7

Using State Functions and MRST's AD-OO Framework to Implement Simulators for Chemical EOR

XIN SUN, KNUT-ANDREAS LIE, AND KAI BAO

Abstract

Surfactant and polymer flooding, alone or in combination, are common and effective chemical enhanced oil recovery (EOR) methods. This chapter reviews the main physical mechanisms and presents how the corresponding mathematical flow models are implemented as an add-on module to the MATLAB Reservoir Simulation Toolbox (MRST) to provide a powerful and flexible tool for investigating flooding processes in realistic reservoir scenarios. Using a so-called limited compositional model, surfactant and polymer are both assumed to be transported in the water phase only but also adsorbed within the rock. The hydrocarbon phases are described with the standard three-phase black-oil equations. The resulting flow models also take several physical effects into account, such as chemical adsorption, inaccessible pore space, permeability reduction, effective solution viscosities, capillary pressure alteration, relative permeability alteration, and so on. The new simulator is implemented using the object-oriented, automatic differentiation (AD-OO) framework from MRST and can readily utilize features such as efficient iterative linear solvers with constrained pressure residual (CPR) preconditioners, efficient implicit and sequential solution strategies, advanced timestep controls, improved spatial discretizations, etc. We describe how the computation of fluid properties can be decomposed into state functions for better granularity and present several numerical examples that demonstrate the software and illustrate different physical effects. We also discuss the resolution of trailing chemical waves and validate our implementation against a commercial simulator.

7.1 Introduction

Mature fields account for a considerable part of the world's current crude oil production. New discoveries are becoming more scarce and it is thus increasingly

https://doi.org/10.1017/9781009019781.013 Published online by Cambridge University Press

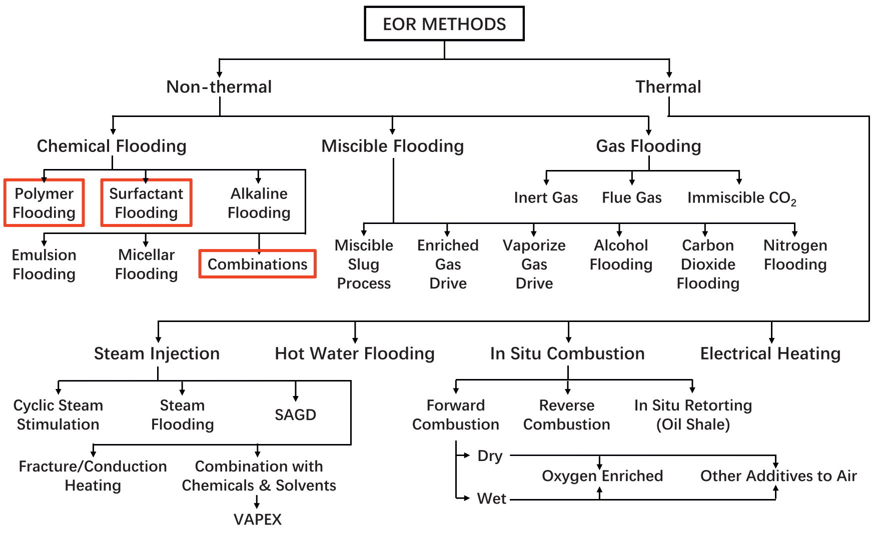


Figure 7.1 Summary of EOR methods, adapted from [2]. Herein, we consider methods marked in red.

important to enhance oil recovery from existing fields in response to global energy demand and depleted reserves. Enhanced oil recovery (EOR) refers to processes that change the physical and chemical properties of the rock and the reservoir fluids to recover more hydrocarbons. When EOR is performed after gas injection or waterflooding, it is referred to as tertiary recovery. EOR methods can roughly be divided into two major categories (Figure 7.1) [2]: thermal and nonthermal methods.

The dominant mechanism of thermal methods is to increase the reservoir temperature by injecting heat into the ground, thereby further reducing the viscosity of the crude oil and improving its fluidity within a high-temperature environment. Thermal methods mainly consist of steam injection (steam flooding or steam huff-n'-puff), in situ combustion, hot water flooding, and electrical heating.

The nonthermal category includes a wider range of methods and more diverse recovery mechanisms, which can be categorized as chemical flooding, gas drives, miscible displacements, and microbial methods. The enhanced recovery mechanisms mainly consist of increasing sweep efficiency, improving the efficiency of the displacement fluid, improving the flow properties of the in situ crude oil by changing its density and viscosity or its interfacial tension with water, and so on. Among the various EOR methods, chemical EOR is very effective but expensive. One particular challenge is that whereas the chemical processes that lead to enhanced recovery are well known in theory and their effect can be proved in laboratory, upscaling and applying them efficiently and economically on a field scale is a complicated balancing act. However, with the gradual depletion of petroleum resources and with oil prices that remained at relatively high and stable levels for a long time, chemical EOR received renewed interest in many oil companies, especially in national oil companies in China [33].

Chemical EOR methods are basically classified into three types, named after the chemical substances involved: polymers, surfactants, and alkali. In addition, people have created new injection methods by adjusting the concentration of the three substances and combining them, such as surfactant–polymer (SP) flooding, alkali–surfactant–polymer (ASP) flooding, emulsion flooding, and micellar flooding [2]. Injection of alkaline liquids can enhance the effect of polymer and surfactant but can also lead to problems such as excessive formation loss and severe scaling. For these reasons, the use of alkali is not as common as polymer and surfactant in chemical EOR processes. This chapter thus focuses on two chemical EOR methods: surfactant and polymer flooding, on their own or in combination.

Polymer flooding: In polymer flooding, water-soluble polymers are added to the injected water to reduce its mobility and hence improve the local displacement and volumetric sweep efficiency of the waterflood [20]. A polymer is generally a chemical compound of large molecular mass, consisting of repetitive structural units, called monomers, bonded together by covalent chemical bonds. These come in two main forms: biopolymers like xanthan gum (a polysaccharide) come as a broth or in powder form, whereas synthetic polymers like partially hydrolyzed polyacrylamides consist of synthetic flexible straight chains of acrylamide monomers and come either as powder or as a water-in-oil microemulsion [36].

The primary mechanism of polymer flooding is that dissolved polymer molecules increase the brine viscosity, which increases the saturation behind the water front and enables the water drive to push more oil through the reservoir, thereby leaving behind less mobile oil in water-swept areas. A higher viscosity also reduces the injected water's tendency of channeling through high-flow zones.

Polymer can be adsorbed onto the surface of the reservoir rock, depending on polymer type and rock and brine properties, which will reduce porosity and permeability of the reservoir rock [36]. Moreover, the diluted polymer solution is in most cases pseudoplastic or shear-thinning and hence has lower viscosity near injection wells and other high-flow zones where shear rates are high. This non-Newtonian fluid rheology improves injectivity and gradually introduces the desired mobility control in terms of a stronger displacement front but may also reduce conformance effects because the polymer solution will have a higher tendency to flow through high-permeability regions. Polymer solutions can also exhibit pseudodilatant or shear-thickening behavior, which improves sweep efficiency and reduces injectivity.

Onshore, polymer flooding is considered a mature technology with well-proven results and relatively low commercial and technical risk. Many countries, especially China, have applied this technology for decades to improve the sweep efficiency for waterfloods with unfavorable mobility ratios and/or to reduce the water mobility in high-permeability zones and improve the displacement of oil in low-permeability zones; e.g., in reservoirs with significant vertical stratification. Offshore, applications of polymer flooding are still few because of challenges related to logistics and platform space, high-salinity formation water, large well spacing, stability under injection, treatment of polymer and produced water, as well as other health, safety, and environment requirements [4].

Surfactant flooding: The main mechanism of surfactant flooding is to mobilize trapped oil by reducing the interfacial tension, which is similar to miscible gas flooding. In more detail, a surfactant consists of two parts, the hydrophilic head group and the lipophilic tail chain [20]. The surfactant can therefore accumulate in a large amount at the oil–water interface, adjusting the polarity difference between the oil and water phases, thereby reducing the oil–water interfacial tension. It can also increase the capillary number (i.e., the relative effect of viscous drag forces versus surface tension forces acting across an interface between oil and water) and thereby improve the relative permeability of the oil and water phases.

Like polymer, surfactant can be adsorbed on the rock surface, depending on the surfactant type and rock properties. This happens when the positively charged hydrophilic head group in the surfactant bonds with a hydroxyl group on the rock surface. This will cause the surfactant to start accumulating on the rock surface and form an adsorption layer. This adsorption has two effects: One is to change the wettability of the rock surface, and the other is to reduce the effective concentration of the surfactant in the solution, thereby reducing its ability to reduce the interfacial tension between oil and water. In general, we treat the adsorption of surfactants as an undesirable behavior that causes a loss of surfactants. As mentioned previously, the surfactant molecules in the solution are able to adsorb at the oil–water interface to reduce the oil–water interfacial tension. On the one hand, low interfacial tension helps to reduce the residual oil saturation in the reservoir and improve the oil washing efficiency. On the other hand, it can increase the capillary number, which appears in the model as an increase in the relative permeability of the oil and water phases.

Combined surfactant and polymer flooding: Decreased interfacial tension and wettability alteration caused by injected low-concentration surfactant will in many cases increase the relative permeability of the aqueous phase without significantly improving the mobility ratio of oil and water. As a result, the pure surfactant slug will finger into the oil bank and significantly decrease the sweep efficiency. A reliable remedy is to add polymer to the surfactant solution to counterbalance the mobility decrease and improve the overall sweep efficiency in the reservoir.

In reality, one rarely implements surfactant flooding alone without adding polymer [34]. This chapter therefore also discusses SP flooding as a combined chemical EOR method. In addition to the pure polymer and surfactant flooding mechanisms just described, the coexistence of polymer and surfactant produces a synergistic effect, resulting in a "one plus one is greater than two" effect. The main synergistic mechanisms are the following:

1. The viscosifying effect of the polymer can reduce the diffusion rate of the surfactant, thereby reducing the loss of surfactant.
2. The polymer can react with calcium and magnesium ions in the formation water and prevent these divalent ions from reacting with the surfactant to form calcium and magnesium salts with low interfacial activity.
3. The coexistence of polymer and surfactant leads to competitive adsorption on the rock surface, which can reduce adsorption loss of surfactant on the rock surface.
4. The polymer can improve the stability of the oil-in-water emulsion formed by the surfactant and further improve the sweep and washing efficiency.
5. Some surfactants can form a complex structure with the polymer to further increase the viscosifying ability of the polymer.

In its most basic form, SP flooding is described by a flow model that consists of two or three phases and three to five fluid (pseudo)components. Compared with the standard black-oil models, the presence of surfactant and long-chain polymer molecules in the water phase introduces a series of new flow effects.

In summary, understanding and being able to accurately simulate the interaction between polymer, surfactant, water, oil, and reservoir rocks on a reservoir scale is important for designing successful surfactant–polymer injection projects. In addition to the basic effects discussed so far, the fluid chemistry of the injected water and the resident water tends to significantly affect the viscosifying ability of the polymer and the interfacial activity of the surfactant. More advanced models of SP flooding should therefore also consider the effects that pH, salt, microemulsion, etc., have on the displacement process and the recovery factor.

In the following, we will review the basic flow equations for surfactant–polymer flooding and show how to use so-called state functions and the object-oriented, automatic differentiation (AD-OO) framework of the MATLAB Reservoir Simulation Toolbox (MRST) to implement a complete and easily extensible, three-phase, fully implicit chemical flooding simulator. At the end of the chapter, we also discuss a few simulation examples and compare the results with a commercial simulator.

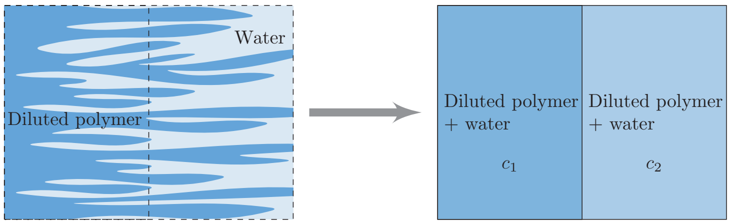


Figure 7.2 Averaging of two partially mixed aqueous fluids over two (large) grid blocks to give a single immiscible fluid phase with polymer concentration c.

7.2 Effective Modeling Using Black-Oil-Type Equations

This section explains physical assumptions and describes the basic flow equations for surfactant and polymer flooding, which are built as extensions of the general black-oil model, which is, e.g., discussed in detail in chapter 11 of the MRST textbook [22]. We will also briefly outline how the various fluid properties that enter the flow equations are interrelated and implemented using so-called state functions, which were recently introduced in MRST. If you are not yet familiar with these, we encourage you to read Chapter 5 before continuing.

7.2.1 Immiscible Flow Models

Polymer flooding involves two different aqueous fluids (water and polymerized water) that can be fully or partially mixed inside the reservoir, depending on heterogeneity and the displacement process. In principle, one should be able to simulate polymer flooding accurately on the laboratory scale, given a sufficiently fine grid and possibly with the use of higher-order numerical discretizations. For field-scale simulations, on the other hand, the required grid resolution is far beyond what is computationally tractable. Instead, it is common to upscale the problem and represent it as an *immiscible* fluid system, in which water and polymerized water are considered as two pseudophases that together form a single immiscible fluid phase; see Figure 7.2. The polymer content of this immiscible fluid phase is represented by a concentration. To account for the effect of partial mixing of the two fluids, we will modify the fluid properties of the pseudophases so that they depend on the polymer concentration.

We use the same approach for the surfactant, which is assumed to exist within both the water and the polymer pseudophases, and thus obtain two different concentrations, c_p and c_s, for polymer and surfactant, respectively.

In the general case of SP flooding, we therefore have a system with five different components – oil (O), gas (G), water (W), polymer (P), and surfactant (S) – that can separate into three phases: an oleic (o), a gaseous (g), and an aqueous

(w) phase. The latter splits into a pure water (ww) and a diluted polymer (wp) pseudophase. Each phase has an associated saturation S_α, a density ρ_α, a viscosity μ_α, and a volumetric flow rate $\vec{v}_\alpha$. These are given by modified versions of Darcy's law:

$$\vec{v}_o = -\frac{k_{ro}(S_o, c_s)}{\mu_o}\mathbf{K}(\nabla p_o - \rho_o g \nabla z) \tag{7.1a}$$

$$\vec{v}_g = -\frac{k_{rg}(S_g)}{\mu_g}\mathbf{K}(\nabla p_g - \rho_g g \nabla z) \tag{7.1b}$$

$$\vec{v}_w = -\frac{k_{rw}(S_w, c_s)}{\mu_{w,\text{eff}}(p, c_s, c_p) R_k(c_p^a)}\mathbf{K}(\nabla p_w - \rho_w g \nabla z) \tag{7.1c}$$

$$\vec{v}_{wp} = -\frac{k_{rw}(S_w, c_s)}{\mu_{wp}(p, c_s, c_p) R_k(c_p^a)}\mathbf{K}(\nabla p_w - \rho_w g \nabla z). \tag{7.1d}$$

Here, $\mathbf{K}$ is permeability, g is gravitational acceleration, and ∇z is the depth gradient, and μ_α, $k_{r\alpha}$, and p_α denote the viscosity, relative permeability, and phase pressure of phase α, respectively. All relative permeabilities, except for the gas phase, depend on both the phase saturation and the surfactant concentration but not on the polymer concentration, whereas the phase pressures are given by the relation

$$p_o = p_w + p_c(S_w, c_s), \tag{7.2}$$

where the capillary pressure function p_c depends on the water saturation and surfactant concentration. Finally, the nondecreasing function $R_k(c_p^a)$ models permeability reduction of the rock to the aqueous phase(s) caused by absorbed polymer.

As already explained, the two aqueous pseudophases combine into an immiscible aqueous phase. For the Todd–Longstaff mixture model, which we will discuss in more detail in Subsection 7.2.2, the volume of the two pseudophases split with the ratio given by $\bar{c}_p = c_p / c_{p,\,\max}$:

$$S_{ww} = (1 - \bar{c}_p)S_w \qquad \text{and} \qquad S_{wp} = \bar{c}_p S_w.$$

Here, $c_{p,\,\max}$ denotes the maximum possible polymer concentration. We assume that the polymer content has a negligible effect on the densities of the aqueous pseudophases, so that $\rho_{ww} = \rho_{wp} = \rho_w$. Finally, we introduce a solid phase (s) to account for the adsorption of polymer and surfactant onto the rock surface, whose amounts are denoted $c_p^a(c_p)$ and $c_s^a(c_s)$, respectively.

