frac{\partial f_n}{\partial x_n} \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \vdots \\ \Delta x_n \end{bmatrix} = - \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_3 \end{bmatrix} \tag{6.31}$$

This system can be written in short notation as

$$\mathbf{J}\Delta\mathbf{x} = -\mathbf{F} \tag{6.32}$$

This linear system of equations is solved for $\Delta\mathbf{x}$, which is the difference between the first approximated solution, $\mathbf{x}_n$, and the "new and better" approximated solution, $\mathbf{x}_{n+1}$. $\mathbf{J}$ is known as the Jacobian, and is a matrix consisting of the n different functions differentiated with respect to the n different variables. $\mathbf{F}$ contains the residuals, which goes to zero as the approximated solution closes on the real solution.

For each iterative loop, a new $\Delta\mathbf{x}$ is calculated to find a new approximated solution.

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}^{n+1} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}^{n} + \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \vdots \\ \Delta x_N \end{bmatrix} \tag{6.33}$$

Convergence
After an unknown number of iterations, the approximated solution is sufficiently close to the real solution and the system is said to have converged. This terminates the iteration algorithm. The criterion may vary (Igarashi, 1984), and a common stop criterion for single variant problems is to look at the relative change solution variable. If it does not change much, then this is interpreted as that a solution is close. The iteration is terminated if the following inequality is fulfilled.

$$\epsilon > \frac{x_{i+1} - x_i}{x_{i+1}} \tag{6.34}$$

Where ϵ is a predetermined tolerance level. The obvious pitfall to this termination criterion is if the approximated solution lands on a value where the solution does not change much, but is in truth not close to the real solution. This will lead to premature termination and erroneous results. This may also lead the user to believe the solution has converged, though in reality it has not.

Another way is to use the function *residuals* as the termination criterion. This means that for each iteration, as a new approximation to the solution is obtained, $\mathbf{x}^{n+1}$, $\mathbf{F}$ is calculated. Since this is zero for the real solution, the infinity norm of $\mathbf{F}$ is compared with a pre-determined tolerance level and the algorithm terminated when or if the criterion is fulfilled.

$$\epsilon > \ ||\mathbf{F}||_\infty \tag{6.35}$$

Something to keep in mind when using this termination criterion, is the relative scale of the equations. This means that if one equation operates on with values several orders of magnitude smaller than the others, the termination criterion has to be specified accordingly.

One of the downsides to the Newton-Raphson method, is that a suitable first approximation to the solution has to be provided in order to arrive at a sufficiently good approximated solution after a reasonable number of iterations.

6.5.2 Solving the Conservation Equations Using Newton-Raphson

Setting up the multi-variant system on the same form as seen in eq. 6.30. Note that the equations have to transformed to the form $f(x) = 0$ to be applicable in the Newton-Raphson solution. This is done by simply subtracting the source terms on both sides of each equation, producing the equations shown below:

Oil, f_1:

$$\frac{V}{\Delta t}\left[\left(\frac{S_o\phi}{B_o}\right)^{l+1} - \left(\frac{S_o\phi}{B_o}\right)^{l}\right] + \sum T_o\frac{\Delta p}{B_o}^{l+1} - \left(\frac{q_o}{B_o}\right) = 0 \tag{6.36}$$

Water, f_2:

$$\frac{V}{\Delta t}\left[\left(\frac{S_w\phi}{B_w}\right)^{l+1} - \left(\frac{S_w\phi}{B_w}\right)^{l}\right] + \sum T_w\frac{\Delta p}{B_w}^{l+1} - \left(\frac{q_w}{B_w}\right) = 0 \tag{6.37}$$

Surfactant, f_3:

$$\frac{V}{\Delta t}\left[\left(\frac{S_w c_s\phi}{B_w}\right)^{l+1} - \left(\frac{S_w c_s\phi}{B_w}\right)^{l}\right] + \sum c_s T_w\frac{\Delta p}{B_w}^{l+1} - \left(\frac{q_w c_s}{B_w}\right) = 0 \tag{6.38}$$

Setting up the solution system on the same form as found in eq. 6.31.

$$\begin{bmatrix} \frac{\partial f_1}{\partial p} & \frac{\partial f_1}{\partial S_w} & \frac{\partial f_1}{\partial c_s} \\ \frac{\partial f_2}{\partial p} & \frac{\partial f_2}{\partial S_w} & \frac{\partial f_2}{\partial c_s} \\ \frac{\partial f_3}{\partial p} & \frac{\partial f_3}{\partial S_w} & \frac{\partial f_3}{\partial c_s} \end{bmatrix} \begin{bmatrix} \Delta p \\ \Delta S_w \\ \Delta c_s \end{bmatrix} = - \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} \tag{6.39}$$

Here, p, S_w and c_s are vectors containing pressures, saturations and concentrations values for each of the N grid cells. This means that $p = [p_1, p_2, \ldots, p_N]$, $S_w = [S_{w1}, S_{w2}, \ldots, S_{wN}]$ and $c = [c_{s1}, c_{s2}, \ldots, c_{sN}]$, which also implies that $\Delta\mathbf{x}$ is $3 \times$ N elements long.

The three conservation equations are evaluated in each grid cell, which means that each element in the Jacobian is a 3N $\times$ 3N matrix and **F** is a $3 \times$ N vector. The linear system of equations in the Newton-Raphson method applied for this problem consists of the following:

- The residual values of each function evaluated at each grid cell:

$$f_1 = \begin{bmatrix} f_{11} \\ f_{12} \\ \vdots \\ f_{1N} \end{bmatrix} \qquad f_2 = \begin{bmatrix} f_{21} \\ f_{22} \\ \vdots \\ f_{2N} \end{bmatrix} \qquad f_3 = \begin{bmatrix} f_{31} \\ f_{32} \\ \vdots \\ f_{3N} \end{bmatrix}$$

- The change in solution approximation vector:

$$\Delta p = \begin{bmatrix} \Delta p_1 \\ \Delta p_2 \\ \vdots \\ \Delta p_N \end{bmatrix} \qquad \Delta s_w = \begin{bmatrix} \Delta S_{w1} \\ \Delta S_{w2} \\ \vdots \\ \Delta S_{wN} \end{bmatrix} \qquad \Delta c = \begin{bmatrix} \Delta c_{s1} \\ \Delta c_{s2} \\ \vdots \\ \Delta c_{sN} \end{bmatrix}$$

- The derivative of the oil-equation with respect to pressure:

$$\frac{\partial f_1}{\partial p} = \begin{bmatrix} \frac{\partial f_{11}}{\partial p_1} & \frac{\partial f_{11}}{\partial p_2} & \cdots & \frac{\partial f_{11}}{\partial p_N} \\ \frac{\partial f_{12}}{\partial p_1} & & & \\ \vdots & & \ddots & \\ \frac{\partial f_{1N}}{\partial p_1} & \cdots & & \frac{\partial f_{1N}}{\partial p_N} \end{bmatrix}$$

- The derivative of the oil-equation with respect water saturation:

$$\frac{\partial f_1}{\partial S_w} = \begin{bmatrix} \frac{\partial f_{11}}{\partial S_{w1}} & \frac{\partial f_{11}}{\partial S_{w2}} & \cdots & \frac{\partial f_{11}}{\partial S_{wN}} \\ \frac{\partial f_{12}}{\partial S_{w1}} & & & \\ \vdots & & \ddots & \\ \frac{\partial f_{1N}}{\partial S_{w1}} & \cdots & & \frac{\partial f_{1N}}{\partial S_{wN}} \end{bmatrix}$$

- The derivative of the oil-equation with respect to surfactant concentration:

$$\frac{\partial f_1}{\partial c_s} = \begin{bmatrix} \frac{\partial f_{11}}{\partial c_{s1}} & \frac{\partial f_{11}}{\partial c_{s2}} & \cdots & \frac{\partial f_{11}}{\partial c_{sN}} \\ \frac{\partial f_{12}}{\partial c_1} & & & \\ \vdots & & \ddots & \\ \frac{\partial f_{1N}}{\partial c_{s1}} & \cdots & & \frac{\partial f_{1N}}{\partial c_{sN}} \end{bmatrix}$$

Here, $\frac{\partial f_1}{\partial p}$, $\frac{\partial f_1}{\partial s_w}$ and $\frac{\partial f_1}{\partial c}$ constitutes the first row in the Jacobian matrix defined in eq. 6.39. In these matrices, the derivative of the f_1 with respect to the pressures, water saturation and surfactant concentration in all the grid cells in the system is evaluated. Obviously, as only the pressure, water saturation and surfactant concentration only affects neighbouring cells, the matrices will be sparse. Similar matrices can be expressed for the second and third row in the Jacobian defined in eq. 6.39

The system in eq. 6.39 containing the elements above is solved[4] for $\Delta\mathbf{x}$. The pressures, saturations and surfactant concentrations are then updated , as shown below in eq. 6.33.

$$\begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_N \end{bmatrix}^{n+1} = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_N \end{bmatrix}^{n} + \begin{bmatrix} \Delta p_1 \\ \Delta p_2 \\ \vdots \\ \Delta p_N \end{bmatrix} \tag{6.40}$$

$$\begin{bmatrix} S_{w1} \\ S_{w2} \\ \vdots \\ S_{wN} \end{bmatrix}^{n+1} = \begin{bmatrix} S_{w1} \\ S_{w2} \\ \vdots \\ S_{wN} \end{bmatrix}^{n} + \begin{bmatrix} \Delta S_{w1} \\ \Delta S_{w2} \\ \vdots \\ \Delta S_{wN} \end{bmatrix} \tag{6.41}$$

[4]In MRST, this is solved by using the backslash operator in Matlab

$$\begin{bmatrix} c_{s1} \\ c_{s2} \\ \vdots \\ c_{sN} \end{bmatrix}^{n+1} = \begin{bmatrix} c_{s1} \\ c_{s2} \\ \vdots \\ c_{sN} \end{bmatrix}^{n} + \begin{bmatrix} \Delta c_{s1} \\ \Delta c_{s2} \\ \vdots \\ \Delta c_{sN} \end{bmatrix} \tag{6.42}$$

When the new approximated solution variables are calculated, they are plugged back into eqs. 6.36, 6.36 and 6.38 to calculate the residual terms on the right hand side in eq. 6.39. The infinity norm of the residual vector is computed and compared with a termination criterion. The iteration process proceeds until the criterion is met.

Automatic differentiation

In conjunction with the Newton-Raphson method, MRST contains a module for efficiently computing function derivative. By incorporating the use of *automatic differentiation* (AD), the Jacobian can be computed for any function that is made up of a number of pre-defined mathematical operators, with the general idea that differentiation rules for can be implemented numerically. The concept of automatic differentiation is explained thoroughly by Neidinger (2010).

When a function is implemented and evaluated at a certain point, the function derivative with respect to the different variables is also computed at the same point. This is therefore useful when implementing new equations to an existing framework of equations in MRST. By associating the solution vectors p, S_w and c_s with an `ADI` Matlab class and implementing the discrete conservation equations in eqs. 6.36, 6.37 and 6.38, the Jacobian matrix is easily computed.

Below is a code section extracted from Matlab with the use of MRST code to illustrate how AD is used in Matlab.

```
P = rand(10,1); % Generate 10 random values for variable #1
S = rand(10,1); % Generate 10 random values for variable #2

[P,S] = initVariablesADI(P,S);  % Associate P and S with the ADI
                                % class

F = @(P,S) P.^2 + S.^2          % F is now a function of P and S, but
                                % is not associated with the ADI
                                % class

VAR = F(P,S)                    % Computing VAR(P,S)

VAR =

  ADI

  Properties:
    val: [10x1 double]
    jac: {[10x10 double]  [10x10 double]}
```

The Matlab variable `VAR` now has function values and the corresponding Jacobi values associated with it, in `VAR.val` and `VAR.jac`. Note that `VAR.jac` has two entries, one for each function variable, namely `P` and `S`.

6.6 Robustness and Optimization

There are several ways to better ensure that the system of equations will arrive at a sufficiently good solution in a practical amount of time. Among these are two methods implemented in the current version of MRST: *1)* step cutting and *2)* line search.

6.6.1 Step Cutting

Step cutting is a way to control the maximum step size, Δx, by imposing an upper pre-defined limit. This can be advantageous if the investigated function has a very flat slope for an approximated solution. Consider the same function as in fig. 6.2.

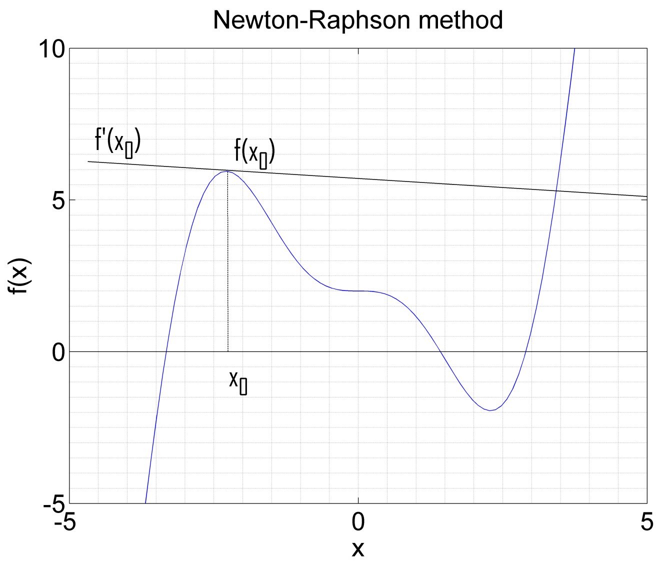


Figure 6.3: *The calculated slope at the initial guess, $f'(x_0)$ results in a large step, bringing the approximated solution further away from the real solution where $f(x) = 0$. It is here beneficial to limit the step length.*

Notice that in fig. 6.3, the initial guess, x_0, is close to the local maximum value of f. The slope at this point $f'(x_0)$. By following the slope until $y = 0$ is reached, this will result in a large step length and bring the new approximated solution further away from the real solution. Cutting the step would in this situation resolve in a more efficient iteration process, obtaining the true solution for a fewer number of iteration steps.