We can now write the mass conservation for component $i \in \{O, G, W, P, S\}$ as

$$\frac{\partial}{\partial t}\left(\sum_{\alpha=o,\,g,\,w} \phi b_\alpha x_{i,\alpha} S_\alpha + \rho_s(1-\phi)x_{i,s}\right) + \nabla \cdot \left(\sum_{\alpha=o,\,g,\,w} b_\alpha x_{i,\alpha}\vec{v}_\alpha\right) = q_i. \tag{7.3}$$

Here, ϕ is the porosity of the reservoir rock, q_i denotes the source of component i, and $x_{i,\alpha}$ is either a volume fraction or concentration of component i in phase

Table 7.1 *Expressions for the volume fractions or concentrations $x_{i,\alpha}$ in the general immiscible surfactant–polymer model (with phases α as rows and components i as columns).*

	O	G	W	P	S
o	1	r_s	0	0	0
g	r_v	1	0	0	0
w	0	0	1	c_p	c_s
s	0	0	0	$c_p^a(c_s)$	$c_s^a(c_s)$

$\alpha \in \{o, g, w, s\}$; expressions for these are summarized in Table 7.1. As in the standard black-oil model, we have introduced pressure-dependent shrinkage factors b_α to relate densities ρ_α at reservoir conditions to densities ρ_α^0 at surface conditions. Likewise, the solution gas–oil ratio r_s accounts for gas dissolved in oil at reservoir conditions, whereas the vaporized oil–gas ratio r_v accounts for oil vaporized in gas. For the polymer component equation, we must additionally replace the water saturation S_w by $S_w(1 - s_{ipv})$, where s_{ipv} is a scalar quantity that accounts for inaccessible pore space (see Subsection 7.2.2), and likewise replace the water flow rate $\vec{v}_w$ by the flow rate of the diluted polymer pseudophase $\vec{v}_{wp}$.

7.2.2 *Physical Effects of Polymer*

In this and the next subsection, we discuss various effective properties for modeling the pertinent EOR mechanisms for polymer and surfactant flooding. We will explain the underlying flow physics, outline how the corresponding effective properties are implemented in MRST, and show how they affect the displacement process. We will then come back to more details of how the various effective properties are integrated into the overall simulator framework to form appropriate model classes in Section 7.3.

Effective Viscosities

Polymer flooding is also called tackifying or thickening waterflooding. This illustrates, from another aspect, the importance of the polymer's viscosifying effect on EOR. Increasing the water viscosity can reduce the water–oil mobility ratio, thereby reducing the fingering effect of water and improving the spreading efficiency of the displacement agent, and ultimately enhance the oil recovery. The molecular structure of the polymer explains the reason for its thickening effect:

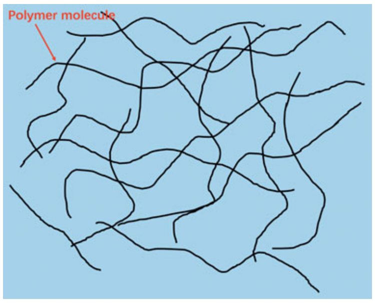


Figure 7.3 Polymer molecules dissolved in water. The long-chain molecules, illustrated by black curves, fully stretch and entangle with each other to increase the viscosity of the aqueous solution.

1. Polymers have a high molecular weight because they are formed from a large number of repeating chain links.
2. The hydrophilic groups in these chain link undergo solvation in water, so that there is a layer of “water sheath” formed by solvated water outside the polymer molecules, which will increase the internal friction when the water phase moves relatively; that is, it increases the water viscosity.
3. Ionic hydrophilic groups dissociate in water, resulting in many chain links with the same type of charge. They repel each other, which makes the polymer molecules more stretchable in water (Figure 7.3) and gives better thickening capabilities.

In addition, if you have deployed polymer solutions in a laboratory experiment or in the oil field, you should find that polymer dissolution is a cautious and slow process. Once a large amount of polymer is added to the water for a short time, the dry polymer powder will aggregate into a mass and become extremely difficult to dissolve. The reason for this phenomenon is that the molecular weight of the polymer is relatively large, so the speed of its various links unfolding in water is relatively slow. For the same reason, the process of diluting high-concentration polymer solutions is also slow. This means that when we inject a relatively high concentration of polymer solution into the formation, we cannot assume that the polymer solution and the formation water are instantaneously miscible and form a single-phase state.

For the effective viscosities of the water–polymer mixture, we use the Todd–Longstaff mixing model [38], which introduces a mixing parameter $\omega \in [0,1]$ that takes into account the degree of mixing of polymer into water. The effect of polymer on the viscosity is included in the model as a multiplicative function γ_p.

For a fully mixed polymer solution, without surfactant and at reference pressure condition, we have

$$\mu_w(c_p) = \gamma_p(c_p)\mu_w. \tag{7.4}$$

For the partially mixed case, the viscosities of pseudophases wp and ww are, in the absence of surfactant and at reference pressure conditions, given by

$$\mu_{wp}(c_p) = \gamma_p(c_p)^{\omega}\gamma_p(c_{p,\,\max})^{1-\omega}\mu_w = \gamma_p^{wp}(c_p)\mu_w, \tag{7.5a}$$

$$\mu_{ww}(c_p) = \gamma_p(c_p)^{\omega}\gamma_p(0)^{1-\omega}\mu_w \qquad = \gamma_p(c_p)^{\omega}\mu_w. \tag{7.5b}$$

The latter equality follows because $\gamma_p(0) = 1$. The effective water viscosity is then calculated as a harmonic average of the contributions from the two pseudophases

$$\mu_{w,\text{eff}} = \left[\frac{1-\bar{c}_p}{\mu_{ww}} + \frac{\bar{c}_p}{\mu_{wp}}\right]^{-1} = \underbrace{\left[\frac{\gamma_p(c_p)^{\omega}}{1-\bar{c}_p+\bar{c}_p/\gamma_{p,\max}^{1-\omega}}\right]}_{\gamma_w^{\text{eff}}(c_p)}\mu_w, \quad \bar{c}_p = \frac{c_p}{c_{p,\,\max}}. \tag{7.6}$$

Microscopic displacement efficiency: To understand how the effective viscosities affect a displacement, we look at the fractional-flow theory for 1D displacements, which is a generalization of the classical Buckley–Leverett theory of pure waterflooding discussed in section 8.4 of the MRST textbook [22]. You can consult [27] for an early introduction to this theory for various displacement types, including polymer and surfactant flooding, and [6] for a more comprehensive overview.

To be specific, we consider polymer flooding consisting of two phases (oil and water) and three components (oil, water, polymer) in a 1D homogeneous, isotropic medium with uniform initial distribution of fluids and continuous injection of constant composition from the left end. We assume incompressible fluids with negligible capillarity, dispersion, and gravity. For simplicity, we only consider the fully mixed case with water viscosity given by (7.4). Finally, we assume constant porosity and rescale the equations to remove the dependence on flow rate, porosity, length, and area. This reduces the flow equations to the following first-order, quasilinear, hyperbolic system (for brevity, we drop subscripts p for polymer and w for water):

$$\partial_t S + \partial_x f(S,c) = 0, \tag{7.7a}$$

$$\partial_t\big(Sc\big) + \partial_x\big(cf(S,c)\big) = 0, \tag{7.7b}$$

$$f(S,c) = \frac{k_{rw}(S)}{k_{rw}(S) + k_{ro}(S)\mu_w(c)/\mu_o}. \tag{7.7c}$$

In fractional-flow theory, we consider initial-boundary data of the following form:

$$(S,c)(x,t) = \begin{cases} (S_L, c_L), & x = 0, \ \ t \geq 0 \\ (S_R, c_R), & x > 0, \ \ t = 0, \end{cases} \tag{7.8}$$

where (S_L, c_L) and (S_R, c_R) are constant states. In mathematical literature, it is more common to consider the *Riemann problem*, with (S_L, c_L) imposed as initial data on the left half-line ($x \leq 0, t = 0$). These two problems are mathematically equivalent and have the same solution, because the system (7.7) only has positive characteristics.

Solutions of the problem (7.7)–(7.8) will generally consist of a set of constant states separated by elementary waves (rarefaction waves, shocks, and contact discontinuities). These waves are functions of x/t only, which makes the overall solution self-similar. To determine the wave structure, we start by expanding the derivatives in (7.7) and writing the system in quasilinear form,

$$\mathbf{u}_t + \mathbf{A}(\mathbf{u})\mathbf{u}_x = \mathbf{0}, \qquad \mathbf{A}(\mathbf{u}) = \mathbf{A}(S,c) = \begin{bmatrix} f_s(S,c) & f_c(S,c) \\ 0 & f(S,c)/S \end{bmatrix}. \tag{7.9}$$

The eigenvalues of $\mathbf{A}$ are $\xi^S = f_s$ and $\xi^c = f/S$. Continuous elementary waves must be integral curves of the two corresponding right eigenvectors $\boldsymbol{\eta}^S$ and $\boldsymbol{\eta}^c$. A complete solution of the Riemann problem for all possible combinations of left and right states was first given by Isaacson [15]. We will briefly introduce some of the ingredients, but we do not explain the general solution in full detail.

For $\xi = \xi^S$, the eigenvector is $\boldsymbol{\eta}^S = (0,1)^T$ so that c is constant along the corresponding integral curve. This means that the *S-waves* can be found by solving the scalar Buckley–Leverett equation, $S_t + f(S)_x = 0$; the solution is discussed in section 8.4 of the MRST textbook [22].

For $\xi = \xi^c$, the eigenvector $\boldsymbol{\eta}^c = (f_c, \xi^c - \xi^S)$ gives a *linearly degenerate* characteristic field, because $\nabla_{(S,c)}\xi^c \cdot \boldsymbol{\eta}^c = 0$, so that the resulting *c-waves* are contact discontinuities and not proper rarefaction or shock waves. Because $\xi^c = f/S$ is constant along a contact discontinuity, these waves have no inherent self-sharpening to counteract numerical smearing, unlike shock waves, for which the characteristics of any intermediate state along the wave curve point into the shock.

Figure 7.4 explains the graphical construction for the special case with $S_L = 1$, $c_R = 0$, and $S_R = S_{wc}$ (connate water). This solution is simple to determine. Given an injection concentration c_L, we first determine the values S_1 and S_2 by solving two scalar equations (where $f(\cdot;c)$ is a function of one variable for each given c):

$$f'(S_1; c_L) = f(S_1; c_L)/S_1, \qquad f(S_2; 0) = f(S_1; c_L)\, S_2/S_1. \tag{7.10}$$

To determine the full solution, all that remains now is to compute the scalar Buckley–Leverett solutions that represent the S-waves for saturation values from 1 to S_1 and from S_2 to S_{wc}. As explained in the MRST textbook [22, section 8.4],

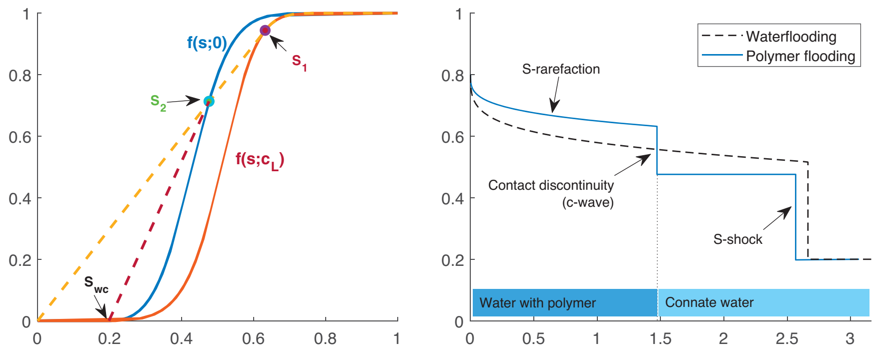


Figure 7.4 Fractional-flow analysis of polymer flooding with increased mobility ratios because of full mixing as the only physical effect included. The left figure shows the construction of the solution in (S, f)-space. The right figure shows the self-similar solution in $(x/t, S)$-space. (The figure assumes a viscosity ratio of 5 between polymer and pure water, equal viscosities of water and oil, Corey exponent 2 for water and 4 for oil, and connate and irreducible water saturation equal to 0.2.)

this is done by constructing the concave envelopes of $f(S;c_L)$, $S \in [S_1, 1]$ and of $f(S;0)$, $S \in [S_{wc}, S_2]$, which both can give a shock followed by a rarefaction. The right plot in Figure 7.4 presents the resulting self-similar solution. Adding polymer to the injected fluid reduces the penetration of water into the reservoir by lowering the saturation and propagation speed of the water front. This delays the water breakthrough and enables us to uphold a higher oil production rate. Behind the polymer front, all connate water initially present has been displaced by water containing diluted polymer in a piston-type displacement. This displacement increases the microscopic displacement efficiency and leaves less residual oil behind compared with the corresponding waterflood, when viewed at a given instance in time. We encourage you to run the script `showRiemannSolution` to see how the solution changes with the viscosity ratio between polymer and pure water, the exponents in the Corey relative permeabilities, and the connate and irreducible water saturation.

Macroscopic displacement efficiency: In addition to improving the injected water's ability to locally displace oil from the pores, polymer can have a significant conformance effect. Improving the viscosity ratio between the displacing and displaced fluids will reduce the tendency of the displacing fluid to form viscous fingers for cases with unfavorable oil–water mobility ratios. You may also observe low sweep efficiency of waterflooding for cases with mobility ratios close to unity as a result of contrasting stratification or large areal permeability variations.

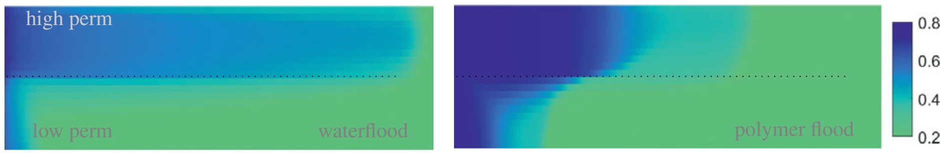


Figure 7.5 Improved vertical sweep efficiency in a layered system with contrasting stratification (upper layer: $\mathbf{K} = (200, 100)$ md, lower layer: $\mathbf{K} = (20, 10)$ md). The plots show water saturations from two high-resolution simulations of waterflooding (left) and polymer flooding (right). (Source code for this example: `runPolymerTwoLayer.m`.)

Adding polymers to the injected water will provide mobility control and counteract undesired effects such as early water breakthrough. Figure 7.5 shows how injection of polymer counteracts the formation of a thief zone in a simple stratified case.

Implementation in MRST: It follows from (7.6) that mixture viscosity is modeled as a *multiplicative* effect, $\mu_{w,\text{eff}} = \gamma_p^{\text{eff}}(c_p)\mu_w$. The state-function framework, which is a relative new addition to MRST, motivated and explained in Chapter 5, contains several hooks that enable you to add in new multiplicative modifications of existing properties by simply defining a new multiplier, implemented as a state function. The viscosity multiplier $\gamma_p^{\text{eff}}(c_p)$ is implemented in the `PVTPropertyFunctions` group as shown in Listing 7.1.

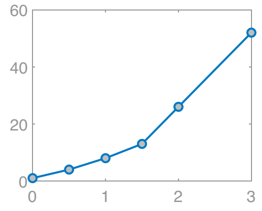
Listing 7.1 *Member functions of* `PolymerViscMultiplier`*.*

```
function gp = PolymerViscMultiplier(model, varargin)
    gp@StateFunction(model, varargin{:});
    gp = gp.dependsOn({'polymer'}, 'state'); % check mechanism
    assert(model.water);
    assert(all(isfield(model.fluid,{'cpmax','mixPar','muWMult'})));
end

function effMult = evaluateOnDomain(prop, model, state)
    fluid   = model.fluid;
    cp      = model.getProp(state, 'polymer');
    cpMax   = repmat(fluid.cpmax, numelValue(cp), 1);
    mult    = prop.evaluateFluid(model, 'muWMult', cp);
    multMax = prop.evaluateFluid(model, 'muWMult', cpMax);
    cpbar   = cp/fluid.cpmax;
    a       = multMax.^(1 - fluid.mixpar);
    b       = 1./(1 - cpbar + cpbar./a);
    effMult = b.*mult.^fluid.mixpar;
end
```

The creator function declares that the viscosity multiplier depends on polymer concentration as well as three functions/values from the fluid object. We also check that the model has a water phase, because γ_p^{eff} only affects this phase. The second function uses model-specific parameters extracted from the fluid object (i.e., the multiplier $\gamma_p(c_p)$ and the two parameters ω and $c_{p,\,\max}$) to evaluate $\gamma_p^{\text{eff}}(c_p)$ for the polymer concentration found on the `state` object. One obvious alternative is to hardcode the necessary functions directly into the fluid object, but by default, the `ad-eor` module assumes that these fluid parameters are specified in an ECLIPSE input file [5, 30], using the following keywords:

```
PLYVISC                    PLMIXPAR
0       1.0                 1.0  /
0.5     4.0
1.0     8.0                PLYMAX
1.5    13.0                 3.00  0  /
2.0    26.0
3.0    52.0 /
```



Because these keywords only contain scalar or two-column tabulated data,[1] MRST’s standard deck reader will simply read them with no further ado. However, to process the content and assign it functionally on the fluid object, we must use the `assignKEYWORD` mechanism from the `ad-props` module. That is, we need to implement three new functions called `assignPLYVISC`, `assignPLMIXPAR`, and `assignPLYMAX` and place them in the subdirectory `props` of the `ad-props` module. (You can find a brief explanation of how such assignment functions work in the MRST textbook [22, section 11.3].) The latter two are simple one-line functions that assign the scalar input value to the corresponding field in the fluid object and are omitted for brevity. The processing function for viscosity, shown in Listing 7.2, may seem somewhat obscure but follows a standard pattern for such assignments. If you have seen one, it is not difficult to implement another.

Listing 7.2 *Processing of the* `PLYVISC` *keyword.*

```
function f = assignPLYVISC(f, plyvisc, reg)
    f.muWMult = getFunction(plyvisc, reg);
end

function muWmult = getFunction(plyvisc, reg)
    muWmult = cell(1, reg.pvt);
    for i = 1:reg.pvt
        t = extendTab(plyvisc{i});
        muWmult{i} = @(c, varargin) reg.interp1d(t(:, 1), t(:, 2), c);
    end
end
```

[1] In this specific example, `PLYVISC` only supplies a single table, but in general there can be multiple tables, one per region, which each is terminated by a /.

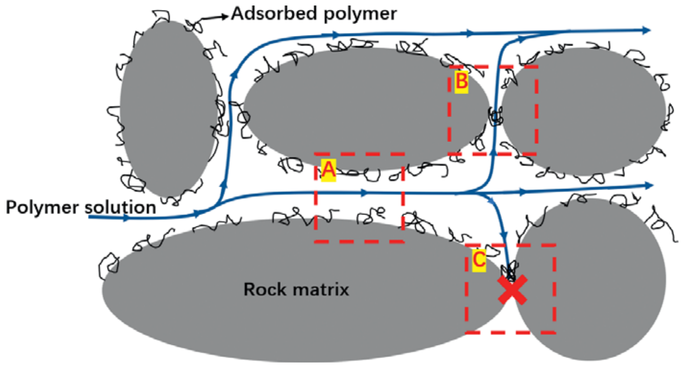


Figure 7.6 Illustration of polymer retention in porous media, inspired by [8, 36]. Polymer molecules, shown as black curly lines, are attached to the surface of the rock particles. The blue lines and arrows indicate the flow path and direction of the polymer solution. The three areas A/B/C correspond to the retention of the polymer solution after passing through the pores of different sizes and their effects on the subsequent fluidity. Zone A: Adsorbed polymer molecules causing increased resistance to flow. Zone B: Molecules mechanically trapped by bridging. Zone C: Molecules trapped by physical plugging. In addition, polymer molecules can become hydrodynamically trapped in stagnant zones.

The first function is called automatically when MRST encounters `PLYVISC` while processing all input keywords. It simply returns a function handle to the internal function `getFunction`, which sets up functions to evaluate the multiplier $\gamma_p(c_p)$ by interpolation in the values given in the `PLYVISC` input tables, one function per pressure–volume–temperature (PVT) region. The first line in the loop modifies the input data for each table so that we can use constant extrapolation for any values outside the data span, whereas the second line constructs a handle to the function that does the actual interpolation.

Polymer Adsorption

Polymer has a tendency to interact with the solid rock and either be adsorbed onto the rock surface or become entrapped between rock particles. Figure 7.6 illustrates various retention mechanisms. Such retention can have strong effect on the overall displacement efficiency of polymer flooding.

Polymer molecules can be bound to the surface of the solid rock by forces such as van der Waals forces (weak, distance-dependent forces acting between atoms and molecules) and hydrogen bonding. This phenomenon is called *adsorption*. The immediate effect on polymer flooding is the loss of the polymer: The polymer stays on the surface of the rock, thereby decreasing the effective concentration of the polymer solution. However, due to effective supplement of displacing fluid, the effect is mainly seen as a backward shift of the front of the polymer solution in the actual displacement process. The adsorption will also reduce the effective rock permeability, because polymer molecules adsorbed to the rock increase the flow resistance of the displacing fluid through the pores, as shown in Zone A of Figure 7.6.

Another form of polymer retention, *entrapment*, refers to the phenomenon that polymer molecules will tend to accumulate if they are too large to pass through a narrow pore throat. This phenomenon causes polymer consumption and the creation of inaccessible pore space for polymer in the reservoir formation. When the radius of the polymer molecule is smaller than the radius of the pore throat, polymers can become entrapped by a gradual bridging effect. As shown in Zone B of Figure 7.6, water may still be able to partially pass through an accumulation of polymer molecules that are blocked outside a throat. However, this will greatly increase the resistance to water flow and cause a permeability reduction. When the radius of the polymer molecule is larger than the radius of the pore throat, the entrapment is called *physical plugging* (see Zone C of Figure 7.6). In this case, polymer molecules and water both cannot pass through the throat.

To make the description of the various polymer effects more structured, we collectively refer to polymer loss caused by retention as *polymer adsorption*, the increase in fluid flow resistance caused by polymer retention as *permeability reduction*, and the reduction of pore space accessible to flow because of polymer molecules that remain outside of pore throats or prefer the "highways" defined by large pore throats as the *inaccessible pore space* effect. We will discuss the two last effects in the following subsections.

As in [30, 31], we assume that adsorption is instantaneous and reversible and model it through the accumulation term $(1-\phi)\rho_s c_p^a(c_p)$ in (7.3) (see also Table 7.1). For the polymer adsorption parameter c_p^a, we use an approach based on Langmuir isotherms, so that the adsorbed concentration is a function of the polymer concentration of the form

$$c_p^a(\hat{c}_p) = \frac{a_p(\hat{c}_p - c_p^a)}{1 + b_p(\hat{c}_p - c_p^a)}, \qquad \hat{c}_p = \min(c_p, c_p^{\max}), \tag{7.11}$$

where a_p, b_p are constants, $c_p^{\max}$ is the maximum polymer concentration, and $c_p - c_p^a$ is the equilibrium concentration in the rock–polymer solution system. Figure 7.7 shows a characteristic form.

We consider the reversible case and the case without desorption. In the irreversible case, the term c_p^a is replaced by $\hat{c}_p^a$, which denotes the maximum value that the adsorption concentration has reached. More precisely, if we define

$$\hat{c}^a_{p,\,\max}(t,x) = \max_{t'<t}(\hat{c}_p^a(t',x)), \tag{7.12}$$

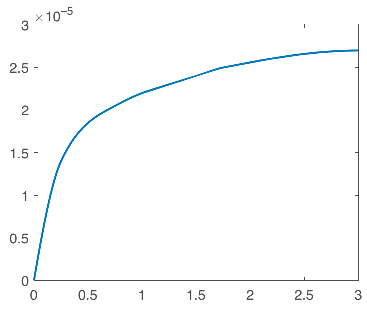


Figure 7.7 Typical shape of the adsorbed concentration $c_p^a(c_p)$ as function of polymer concentration c_p.

we can define the cumulative amount of irreversible adsorption as

$$\hat{c}_p^a(t,x) = \max(c_p^a(c(t,x)), \hat{c}_{p,\,\max}^a(t,x)). \tag{7.13}$$

Microscopic displacement efficiency: To perform the same type of fractional flow analysis as in Subsection 7.2.2, we consider the following 1D problem:

$$\partial_t S + \partial_x f(S,c) = 0, \tag{7.14a}$$
$$\partial_t\big[Sc + a(c)\big] + \partial_x\big(cf(S,c)\big) = 0, \tag{7.14b}$$

where $a(c)$ represents the adsorption term. Johansen and Winther [16] presented complete solutions for the corresponding Riemann problem for all possible combinations of left and right states under the assumption that $a(c)$ is smooth, increasing, and concave; the extension to multicomponent polymers is developed in [17]. The solution is largely similar to the one we discussed earlier, except for one important difference. The second eigenvalue is now $\xi^c = f/(S + a'(c))$ and corresponds to a *genuinely nonlinear* characteristic field, because $\nabla_{(S,c)}\xi^c \cdot \boldsymbol{\eta}^c > 0$. Any c-wave will therefore be a *shock* if $c_L > c_R$ and a *rarefaction* otherwise, and Riemann solutions will at most consist of four constant states.

Let us look at how to construct this solution for the same setup as in Figure 7.4; i.e., left state $(1, c_L)$ and right state $(S_{wc}, 0)$. Now, the contact discontinuity will be replaced by a c-shock that satisfies the so-called Rankine–Hugoniot conditions

$$f(S_R, c_R) - f(S_L, c_L) = \sigma\big(S_R - S_L\big), \tag{7.15a}$$
$$c_R f(S_R, c_R) - c_L f(S_L, c_L) = \sigma\big[S_R c_R + a(c_R) - S_L c_L - a(c_L)\big]. \tag{7.15b}$$

Here, the scalar constant σ denotes the shock speed. We can now use (7.15a) to eliminate f_L or f_R from (7.15b), which gives

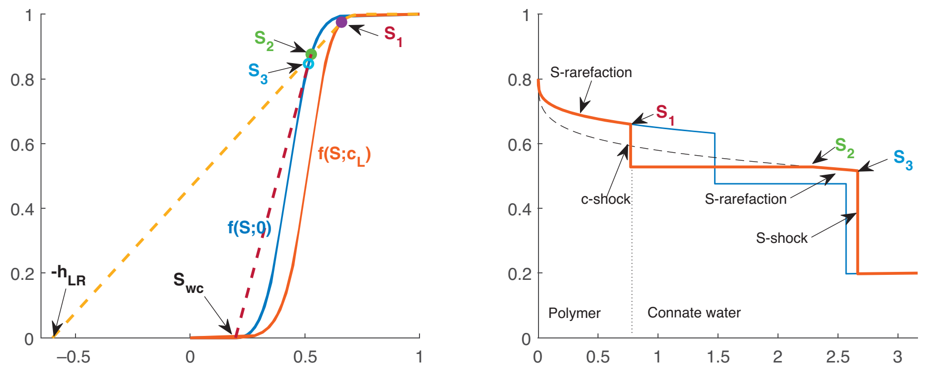


Figure 7.8 Fractional-flow analysis of polymer flooding with adsorption and increased mobility ratios because of full mixing. The left figure shows the construction of the solution in (S, f)-space. The right figure shows the self-similar solution in $(x/t, S)$-space. (The figure is generated with the `showRiemannSolution` script and assumes a viscosity ratio of 5 between polymer and pure water, Corey exponent 2 for water and 4 for oil, connate and irreducible water saturation equal to 0.2, and strong adsorption with $h_{LR} = 0.6$.)

$$\big(c_R - c_L\big) f_L = \sigma S_L \big(c_R - c_L\big) + \sigma \big(a_R - a_R\big), \qquad (7.16a)$$

$$\big(c_R - c_L\big) f_R = \sigma S_R \big(c_R - c_L\big) + \sigma \big(a_R - a_R\big). \qquad (7.16b)$$

If we solve both of these equations for σ, we can express the Rankine–Hugoniot conditions as

$$\sigma = \frac{f(S_L, c_L)}{S_L + h_{LR}} = \frac{f(S_R, c_R)}{S_R + h_{LR}}, \qquad h_{LR} = \frac{a(c_R) - a(c_L)}{c_R - c_L}. \qquad (7.17)$$

Figure 7.8 illustrates the graphical construction of the corresponding solution. Here, we have chosen the adsorption to be so strong that the leading S-wave is a composite wave consisting of a shock wave from S_{wc} to S_3, followed by a (small) rarefaction wave from S_3 to the right state, S_2, of the c-shock. (Notice, however, that S_3 is not a constant state, because the wave speeds of the shock and the rarefaction wave coincide.) Because a large fraction of the polymer is adsorbed, the polymer flooding loses much of its efficiency. Not only does the polymer front infiltrate less deeply into the reservoir, but the leading water front retains its front saturation and propagation speed from the pure waterflooding case.

Macroscopic displacement efficiency: Adsorption and mechanical trapping can also have an advantageous effect on sweep efficiency for cases with strong spatial heterogeneity. Pore blocking and local reduction in permeability will effectively alter the preferential flow paths through high-permeability regions and deviate more flow to regions with lower permeability and thereby improve sweep efficiency and

Listing 7.3 *Member functions of* `PolymerAdsorption`.

```
function gp = PolymerAdsorption(model, varargin)
    gp@StateFunction(model, varargin{:});
    gp = gp.dependsOn({'polymer', 'polymermax'}, 'state'); % check mechanism
    assert(model.water && isfield(model.fluid, 'adsInx'));
end

function ads = evaluateOnDomain(prop, model, state)
    [cp, cpmax] = model.getProps(state, 'polymer', 'polymermax');
    if model.fluid.adsInx == 2
        ce = max(cp, cpmax);
    else
        ce = cp;
    end
    ads = prop.evaluateFluid(model, 'ads', ce);
end
```

contribute to homogenize the lengths of flow paths from injector to producer. We show an example in the next subsection.

Implementation in MRST: The implementation of adsorption essentially follows the same lines as the evaluation of the viscosity multiplier, described in the previous subsection. That is, the reversible adsorption function $c_p^a(c_p)$ or the cumulative irreversible adsorption $\hat{c}_p^a(t,x)$, depending on what type of process we are modeling, is evaluated by the state function in Listing 7.3. The constructor declares that the function depends on two state variables, the current concentration and the maximum concentration observed so far over the simulation history. The second function performs the actual evaluation of $c_p^a(c_p)$ or $\hat{c}_p^a(t,x)$. The necessary input are given in two different ECLIPSE keywords [5, 30]: `PLYADS` describes the adsorption function $c_p^a(c_p)$ and comprises one or more tables given on the same form as `PLYVISC`. The corresponding `assignPLYADS` function is essentially the same as `assignPLYVISC`. In addition, `adsInx` defines whether the adsorption process is reversible or irreversible. The value of this variable is found in the fourth entry of the polymer–rock keyword (with one data record/line for each rock type):

```
PLYROCK
--IPV   RRF   dens   AI   max ads
 0.05   1.3   2600   2   0.000025 /
```

If the adsorption index (AI) equals 1, the adsorption process is reversible so that the adsorption isotherm is retracted whenever c_p decreases. It the index equals 2, the process is irreversible. The other four entries give the fraction of inaccessible pore volume for each rock type, the residual resistance factor, the rock density, and the maximum polymer adsorption used to evaluated permeability reduction. These will be discussed in more detail in the next two subsections.

Permeability Reduction

In the previous section, we saw that adsorption of polymer on the pore surface and mechanical trapping by bridging will both lead to a decrease in the absolute permeability of the rock. We use the permeability reduction factor R_k to represent this phenomenon, which is given as

$$R_k(c_p) = 1 + (RRF - 1)\, c_p^a(c_p)/c_{p,max}^a. \tag{7.18}$$

Here, $c_{p,max}^a$ is the maximum adsorbed concentration, whereas the (hysteretic) residual resistance factor $RRF \geq 1$ is defined as the ratio between water permeability measured before and after polymer flooding. Both of these quantities depend on the rock type and will thus generally vary in space. Because $R_k(c_p)$ is a translation and scaling of c_p^a, it will essentially have the same shape; see Figure 7.7.