This is implemented in MRST along with a *capping* of solution values. What this does is ensuring that solution variables make physical sense, i.e. to cap saturation values between zero and one and ensure that surfactant concentration is always larger than zero. This ensures code robustness and helps to optimize the solution.

6.6.2 Line Search

Line search is an algorithm often applied in optimization theory in numerical solutions. It is as a way to find the most suitable step length in any numerical method, not necessarily just Newton-Raphson. The most suitable step length is the one that minimizes the residuals along the step direction. Sun and Yuan (2006) uses the following notation. Let d_k be the direction vector and α_k be the step length in that direction in a multi-variant system.

$$\phi(\alpha_k) = f(x_k + \alpha_k d_k) \tag{6.43}$$

The idea is to find the step length, α_k, which minimizes the residual along the step.

$$f(x_k + \alpha_k d_k) = \min_{\alpha>0} f(x_k + \alpha d_k) \tag{6.44}$$

A more elaborate explanation is given by Sun and Yuan (2006).

During the Newton-Raphson algorithm, inverting and solving the linear system in eq. 6.32 is far more computationally expensive than calculating the residuals **F**. Being able to reduce the number of iterations by calculating less computationally expensive residuals along the step is a good optimization strategy. Line search is implemented as an option in MRST, but has been switched off for this work.

Chapter 7

Surfactant Model Specifics

This chapter gives the reader insight to the physical effects and keyword implementations required in the surfactant model. The reader is expected to have a minimum knowledge regarding Eclipse® input formatting.

7.1 Keywords

MRST is designed to read deck input in the same format as for Eclipse®. This means that all information required to run the simulation is initialized by specifying a set of keywords. Since the model implemented in MRST during in this study is based on the surfactant model included in Eclipse®, the same keywords were used to define necessary parameters. This is convenient for model verification since the same input file can be used in MRST and Eclipse® simultaneously without any major changes necessary. Apart from the standard keywords used for grid and reservoir initialization, well data etc., the surfactant model specific keywords are listed in table 7.1. Note that some of these keywords are optional to use. The Matlab codes for different keyword implementations is provided in appendix A.

Table 7.1: *Surfactant model specific keywords. Presented in Eclipse ® Technical Description.*

Keyword	**Description**	**Notes**
SURFACT	Model initialization	Obligatory
SURFST	IFT data	Obligatory
SURFVISC	Viscosity modifier	Obligatory
SURFCAPD	Capillary de-saturation	Obligatory
SURFADS	Surfactant adsorption	Optional
SURFROCK	Rock properties	Obligatory if SURFADS is used
SURFNUM	Grid region specification	Obligatory
WSURFACT	Injected concentration	Optional

SURFACT
This is a keyword for initialization of the surfactant model in Eclipse®. Though not strictly required in MRST, it has been implemented for an easy transition.

SURFST
This keywords controls the IFT between oil and water as a function of surfactant concentration. The data is supplied in a set of tabulated values with surfactant concentration in kg/Sm^3 and IFT in N/m.

SURFVISC
Tabulated data determines the viscosity of the aqueous solution as a function of concentration. The equation used to determine the effective aqueous phase viscosity is given by eq. 7.11.

SURFCAPD
This keyword largely controls the overall effects of the surfactant on transport properties. It contains logarithmic values for the capillary number, N_c, tabulated against a miscibility factor, (m, ranging from 0 (immiscible conditions) to 1 (fully miscible conditions). This is an alternative representation of the commonly used capillary de-saturation curve. The miscibility factor is used to quantify the changes in transport properties as the IFT decreases.

SURFADS
Surfactant retention is assumed to only be caused by adsorption, where surfactant molecules are adsorbed onto the rock surface and not partitioned in the

water. The mass of adsorbed surfactant is dependent on the amount in solution and the rock density, so this keyword contains tabulated values for surfactant concentration against adsorbed surfactant mass per reservoir rock mass, kg/kg. Adsorbed amount of surfactant is calculated using eq. 7.1.

SURFROCK
There is two types of surfactant adsorption mechanisms: *1)* only adsorption and *2)* both adsorption and desorption. Desorption is the opposite of adsorption, meaning that previously adsorbed surfactant molecules can go from the rock back into solution. This keyword contains two parameters: adsorption index and rock density in kg/m^3. The index values can be either 1 or 2, where 1 means that surfactant desorption may occur and 2 means that it can not. The different rock densities and adsorption indices are used in the different grid regions defined with the keyword SURFNUM.

SURFNUM
This keyword specifies in which regions the fully miscible conditions may apply. In the Eclipse® input deck, the keyword TABDIMS controls the number of entries in tabulated data. The first parameter in TABDIMS is NTSFUN, which specifies how many saturation dependent functions each keyword hold. For instance, relative permeability curves hold NTSFUN sets of tabulated data for relative permeability. The keyword SURFNUM is specified by an integer for the number of cells (from 1 to N) in the same manner as the SATNUM keyword is used. The SURFNUM keyword tells MRST which active cells are prone to interpolated relative permeability and other functions dependent on saturation. Among the discussed keywords do SURFST, SURFVISC, SURFCAPD and SURFADS contain two sets of tabulated data.

WSURFACT
In the schedule section in the input file, this keyword may be used to define what wells are injecting surfactants. The input data under this keyword are the name(s) of the specific well(s) matching the names of previously defined wells under the WELSPECS and COMPDAT keywords, followed by an injected surfactant concentration. If not defined, MRST will set the injected concentration to 0.0 by default.

7.2 Physical Effects

The physical effects implemented in this model for describing effect of surfactant injection is described in this section. All of the described effect occur on a cell-to-cell basis, meaning that the presented equations are evaluated for all grid cells in the model. The Matlab code is provided in appendix. B.

7.2.1 Oil-Water IFT

For a given surfactant concentration, the IFT value is obtained by interpolating the tabulated data supplied under the SURFTST keyword.

7.2.2 Surfactant Adsorption

The surfactant adsorption follows the tabulated data provided under the SURFADS keyword. The adsorbed surfactant is obtained by interpolated the tabulated data for a given surfactant concentrations. Depending on the SURFROCK adsorption index, the amount of adsorbed surfactant may decrease if the injected concentration decreases. After the amount of adsorbed surfactant computed, it is added to the left hand side in the discrete conservation equation for surfactant in eq. 6.38 that needs to be fulfilled in order to solve for the new concentration. The amount of surfactant adsorbed is calculated using eq. 7.1, presented in the Eclipse® Technical Description.

$$\text{ADS} = \text{P.V.}\ \frac{1-\phi}{\phi}\rho_{rock}\ f(c_s) \tag{7.1}$$

The $f(c_s)$ is the interpolated tabular value supplied under the SURFADS keyword and P.V. is the pore volume of the cell.

7.2.3 Miscibility

As the surfactant reduce the IFT, it also increase the capillary number. From the description of capillary de-saturation curves, the same effect on the residual oil saturation is expected.

$$N_c = \frac{|\mathbf{K} \cdot \nabla P|}{\sigma} \tag{7.2}$$

The capillary number is calculated with eq. 7.2, provided in the Eclipse® Technical Description. To find the miscibility condition, the logarithmic value for the capillary number is interpolated in the supplied table under SURFCAPD.

7.2.4 Relative Permeability Alterations

Another effect of lowered IFT is the changes in relative permeability, as presented in section 5.4.1. To account for this, there are supplied two sets of relative permeability curves in the input deck. One for immiscible conditions and one for fully miscible conditions. Consider the relative permeability curves for water in fig. 7.1.

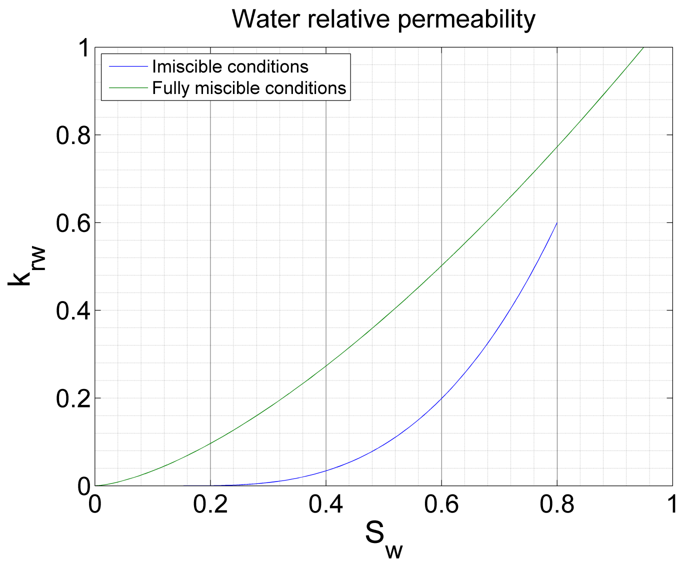


Figure 7.1: *Two sets of relative permeability curves for water. One curve for immiscible conditions (m = 0) and one curve for fully miscible conditions (m = 1).*

After m has been determined, the two relative permeability are scaled and averaged according to a specific method. The lower and upper end-points on the two relative permeability curves are used to calculate new end-point saturations by a weighted average with m (eqs. 7.3 and 7.4). This now constitutes the mobile saturation region, and an effective saturation variable is introduced (eq. 7.5).

$$S_{wc,new} = mS_{wc,misc} + (1-m)S_{wc,immisc} \tag{7.3}$$

$$1 - S_{or,new} = m(1 - S_{or,misc}) + (1-m)(1 - S_{or,immisc}) \tag{7.4}$$

$$S_{eff} = \frac{S_w - S_{wc,new}}{1 - S_{or,new} - S_{wc,new}} \tag{7.5}$$

As m vary between 0 and 1, the relative permeability calculations vary. Since the relative permeability defined in the input deck consist of a number of discrete points and is not described by a continuous function, direct scaling of the curves to fit the new end-point saturations is not possible. Instead, a code that creates two new saturation variables is implemented(eqs. 7.6 and 7.7). These two saturation values are used to calculate the relative permeability for both miscible and immiscible conditions at the target saturation, S_w.

$$\bar{S_1} = S_{eff}\big[(1 - S_{or,immisc}) - S_{wc,immisc}\big] + S_{wc,immisc} \tag{7.6}$$

$$\bar{S_2} = S_{eff}\big[(1 - S_{or,misc}) - S_{wc,misc}\big] + S_{wc,misc} \tag{7.7}$$

Example
Assume $S_w = .15$ and the original saturation end-point are also 0.15 and 0.8 (initially immobile water). Assuming that the end-points for miscible conditions are 0 and 1, the new lower saturation end-point is .12 for the semi miscible case with $m = .2$ (eq. 7.3). According to eq. 7.5, $S_{eff} = 0.047$. Because of the semi-miscible conditions, $S_w = .15$ is now inside the mobile region. With the values for S_{eff}, lower and upper limit previously calculated, the new saturation values $\bar{S_1}$ and $\bar{S_2}$ are calculated from eqs. 7.6 and 7.7 to be 0.1805 and 0.047 respectively. These saturation points correspond to $S_w = 0.15$ for the scaled immiscible and miscible curves.

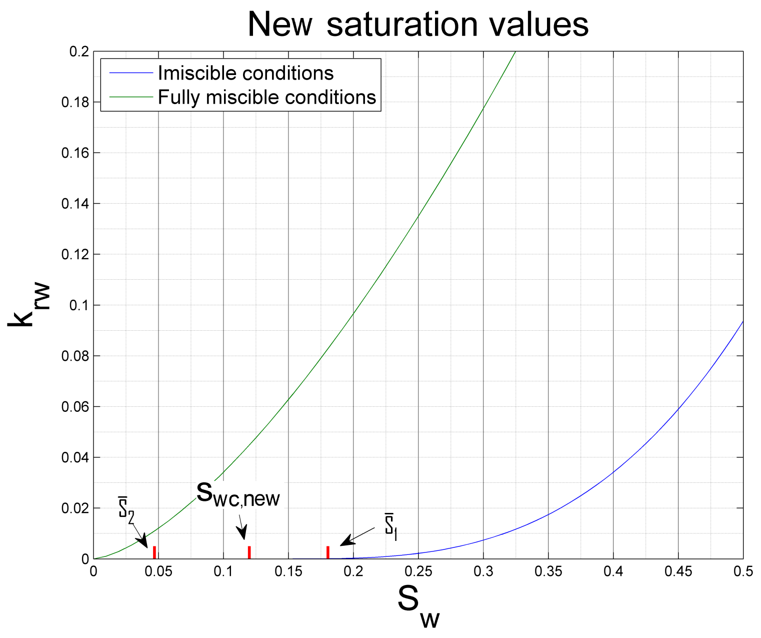


Figure 7.2: *The calculated parameters shown on the relative permeability curves plotted in fig. 7.1. $\bar{S}_1$ and $\bar{S}_2$ are the scaled water saturation value for the case where $m = 0.2$. They are used to calculate a weighted average of relative permeability*

In short: new saturation values are calculated, one for the miscible curve and one for the immiscible curve. The two new saturation values correspond to the input saturation value, in this case 0.15, for the scaled miscible and immiscible curve. Using these new saturation values, the relative permeability is interpolated in the miscible and immiscible table. The approach is visualized in the fig. 7.2.