Displacement efficiency: The microscopic effect of reduced permeability was discussed in the previous subsection. Going back to the fractional-flow analysis, we see that the only microscopic effect of reduced permeability is to improve the mobility ratio between the displacing and displaced fluid. Figure 7.9 shows

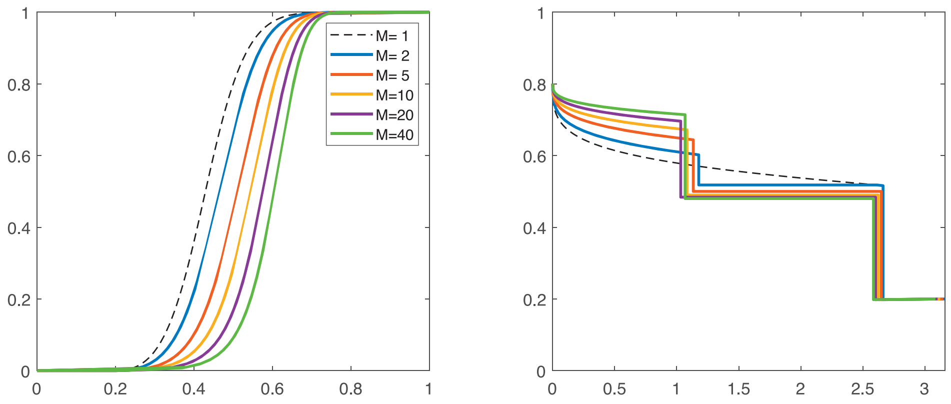


Figure 7.9 Illustration of how the viscosity ratio M between polymer and pure water affects the displacement profile. The figure assumes Corey exponent 2 for water and 4 for oil, connate and irreducible water saturation equal to 0.2, and adsorption with $h_{LR} = 0.2$; see Figure 7.8.

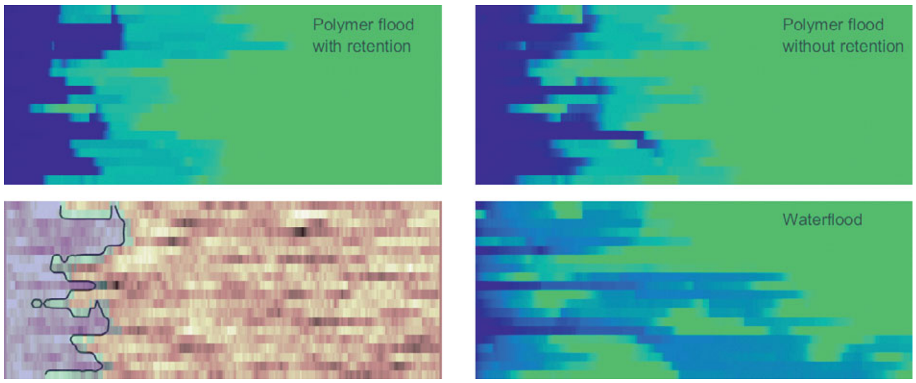


Figure 7.10 Conceptual simulations demonstrating the conformance effect of reduced permeability by polymer adsorption for an unfavorable displacement. The lower-left plot shows the permeability (on a logarithmic color scale) with the area affected by permeability reduction indicated in bluish colors inside a contour line. (Source code: `verticalSPE10.m`.)

how the displacement profile changes with the viscosity ratio between displacing fluid with and without polymer. We encourage you to experiment with the script `showMobilityEffect` to see how the figure changes with different choices of the fluid parameters.

On a macroscopic scale, the permeability reduction will have a conformance effect by diverting flow from highly conductive to less permeable regions. This is illustrated in Figure 7.10. The vertical cross section is sampled from the SPE 10 benchmark [10] and has large permeability contrasts, with five orders of magnitude variations in the lateral direction and eight orders in the vertical direction. The oil is significantly more viscous than the injected water, and this gives an unfavorable displacement in which the water has a strong tendency to form long fingers induced by heterogeneity. Injecting polymer improves mobility control, but the displacing fluid still shows a relatively strong affinity to follow high-permeability pathways. If adsorption induces a significant permeability reduction (here, we have used an exaggerated *RRF* value of 8 to demonstrate the effect), the conformance improvement is pronounced.

Implementation in MRST: Going back to (7.1c), we see that permeability reduction is another example of a multiplicative modification of a physical property, this time for the relative permeability of water. Permeability reduction is therefore implemented as a multiplier, defined in the `PolymerPermReduction` state function

```
function permRed = evaluateOnDomain(prop, model, state)
    ads     = prop.getEvaluatedDependencies(state,'PolymerAdsorption');
    fluid   = model.fluid;
    permRed = 1 + ((fluid.rrf - 1)./fluid.adsMax).*ads;
end
```

If we know that all dependencies are evaluated, we can bypass much of the dependency checking in `getProp` and use `getEvaluatedDependencies` to fetch polymer adsorption that has already been evaluated (and cached). Values on the fluid object are usually set by the `PLYROCK` keyword discussed in the previous subsection.

Inaccessible Pore Space

Many polymer flooding experiments show that polymers tend to migrate faster than other components (such as the chromatographic separation in ASP flooding). The main reason for this is that polymer molecules are so large that they can only move through larger pores (see Figure 7.6, Zone C). Because the small pores constitute the lower part of the underlying velocity spectrum, the effective Darcy velocity of a polymer mixture increases compared to that of pure water. The region of pore space not accessible to polymer is known as the inaccessible pore volume, or the dead pore space. We can introduce a scalar rock parameter, s_{ipv}, to represent the fraction of the pore volume that is inaccessible to the polymer solution for each specific rock type.

Conventional polymer models used in many commercial reservoir simulators then simply use s_{ipv} to reduce the accessible pore volume for polymer in the accumulation term, $(1 - s_{ipv})\phi b_w S_w c_p$, of (7.3). The resulting model is unfortunately not well posed and can give instabilities and predict unphysical accumulation of polymer at the water front or infinite polymer concentrations. The reason is that the model allows polymer to travel independent of the concentration and the water saturation, which can result in polymer traveling beyond the existence of water. A number of alternative models have therefore been proposed by various authors; see [14] for an overview. In practice, it may still be reasonable to use the simple, but incorrect, model, because other physical effects and the significant smearing induced by the coarse grid resolution of most sector or field models tend to mask and dampen the unphysical behavior.

Non-Newtonian Rheology of the Polymer Solution

Polymer molecules can be entangled with each other to form a 3D network structure, thereby increasing the viscosity of the aqueous solution. This can improve the water–oil mobility ratio and the sweep efficiency of the flooding fluid. On the other

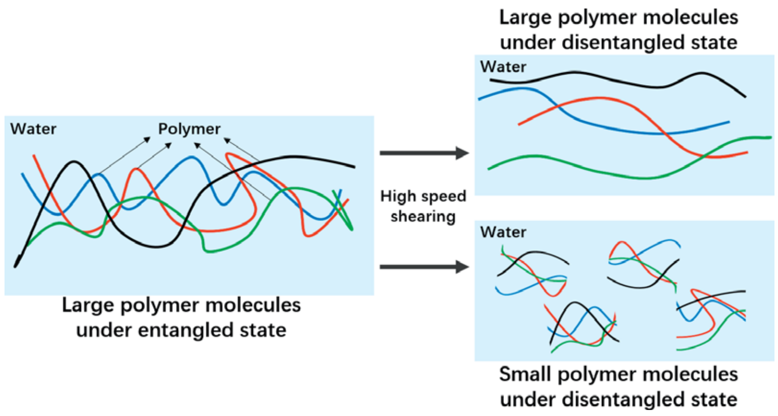

Water
Polymer
Large polymer molecules under entangled state
High speed shearing
Large polymer molecules under disentangled state
Water
Water
Small polymer molecules under disentangled state

Figure 7.11 Schematic of polymer in water under shearing, inspired by [19, 29]. The different colored lines in the figure represent polymer molecules. In static state, large polymer molecules entangle with each other, which greatly increases the viscosity of the aqueous solution (left diagram). During high-speed shearing, polymer molecules may align along the shearing direction and disentangle, which reduces their tackifying ability (upper right). If the shear rate is high enough, the polymer molecular chain may even break, which reduces the molecular weight of the polymer solution and its tackifying ability (lower right).

hand, one should expect a need to increase the injection pressure at the wellhead considerably to be able to push the highly viscous, polymeric fluid through the well pipe and into the rock formation.

In practice, the wellhead pressure is often much lower than what one should expect from the polymer's static viscosity. The reason is that diluted polymer generally has non-Newtonian rheology. When a polymer solution moves at a high speed in a certain direction, the polymer molecules will elongate and align with the flow direction or even unwind (see Figure 7.11). These phenomena cause a decrease in the viscosity of the solution. We call this behavior *shear thinning*. Of course, there are also some polymers that thicken as the shear rate increases, but such polymers are not commonly used. When the flow speed is high enough, strong shear force may also break the molecular chain and cause irreversible mechanical degradation of the polymer.

Here, we follow [31] and set the shear viscosity of the polymer to be a function of flow rate. (See also the discussion in section 7.4 of the MRST textbook [22].) First, we introduce the shear factor Z to describe the shear effect,

$$Z = \frac{\mu_{w,sh}(u_{w,sh})}{\mu_{w,\text{eff}}} = \frac{1 + (\gamma_p^{\text{eff}}(c_p) - 1)\gamma_{sh}(u_{w,sh})}{\gamma_p^{\text{eff}}(c_p)}. \qquad (7.19)$$

The multiplier γ_{sh} is a user-prescribed function of the unknown shear-modified water velocity $u_{w,sh}$ and $\mu_{w,\text{eff}}$ is the effective water viscosity (7.6) without considering the shear effect. The multiplier takes values $\gamma_{sh} > 1$ when shear thickening occurs and $\gamma_{sh} \in [0, 1)$ for shear thinning. With no shear effect ($\gamma_{sh} = 1$), we use the effective water viscosity, whereas the shear viscosity equals $\mu_{w,\text{eff}}/\gamma_p^{\text{eff}}(c_p)$ in the case of maximum shear thinning ($\gamma_{sh} = 0$). To calculate the unknown velocity $u_{w,sh}$, we follow [4] and first introduce the effective, unsheared water velocity $u_{w,0}$, calculated from (7.1c) without shear effect, and then use the relation

$$u_{w,sh} = u_{w,0}\frac{\mu_{w,\text{eff}}}{\mu_{w,sh}(u_{w,sh})} \tag{7.20}$$

combined with (7.19) to derive the following implicit equation for $u_{w,sh}$:

$$u_{w,sh}\left(1 + (\gamma_p^{\text{eff}}(c_p) - 1)\gamma_{sh}(u_{w,sh})\right) - \gamma_p^{\text{eff}}(c_p)u_{w,0} = 0. \tag{7.21}$$

A standard Newton's method is used to solve (7.21) for $u_{w,sh}$. With $u_{w,sh}$, we can calculate shear factor Z from (7.19) and calculate the shear-modified viscosities $\mu_{w,sh}$ and $\mu_{p,sh}$ as

$$\mu_{w,sh} = \mu_{w,\text{eff}}Z \quad \text{and} \quad \mu_{p,sh} = \mu_{wp}Z. \tag{7.22}$$

In practice, we update the phase fluxes directly as

$$\vec{v}_{w,sh} = \vec{v}_w/Z \quad \text{and} \quad \vec{v}_{wp,sh} = \vec{v}_{wp}/Z \tag{7.23}$$

to avoid repeated computation. Alternatively, we could also calculate shear factors for $\mu_{w,\text{eff}}$ and μ_{wp} separately, but at the moment we use a single shear factor.

Implementation in MRST: The `ad-eor` module currently implements two alternative methods to calculate the shear factor (7.19). The first method is activated with the keyword PLYSHEAR, which inputs the shear multiplier γ_{sh} in tabulated form. With this keyword, the solution procedure just described with (7.19) and (7.21) will be employed.

```
PLYSHEAR
-- Vw.      VRF
0             1
0.00002     0.9
0.00008     0.8
:           :
0.22        0.48
2.2         0.41  /
```

The keyword PLYSHLOG activates an alternative logarithm-based description to calculate the shear factor (see [30] for more details about this keyword). With (7.19) and (7.20), we have

$$u_{w,sh}Z = u_{w,0}. \tag{7.24}$$

Taking the natural logarithm of both sides of (7.24), we obtain

$$\ln u_{w,sh} + \ln Z = \ln u_{w,0}. \tag{7.25}$$

The nonlinear system associated with (7.19) thus becomes a line intersection problem, which can be more robust when Newton's method faces convergence difficulties.

The corresponding computations of shear multipliers are implemented in two private functions in the `ad-eor` module, which we do not present for brevity. More implementation details about the first method can be found in [4].

7.2.3 Physical Effects of Surfactants

Surfactants (short for surface active agents) are chemical compounds that are capable of lowering the interfacial tension between two liquids, between a gas and a liquid, or between a liquid and a solid. They are generally injected along with a displacing fluid to increase the mobility of hydrocarbons that would otherwise be residually trapped. Depending on the surfactant concentration and the type of additives in the displacing fluid, surfactant flooding can be roughly divided into conventional surfactant flooding, micellar solution flooding, and microemulsion flooding. If mixed gas is considered, foam injection is also included. Herein, we will only consider models for conventional surfactant flooding, which has relatively low cost and widespread application.

Relative Permeability Alterations

Wettability is the ability of a liquid to maintain contact with a solid surface. When an oil drop is placed on a smooth, horizontal surface covered by water, intermolecular adhesive forces between the oil and the solid will cause the drop to spread out, whereas cohesive forces inside the fluid will try to make the drop avoid the contact and stay in a spherical form. This process will reach an equilibrium when a force balance is established among the interfacial tensions[2] between oil and water, water and solid, and oil and solid; see Figure 7.12. We say that the system is preferably oil wet if the contact angle θ is less than 90° and water wet in the opposite case.

The strength of the contact between the oil droplet and the rock surface is measured by the work of adhesion, defined as the work one must do to pull an oil droplet in water with a unit contact area away from the rock surface, as illustrated in Figure 7.13. Mathematically this work is given as

$$W_a = \sigma_{ow} + \sigma_{ws} - \sigma_{os}. \tag{7.26}$$

The interfacial tension is by definition directed along the contact surface. Combined with Figure 7.12, we find that the force at point O is balanced, and the interfacial tension here satisfies the following relationship:

$$\sigma_{ws} = \sigma_{os} + \sigma_{ow} \cos\theta, \tag{7.27}$$

[2] Interfacial tension, or surface energy, is the work that must be expended to increase the interface between two different phases.

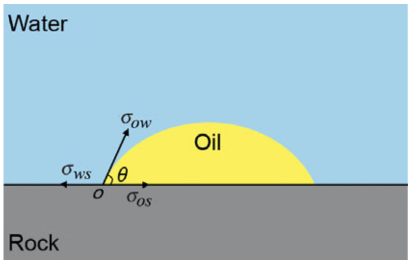


Figure 7.12 For an oil droplet attached to a solid surface covered by water, there is a force equilibrium between three interfacial tensions: σ_{ow} between oil and water, σ_{ws} between water and solid, and σ_{os} between oil and solid. This equilibrium defines the contact angle θ.

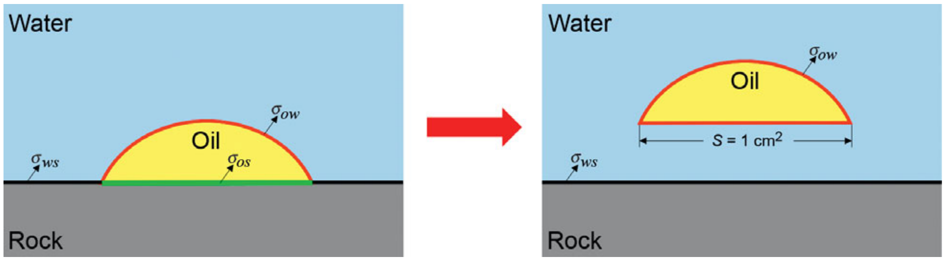


Figure 7.13 Schematic diagram of work of adhesion. Black lines denote the water–solid contact, red lines the oil–water contact, and the green line the oil–solid contact. Work of adhesion is the energy required to pull the droplet free from the surface, illustrated by the hovering drop in the right plot.

where θ is the contact angle. Inserting (7.27) into (7.26) gives the so-called Young–Dupré equation for the work of adhesion,

$$W_a = \sigma_{ow}(1 + \cos\theta). \tag{7.28}$$

To recover an oil drop attached to the solid rock, we must apply work that exceeds this work per unit oil–solid interface. Surfactants will lower σ_{ow}, and hence we should expect that the work applied by the invading fluid should be able to displace more droplets of oil.

Let us now see how this process works at reservoir conditions [11], as illustrated in Figure 7.14. Natural active substances in crude oil adsorb on the rock surface at reservoir conditions and the rock therefore shows an oleophilic state with contact angle less than 90°. If a surfactant solution invades the medium, surfactant molecules will adsorb on the contact surface between oil and water and reduce the oil–water interfacial tension. Likewise, an additional adsorption layer will form on top of the adsorption layer of natural active substances on the rock surface, thereby changing the rock wettability. These effects reduce the work of adhesion of crude

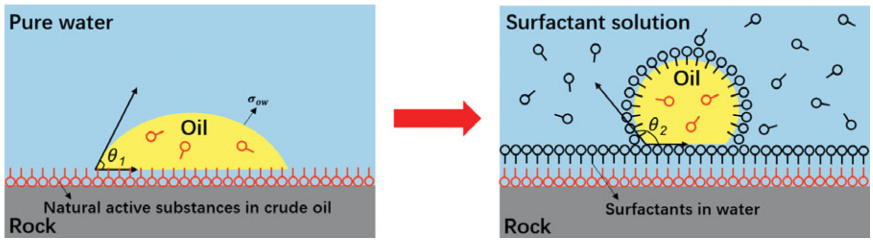


Figure 7.14 Adhesion change at reservoir conditions. The red ball-stick represents the natural active substance in crude oil, whereas the black ball-stick represents the surfactant molecules in the surfactant solution. Left picture: The rocks are oil wet due to the adsorption of natural active substances on the rock surface in the original reservoir state. Right picture: When the surfactant solution invades, the surfactant is adsorbed on the oil–water interface and on the rock surface. The oil–water interface adsorption reduces the oil–water interface tension, and adsorption on the rock surface changes the rock wettability from oil wet to water wet.

oil, making it easier to remove from the rock surface, which improves displacement efficiency.

In our simulator, we use alteration in the relative permeability between two phases to model these phenomena [31]. The surfactant will reduce the adhesion and hence reduce the flow resistance of water and oil. Changes in the oil–water interfacial tension can be orders of magnitude and are considered to be the primary effect. Changes in the wetting angle can change the rock from being primarily water wet to primarily oil wet, or vice versa, and also affect the adhesion. However, this is considered a secondary effect and is neglected in the following.

At the upscaled level, the change in relative permeability depends on the capillary number, which measures the ratio between the viscous and capillary forces

$$N_c = \frac{|K\nabla p|}{\sigma}. \tag{7.29}$$

The interfacial tension σ is assumed to be a known function of surfactant concentration, typically given in tabulated form.

We now outline the procedure for computing relative permeability as function of surfactant concentration; see Figure 7.15. In addition to σ, we assume that the relative permeabilities $k_{r\alpha}^{ns}(S_\alpha)$ and $k_{r\alpha}^{s}(S_\alpha)$, and the accompanying residual saturations $S_{\alpha r}^{ns}$ and $S_{\alpha r}^{s}$, for zero and maximal surfactant concentrations are known input. These correspond to minimal and maximal capillary numbers, N_c^{ns} and N_c^{s}.

We first compute σ for the given surfactant concentration, which gives us a new N_c value. To calculate the corresponding relative permeability curve $k_{r\alpha}(S_\alpha, N_c)$, we first use logarithmic interpolation in N_c to calculate a new residual saturation,

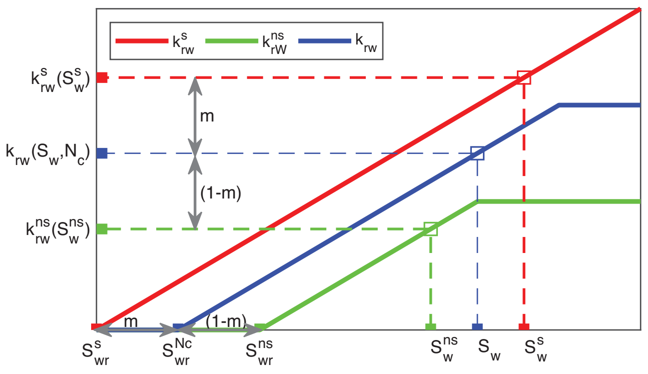


Figure 7.15 A simple example for the relative permeability calculation method. The green line represents water relative permeability with zero surfactant, the red line represents water relative permeability with maximum surfactant, and the blue line represents the final calculated water relative permeability with specific surfactant concentration.

$$S^{N_c}_{\alpha r} = m(N_c)S^s_{\alpha r} + \big[1 - m(N_c)\big]S^{ns}_{\alpha r}, \qquad m(N_c) = \frac{\log_{10} N_c - \log_{10} N^{ns}_c}{\log_{10} N^s_c - \log_{10} N^{ns}_c}. \tag{7.30}$$

We then compute two saturation values, S^s_α and S^{ns}_α, which we will use to interpolate the relative permeabilities $k^s_{r\alpha}$ and $k^{ns}_{r\alpha}$ for each S_α-value:

$$S^j_\alpha = S^j_{\alpha r} + \frac{S_\alpha - S^{N_c}_{\alpha r}}{1 - S^{N_c}_{\alpha r} - S^{N_c}_{\beta r}}\big(1 - S^j_{\alpha r} - S^j_{\beta r}\big), \qquad j = s, ns. \tag{7.31}$$

Here, α and β denote different phases. With these two new saturations, the interpolation in N_c using the coefficient $m(N_c)$ reads

$$k_{r\alpha}(S_\alpha, N_c) = m(N_c)k^s_{r\alpha}(S^s_\alpha) + \big[1 - m(N_c)\big]k^{ns}_{r\alpha}(S^{ns}_\alpha). \tag{7.32}$$

Microscopic displacement efficiency: Because the primary effect of injecting surfactant is to lower the residual saturations, it is usually applied after a waterflood or in conjunction with polymer flooding to mobilize residually trapped oil. We will nonetheless first consider the case of secondary recovery by surfactant injected into a reservoir initially filled with oil. The fractional-flow analysis is essentially the same as in Subsection 7.2.2, and we therefore do not present any details.

Figure 7.16 shows displacement profiles for two different fluid parameters. In Case 1, the profile is quite similar to the polymer flooding example presented in Figure 7.4, except that the trailing S-rarefaction now goes all the way up to 0.975 because more oil is mobilized as a result of reduced interface tension. In Case 2,

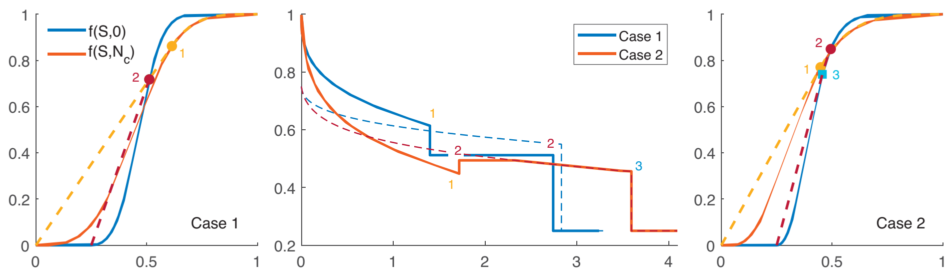


Figure 7.16 Secondary surfactant flooding into an oil-filled reservoir containing 30% connate water. Left figure: Solution in (S, f)-space with Corey exponents $n_w = 2$ and $n_o = 3$ and $\mu_o : \mu_w = 1$. Right figure: $n_w = 2$, $n_o = 2.5$, and $\mu_o : \mu_w = 0.225$. The middle plot shows the corresponding self-similar solutions in $(x/t, S)$-space. (The figure is generated with the `showRiemannSolution` script, assuming that the surfactant reduces the residual saturations of 0.25 for pure waterflooding to 0.05 for water and 0.025 for oil. No adsorption or viscosity effects are included for the surfactant.)

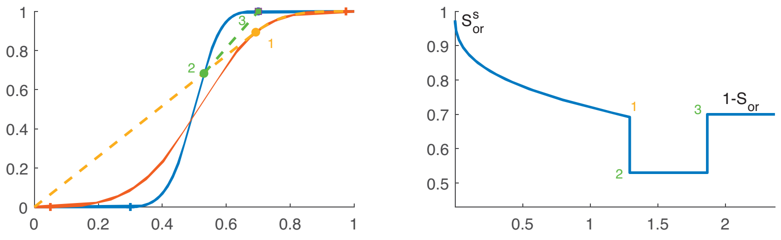


Figure 7.17 Tertiary surfactant flooding for Case 1 from Figure 7.16 but with $\mu_w : \mu_o$ set to 2 to better separate the leading leading S-shock and the trailing c-wave.

the trailing c-wave gives rise to an “oil wall” that increases the oil production momentarily before it again decays in the trailing S-rarefaction.

For surfactant injected into a waterflooded reservoir (tertiary case), the solution is almost identical, except that the solution ahead of the leading S-shock now equals the irreducible oil saturation S_{or}. Figure 7.17 shows the solution for a fluid model similar to Case 1 from Figure 7.16. To construct this solution, we modify the construction of the S-wave and now use the *lower convex* envelope of the fractional-flow function for water. (Source code: `showSurfTertiary.m`.)

Implementation in MRST: The procedure described earlier in this section for interpolating relative permeabilities is implemented in two separate state functions: `CapillaryNumber` and `SurfactantRelativePermeability`. The former computes the capillary number, whereas the latter computes the new residual

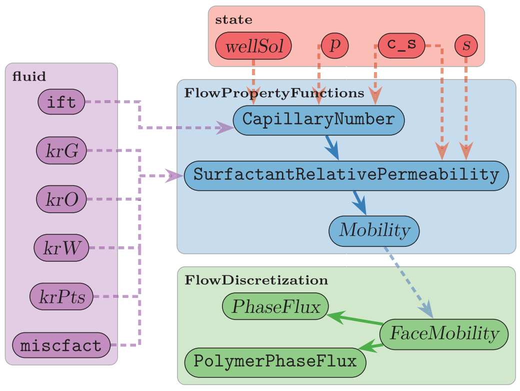


Figure 7.18 The dependency graph for the two state functions that compute relative permeabilities with alterations due to the presence of surfactant. The figure also shows how results are fed forward to routines computing interface fluxes. Slanted text means that this part is inherited from the underlying black-oil model.

saturations and performs the actual interpolation between the $k^{ns}_{r\alpha}$ and $k^{s}_{r\alpha}$ curves. Figure 7.18 shows their dependence graphs. The functions consist of many lines of code, which will not be discussed in detail.

Let us instead look at the ECLIPSE keywords you can use to specify the necessary input for the fluid object. The relative permeability curves (`krG,krO,krW`) and their endpoints (`krPts`) can be given using either of the two alternative keyword families (`SWOF, SGOF`) or (`SWFN, SGFN, SOF3`) [5, 30]. Each keyword must specify two tables for each fluid region; the first gives curves for the immiscible system without surfactant and the second gives curves for the miscible system with maximum surfactant concentration (these will be associated to two different surfactant regions). The first family is discussed in detail in the MRST textbook [22, section 11.3].

The `SURFST` keyword gives input to `fluid.ift`, which describes the water–oil surface tension as a function of surfactant concentration in the water phase. The data are given as a two-column table, in the same form as described earlier for `PLYVISC` and `SHEARVISC`, with concentration values increasing monotonically downward. Last but not least, `fluid.miscfact` contains the surfactant capillary desaturation function, which takes values in the interval [0,1] and describes the transition from immiscible to miscible conditions as a function of $\log_{10}(N_c)$. (Notice that this represents a generalization of the linear relation $m(\log_{10} N_c)$ given in (7.30).) The corresponding keyword `SURFCAPD` should specify the same number of two-column tables with $\log_{10} N_c$ versus m as the number of fluid regions.

Capillary Pressure Alterations

When two phases coexist in a porous medium, interfacial tension and difference in wettability of the fluids on the rock surface will cause fluid interfaces to be curved and be accompanied by a finite pressure difference, referred to as capillary pressure (see, e.g., subsection 8.1.3 of the MRST textbook [22] for more details). This pressure is defined as the difference between the phase pressures of the nonwetting and wetting fluids, $p_c = p_n - p_w$, because $p_n > p_w$. For a given interfacial tension σ, the capillary pressure is also given by the Young–Laplace equation

$$p_c = \sigma \left(\frac{1}{R_1} + \frac{1}{R_2} \right), \tag{7.33}$$

where R_1 and R_2 denote the principal radii of curvature for the interface.

In a three-phase black-oil model, the capillary pressures are defined for the oil–water and gas–oil subsystems,

$$p_{cow} = p_o - p_w \quad \text{and} \quad p_{cog} = p_g - p_o.$$

The two capillary functions p_{cow} and p_{cog} are usually assumed to be functions of saturation. Because surfactant influences the interfacial tension between oil and water, it follows from (7.33) that it will also affect the capillary pressure. In our model, we assume a simple scaling relationship,

$$p_{cow}(S_w, c_s) = p_{cow}(S_w) \frac{\sigma(c_s)}{\sigma_0}, \tag{7.34}$$

where $p_{cow}(S_w)$ and σ_0 denote oil–water capillary pressure and interfacial tension without surfactant, respectively.

```
SURFST
-- Cs     Surf.tens
0         0.05
0.1       0.0005
0.5       1e-5
1         1e-6
30        1e-6
100       1e-6 /
```

Implementation in MRST: The evaluation of capillary pressures is implemented in a state function called `SurfactantCapillaryPressure` (Listing 7.4), which is built on top of the existing state function from the `ad-blackoil` module. This demonstrates one of the advantages of using state functions based on inheritance. The constructor simply says that this state function depends on surfactant concentration and whatever the underlying state function from `ad-blackoil` depends on.

The evaluation function first computes capillary pressures for the black-oil system, and if this system has a capillary function for a water–oil system (`ph=='W'`), we scale it by interfacial tension according to (7.34). Tabulated values for $\sigma(c_s)$ are usually given by the `SURFST` keyword [5, 30].

Listing 7.4 *Member functions of* `SurfactantCapillaryPressure`.

```
function prop = SurfactantCapillaryPressure(model, varargin)
    prop@BlackOilCapillaryPressure(model, varargin{:});
    prop = prop.dependsOn('surfactant', 'state');
    assert(model.water && isfield(model.fluid,'ift'))
end

function pc = evaluateOnDomain(prop, model, state)
    pc = evaluateOnDomain@BlackOilCapillaryPressure(prop, model,state);
    ph = model.getPhaseNames(); iW = find(ph=='W');
    c  = model.getProps(state, 'surfactant');
    pc{iW} = pc{iW}.*model.fluid.ift(c)/model.fluid.ift(0);
end
```

Effective Viscosities

Addition of surfactant may increase the viscosity of the injected aqueous phase. As for a fully mixed polymer solution in (7.4), this is modeled by introducing a multiplier function, $\gamma_s(c_s)$,

$$\mu_w(c_s) = \gamma_s(c_s)\mu_w^0. \tag{7.35}$$

Here, μ_w^0 denotes the viscosity of pure water at reference pressure condition. Unlike polymers, however, the effect of surfactants on the viscosity of the aqueous phase is relatively minor and will in many cases be comparable to pressure effects in magnitude. Water viscosibility, the dependence of the viscosity with respect to pressure, is usually also introduced as a multiplier $\gamma(p)$,

$$\mu_w(p) = \gamma(p)\mu_w^0. \tag{7.36}$$

If we now consider a system containing both polymer and surfactant and include all effects of polymer/surfactant concentration and pressure dependence, it follows from (7.6), (7.35), and (7.36) that the effective water viscosity becomes

$$\mu_{w,\text{eff}} = \gamma(p)\,\gamma_w^{\text{eff}}(c_p)\,\gamma_s(c_s)\,\mu_w^0. \tag{7.37}$$

Microscopic displacement efficiency: In Figure 7.9, we saw that increasing the viscosity ratio between a polymer mixture and pure water will shift the fractional-flow curve of polymer to the right and increase the frontal saturation of the chemical wave. Exactly how much this increases the oil recovery will, of course, depend on the shape of the fractional-flow functions. The effects on secondary surfactant flooding are similar and are not included for brevity.