The *effective* relative permeability in the grid cell with saturation S_w and miscibility factor m is then a weighted average of the two curves (eqs. 7.8 and 7.9).

$$k_{r,w}\big|_{S_w} = m \cdot k_{r,w,misc}\big|_{\bar{S_2}} + (1-m) \cdot k_{r,w,immisc}\big|_{\bar{S_1}} \tag{7.8}$$

$$k_{r,o}\big|_{S_w} = m \cdot k_{r,o,misc}\big|_{\bar{S_2}} + (1-m) \cdot k_{r,o,immisc}\big|_{\bar{S_1}} \tag{7.9}$$

7.2.5 Capillary Pressure Alterations

When the IFT decreases, so does capillary pressure, according to eq. 2.5. The new capillary pressure is found by multiplying the capillary pressure with the ratio of IFT at a given saturation to the IFT at zero surfactant concentration.

$$p_{cow} = p_{cow}(S_w)\frac{\sigma_{c_s}}{\sigma_{(c_s=0)}} \tag{7.10}$$

7.2.6 Water Viscosity Alterations

Adding surfactant to the injected brine will increase its viscosity. The viscosity is related to the pressure dependent viscosity function supplied under the keyword PVTW, reference water viscosity and the surfactant viscosity tabulated from the SURFVISC keyword. An *effective* water viscosity is then calculated with eq. 7.11.

$$\mu_{ws}(c_s, p) = \mu_w(p)\frac{\mu_s(c_s)}{\mu_w(p_{ref})} \tag{7.11}$$

Chapter 8

Simulation Results and Discussion

As the equations, numerical model and solution method have been presented, this chapter focuses on simulation results obtain with the surfactant model implemented in MRST. These results will be compared with simulation results obtained using the Eclipse® surfactant model and a qualitatively and quantitatively analysis will be given. The goal here is to validate the model created in this thesis by applying simple surfactant injection schemes in a generic 1D, 2D and 3D reservoir model.

The simulations are first conducted with Eclipse®. The deck files are then run with the newly implemented surfactant model and a set output parameters are compared for the two runs. These two data sets are then presented in a way to better see any differences that might arise.

Along with studying the production data for the 1D validation case the oil saturation, pressure and surfactant concentration profiles are compared to better understand differences that might arise.

For the 2D and 3D case, surfactant distribution in the reservoir at different times during a cycling surfactant injection from four injection wells is visualized. The grid cells where ultra-low IFT conditions have been met are visualized and the results linked to distinct features in the production data.

8.1 Input Data

Before running the surfactant model in both Eclipse® and the newly implemented surfactant model, the appropriate input data have to be provided. The surfactant specific data consists of tabulated values of different parameters supplied under each of the surfactant model specific keywords.

1. Oil-water IFT tabulated against surfactant concentration
2. Water viscosity tabulated against surfactant concentration
3. A miscibility factor between 0 and 1 tabulated against logarithmic value of N_c
4. Surfactant adsorption isotherm tabulated against surfactant concentration

Feng et al. (2011) provided all the necessary data for the surfactant model, which are listed in tables 8.1, 8.2, 8.3 and 8.4.

Table 8.1: *Oil-water IFT at different surfactant concentration*

Surfactant concentration [kg/m^3]	0	0.1	0.5	1	30	100
Interfacial tension [N/m]	0.05	0.0005	1E-5	1E-6	1E-6	1E-6

Table 8.2: *Water viscosity at different surfactant concentrations*

Surfactant concentration [kg/m^3]	0	30	100
Water viscosity [mPa s]	0.61	0.8	1

Table 8.3: *Miscibility condition for different capillary numbers*

log (N_c)	-10	-5.5	-4	-3	2
Miscibility factor	0	0	0.5	1	1

Table 8.4: *Surfactant adsorption at different concentrations*

Surfactant concentration [kg/m^3]	0	1	30	100
Surfactant adsorption [kg/kg]	0.0005	0.0005	0.0005	0.0005

The two relative permeability curves used for the miscible and immiscible conditions in these simulations have been obtained by assuming the Corey type equations are valid.

By using eqs. 2.1 and 2.2 presented in section 2, the end-point saturation, end-point relative permeability for both phases and curve exponent for the two conditions have to be specified prior to conducting the simulations. Keep in mind that this, in a way, determines the ultimate recovery potential for surfactant injection in terms of residual oil saturation as the oil saturation can not go below the lowest specified value in these tables. The values chosen for this study are shown in table 8.5. These values produce the relative permeability curves seen in fig. 8.1.

Table 8.5: *Corey type equation input values*

	n	S_{wc}	$1 - S_{or}$	k^0_{ro}	k^0_{rw}
Immiscible	3	0.2	0.8	0.5	0.6
Miscible	1.5	0.05	0.95	1	1

Evans (1970) tabulated capillary pressure against water saturation for a water flood. This data was converted to SI units and then fitted with a sixth degree polynomial. This polynomial equation was then used to calculate the capillary pressure value at the same saturation points as the relative permeability values are specified. This produced the capillary pressure curve seen in fig. 8.2.

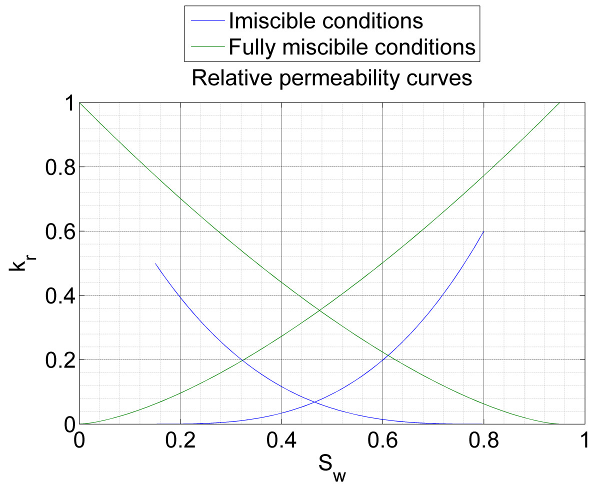


Figure 8.1: *The two sets of relative permeability curves used as input values. Curves were constructed by assuming Corey equations were valid and using the parameters listen in table 8.5.*

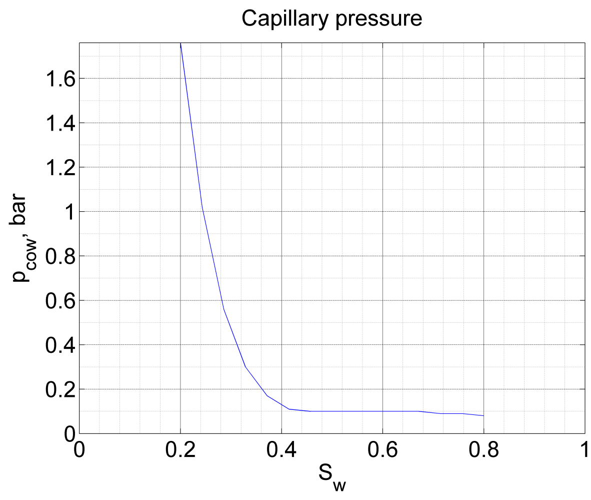


Figure 8.2: *Capillary pressure data used as input values. Tabulated data were extracted from Evans (1970) and fitted with a polynomial.*

Fluid and rock properties such as compressibility, density, viscosity and formation volume factor was provided for this study and their validity is not subject to discussion. Reservoir and fluid parameters used in all three validation cases are presented in tables 8.7 and 8.6 respectively.

Table 8.6: *Fluid parameters used in the validation cases*

Reservoir parameters	
ϕ	0.3
k_{xx}	100 mD
k_{yy}	100 mD
k_{zz}	20 mD
P_i	300 bar
Top depth	1000 m

Table 8.7: *Reservoir parameters used in the validation cases*

Fluid parameters	
$S_{w,i}$	0.2
$\rho_{w,sc}$	1080 kg/Sm^3
$\rho_{o,sc}$	800 kg/Sm^3
$\mu_{w,ref}$	0.61 mPa s
$\mu_{o,ref}$	5.0 mPa s
P_{ref}	300 bar

8.2 Horizontal Displacement Validation Case

In the first validation case, a simple 1D horizontal displacement is investigated. In this case, an initial water flood is first conducted for two years at a constant injecting and producing bottomhole pressure of 320 and 280 bar respectively. This is followed by a constant surfactant injection with a concentration of 50 kg/m^3 for another two years, while keeping the well pressures unchanged. The simulation is concluded with a two year long water flood after the surfactant slug, to see how the surfactant behaves when injected as a slug. This way, the adsorption effects on the slug's "head" and "tail" will be observed.

Table 8.8: *Grid properties*

$N_x \times N_y \times N_z$	$100 \times 1 \times 1$
$\Delta x \times \Delta y \times \Delta z$	$1 \times 5 \times 5$
Physical dimensions	100m $\times$ 5m $\times$ 5m

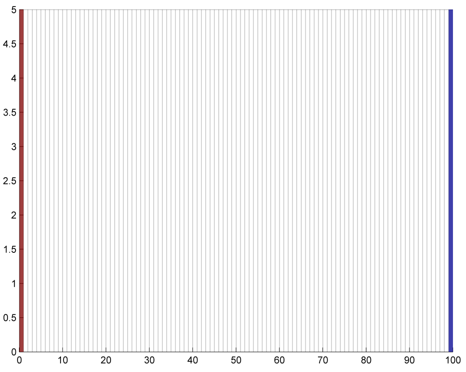


Figure 8.3: *The grid used in this case. Red denotes injection well placement and blue production well. Grid and well configuration chosen to investigate the validity of 1D flow in the surfactant model implemented in MRST*

The physical dimensions and an overview of the reservoir is given in table 8.8 and fig. 8.3 respectively. Injection well is located in the cell highlighted in red and the producing well in the cell highlighted blue.

After the simulation, surfactant concentration, oil saturation and oil pressure profile is plotted for two different times, 848 and 1548 days. These times were chosen to capture the effects of an increasing and decreasing surfactant concentration, as the surfactant slug is *entering* and *leaving* the reservoir at these times. The simulation results obtained from the newly implemented surfactant model are plotted together with the simulation results from Eclipse® in dotted and full lines respectively.

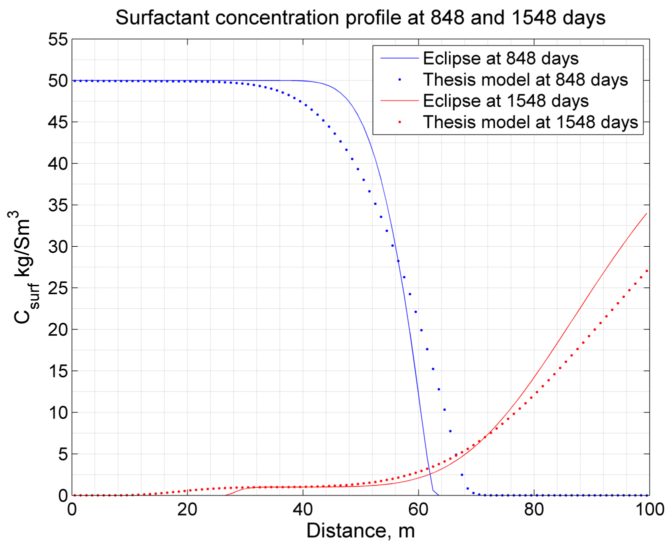


Figure 8.4: *Surfactant concentration profile. After 848 days denoted with blue markers and 1548 days with red markers. A clear difference is seen as the surfactant model implemented in MRST show a more smeared out profile than Eclipse®*

From figs. 8.4 an obvious discrepancy is seen and a few key observations can be noted. As the injected surfactant slug enters and propagates the reservoir (from the left to right), the model implemented in MRST predicts a less sharp concentration front than Eclipse®. This is illustrated at 848 days with blue lines. As the slug flows through the reservoir and exiting on the right side, the surfactant model predicts a longer concentration "tail" where the surfactant slug have moved through. The end effects of the concentration in the surfactant slug can be interpreted as a dispersion effect, though the physics behind this has not been implemented in this study. Instead, the dispersion effect on the ends of the slug are interpreted as *numerical dispersion*, an artefact of implicit numerical solutions which cause the results to smear out.

Overall, the model developed in this thesis seem to predict a more smeared out surfactant concentration profile with distance than Eclipse® which will produce a more "bell-shaped" curve. Eclipse® predicts a steeper concentration decrease and increase around the slug ends and will produce a more column like concentration profile, more like a shock front.

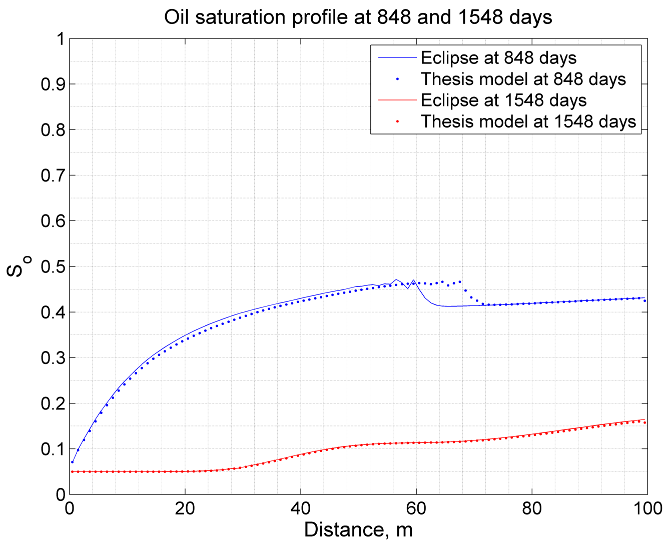


Figure 8.5: *Oil saturation at 848 and 1548 days. The surfactant model implemented in MRST shows the same qualitative trend as Eclipse®, but a faster moving oil bank*

The saturation profiles in fig. 8.5 show the same similarities and differences as the surfactant concentration profile. The oil saturations at 848 days show the same qualitative trend, but compared with Eclipse®, the oil saturations leading up to the ultimate oil bank saturation is under-predicted in thesis model. On the other hand, the oil bank is moving faster in the thesis model, reaching a longer distance at 848 days. It is worth mentioning that the formation of the oil bank is consistent with the theory presented in section 5.5.

At 1548 days, the oil saturation profile coincide well, although a somewhat smaller value can be seen at around 40 meters and from 70 to 100 meters in the implemented surfactant model. Using Matlabs' embedded `trapz` function to numerically integrate the plotted function, and thus calculating the area under the curves, the total amount of oil can be calculated. By subtracting the volume of the immobile oil (under $S_o = 0.05$), the oil in the oil bank is found.

$$\text{Oil bank volume} = A\phi \int\limits_{x=0}^{L} S_o(x)\ dx - 25m^2 \cdot 100m \cdot 0.3 \cdot 0.05 \tag{8.1}$$

Table 8.9: *Table showing oil bank volume at 848 days and 1548 days and relative error to Eclipse®. The difference at 848 days is smaller than at 1548 days because of volume balancing each other out.*

	Eclipse®	**Thesis model**	**Relative error, %**
848 days	250.05	249.81	0.096
1548 days	35.04	34.09	2.787

From the results in table 8.9, we can see that when the oil bank propagates, the same volume of oil is moved within a margin of less than 0.1 percent. This is because the smaller saturation values from the thesis model leading up to the oil bank saturation is to a certain degree balanced out by the fact that the oil bank is longer. So the excess volume that arises from having a longer oil bank is almost the same as the smaller area during the build up of saturation.

However, when the oil bank breaks through, there is no excess volume to balance out the error. This means that since the thesis surfactant model predicts the oil to move faster through the reservoir, the amount of produced oil at any given time after the oil bank break through, will be larger than for Eclipse®. This is why the relative error increases after break through.

Since the oil bank moves faster in the thesis model, the amount of produced oil will always be a few "steps ahead" of Eclipse®. This means that the response from the oil bank break through will be observed at different times on the cumulative oil production plot, shown below, and, as mentioned above, at any given time after oil bank break through, the thesis model will predict a higher recovery than Eclipse®, though only marginal. Another observation is the fluctuating oil saturation values near the saturation front. This effect is also reproduced by the surfactant model implemented in MRST, although at a distance further away, coinciding with the above discussion.

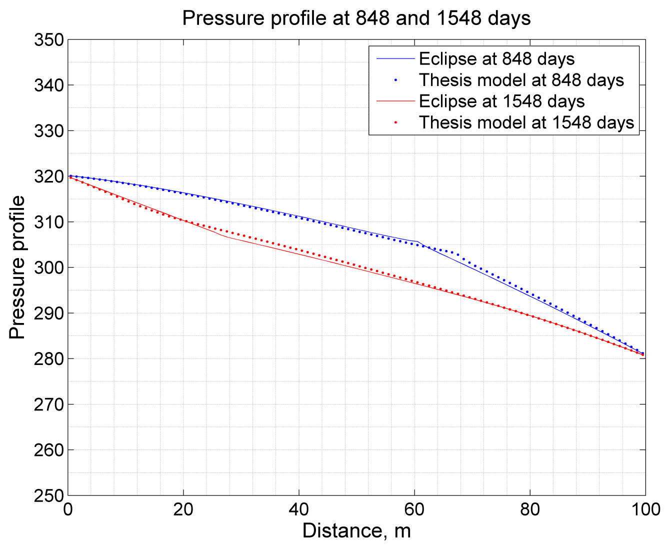
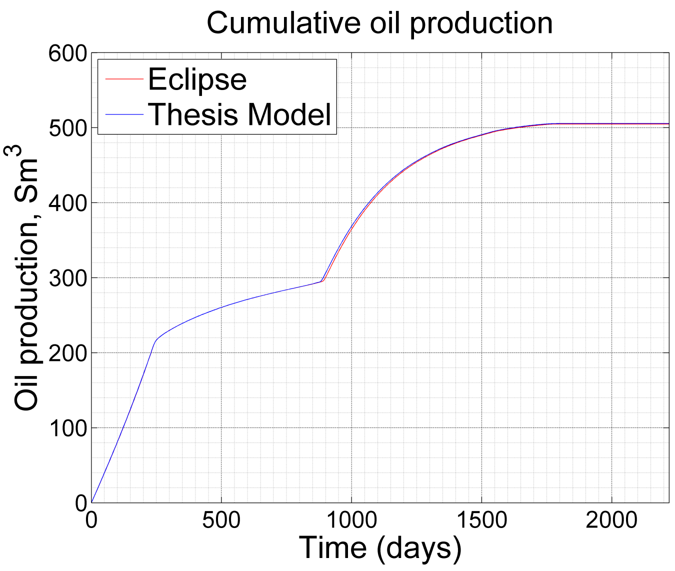
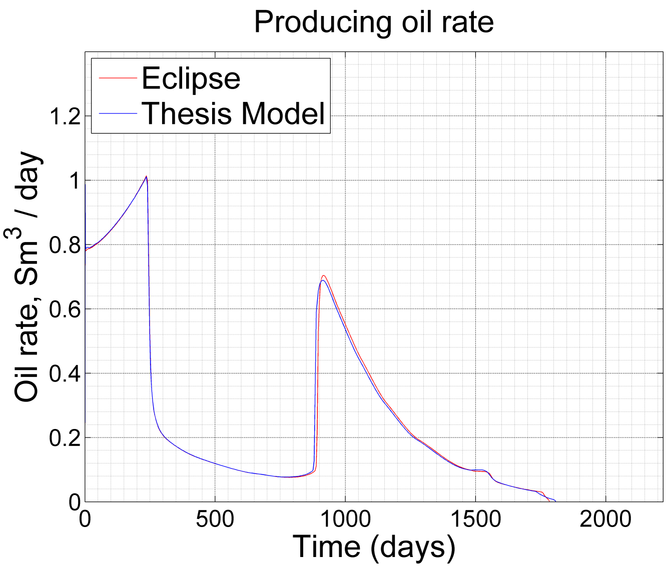


Figure 8.6: *Pressure at 848 and 1548 days. Eclipse® is plotted with red markers and the implemented surfactant model in MRST with blue markers. An obvious discrepancy is seen, coinciding with the concentration and saturation profile in figs. 8.4 and 8.5.*

The pressure profile in fig. 8.6 reflects the saturation distribution from fig. 8.5. At 848 days, the change in pressure gradient coincides with the different oil bank saturation fronts. As water viscosity increases moderately with surfactant concentration, the water viscosity is expected to gradually change over the changing regions of the surfactant concentration. Since the two surfactant profiles at 848 days are not equal, the spatial variation in water viscosity is not expected to be the same. Because of this, and the fact that the thesis model predicted a slightly smaller oil saturation leading up to the oil bank front, the pressure profile at 848 days predicted by the thesis model and Eclipse® are different. The thesis model predicts a slightly more curved pressure profile in the region before the oil bank front. The pressure profile at 1548 days also deviates from Eclipse®, even though the saturation profiles match rather well. This might also be related to the surfactant concentration once again, resulting in a region of higher water viscosity in the thesis model and a higher pressure drop.

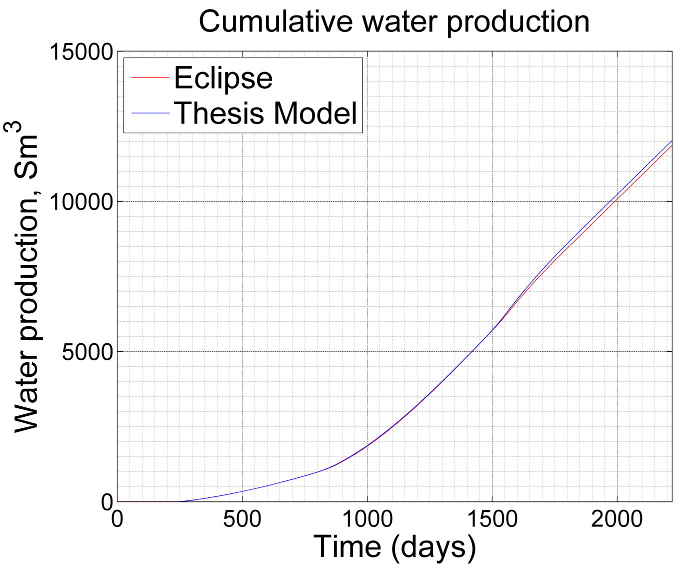


(a) *Cumulative oil production with time.*

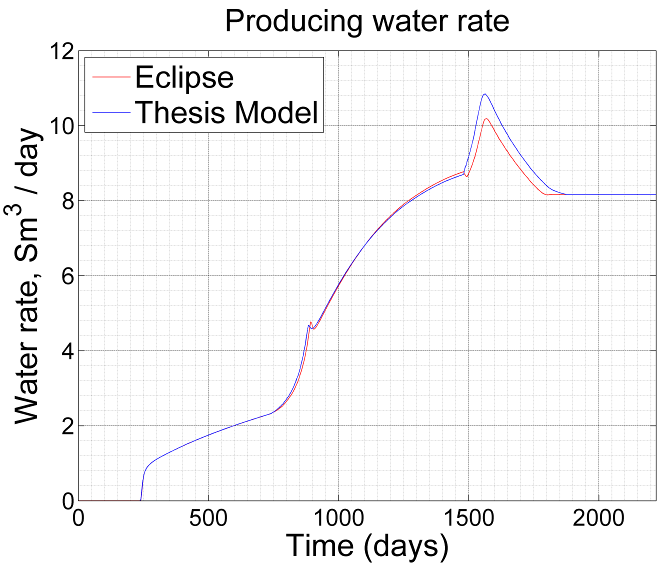


(b) *Producing oil rate with time*

Figure 8.7: *Cumulative and producing oil rate plotted against time*



(a) *Cumulative water production with time*



(b) *Producing water rate with time*

Figure 8.8: *Cumulative and producing water rate plotted against time*

In terms of production data, figs. 8.7a, 8.7b, 8.8a and 8.8b suggest that the surfactant model implemented in MRST is able to reproduce Eclipse®'s surfactant model with great accuracy, being almost identical for oil production. As mentioned in the discussion on the oil bank saturation front, the break through of the oil bank is seen earlier in fig. 8.7a at around 950 days. This causes the predicted oil production curve to always lie "on top" of the one predicted by Eclipse®. This effect can also be seen in fig. 8.7b, where the sharp increase in oil rate (due to oil bank break through) comes earlier in the data from the thesis model than from Eclipse®.

The highest oil rate predicted for the thesis model is seen to be moderately lower than for Eclipse®. This can be seen in context with fig. 8.5, where the oil saturation in the oil bank is lower for the thesis model than for Eclipse®. This also coincides well with the water rate plotted in fig. 8.8b, which shows a larger water rate from the thesis model than Eclipse® for the same period in time.

At around 1500 days the oil production rate plateaus out more for the thesis model than for Eclipse®, though not a major difference. This time coincides with the time when the surfactant is stopped being injected. The calculated cumulative oil and water at the end of the simulation is shown in table 8.10, presented in Sm^3. The relative error is calculated using eq. 8.2.

$$\left| \frac{\{\text{Thesis Model}\} - \{\text{Eclipse®}\}}{\{\text{Eclipse®}\}} \right| \tag{8.2}$$

Table 8.10: *Oil and water cumulative production in standard cubic meters and relative error to Ecipse®.*

	Eclipse®	**Thesis model**	**Relative error, %**
Oil	504.9	505.7	0.157
Water	1.1873E4	1.2033E4	1.348

As a final remark, the author would like to point out that a shock front would be expected in the surfactant and oil saturation profiles. However, this is not observed and is attributed to the numerical solution. As pointed out by Swaminathan (1994), an artefact of implicit methods is smearing, referred to as *false diffusion* or *artificial dispersion*. This accounts for the absence of shock fronts in the profiles and can also be seen in fig. 3.2, as the results are obtained in the thesis model. The results presented in this section indicate that Eclipse® has less numerical dispersion than the implemented surfactant model in MRST.

8.3 Five Well Spot Pattern Validation Case

In the next case, a five well spot pattern configuration is investigated. Here, a producing well is located near the centre of the reservoir surrounded by an injection well in each corner, A, B, C and D. The placement of the wells are shown in fig. 8.9. Red denotes injection well placement and blue denotes the production well. The physical dimensions of the simulation grid are given in table 8.11.