Let us instead look at the tertiary flooding case from Figure 7.17. Figure 7.19 shows fractional-flow curves and solution profile for a case with a fourfold increase

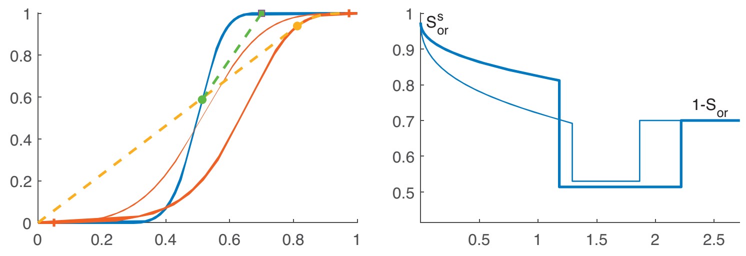


Figure 7.19 Tertiary surfactant flooding for the case from Figure 7.17 but with a fourfold increase in the water viscosity as a result of surfactant and pressure effects. (Fractional-flow curve and solution with no viscosity increase are shown as thin lines.)

in the viscosity of the surfactant solution. Here, we observe the interesting effect that the increased viscosity not only improves the displacement efficiency behind the surfactant front but also significantly accelerates the oil wall and gives an earlier increase in the oil recovery.

Implementation in MRST: The surfactant multiplier $\gamma_s(c_s)$ is implemented as a state function in the same way as the polymer multiplier from Listing 7.1:

```
function mSft = evaluateOnDomain(prop, model, state) % SurfactantViscMultiplier
    cs   = model.getProps(state,'surfactant');
    mSft = prop.evaluateFluid(model, 'muWSft', cs) / model.fluid.muWr;
end
```

The combined evaluation of all terms in (7.37) is handled generically by the underlying state-function framework. For the special case of a polymer–surfactant model, this effectively amounts to first evaluating the standard black-oil viscosities that include any pressure dependencies, checking which chemical components are part of the model, computing the corresponding multipliers, and overwriting the viscosity for the water phase. The other viscosities as kept as for the black-oil model.

Figure 7.20 shows the dependence graph for all functions involved in the computation of effective viscosities for a two-phase surfactant–polymer system. The necessary functional relationships for the fluid object are easiest to give as tabulated data through an ECLIPSE input deck. The PVTW and PVTO keywords specify viscosities at reference conditions and curves for the dependence on pressure. These are part of the standard black-oil support in MRST and are discussed in the MRST textbook [22, section 11.5]. The dependence on viscosity can be prescribed through the SURFVISC keywords; these data must be consistent with the input from PVTW. The specification and processing of SURFVISC follows the exact same pattern as PLYVISC discussed in Subsection 7.2.2.

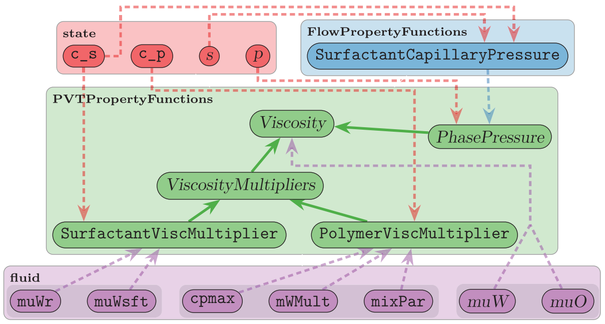


Figure 7.20 The dependency graph for the state functions implementing the viscosity relationship in (7.37) for a two-phase surfactant–polymer system. Dependencies on the fluid object are not usually included in the dependency graph generated by MRST. Slanted text indicates inherited entities.

Surfactant Adsorption

We have already discussed how surfactant molecules tend to accumulate at the oil–water interface and rock surface as a result of the coexistence of specific hydrophilic and lipophilic groups (Figure 7.14), which has the beneficial effect of reducing the oil–water interfacial tension. However, adsorption on the rock surface also reduces the effective concentration of surfactant in the solution, resulting in loss of surfactant and decreased displacement efficiency. Both the modeling and the fractional-flow analysis are virtually identical to those of polymer and details are omitted for brevity. (The interested reader can use the scripts `showRiemannSolution` and `showSurfTertiary` to study the effects on secondary and tertiary displacements.)

7.3 The Surfactant–Polymer Flooding Simulator

So far, we have introduced different physical mechanisms that affect surfactant and polymer flooding, described pertinent mathematical models, and outlined their implementation as state (or utility) functions in the `ad-eor` module. In doing so, we have also mentioned relevant ECLIPSE keywords you can use to set all of the different parameters and functional relationships.

In this section, we explain how all of the different pieces outlined so far are put together to make classes in the AD-OO simulation framework (a first version was released in MRST 2019b). We will not present all details but focus on the main points. As always with MRST, you can find full details by querying the latest version of the `ad-eor` module. We end the section by explicitly describing the necessary steps to run simulations based on an ECLIPSE input deck.

7.3.1 Design of Flexible Model Classes

The `ad-eor` module offers different model classes that implement chemical flooding simulators. In a previous paper [4], we discussed the implementation of a fully implicit, three-phase polymer simulator. (This is built in the same way as the `ThreePhaseBlackOilModel` model from the original AD-OO framework discussed in chapter 12 of the MRST textbook [22].) More recently, we developed a similar model for surfactant–polymer flooding. The only conceptual difference is that various fluid properties, which in the polymer model were evaluated by functions held by the fluid object (see [4]), now are evaluated using state functions.

In this model, the evaluation of residual flow equations is essentially located to a single function, which makes it simpler to follow the logic of the computation. The disadvantage is that if you want to simulate, say, a two-phase oil–water–surfactant problem, you would either use the full model and always solve for five unknowns per cell or develop a specialized class (`OilWaterSurfactantModel`) that uses only three unknowns per cell. With three phases and five components, there will be many special cases, and if we later wish to expand the module by adding additional chemicals like alkali or solvents, we quickly end up with a combinatorial explosion of model classes.

Component (or generic) models, discussed in Chapter 5, were introduced to overcome this limitation. They offer finer granularity and eliminate the need for specialized classes for different combinations of phases and components. Instead, combinations can be specified at runtime to form models having only the necessary unknowns. The underlying framework is designed so that it is easy to specify alternative spatial or temporal discretizations; e.g., to replace the standard fully implicit formulation by an adaptive implicit method, replace the two-point flux approximation scheme by a multipoint discretization, or replace the single-point upwind mobility scheme by a high-resolution method; see Chapter 5.

7.3.2 The Full Three-Phase, Five-Component Model

We start by briefly describing the full model having three phases and all five components. The model is derived and inherits a lot of functionality from the `ThreePhaseBlackOilModel`. We therefore assume that you are already familiar with this model and the underlying `ReservoirModel` and `PhysicalModel`; all three are described in great detail in chapter 12 of the MRST textbook [22]. The model has five properties that decide which components are active and what type of shear model to use, if any. These are set in the constructor based on an ECLIPSE-type input deck:

```
classdef ThreePhaseSurfactantPolymerModel < ThreePhaseBlackOilModel
   properties
      polymer
      surfactant
      usingShear, usingShearLog, usingShearLogshrate
   end
```

Variables and updating: The model must define necessary variables: polymer and surfactant concentrations c_p and c_s are primary variables, whereas maximum concentrations $c_{p,\max}$ and $c_{s,\max}$ are used to calculate adsorption:

```
function [fn, index] = getVariableField(model, name, varargin)
   switch(lower(name))
      case {'polymer', 'polymermax'}
         index = nnz(strcmpi(model.getComponentNames(), 'polymer'));
         if strcmpi(name, 'polymer')
             fn = 'cp';
         else  fn = 'cpmax'; end
        :
      otherwise
         [fn, index] = getVariableField@ThreePhaseBlackOilModel...
                    (model, name, varargin{:});
   end
```

Here, `fn` and `index` are the field name and column index that will extract the correct variable from the state struct. The last two lines access all variables defined in the black-oil model. The function also defines two extra variables, `qWPoly` and `qWSft`, for the surface rate of polymer and surfactant in and out of wells.

We also need to implement two update functions. The first, `updateState`, is run after each iteration step to update the reservoir state and check that it is within physical limits; that is, we let the underlying black-oil model check the states defined in this model and then check that $c_{p/s} \in [0, c_{p/s,\max}]$; see [4] for details. The second function, `updateAfterConvergence`, is almost identical and is run after the nonlinear equation has converged to track the maximum concentrations $c_{p/s,\max}$. The class also implements an enhanced `validateState` function, which checks that concentration variables are present in the state and have the correct dimensions.

State functions: By and large, the member functions are straightforward extensions to those discussed in [4] and [22, chapter 12]. The new part is how to incorporate the state functions for computing fluid properties described in Subsections 7.2.2 and 7.2.3. These are not created by the class constructor, because the properties of the model can change after the class is instantiated, but are instead instantiated at the start of the simulation by the `validateModel` function, which in turn calls the `setupStateFunctionGroupings` function. We go through some parts of this to give you the general idea. We start by calling upon the underlying black-oil model to create the necessary state-function groupings:

```
function model = setupStateFunctionGroupings(model, varargin)

    model=setupStateFunctionGroupings@ThreePhaseBlackOilModel(model,varargin{:});

    fp = model.FlowPropertyFunctions;
    pp = model.PVTPropertyFunctions;
    fd = model.FlowDiscretization;

    pvtreg  = pp.getRegionPVT(model);
    satreg  = fp.getRegionSaturation(model);
    surfreg = fp.getRegionSurfactant(model);
```

The last three lines extract region identifiers that will be used to model spatial dependencies in PVT properties, capillary pressures, and relative permeabilities. Saturation and PVT regions are used as in a standard black-oil model from the `rock` struct stored by the model, if available. For relative permeabilities we use `satreg` to get the value of $k_{r\alpha}^{ns}(S_\alpha)$ and use `surfreg` to get the value of $k_{r\alpha}^{s}(S_\alpha)$, whereas `pvtreg` is used when computing the viscosities.

Let us start by state functions modeling adsorption. Provided that polymer is present, the corresponding function is added to the flow property group as follows (setup for surfactant adsorption is analogous):

```
fp = fp.setStateFunction('PolymerAdsorption', PolymerAdsorption(model, satreg));
```

The state functions necessary to compute viscosities for the three physical phases are set up in the PVT property group:

```
pp = pp.setStateFunction('Viscosity', EORViscosity(model, pvtreg));
pp = pp.setStateFunction('BaseViscosity', BlackOilViscosity(model));

viscmult = PhaseMultipliers(model);
viscmult.label = 'M_\mu';
```

The evaluation is done by the general class `EORViscosity`. The function first evaluates a base viscosity, which here is the standard, pressure-dependent viscosity from the black-oil model – i.e., $\mu_\alpha(p) = \gamma(p)\mu_\alpha^0$ – and then applies any multipliers defined by chemical effects in the EOR model. The `PhaseMultipliers` class is a simple container that collects state functions defining multipliers. These are added as follows, conditional on the presence of polymer and surfactant components:

```
if model.polymer
    peffmult = 'PolymerEffViscMult';
    pp = pp.setStateFunction(peffmult, PolymerEffViscMult(model, pvtreg));
    viscmult = viscmult.addMultiplier(model, peffmult, 'W');
end
if model.surfactant
    smult = 'SurfactantViscMultiplier';
    pp = pp.setStateFunction(smult, SurfactantViscMultiplier(model, pvtreg));
    viscmult = viscmult.addMultiplier(model, smult, 'W');
end
pp = pp.setStateFunction('ViscosityMultipliers', viscmult);
```

Notice here the last argument to `addMultiplier`, which specifies that multipliers will only be applied to the water phase.

State functions for relative permeability, with reduced permeability as multiplier, and for the polymer pseudophase are defined in the much the same way. We must also set up state functions for evaluating interface fluxes for the polymer pseudophase:

```
fd = fd.setStateFunction('PolymerPhaseFlux' , PolymerPhaseFlux(model));
fd = fd.setStateFunction('FaceConcentration', FaceConcentration(model));
```

Residual equations: With state functions set up, we can compute the discretized flow equations. This is done in `equationsThreePhaseSurfactant Polymer`, which follows the same general pattern as in [4, 22]. The function is admittedly complicated because of wells, source terms, boundary conditions, shear-thinning effects (see discussion in [4]), and the possibility of formulating reverse equations for computing adjoints. Going through all these details is outside the scope of this introductory chapter. Instead, we briefly explain how state functions and extended functionality from the new AD backends discussed in Chapter 6 are used to evaluated the basic flow equation for the polymer component.

We start by extracting the necessary physical variables, selecting and setting up primary unknowns as AD variables, and evaluating basic fluid properties:

```
[p, sW, sG, rs, rv, cp, cpmax] = model.getProps(state, ...
            'pressure', 'water', 'gas', 'rs', 'rv', 'polymer', 'polymermax');

[p, sW, x, cp] = model.AutoDiffBackend.initVariablesAD(p, sW, x, cp);

[b, pv]           = model.getProps(state, 'ShrinkageFactors', 'PoreVolume');
[phFlux, flags]   = model.getProps(state, 'PhaseFlux',  'PhaseUpwindFlag');
[pres, mob, rho]  = model.getProps(state, 'PhasePressures', 'Mobility', 'Density');
vP                = model.getProps(state, 'PolymerPhaseFlux');
muWeffMult        = model.getProp(state, 'PolymerEffViscMult');
adsp              = model.getProp(state, 'PolymerAdsorption');
[bW, bO, bG]      = deal(b{:});
[vW, vO, vG]      = deal(phFlux{:});
                  :
```

For brevity, we have skipped code for selecting the primary variable `x` representing free or dissolved gas, code computing values at the previous time step (`adsp0`, `cp0`, `p0`, `sW0`), and so on. We then compute the accumulation and flux terms:

```
plyAcc  = ((1-fluid.dps)/dt).*(pv.*bW.*sW.*cp - pv0.*fluid.bW(p0).*sW0.*cp0) ...
           + (s.pv/dt).*fluid.rhoR.*((1-poro)./poro).*(adsp - adsp0);
plyFlux = s.faceUpstr(upcw, bW).*vP;
```

These are collected along with similar terms from the other equations and passed on to the AD backend, which combines the accumulation and the divergence of the flux term as efficiently as possible:

```
eqs    = {wAcc,  oAcc,  gAcc,  plyAcc,  sftAcc};
fluxes = {wFlux, oFlux, gFlux, plyFlux, sftFlux};

for i = 1:numel(fluxes)
    eqs{i} = s.AccDiv(eqs{i}, fluxes{i});
end
```

One word of caution at the end: During simulation, states with zero or almost zero water saturation and nonzero polymer/surfactant concentrations may lead to bad condition numbers for the corresponding Jacobian matrices. We therefore post-process the Jacobians and reinstate original values in cells with problematic values.

7.3.3 A Generic Surfactant–Polymer Model

So-called generic models offer a systematic and relatively simple approach to formulate multiphase, multicomponent systems. The basic idea is to build the system at runtime as a collection of components that each has a well-known PVT behavior.

Examples of such components are *immiscible components* that only exist in one phase that is made up entirely of the specific component; *oil or gas components* of the type found in black-oil models that can exist in all hydrocarbon phases; or *concentration components* that are transported in another phase without changing the mass and density of the phase but affecting other properties like viscosity. Each component type is implemented as a class that offers routines for computing phase composition, component density and mass, component mobility, etc.; see Chapter 5.

Surfactant and polymer models can generally contain components of all of these types. Water is an immiscible component, polymer and surfactants are concentration components, whereas oil is an "oil component" in a system with live oil or wet gas and an immiscible component in a dead-oil system without gas or with dry gas.