Table 8.11: *Grid and reservoir properties*

$N_x \times N_y \times N_z$	$20 \times 20 \times 1$
$\Delta x \times \Delta y \times \Delta z$	$5 \times 5 \times 5$
Physical dimensions	100m × 100m × 5m

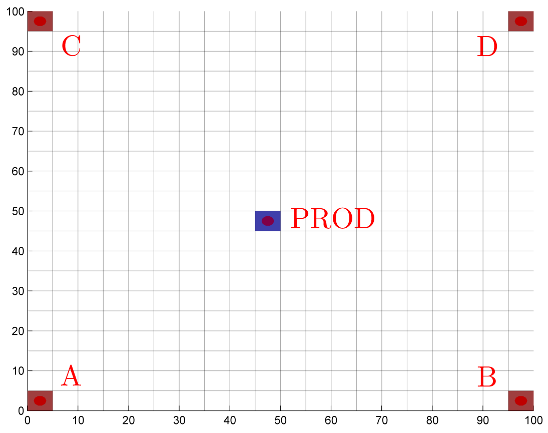
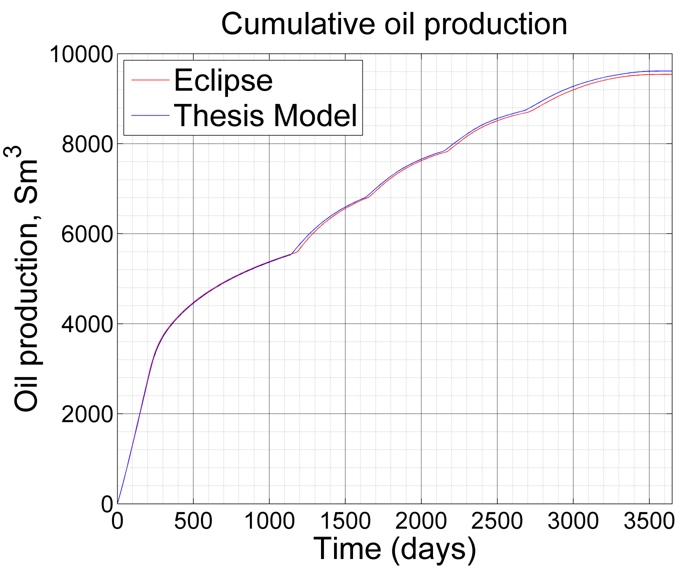
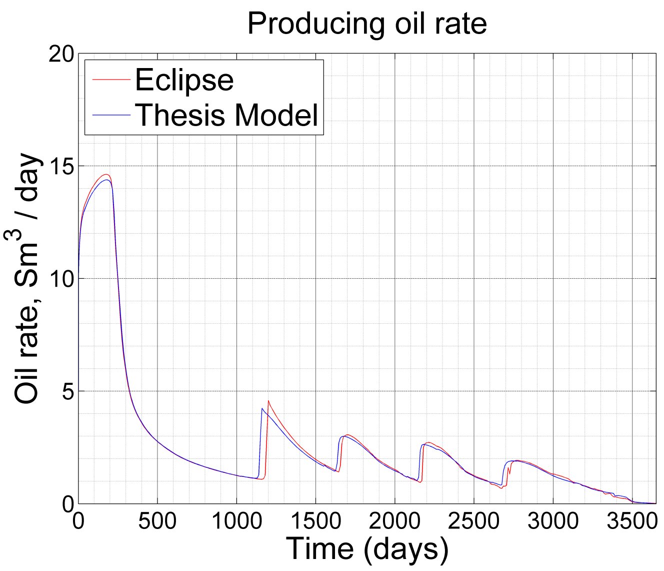


Figure 8.9: *Cells where injection wells (red) and producer (blue) are located. Grid and well placement chosen to investigate the validity of 2D flow in the surfactant model implemented in MRST*

Similar to the previous case, injection wells operate at a fixed bottomhole pressure of 320 and 280 bar. In this case, the reservoir first is water flooded for 1025 days. After the water flood, surfactant is injected with a concentration of 50 kg/m^3 from one well at the time for 525 days each. While one well injects surfactant, the other three inject water. When one well stops the surfactant injection, a new well commences its. This continues until all four injection wells have injected one slug each, and the cycle is A, B, C and D. The simulation is concluded with a 525 days long water flood to see the end effects on the surfactant slug from well D.

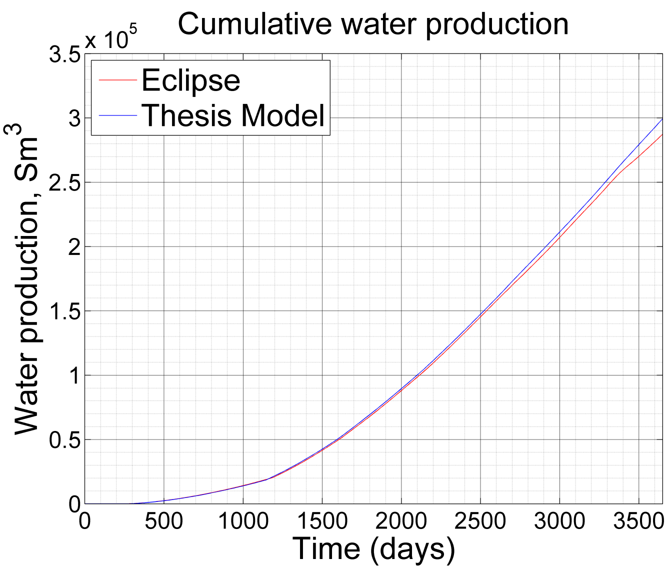


(a) *Cumulative oil production with time*

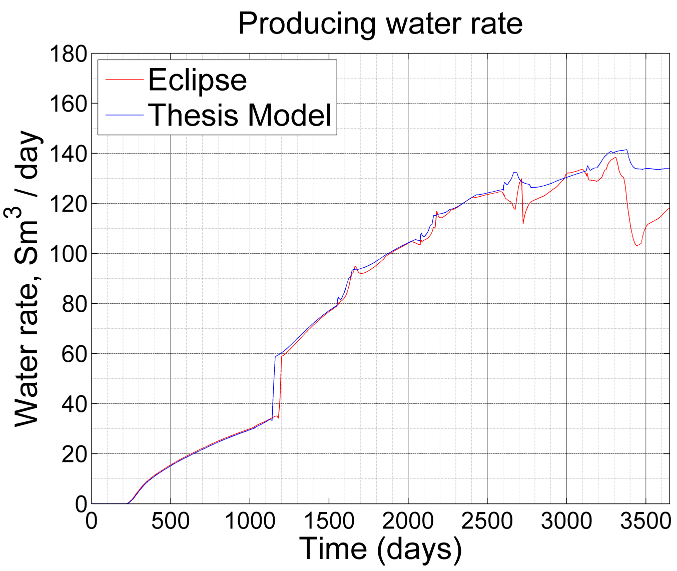


(b) *Oil production rate with time*

Figure 8.10: *Cumulative and producing oil rate plotted against time*



(a) *Cumulative water production with time*



(b) *Water production rate with time*

Figure 8.11: *Cumulative and producing water rate plotted against time*

By studying the producing oil rate in fig. 8.10b, the breakthrough of four oil banks can clearly be seen between 1000 and 3000 days as distinct spikes in the producing rate. As discussed in section 8.2, the oil bank move faster in the thesis model than in Eclipse®, and the effect of this can be as the blue line (thesis model) spikes and subsequently decreases before the red line (Eclipse®).

Also, as mentioned in the previous section, the oil saturation in the oil bank is smaller in the thesis model than in Eclipse® and is why the ultimate production rate at these spikes is lower for the thesis model. At the time when the last oil bank breaks through, Eclipse® predicts somewhat fluctuating production rates which can not be seen in the results from the thesis model. The reason for this is not known. The four spikes can also be located in the production profile in fig. 8.10a as incremental production as the curve starts to flatten out. Since each oil bank front arrives at the producing well earlier in thesis model than in Eclipse®, the amount of produced oil will be higher for the thesis model at any given time after break through of one bank. In this case, however, the producing reservoir volume is greater and there are formed four oil banks, so this effect will be even greater and result in a larger discrepancy in cumulative produced oil at the final plateau than for the 1D case. In table 8.12, the cumulative oil and water produced is presented in Sm^3 at the end of the simulation run, calculated for Eclipse® and the thesis model. The relative error to Eclipse® is also calculated.

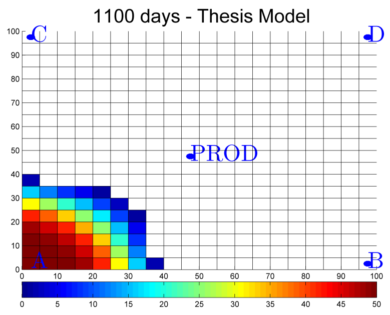
Table 8.12: *Cumulative oil and water produced in standard cubic meters at the end of the simulation. Error is relative to Eclipse®*

	Eclipse®	**Thesis model**	**Relative error, %**
Oil	9.543E3	9.617E3	0.775
Water	2.875E5	2.994E5	4.158

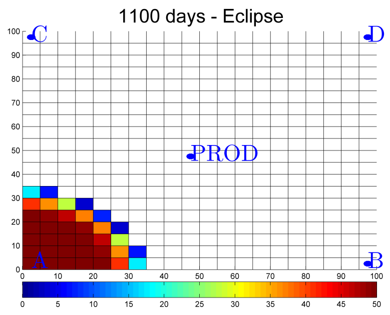
The production data for oil is well matched. The small discrepancy may be related to the discussion for producing oil in the 1D case, and may give a false impression.

Considering the production profile for water, the cumulative water production is well matched, but due to the scaling of the axis, minor differences between the two simulation results are difficult to detect. Considering the producing water rate in fig. 8.11b, a clear difference is easily spotted. As the first oil bank arrives at the producing well, the water rate also increases. The general trend is that the thesis model predicts a higher water rate than Eclipse® for the entire simulation run. The thesis model is also not able to capture the effects of the fluctuating water rates as seen from around 2600 days. The reason for this is now known.

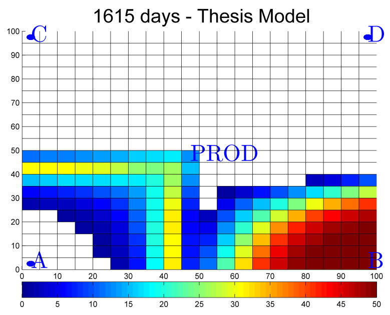
In figs. 8.12 and 8.13 below, surfactant concentration at different times is visualized to see how surfactant is transported throughout the reservoir. A lower limit cut off value of 1 kg/Sm^3 is used to ensure that only the grid cells where the IFT has been reduced to the minimum value, as according to the data in table 8.1, is visualized.



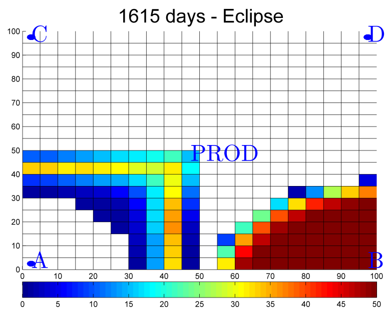
(a) *Surfactant distribution at 1100 days*



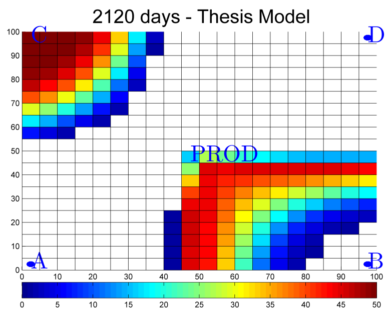
(b) *Surfactant distribution at 1100 days*



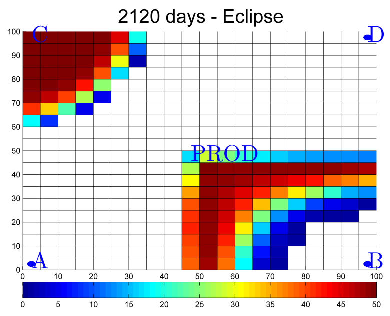
(c) *Surfactant distribution at 1615 days*



(d) *Surfactant distribution at 1615 days*

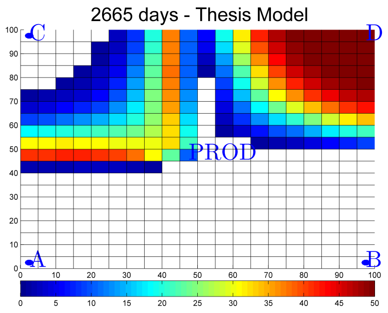


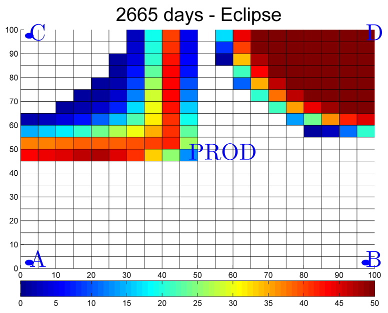
(e) *Surfactant distribution at 2120 days*



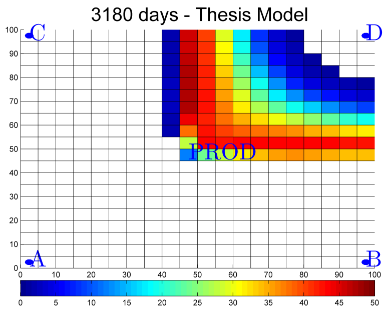
(f) *Surfactant distribution at 2120 days*

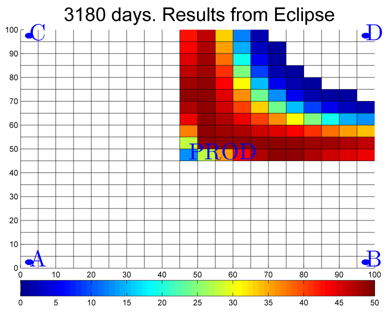
Figure 8.12: *Surfactant distribution at different times during the simulation*





(a) *Surfactant distribution at 2665 days* **(b)** *Surfactant distribution at 2665 days*





(c) *Surfactant distribution at 3180 days* **(d)** *Surfactant distribution at 3180 days*

Figure 8.13: *Surfactant distribution at different times during the simulation*

From the surfactant concentration distribution plots in figs. 8.12 and 8.13 show a difference in surfactant distribution throughout the simulation. The thesis model predicts, as concluded before, that the surfactant has more gradual surfactant concentration fronts around the edges of the slug. Considering fig. 8.12a, Eclipse® shows a more concentrated slug, with a sharper, more well defined front than the thesis model, but this model predicts the slug to reach more grid cells at by the same time. Since the IFT reaches a minimum at 1 kg/Sm^3, the entire plotted regions in figs. 8.12 and 8.13 represent areas with minimum IFT values.

Since the thesis model predicts a more dispersed concentration front, a larger portion of the reservoir is exposed to the surfactant and is thus at minimum IFT conditions. This is the general trend at all times plotted in figs. 8.12 and 8.13. In practice, what this means is that a greater portion of immobile oil has been mobilized and is being produced in the thesis model than in Eclipse®. These results are also supported by the concentration profiles plotted in section 8.2 as well as the production data plotted in figs. 8.10a and 8.10b.

8.4 Layered Five Well Spot Pattern Validation Case

In the final validation case, the same five spot well configuration is investigated, though this time the injection wells are perforated only in the upper most layer and the production well only in the lower most layer. This way, the full effect of 3D flow can be investigated as the stream will flow diagonally from the corners towards the centre and also vertically from the upper to the lower layer. The physical dimensions of the reservoir grid is listen in table 8.13 and the grid cells where producing and injection wells are perforated is show in fig. 8.14. The run schedule is the same as in the previous case.

Table 8.13: *Grid and reservoir properties*

$N_X \times N_Y \times N_Z$	$10 \times 10 \times 5$
$\Delta x \times \Delta y \times \Delta z$	$10 \times 10 \times 1$
Physical dimensions	100m × 100m × 5m

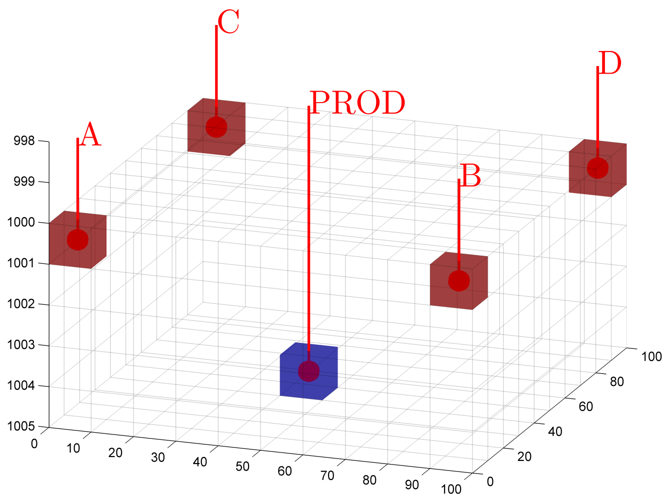
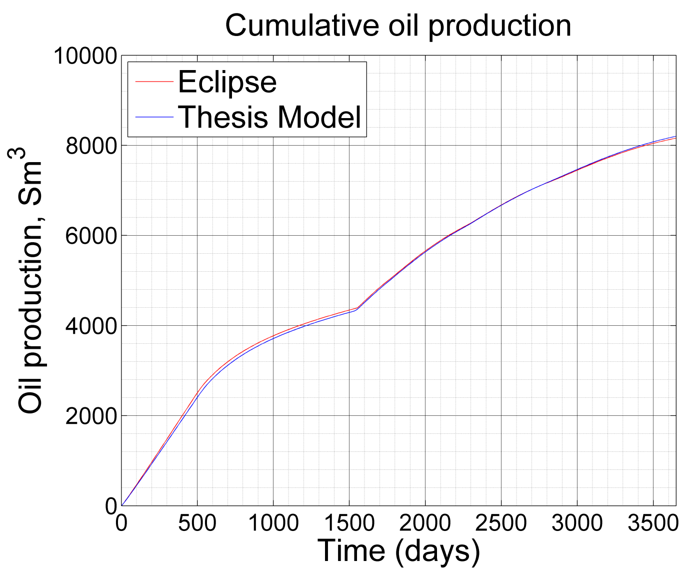
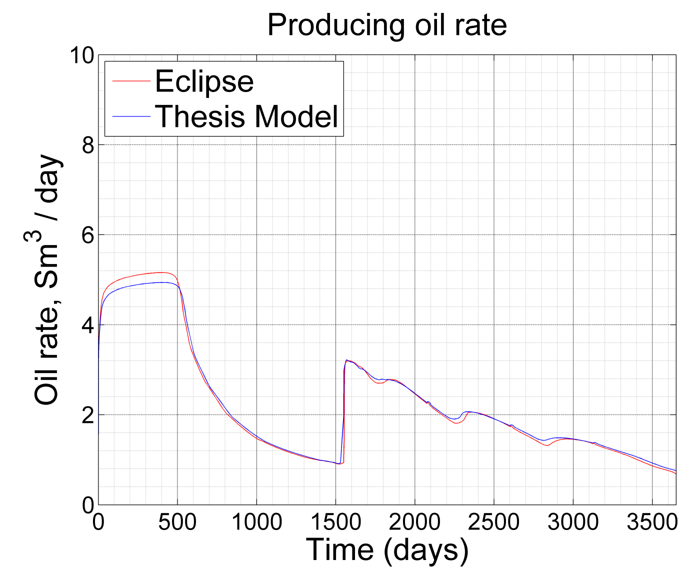


Figure 8.14: *Cells where injection wells (red) and producer (blue) are perforated. Grid and well placement chose to see the validity of 3D flow in the implemented surfactant model in MRST*

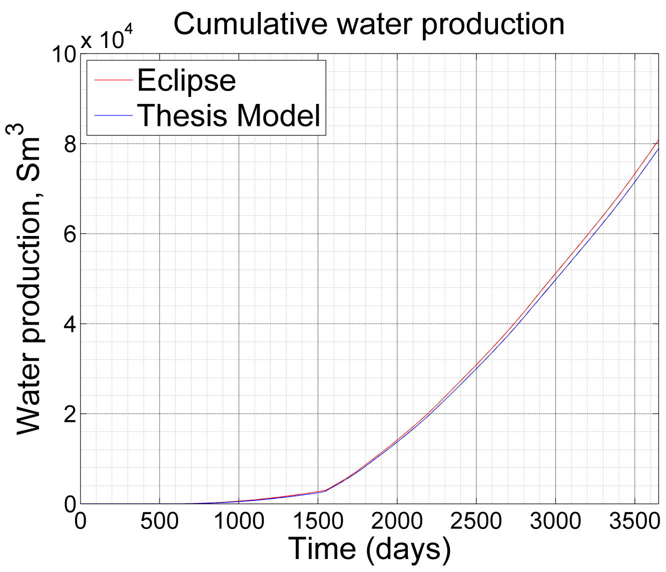


(a) *Cumulative oil production with time*

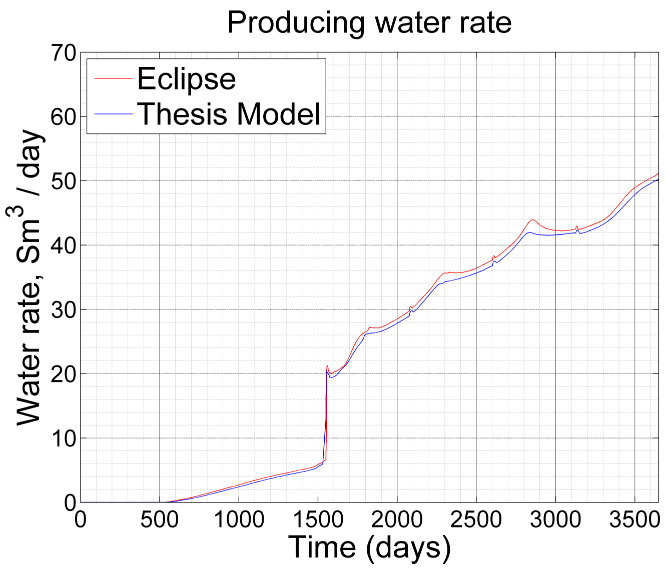


(b) *Oil production rate with time*

Figure 8.15: *Cumulative and producing oil rate plotted against time*



(a) *Cumulative water production with time*



(b) *Water production rate with time*

Figure 8.16: *Cumulative and producing water rate plotted against time*

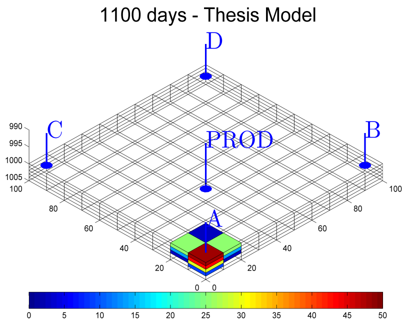
As can be seen from the cumulative oil production profile in fig. 8.15a, there is a discrepancy between the results from newly implemented surfactant model and Eclipse®, not only limited to the times after oil bank break through, but also during the water flooding. This is a deviation from the observed trend, as the production profiles aligned perfectly during the times prior to oil bank break through in the two other cases. It is therefore reasonable to suspect that the a problem might have been caused in the fluid potential calculations.

At around 1600 days, one can see that the thesis model predicts the oil banks to break through earlier than in Eclipse® as is consistent with the trend presented in the two previous cases. As the effect of the injected surfactant is observed on the oil production curve, the results from the thesis model and Eclipse® seem to coincide well. The producing oil rate in fig. 8.15b shows a significant difference in the early times as the oil rates predicted by Eclipse® exceeds the thesis model's. The reason for this is undetermined. After the break through of oil banks, the thesis model predicts the same qualitative trend as Eclipse®. The most significant difference is that the oil rate for the thesis model is always larger than for Eclipse® and does not drop as much in the "valleys". For the producing water rate in fig. 8.16b, the situation is opposite as the water rate for Eclipse® is overall larger than for the thesis model, predicting the same qualitative trend, but not capturing the fine details.

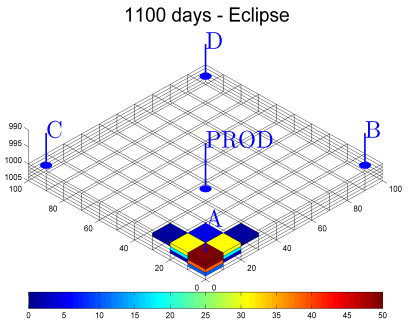
Table 8.14: *Cumulative oil and water produced in standard cubic meters. Error is relative to Eclipse®*

	Eclipse®	**Thesis model**	**Relative error, %**
Oil	8.161E3	8.203E3	0.517
Water	8.948E4	7.894E4	11.776

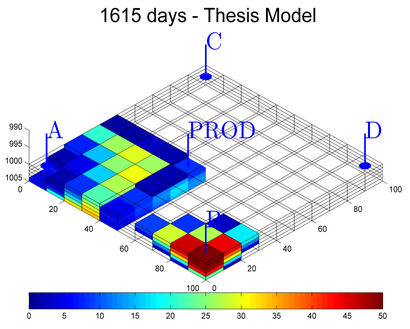
In table 8.14 can see that the cumulative oil production only has a relative error of .517%. On the other side, the cumulative water produced has a relative error of 11.776 %, which is very significant.



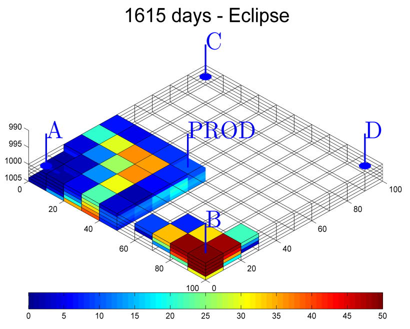
(a) *Surfactant distribution at 1100 days*



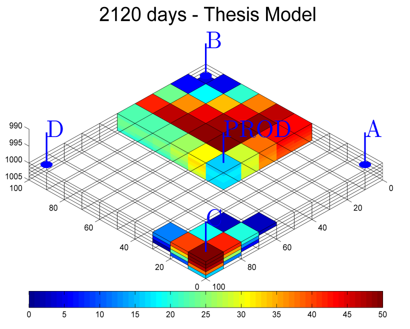
(b) *Surfactant distribution at 1100 days*



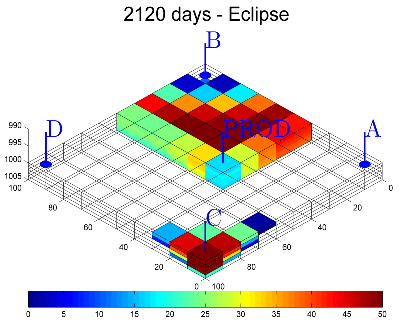
(c) *Surfactant distribution at 1615 days*



(d) *Surfactant distribution at 1615 days*

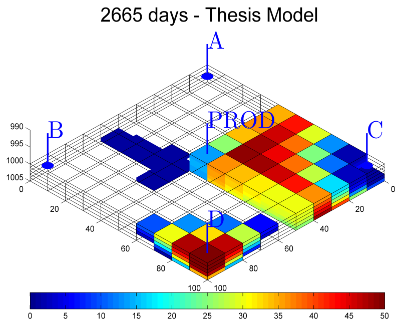


(e) *Surfactant distribution at 2120 days*

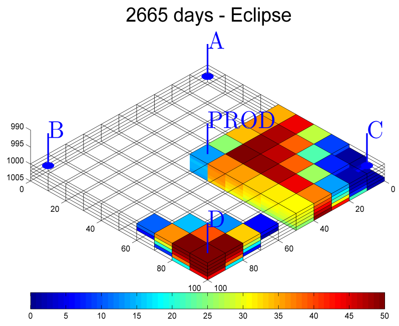


(f) *Surfactant distribution at 2120 days*

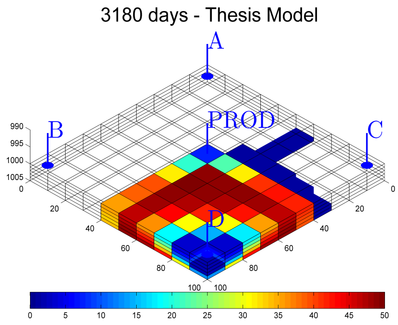
Figure 8.17: *Surfactant distribution at different times*



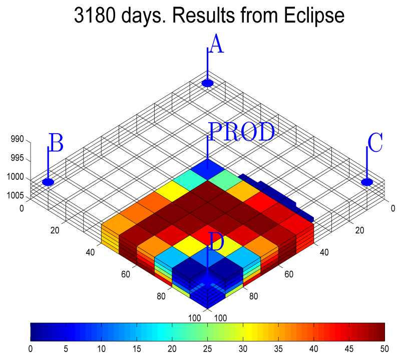
(a) *Surfactant distribution at 2665 days*



(b) *Surfactant distribution at 2665 days*



(c) *Surfactant distribution at 3180 days*



(d) *Surfactant distribution at 3180 days*

Figure 8.18: *Surfactant distribution at different times*

In the figs. 8.17 and 8.18, the model has been rotated such that the well that is injecting surfactant is facing the viewer. As in the previous section, cells with surfactant concentration above 1 kg/Sm^3 are highlighted. The results in figs. 8.17 and 8.18 show that the surfactant propagation for the 3D validation case differs from the 1D and 2D cases. Here, Eclipse® is observed to predict a larger area in the reservoir exposed to surfactant at 1100 days in fig. 8.17a. At the other times, the thesis model predicts a larger number of grid cells with a concentration above 1, consistent with the previous observations.