Polymer/surfactant components: The `ConcentrationComponent` neither implements the density (PVT behavior) of the carrying aqueous phase by itself nor implements how the component concentration affects the component mobility. To get the correct behavior, we must therefore implement two derived classes. Let us consider surfactant as an example; the class for polymer is almost identical but with the obvious modifications due to different flow physics. The class has no properties and the constructor only states that the surfactant is carried by the aqueous phase (which by convention is the first phase) and then registers functional dependencies:

```
classdef SurfactantComponent < ConcentrationComponent
    properties
    end
    methods
        function c = SurfactantComponent()
            c@ConcentrationComponent('surfactant', 1);
            c = c.functionDependsOn('getComponentDensity', 'surfactant');
            c = c.functionDependsOn('getComponentDensity', 'ShrinkageFactors',...
                                    'PVTPropertyFunctions');
            :
        end
```

The component density (relative to the constant surface density of the aqueous phase) is given by the usual shrinkage factor times the surfactant density:

```
function c = getComponentDensity(component, model, state, varargin)
   [cs, b] = model.getProps(state, 'surfactant', 'ShrinkageFactors');
   c       = cell(1, numel(b));
   c{1}    = cs.*b{1};
end
```

Surfactant mass is found in two different forms: dissolved in the mobile aqueous phase and adsorbed in the rock:

```
function c = getComponentMass(component, model, state, varargin)
     [cs, pv, b, sw, ads] = model.getProp(state, 'surfactant', 'PoreVolume', ...
                                 'ShrinkageFactors', 'sW', 'SurfactantAdsorption');
     c        = cell(1, model.getNumberOfPhases);
     mobile   = sw.*cs.*b{1};
     poro     = model.rock.poro;
     adsorbed = model.fluid.rhoR .* ((1-poro)./poro) .* ads;
     c{1}     = pv.*(adsorbed + mobile);
end
```

Finally, the mobility of the surfactant component is the mobility of the water phase times the component density:

```
function cmob = getComponentMobility(component, model, state, varargin)
     [mob, b, c] = model.getProps(state, 'Mobility', 'ShrinkageFactors', 'surfactant');
     cmob    = cell(1, model.getNumberOfPhases);
     cmob{1} = c.*b{1}.*mob{1};
end
```

Because the polymer component only flows with the polymer pseudophase, its mobility must be multiplied by $\gamma_p^{\text{eff}}(c_p)/\gamma_p^{wp}(c_p)$ – i.e., the ratio between the polymer effective viscosity multiplier and the polymer multiplier.

The `GenericSurfactantPolymerModel` class: Any generic model must be derived from the dummy class `ExtendedReservoirModel` to signify that it is generic. Here, we also derive the new class from `ThreePhaseSurfactantPolymerModel` so that we can inherit properties, the constructor, setup of state functions, and necessary update functions. As with state functions, the component functions are initiated during runtime by the `validateModel` member function and not by the constructor. The following is a slightly edited excerpt of this member function:

```
if isempty(model.Components)
    nph = model.getNumberOfPhases();
    names = model.getPhaseNames();
    model.Components = cell(1, nph + model.polymer + model.surfactant);
    for ph = 1:nph
        switch names(ph)
            case 'W', c = ImmiscibleComponent('water', ph);
            case 'O'
                if disgas || vapoil, c = OilComponent('oil', ph, disgas, vapoil);
                else,                c = ImmiscibleComponent('oil', ph); end
              :
        end
        model.Components{ph} = c;
    end
    index = nph;
    if model.polymer
        index = index + 1;
        model.Components{index} = PolymerComponent();
    end
    :
end
model = validateModel@ThreePhaseSurfactantPolymerModel(model, varargin{:});
```

The function first loops through all phases present to construct the corresponding components. Then, we construct any concentration components. Finally, we call upon any underlying classes to perform additional validations, which in this case are carried out in the `ReservoirModel` and `PhysicalModel` classes.

The residual equations are evaluated by the `getModelEquations` function, which, in the case with no wells and boundary conditions, only consists of a few lines:

```
function [eqs, names, types, state] = ...
              getModelEquations(model, state0, state, dt, drivingForces)
[eqs, flux, names, types] = ...
   model.FlowDiscretization.componentConservationEquations(model,state,state0,dt);
for i = 1:numel(eqs)
   eqs{i} = model.operators.AccDiv(eqs{i}, flux{i});
end
```

The first statement calls a member function of the flux-discretization group, which essentially loops through all phases and components to evaluate and accumulate the component phase masses and the component phase fluxes. These fluxes are in turn evaluated from the component mobilities of the type we just saw explained for the `SurfactantComponent` class.

Figure 7.21 shows the dependency between all state variables and the various state functions used to evaluate the component fluxes and masses. The diagram for fluxes includes 28 different state functions and eight different state variables. The diagrams are very complex, but if you look in detail, you will probably recognize many of the state functions we discussed in Subsections 7.2.2 and 7.2.3.

The point of including this figure, however, is not to discuss details in the dependencies but rather highlight the fine granularity of the generic implementation. This offers a lot of flexibility but also makes it more challenging to figure out exactly where in the code each model or formula is evaluated.

This ends our discussion of the general setup. If you want to learn more technical details about how to include wells and surface facilities or source terms and boundary conditions or how to account for non-Newtonian fluid effects, you must look in

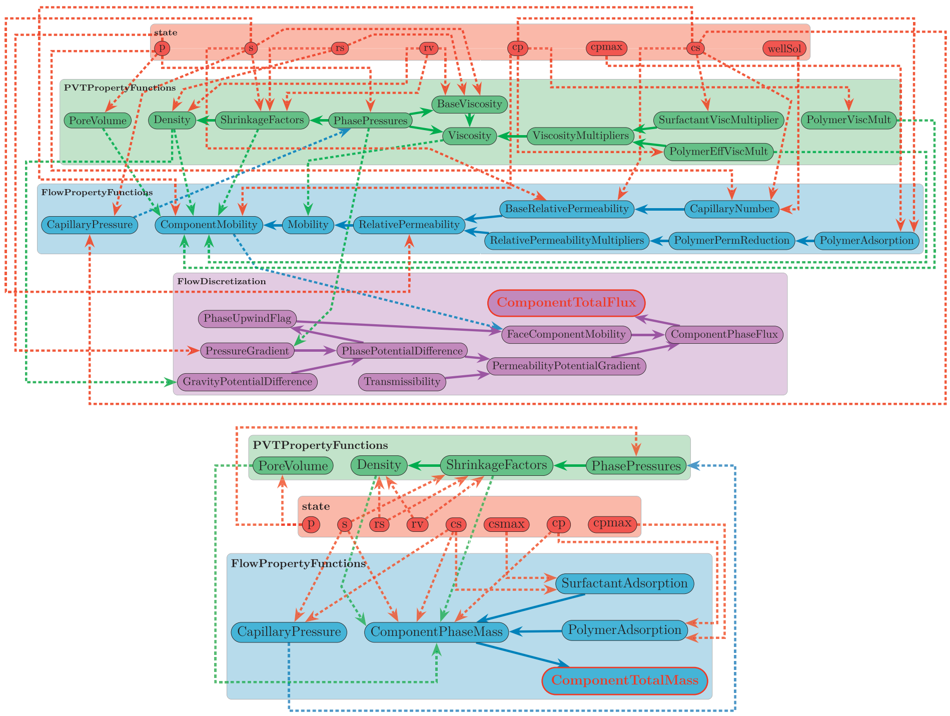


Figure 7.21 Dependency diagrams for the calculation of total component flux (top) and total component mass (bottom) for the generic surfactant–polymer model.

the code itself. It is admittedly complicated, but all of the necessary details are there. If you want to explore the code, your experience is that the powerful functionality for displaying state-function diagrams and groupings is an indispensable tool to quickly develop an understanding of how things are tied together.

7.3.4 Running the Simulator from an Input Deck

The first thing you need to do to run an EOR simulation with the `ad-eor` module is to load the necessary modules:

```
mrstModule add ad-core ad-blackoil ad-eor ad-props deckformat mrst-gui
```

The first three modules contain the necessary simulator classes, the next two supply routines for reading ECLIPSE input decks [30] and building fluid objects, and the last module contains useful plotting tools.

Here, we simply assume that you have made an appropriate input function according to the specifications in [30]. We have already discussed most of the keywords necessary to specify polymer and surfactant behavior in Section 7.2. Likewise, section 11.5 of the MRST textbook [22] gives a brief and MRST-centric introduction to how you can specify properties of the underlying black-oil model. For a more comprehensive discussion of the input format, you should consult the documentation of ECLIPSE [30], the lecture notes of Pettersen [26], or the documentation of OPM Flow [5]; we recommend the latter two, because they are freely available online.

Given an input deck, you can use a function that was recently introduced in the AD-OO framework to read the deck and construct an initial state, a model, a simulation schedule, and a nonlinear solver class:

```
fn = fullfile('path','to','datafile', 'filename');
[state0, model, schedule, nonlinear] = initEclipseProblemAD(fn);
```

The init function uses `selectModelFromDeck` to pick the appropriate model, and if you write a new model class, you must make sure that the logic of this latter function is able to select your model. Likewise, the init function uses `getNonlinearSolver` and `selectLinearSolverAD` to set up reasonable defaults for the nonlinear solver, the linear solver, and the timestep selector classes. You can find more details about this functionality in Chapter 6. (You can, of course, also perform the setup manually if you want more direct control.)

We can now simulate the model and plot the well responses (bottom-hole pressures, reservoir and surface rates, water cut, etc.):

```
[wellSols, states, report] = ...
   simulateScheduleAD(state0, model, schedule, 'NonLinearSolver', nonlinear);
plotWellSols(wellSols,cumsum(schedule.step.val))
```

By default, the simulator will report progress by printout, but you can also use the `getPlotAfterStep` function to set up a graphical user interface that provides simple means for computational steering; see section 12.1 of the MRST textbook [22]. Chapter 6 describes more advanced functionality for setting up so-called packed simulation problems, which provide a means for automatic restarts of aborted simulations and storage and retrieval of simulation results from disk.

7.4 Numerical Examples

In this section, we go through a few examples to demonstrate how you can use the models and solvers from the `ad-eor` module to set up simulations. All examples come with complete source code and ECLIPSE input files, all found in the `book-ii` example directory of the `ad-eor` module. We have already discussed how to run the `ad-eor` simulators from an input deck; hence, we will not discuss code details in the following.

However, before you continue, we would like to add a few words of caution. The examples presented in this section have been designed to highlight particular features of chemical flooding. In most of the examples, we have therefore modified fluid and reservoir parameters to visually highlight specific effects of polymer and surfactant on enhanced recovery or to demonstrate effects in the numerical solvers you should be aware of. In particular, we have scaled the concentration of chemical agents (compared to actual reservoir development parameters). When reading the examples, you should therefore look at trends and features and not focus on specific numbers such as fluid rates, pressure values, temporal and spatial scales, and so on. Likewise, parameters found in the input files should not be applied uncritically to model real-life scenarios.

7.4.1 Numerical Resolution of Trailing Waves

We have seen in previous sections that trailing waves in a displacement profile in many cases are a main cause of the EOR effect observed for polymer and surfactant flooding. (As an example, you can think of the c-waves in Figures 7.4 and 7.8.) To correctly predict potential improvements in microscopic and macroscopic displacement efficiency it is therefore important to resolve these waves as accurately as possible. This may prove quite challenging if the trailing wave is linear or weakly nonlinear, which means that the wave has no or very little self sharpening to counteract the strong numerical smearing seen in the standard first-order discretization methods used in reservoir simulation. Reducing this smearing is one of the main motivations for developing high-resolution schemes such as the weighted essentially nonoscillatory (WENO) schemes outlined briefly in Chapter 5 and higher-order discontinuous Galerkin methods discussed in Chapter 3.

Conceptual analysis: To illustrate this point, let us first look at a conceptual problem describing the transport of a scalar quantity $u \in [0,1]$ subject to a convex flux function $h(u)$, for which $h(0) = 0$ and $h(1) = 1$. A discontinuity $\hat{u}(x,t)$ with left state $u_L = 1$ and right state $u_R = 0$ will then propagate with constant unit speed. In Lagrangian coordinates (i.e., coordinates that follow the discontinuity), we can write the effective quasilinear equation as

$$u_t + \big[h'(u) - 1\big]u_x = 0 \tag{7.38}$$

and let the stationary discontinuity $\hat{u}$ be centered at the origin, so that $\hat{u}(x,t)$ equals one for $x < 0$ and zero for $x > 0$. Consider now a smeared version $\tilde{u}(x,t)$ of the stationary solution $u(x,t)$. If we assume $h''(u) > 0$, there exists a u^* such

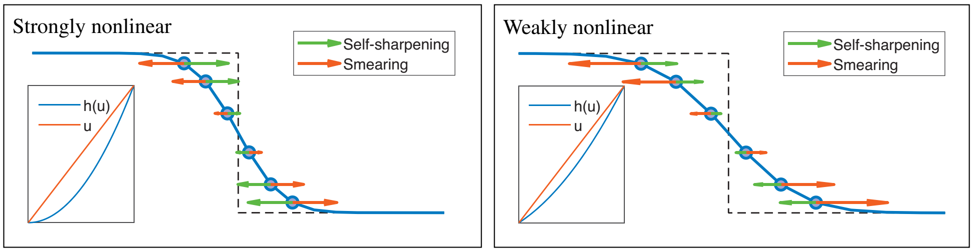


Figure 7.22 Conceptual illustration of the balance between self-sharpening and numerical smearing in the transport equation (7.38) for a stationary shock (dashed line). The circles denote constant u-values and the arrows indicate the effects that try to move these values in opposite directions.

that $h'(u) < 1$ for $u \in [0, u^*)$ and $h'(u) > 1$ for $u \in (u^*, 1]$. Using the method of characteristics, we then have that any value $\tilde{u}(x,t) < u^*$, found to the right of the discontinuity, will propagate leftward toward $x = 0$. Likewise, any value $\tilde{u}(x,t) > u^*$, found to the left of the discontinuity, will propagate rightward toward $x = 0$. As a result, the smeared profile will eventually sharpen up again into the stationary discontinuity $\hat{u}(x)$. This is what we refer to as *self-sharpening*; see Figure 7.22 for a conceptual illustration.

When simulating this transport with a finite-volume or finite-difference scheme, we effectively introduce some numerical smearing, so that instead of solving (7.38), we solve $u_t + [h'(u) - 1]u_x = \varepsilon u_{xx}$, where the magnitude of the diffusion coefficient ε depends on the specific scheme, the grid resolution, and the timestep. The smearing can be thought of as a “force” that pushes the states $\tilde{u}(x,t) < u^*$ rightward and the states $\tilde{u}(x,t) > u^*$ leftward. To what degree the profile stays sharp or continues to be smeared out depends on the magnitude of $|h'(u) - 1|$ compared with the smearing. In other words, the more *nonlinear* $h(u)$ is, the less the discontinuous profile will be affected by numerical smearing. We also see that in the case of $h(u) = u$, which corresponds to a contact discontinuity, there is *no* self-sharpening to counteract the smearing, and in Lagrangian coordinates this discontinuity will effectively evolve according to a heat equation $u_t = \varepsilon u_{xx}$.