In figs. 8.18a and 8.18c, there are regions in the central parts of the lower most layer with surfactant concentration values above 1. These values are not consistent with Eclipse®, and is an indication of a erroneous result during the simulation. It might also originate from adsorption calculation, as the thesis model predicts a longer "tail" on the surfactant concentration profile.

Chapter 9

Conclusion

Based on the observations made and discussed in sections 8.2, 8.3 and 8.4, a few key conclusions can be drawn.

1. From the presented results, the author concludes that the work of implementing a surfactant model as an extension to a black-oil model on general grids in MRST has been successful. The use of automatic differentiation has proved invaluable in the work of implementing new equations.

2. Comparing key features such as surfactant transport, oil saturation profile, pressure profile and production data, the implemented model compares well with the formulation from Eclipse®. The model is able to qualitatively predict the same behaviour as Eclipse®, however it is not able to reproduce the results in full detail.

3. The most interesting difference, which largely can explain the other discrepancies, is the smeared out surfactant concentration profile presented in section 8.2. This results in a larger region where the IFT has reaches ultra-low levels and thus improving the flow properties in a larger region in the implemented surfactant model than in Eclipse®. This is manifested in the production curves as an earlier break through in oil bank and a marginally larger cumulative oil production at any given time after break through. It is reasonable to conclude that the difference is surfactant distribution between the two models is the reason for all the other observed differences.

4. The reasons for the observed differences are not determined, but the author points out the most likely reasons: *1)* The smeared nature of the surfactant concentration profile is indicative of a differences in the numerical solu-

tion which results in more numerical dispersion in the newly implemented surfactant model in MRST than in Eclipse® (Swaminathan, 1994). *2)* Implementation of fluid transport properties, mainly relative permeability, in the ultra-low IFT zones ar also considered a reason for the observed differences.

5. The work in this thesis is a solid foundation for further research and development in surfactant injection EOR modelling in an open-source framework.

Chapter 10

Future Improvements

Concluding this thesis, suggestions to elements that would further improve the authors implementation of a surfactant model to the MRST source code is noted. The use of the automatic differentiation is invaluable for including new physics to an existing framework of equations.

- **Three-phase flow** - In the present model, formation of a microemulsion phase is not considered. For a more realistic modelling approach, the formation of type II(-), II(+) and III microemulsions should be considered.
- **Salinity** - the author suggests implementing conservation and mixing equations for brine salinity.
- **Ions** - The effect of ions (especially divalent ions) on surfactant's ability to reduce IFT and increase recovery is quite significant (Jamaloei and Rafiee, 2008). This is effect could also be combined with salinity effects, to see the effect of *effective* salinity.
- **Temperature** - Surfactant adsorption which are dependent on temperature conditions (Ziegler and Handy, 1981) is suggested to be implemented.
- **Compositional effects** - Implementing a compositional module to the developed surfactant model would allow for compositional changes in oil and gas. Surfactants ability to reduce IFT is dependent on the oil composition, so this would allow for a more representative model. How well the surfactant properties are documented for compositional changes in oil is unknown to the author.

Nomenclature

$\vec{v_i}$	Velocity vector of fluid i [m/s]
α_k	Step length along direction vector
α	Dip angle [radians]
$\Delta\mathbf{x}$	Vector of incremental change to solution variables
$\delta p_{m,n}$	Discrete pressure drop from cell centred computational nodes in cell m to n [Pascal]
Δt	Time step length [s]
Δx_n	Distance from cell centre to cell interface in cell n [m]
ϵ	Predetermined tolerance level
$\gamma_{n,m}$	Interface between two grid cells, m and n, $\partial\Omega_n \cap \partial\Omega_m$
$\mathbf{F}$	Residuals
$\mathbf{J}$	Jacobian
$\mathbf{K}$	Permeability tensor [m^2]
μ_s	Surfactant viscosity [mPa s]
μ_i	Viscosity of fluid i [mPa s]
μ_{ws}	Viscosity of the surfactant/water mixture [mPa s]
∇z	Depth gradient [-]
Ω	Control volume
$\partial\Omega$	Confining boundary of control volume Ω
Φ_i	Potential of fluid i [Pascal]

ϕ	Porosity [-]
ρ_i	Density of fluid i [kg/m^3]
$\sigma_{i,j}$	Interfacial tension between fluid i and fluid j [N/m]
$\tilde{q}_i$	Volumetric flow rate of phase i per volume in surface conditions [Sm^3/s]
A	Area available to fluid flow [m^2]
ADS	Surfactant adsorption [kg]
B_i	Formation volume factor of phase i [m^3/Sm^3]
c_s	Surfactant concentration [kg/Sm^3]
d_k	Direction vector
E_m	Microscopic sweep [-]
E_s	Sweep efficiency [-]
F_c	Capillary forces [Newton]
f_i	Flux of phase i across the control volume boundaries [m^3/s]
F_v	Viscous forces [Newton]
f_w	Flow fraction of water [-]
$k_{e,i}$	Effective permeability of phase i [-]
$k_{r,i}$	Relative permeability of phase i [-]
$k^0_{r,i}$	End-point relative permeability of phase i, [-]
l	Discrete time step in the finite-volume evaluation [-]
m	Miscibility [-]
n	Corey curve exponent, [-]
N_c	Capillary number [-]
N_c^*	Critical capillary number [-]
N_p	Dimensionless incremental oil from an EOR process [-]
$P.V.$	Cell pore volume [m^3]
p_c	Capillary pressure [Pascal]
p_i	Pressure of phase i [Pascal]
p_{nw}	Non-wetting phase pressure [Pascal]

p_{ref}	Reference pressure [Pascal]
p_w	Wetting phase pressure [Pascal]
Q_i	Average generation of mass of phase i per volume time [kg/m^{3}s]
q_i	Volumetric flow rate of fluid i [m^3/s]
q_t	Total flow rate [m^3/s]
R	Remaining terms in a Taylor series
$r_{1,2}$	Principal radii of curvature of an interface [m]
S_i	Saturation of phase i [-]
S_{eff}	Effective saturation [-]
S_{orc}	Residual oil saturation to chemicals [-]
S_{orw}	Residual oil saturation to water [-]
S_{or}	Residual oil saturation [-]
S_{wf}	Shock front water saturation [-]
S_{wi}	Irreducible water saturation [-]
S_{wn}	Normalized water saturation [-]
T	Transmissibility [m^3/Pascal s]
t	Time [s]
V	Grid cell volume [m^3]
N	Number of grid cells

Bibliography

A. W. Adamson and A. P. Gast. *Physical Chemistry of Surfaces - sixth edition.* 2010.

A. Bashiri and N. Kasiri. Properly Use Effect of Capillary Number on Residual Oil Saturation. *SPE*, 2011.

J. P. Batycky and F.G. McCaffrey. Low Interfacial Tension Displacement Studies. 1978.

P. Behrenbruch and H.M. Goda. Two-Phase Relative Permeability Prediction: A Comparison of the Modified Brooks-Corey Methodology With a New Carman-Kozeny Based Flow Formulation. *SPE*, 2006.

S. E. Buckley and M. C. Leverett. Mechanisms of Fluid Displacement in Sands. 1941.

LP. Dake. *Fundamentals of Reservoir Engineering.* Shell Learning and Development, 1985.

M. Delshad, D. Bhuyan, G. A. Pope, and L. W. Lake. Effect of Capillary Number on Residual Saturation of a Three-Phase Micellar Solution. *SPE/DOE*, 1986.

Daved E. Evans. Numerical Simulation of Waterflooding in a Pancake Oil Column. 1970.

Xu Feng, Xiao Guo, Wanbin Wang, Nan Zhang, Sha Jia, and Xiaoqin Wang. Case Study: Numerical Simulation of Surfactant Flooding in Low Permeability Oil Field. 2011.

L. W. Harbert. Low Interfacial Tension Relative Permeability. *SPE*, 1983.

Masao Igarashi. A Termination Criterion for Iterative Methods Used to Find the Zeros of Polynomials. *Mathematics of Computation*, 42:165–171, 1984.

B. Yadali Jamaloei and M. Rafiee. Effect of Monovalent and Divalent Ions on Macroscopic Behaviour of Surfactant Flooding. 2008.

Larry W. Lake. *Enhanced Oil Recovery.* Prentice Hall Incorporated, 1989.

Knut-Andreas Lie, Stein Krogstad, Ingeborg S. Ligaarden, Jostein R. Natvig, Halvor M. Nilsen, and Bård Skaflestad. Open-source MATLAB Implementation of Consistent Discretisations on Complex Grids. 2012.

Thomas Lunde. Comparison Between Mimetic and Two-Point Flux-Approximation Schemes on PEBI-Grids. Master's thesis, UiO, 2007.

Fatemi S. Mobeen and Riyazz Kharrat. Investigation of Steam Assisted Gravity Drainage (SAGD) and Expanding Solvent-SAGD (ES-SAGD) Proesses in Complex Fractured Moldes: Effects of Fracture's Geometrical Properties. 2011.

Richard D. Neidinger. Introduction to Automatic Differentiation and MATLAB Object-Oriented Programming. *SIAM*, 2010.

L. Scandellari Nilsen, P. E. Øren, S. Bakke, and A. Henriquez. Prediction of Relative Permeability and Capillary Pressure from a Pore Model. *SPE*, 1996.

S. Njå. Improves Oil Recovery - The Norwegian Case. 1994.

Norwegian Petroleum Directorate. URL `http://www.npd.no/en/Publications/Facts/Facts-2013/Chapter-3/`.

OPM. URL `http://www.opm-project.org/`.

Donald W. Peaceman. *Fundamentals of Numerical Reservoir Simu.* Elsevier, 1977.

Gary A. Pope. Course Notes 323L - Chapter 3 - Multiphase and Mulicomponent Unsteady State Flow in Permeable Media.

Gary A. Pope. The Application of Fractional Flow Theory to Enhanced Oil Recovery. *SPE*, 1980.

Milton J. Rosen and Joy T. Kunjappu. *Surfactants and Interfacial Phenomena.* Wiley, 2012.

Michael Schafer. *Computational Engineering - Introduction to Numerical Methods.* Springer Berlin Heidelberg, 2006.

Schlumberger. *Eclipse Technical Description.* Schlumberger, 2011a.

Schlumberger. *Eclipse Reference Manual.* Schlumberger, 2011b.

P. Shen, B. Zhu, X. B. Li, and Y. S Wu. The Influence of Interfacial Tension on Water/Oil Two-Phase Relative Permeability. *SPE*, 2006.

James J. Sheng. *Modern Chemical Enhanced Oil Recovery - Theory and Practice.* Gulf Professional Publishing, 2011.

Wan Rosli Wan Sulaiman and Euy Soo Lee. Simulation of Surfactant Based Enhanced Oil Recovery. *The Open Petroleum Engineering Journal*, 2012.

W. Sun and Y. X. Yuan. Optimization Theory and Methods - Nonlinear Programming. pages 71–72, 2006.

C. R. Swaminathan. A Time-Implicit Filling Algorithm. 1994.

Frank M. White. *Fluid Mechanics.* McGraw - Hill. International Edition, 2008.

www.sintef.no/MRST. URL `www.sintef.no/MRST`.

Victor M. Ziegler and Lyman L. Handy. Effect of Temperature on Surfactant Adsorption in Porous Media. 1981.

Anatoly B. Zolotukhin and Jann-Rune Ursin. *Introduction to Petroleum Reservoir Engineering.* 2000.

Appendices

Appendix A

Implemented Keywords

A.1 SURFST

Function interpolating tabulated IFT data at a given surfactant concentration.

```
function f = assignSURFST(f, surfst, reg)
f.iftOWsurf = @(c) iftOWsurf(c, surfst, reg);
end

function v = iftOWsurf(c, surfst, reg)
satinx = getRegMap(c, reg.SATNUM, reg.SATINX);
surfst = extendTab(surfst);
v = interpReg(surfst, c, satinx);
end
```

A.2 SURFVISC

Function interpolating tabulated surfactant viscosity data at a given surfactant concentration.

```
function f = assignSURFVISC(f, surfvisc, reg)
f.muS = @(c) visc(c, surfvisc, reg);
end

function v = visc(c, surfvisc, reg)
satinx = getRegMap(c, reg.SATNUM, reg.SATINX);
surfvisc = extendTab(surfvisc);
v = interpReg(surfvisc, c, satinx);
end
```

A.3 SURFROCK

Function assigning rock density and adsorption index in different regions in the grid. This function is used in adsorption calculations.

```
function f = assignSURFROCK(f, surfrock, reg)

f.surfrock=[];
ind = 1;

if ischar(reg.SATINX)
    if reg.SATINX == ':'
        f.surfrock.type = surfrock{1}(1);
        f.surfrock.rhoR = surfrock{1}(2);
    else
    end
else
    for i = 1:numel(reg.SATINX)
        for j = 1:numel(reg.SATINX{i})
            f.surfrock.type(ind,1) = surfrock{i}(1);
            f.surfrock.rhoR(ind,1) = surfrock{i}(2);
            ind = ind + 1;
        end
    end
end
end
```

A.4 SURFADS

Function to interpolate adsorption data at a given surfactant concentration

```
function f = assignSURFADS(f, surfads, reg)
f.surfads = @(c, varargin) ads(c, surfads, reg, varargin{:});
end

function v = ads(c, surfads, reg, varargin)
satinx = getRegMap(c, reg.SATNUM, reg.SATINX, varargin{:});
surfads = extendTab(surfads);
v = interpReg(surfads, c, satinx);
end
```

A.5 SURFCAPD

Function for interpolating the logarithmic value of N_c in tabulated data fo find the miscibility conditions.

```
function f = assignSURFCAPD(f, surfcapd)

    f.misc = @(logCAPN)misc(logCAPN, surfcapd);
end

function v = misc(logCAPN, surfcapd, varargin)
surfcapd = extendTab(surfcapd);

logCAPN(isinf(logCAPN))= min(surfcapd{1}(:,1));
v = interptable(surfcapd{1}(:,1), surfcapd{1}(:,2), logCAPN);
end
```

Appendix B

Physical effects

B.1 Surfactant adsorption

This function calculates the adsorption term in the discrete conservation equation for surfactant. `tmp` is an output variable used for de-bugging.

```
function [ads,tmp] = ...
    surfactantAds(f,s,poro,cmax,state0,pvMult,pvMult0,dt,c,c0)
        ads.type2 = ...
            (s.pv/dt).*(f.surfrock.rhoR.*((1-poro)./poro). ...
        *(pvMult.*f.surfads(c) - pvMult0.*f.surfads(state0.cmax)));
                    ind=c<state0.cmax;
                    ads.type2(ind)=ads.type2(ind)*0;
        tmp.type2 = (f.surfrock.rhoR.*((1-poro)./poro). ...
        *(pvMult.*f.surfads(cmax)));

        ads.type1 = ...
            (s.pv/dt).*(f.surfrock.rhoR.*((1-poro)./poro). ...
        *(pvMult.*f.surfads(c) - pvMult0.*f.surfads(c0)));
        tmp.type1 = (f.surfrock.rhoR.*((1-poro)./poro).*(pvMult. ...
        *f.surfads(c)));


        ads = (1-rem(1,f.surfrock.type)).*ads.type1 + ...
            rem(1,f.surfrock.type).*ads.type2;
        tmp = (1-rem(1,f.surfrock.type)).*tmp.type1 + ...
            rem(1,f.surfrock.type).*tmp.type2;
end
```

B.2 Computing Capillary Number

Function for computing N_c.

```
function CAPN = computeNcSurf(p, f, state, g, G, s)

bO = f.bO(p);
bO = bO.val;
p_temp   = state.pressure + bO.*f.rhoOS.*g.*(G.cells.centroids(:,3));
gradp    = s.grad(p_temp);
Kgradp   = s.T.*gradp;
term     = s.divnc(((1./G.faces.areas(s.internal)).*Kgradp).^2);

CAPN     = ((term./2).^0.5)./f.iftOWsurf(state.c);
end
```

B.3 Capillary Pressure and Relative Permeability Calculations

The SWOF keyword is used for saturation dependent relative permeability and capillary pressure data. This function calculates and averaged relative permeability for the miscible and immiscible curve. It also interpolates capillary pressure from tabulated data at a given surfactant concentration.

```
function f = assignSWOF(f, swof, reg)

f.krW  = @(sw, varargin)krW(sw, swof, reg, varargin{:});
f.krWmisc = @(sw, varargin)krWmisc(sw, swof, reg, varargin{:});

f.krOW = @(so, varargin)krOW(so, swof, reg, varargin{:});
f.krOWmisc = @(so, varargin)krOWmisc(so, swof, reg, varargin{:});


f.avgRelPerm = @(m, sw)avgRelPerm(m, sw, swof, reg);


f.pcOW = @(sw, varargin)pcOW(sw, swof, reg, varargin{:});
swcon  = cellfun(@(x)x(1,1), swof);
ntsat = numel(reg.SATINX);
if ntsat == 1
    f.sWcon = swcon(1);
else
    f.sWcon = swcon(reg.SATNUM);
end
end

function [u, v] = avgRelPerm(m, sw, swof, reg)
satinx = getRegMap(sw, reg.SATNUM, reg.SATINX);
surfinx = getRegMap(sw, reg.SURFNUM, reg.SURFINX);

lower   = m*swof{2}(1,1) + (1-m)*swof{1}(1,1);
upper   = m*swof{2}(end,1) + (1-m)*swof{1}(end,1);

Seff = (sw - lower)./(upper - lower);
Sbar_1 = Seff.*(swof{1}(end,1) - swof{1}(1,1)) + swof{1}(1,1);
Sbar_2 = Seff.*(swof{2}(end,1) - swof{2}(1,1)) + swof{2}(1,1);

Tw = cellfun(@(x)x(:,[1,2]), swof, 'UniformOutput', false);
Tw = extendTab(Tw);

To = cellfun(@(x)x(:,[1,3]), swof, 'UniformOutput', false);
To = extendTab(To);
```

```
krMiscW = interpReg(Tw,Sbar_2,surfinx);
krImmiscW = interpReg(Tw,Sbar_1, satinx);
krMiscO = interpReg(To,Sbar_2, surfinx);
krImmiscO = interpReg(To,Sbar_1, satinx);


u = m.*krMiscW + (1-m).*krImmiscW;
v = m.*krMiscO + (1-m).*krImmiscO;

end


function v = krW(sw, swof, reg, varargin)
satinx = getRegMap(sw, reg.SATNUM, reg.SATINX, varargin{:});
T = cellfun(@(x)x(:,[1,2]), swof, 'UniformOutput', false);
T = extendTab(T);
v = interpReg(T, sw, satinx);
end

function v = krWmisc(sw, swof, reg, varargin)
surfinx = getRegMap(sw, reg.SURFNUM, reg.SURFINX, varargin{:});
T = cellfun(@(x)x(:,[1,2]), swof, 'UniformOutput', false);
T = extendTab(T);
v = interpReg(T, sw, surfinx);
end

function v = krOW(so, swof, reg, varargin)
satinx = getRegMap(so, reg.SATNUM, reg.SATINX, varargin{:});
T = cellfun(@(x)x(:,[1,3]), swof, 'UniformOutput', false);
T = extendTab(T);
v = interpReg(T, 1-so, satinx);
end

function v = krOWmisc(so, swof, reg, varargin)
surfinx = getRegMap(so, reg.SURFNUM, reg.SURFINX, varargin{:});
T = cellfun(@(x)x(:,[1,3]), swof, 'UniformOutput', false);
T = extendTab(T);
v = interpReg(T, 1-so, surfinx);
end



function v = pcOW(sw, swof, reg, varargin)
satinx = getRegMap(sw, reg.SATNUM, reg.SATINX, varargin{:});
T = cellfun(@(x)x(:,[1,4]), swof, 'UniformOutput', false);
T = extendTab(T);
v = interpReg(T, sw, satinx);
end
```

Appendix C

Implemented conservation equations and surfactant specific mechanisms

This is the `.m` file containing the discrete conservation equations for oil, water and surfactant. It also contain calculations for transport properties and implementation of the surfactant specific mechanisms described in chapter. 7.

```
function [eqs, hst] = eqsfiOWSurfactantExplicitWells(state0, ...
    state, dt, G, W, s, f, varargin)
%% Generate equations for a Oil—Water—Surfactant system.

opt = struct('Verbose', mrstVerbose, ...
             'reverseMode', false, ...
             'scaling', [], ...
             'resOnly', false, ...
             'history', []);

opt = merge_options(opt, varargin{:});

if ¬isempty(opt.scaling)
    scalFacs = opt.scaling;
else
    scalFacs.rate = 1; scalFacs.pressure = 1;
end

hst = opt.history;
```

```
%% current variables:
p     = state.pressure;
sW    = state.s(:,1);
c     = state.c;
g = gravity();

pBHP = vertcat(state.wellSol.pressure);
qWs  = vertcat(state.wellSol.qWs);
qOs  = vertcat(state.wellSol.qOs);
% surf_num  = vertcat(state.wellSol.surf);
% surf = surf_num;

wciSurf_num = getWellSurfactant(W);
wciSurf = wciSurf_num;

%% previous variables
p0  = state0.pressure;
sW0 = state0.s(:,1);
c0  = state0.c;


%% Initialization of independent variables
zw = zeros(size(pBHP));

if opt.resOnly
    % ADI variables aren't needed since we are only computing the ...
        residual.
elseif ¬opt.reverseMode
    [p, sW, c, qWs, qOs, wciSurf, pBHP]  = initVariablesADI(p, ...
        sW, c, qWs, qOs, wciSurf_num, pBHP);
else
    [p0, sW0, c0, ¬, ¬, ¬, zw] = initVariablesADI(p0, sW0, c0,...
        zeros(size(qWs)), ...
        zeros(size(qOs)), ...
        zeros(size(wciSurf_num)), ...
        zeros(size(pBHP))...
        );
end

cmax  = max(c,state0.cmax);

g  = norm(gravity);
[Tw, dzw, Rw, wc, perf2well, pInx, iInxW] = getWellStuff(W);

%---------------------
% Check for pressure-dependent transmissibility multiplier
trMult = 1;
if isfield(f, 'tranMultR')
    trMult = f.tranMultR(p);
end
```

```
% Check for pressure—dependent pore—volume multiplier
pvMult = 1; pvMult0 = 1;
if isfield(f, 'pvMultR')
    pvMult =  f.pvMultR(p);
    pvMult0 = f.pvMultR(p0);
end



% Surfactant injection well:
cw         = c(wc);
cw(iInxW) = wciSurf;

%% Compute surfactant specific mechanisms
% Compute capillary pressure
pcOW = 0;

if isfield(f, 'pcOW');
    pcOW = ...
        f.pcOW(sW).*f.iftOWsurf(c0)./f.iftOWsurf(zeros(G.cells.num,1));
    pcOWw = pcOW(wc);
end

% Compute capillary number and determine miscibility
CAPN = computeNcSurf(p, f, state0, g, G, s);
logCAPN = log10(CAPN);
m = f.misc(logCAPN);

% Determine rel. perms for near miscible conditions
[krW, krO] = f.avgRelPerm(m, sW);


% Viscosities
muW = f.muW(p—pcOW); %Added "—pcOW" term to evaluate at water ...
    pressure
muS = f.muS(c0);    % Interpolerer fra tabell
muWeff = muW.*muS./f.muWref;

%% Water props

bW = f.bW(p—pcOW);
rhoW   = bW.*f.rhoWS;

rhoWf  = s.faceAvg(rhoW);
mobW   = trMult.*krW./muWeff;
dpW     = s.grad(p—pcOW) — g*(rhoWf.*s.grad(G.cells.centroids(:,3)));

% water upstream—index
upc = (double(dpW)≥0);
bWvW = s.faceUpstr(upc, bW.*mobW).*s.T.*dpW;
bWvS = s.faceUpstr(upc, bW.*mobW.*c).*s.T.*dpW;
```

```
%% Oil props
bO      = f.bO(p);
rhoO    = bO.*f.rhoOS;
rhoOf   = s.faceAvg(rhoO);
mobO    = trMult.*krO./f.BOxmuO(p);
dpO     = s.grad(p) - g*(rhoOf.*s.grad(G.cells.centroids(:,3)));

% oil upstream-index
upc = (double(dpO)≥0);
bOvO    = s.faceUpstr(upc, mobO).*s.T.*dpO;


%% Values for cells containing wells
bWw      = bW(wc);
bOw      = bO(wc);

mobWw    = mobW(wc);
mobOw    = mobO(wc);
mobSw    = mobW(wc).*cw;

%producer mobility
bWmobWw   = bWw.*mobWw;
bOmobOw   = bOw.*mobOw;

bWmobSw = bWw.*mobSw;

%set water injector mobility: mobw = mobw+mobo, mobo = 0;
bWmobWw(iInxW) = bWw(iInxW).*(mobWw(iInxW) + mobOw(iInxW));
bOmobOw(iInxW) = 0;

pw   = p(wc);

% Residual equations for source terms
% Transmissibility and pressure differential is common to water and
% surfactant

%% Source terms
% Water
tmp = Tw.*(pBHP(perf2well) - pw + pcOWw + g*dzw.*rhoW(wc));
bWqW   = -bWmobWw.*tmp;

% Surfactant
bWqS = -bWmobSw.*tmp;

% Oil
bOqO   = -bOmobOw.*Tw.*(pBHP(perf2well) - pw + g*dzw.*rhoO(wc));

%% EQUATIONS
% oil:
eqs{1} = (s.pv/dt).*( pvMult.*bO.*(1-sW) - ...
    pvMult0.*f.bO(p0).*(1-sW0) ) + s.div(bOvO);
eqs{1}(wc) = eqs{1}(wc) + bOqO;
```

```
% water:
eqs{2} = (s.pv/dt).*( pvMult.*bW.*sW     - pvMult0.*f.bW(p0).*sW0 ...
         ) + s.div(bWvW);
eqs{2}(wc) = eqs{2}(wc) + bWqW;      % bWqW = qW/Bw (bW = 1/Bw)

% Surfactant:
poro =  s.pv./G.cells.volumes;

% Calculates adsorption term
[ads,tmp] = surfactantAds(f, s, poro, cmax, state0, pvMult, ...
    pvMult0, dt, c, c0);

eqs{3} = (s.pv/dt).*(pvMult.*bW.*sW.*c - ...
    pvMult0.*f.bW(p0).*sW0.*c0) +  s.div(bWvS) + ads;
eqs{3}(wc) = eqs{3}(wc) + bWqS;


% Well equations
zeroW = 0*zw;

eqs{4} = Rw'*bWqW + qWs + zeroW;
eqs{5} = Rw'*bOqO + qOs + zeroW;


% Trivial constraint - this is only to get the adjoint partial ...
    derivatives
eqs{6} = wciSurf - wciSurf_num + zeroW(iInxW);

% Last eq: boundary cond
eqs{7} = handleBC(W, pBHP, qWs, qOs, [], scalFacs);% + zeroW;
end

%% Functions
function wciSurf = getWellSurfactant(W)
if isempty(W)
    wciSurf = [];
    return
end
inj    = vertcat(W.sign)==1;
surfInj = cellfun(@(x)¬isempty(x), {W(inj).surf});
surfVal = zeros(nnz(inj), 1);
surfVal(surfInj) = vertcat(W(inj(surfInj)).surf);
wciSurf = rldecode(surfVal, cellfun(@numel, {W(inj).cells}));
end
```