1D polymer flooding: Let us now use this insight to study the resolution of the polymer front in a 1D pure polymer flooding. For this, we use a setup consisting of a $50 \times 3 \times 3$ m^3 horizontal reservoir with permeability 100 md and porosity 0.2, discretized by a 1D grid. The reservoir is initially filled with oil and 20% connate water. We continuously inject a polymer solution with concentration 3.0 kg/m^3 at a constant rate along the left edge and produce fluids at a constant rate from the right edge. We neglect all polymer effects except for viscosity enhancement and consider varying degrees of mixing, from fully mixed to no mixing. You can find the source

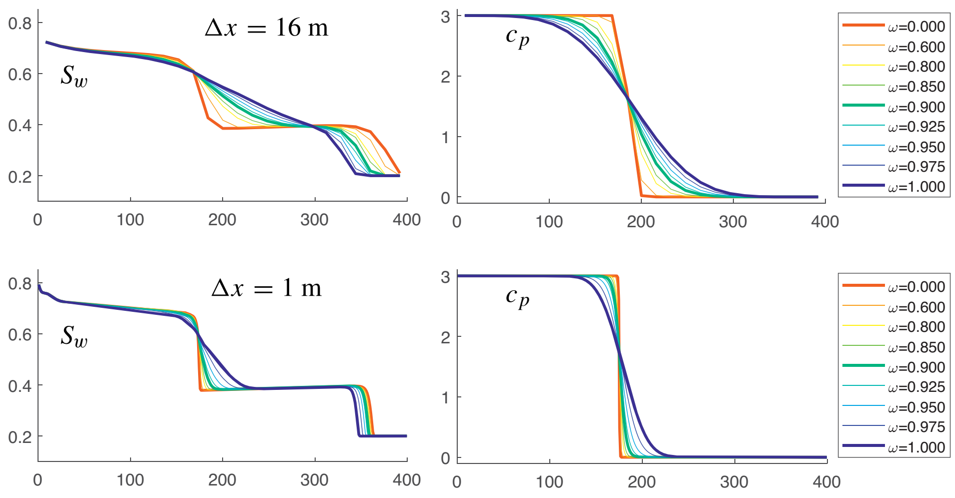


Figure 7.23 Polymer injection for various degrees of mixing, from no mixing ($\omega = 0$ in (7.6)) to full mixing ($\omega = 1$), simulated with two different grid resolutions. The fluid model only accounts for changes in effective viscosities resulting from the polymer.

code in the script `runPolyMixParam.m` and the setup in two ECLIPSE input files called `MixPar25.DATA` and `MixPar400.DATA`, in which 25 and 400 refer to the number of cells in the grid.

Figure 7.23 reports saturation and concentration profiles for a variety of mixing parameters ω in (7.6), simulated on the two different grids. The analytic solution consists of a leading S-shock, followed by a discontinuous c-wave and an S-rarefaction. On the coarsest grid, it is difficult to distinguish the c-wave from the trailing S-rarefaction in the saturation profile for values of ω close to one, because the concentration front is smeared out over large distances. A grid resolution of 16 m may be possible for onshore EOR applications, where well distances are not too large, but for offshore applications this would be considered a *high-resolution grid*. Even on the much finer grid (with $\Delta x = 1$ m), we see that the c-wave is smeared much more than the leading shock. You may recall from Subsection 7.2.2 that the c-wave is a contact discontinuity for the special case of $\omega = 1$. For $\omega < 1$, the polymer equation (7.7b) is replaced by

$$\partial_t\big(Sc\big) + \partial_x\big[cm(c)\,f(S,c)\big] = 0, \tag{7.39}$$

which means that the contact discontinuity turns into a shock. Here, $cm(c) \leq c$ is a convex function of a form similar to $h(u)$ from our conceptual analysis. The function $cm(c)$ becomes more convex with decaying values of ω. This increases

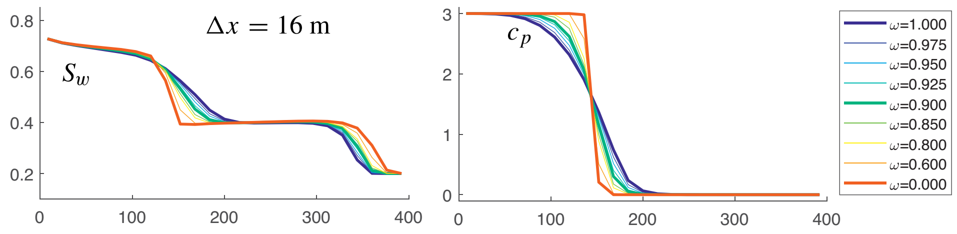


Figure 7.24 Polymer injection for varying degrees of mixing, from no mixing ($\omega = 0$ in (7.6)) to full mixing ($\omega = 1$). In addition to changes in effective viscosities, we account for polymer adsorption and permeability reduction but not inaccessible pore volume.

the self-sharpening of the c-wave, and for $\omega \lesssim 0.6$, you can see that the c-wave is resolved approximately as accurately as the leading S-shock.

If the fluid model also accounts for polymer retention, the c-wave is no longer linear or weakly nonlinear, and the need for high grid resolution is not as imminent. In Figure 7.24, we have repeated the simulation from Figure 7.23 with the effects of adsorption and permeability reduction included on the coarsest grid. The degree of mixing still influences how accurately we resolve the trailing c-discontinuity, but the effect is not as evident as for the case without polymer retention.

Improved discretizations: To improve the resolution we will use the implicit, second-order WENO scheme developed in [23, 24], which is explained briefly in Chapter 5. Instead of computing fluxes from cell-averaged quantities, one uses a *nonlinear* procedure to reconstruct a piecewise linear approximation to the mobilities inside each cell and then uses reconstructed point values on opposite sides of each cell interface to evaluate the numerical fluxes.

To replace the standard single-point upwind (SPU) mobility scheme by a WENO scheme, we simply select a different set of state functions in the generic model class. These are set up by the helper function

```
model = setWENODiscretization(model);
```

To further reduce numerical smearing, we can also use an adaptive-implicit temporal discretization [37], which uses an estimate of the Courant number to determine whether a mobility is evaluated explicitly or implicitly. AIM is set up as follows:

```
model = setTimeDiscretization(model, 'aim', 'saturationCFL', 0.75);
```

The estimated Courant number is not necessarily accurate, and to be on the safe side we use a conservative limit of 0.75; this is to prevent the method from introducing

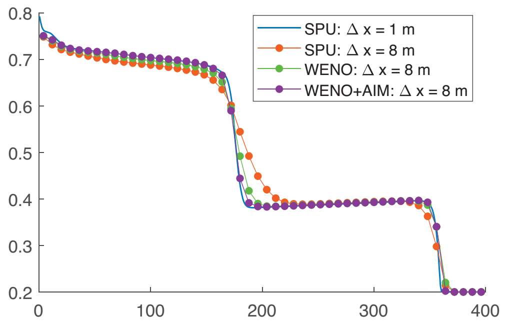


Figure 7.25 Polymer injection with mixing coefficient $\omega = 0.85$ simulated on a fine grid and with the standard single-point upwind scheme and two high-resolution schemes on a coarser grid. (Source code: `runPolyMixHiRes.m`.)

explicit discretizations too aggressively, which would lead to instabilities in the form of spurious oscillations. Figure 7.25 shows how this improves resolution; with AIM and WENO, the trailing waves are resolved almost as accurately as on the fine grid.

7.4.2 Subset from SPE10: Conformance Improvement

We consider a subset of the 64th layer of Model 2 from the SPE10 benchmark [10]. Our reservoir is described by a uniform $50 \times 50 \times 1$ rectangular grid that covers an area of size $1\,000 \times 500 \times 2\ m^3$. Initially, the reservoir has a uniform pressure equal 423 bar and is filled with 33% water and 67% oil that contains a lot of dissolved gas. We set a water injection well in the center of the model, operating at a fixed injection rate, and two production wells on the upper and lower sides, operating at a fixed bottom-hole pressure of 420 bar. All wells are shown as black dots in Figure 7.26. We consider two different injection scenarios: pure waterflooding and continuous polymer flooding. For the polymer, we include all effects outlined in Subsection 7.2.2: viscosity enhancement due to full mixing of polymer and water, adsorption, permeability reduction with $RRF = 1.3$, 5% inaccessible pore space, and shear-thinning polymer rheology. (The complete source code is found in the script `plyConformSPE10.m`.)

Here, we only review a few conspicuous points for the two scenarios. Figure 7.26 clearly shows how the injected water will finger rapidly through the high-permeability channel that connects the injector and the two producers. Adding polymer reduces the frontal saturation and retards the water front as observed in the fractional-flow analysis in Figure 7.4 but also improves flow conformance in terms of an improved areal sweep. This is also seen in the water cuts of the two producers and the cumulative oil production reported in Figure 7.27. However, this comes

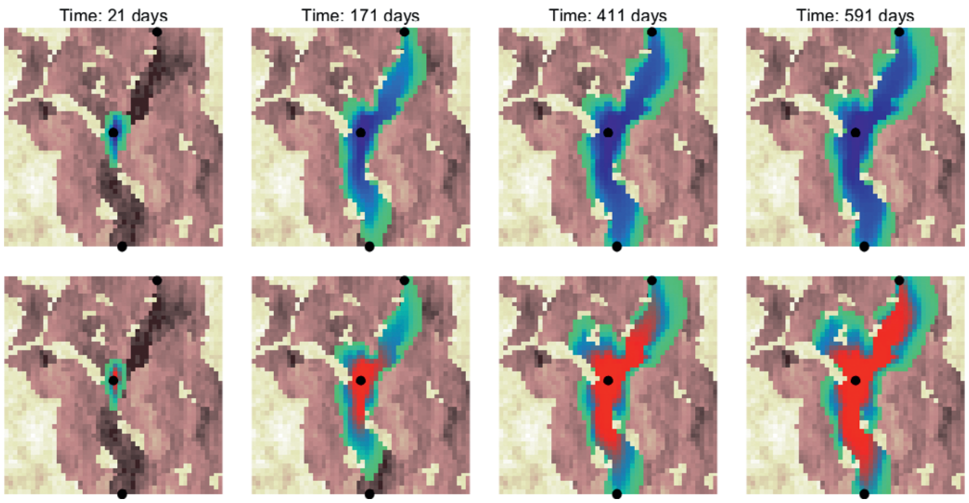


Figure 7.26 Comparison of waterflooding (top) and polymer flooding (bottom) in a fluvial reservoir. The brown background colors show the permeability on a logarithmic scale, with light colors signifying low permeabilities and dark colors signifying high permeabilities. Water saturation in flooded cells are shown using a blue-to-green color scale, and polymer concentration is shown in red.

at a cost: The increased viscosity of the diluted polymer causes a considerable reduction in injectivity despite its shear-thinning rheology, and to maintain the prescribed injection rate, the bottom-hole pressure of the injector must be increased significantly above the injection pressure used in the waterflooding scenarios. Notice also that the injection pressure continues to increase as the reservoir is filled with more water containing diluted polymer. Without the shear-thinning effect, the injection pressure would have to be increased another 10 bar or so.

Computer exercises

To get more familiar with the simulator, we suggest a few exercises:

1. Run the simulations yourself and compare the number of iterations and the consumed CPU time. (Hint: look in the `reports` structures.)
2. Run a simulation to verify the difference in injection pressure with and without shear thinning effects. How much does this effect the overall displacement? (Hint: to turn off shear thinning, you can set `model.usingshear` to `false`.)
3. How is the displacement affected if the assumption of full mixing of water and diluted polymer is not correct? (Hint: you can control the degree of mixing using the ω parameter `model.fluid.mixPar`.)

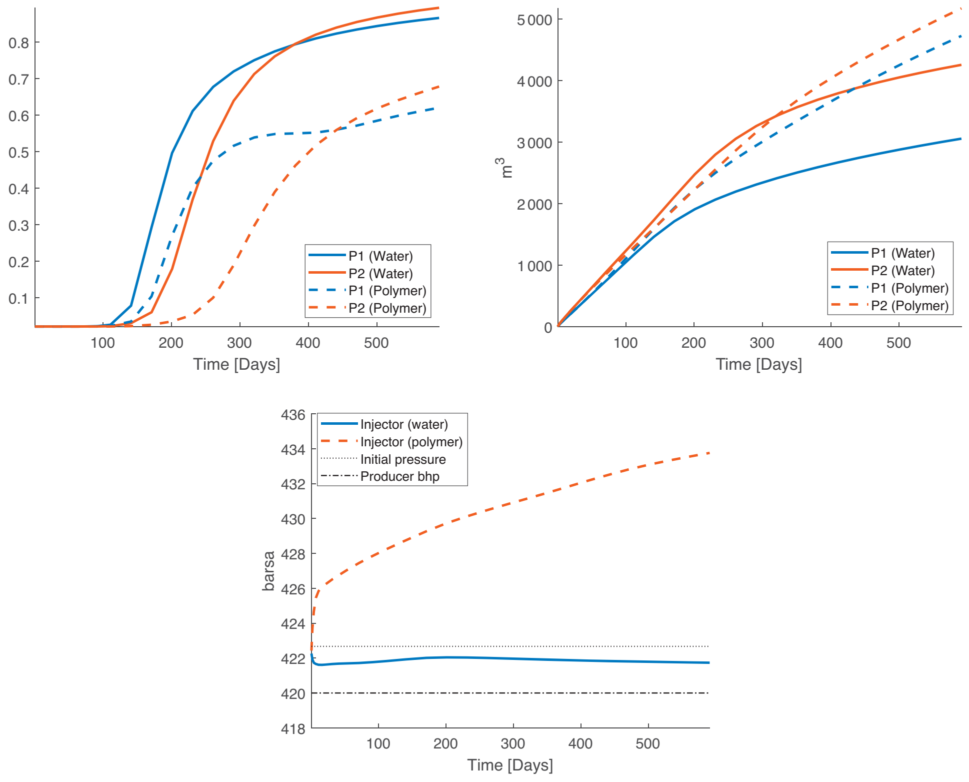


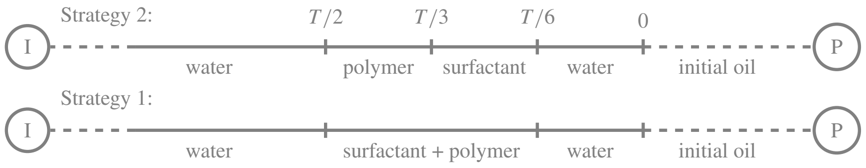
Figure 7.27 Well responses for the simulations in Figure 7.26. The upper-left plot shows water cut in the two producers, the upper-right plot shows cumulative surface production of oil, and the lower plot shows bottom-hole pressure in the injector.

4. The pressure control on the two producers is set so high that the oil stays above the bubble point. Lower the bottom-hole pressure and see how this affects the production process.
5. Use tools from the `diagnostics` module to investigate how polymer injection affects the *dynamic heterogeneity* of the displacement; see section 13.2 of the MRST textbook [22].

7.4.3 The Dynamics of Slug Injection

In all examples so far, we have considered continuous injection of chemical agents as part of a secondary or tertiary recovery process. Polymers and surfactants are rarely used in this way. A usual setup is instead to start off recovering oil by natural drive mechanisms (primary recovery), followed by injection of water to maintain pressure and push oil toward the producers (secondary recovery). In the tertiary recovery period, one first injects water containing surfactants to reduce interfacial tension between water and residual oil and to alter the wettability so that more oil can be recovered. Polymer is then injected for an extended period of time to provide mobility control for the mobilized oil bank. After a significant fraction of the sweep region has been flooded by chemicals, a pure water drive is set up to push the chemical slugs and the enhanced oil bank toward the producers.

In this example, we consider a conceptual 1D slug injection with an injector at the left and a producer at the right end. (Source code and ECLIPSE input files are found in the `book-ii/slug1D` folder of the `ad-eor` module.) Supporting fractional-flow analysis for such cases is discussed in chapters 7–9 of the textbook by Bedrikovetsky [6], and herein we only present results from numerical simulations. We consider two different EOR scenarios for our virtual recovery project, visualized to illustrate how the fluid distribution would be inside the reservoir, from injector I to producer P, if each slug represented a piston-type displacement:



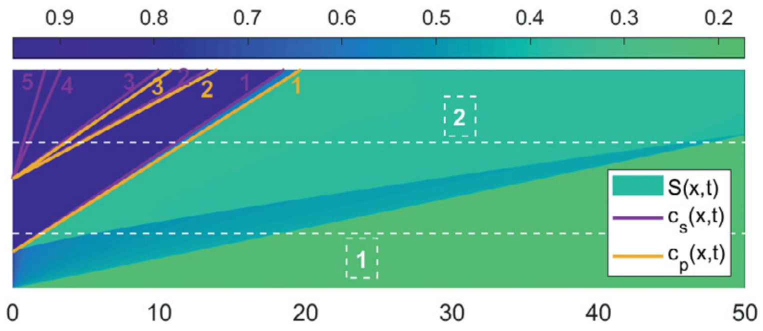
Strategy 1: Instead of injecting surfactant and polymer as two consecutive slugs, we first consider coinjection of a single slug. During the initial secondary-recovery period, the solution is a standard Buckley–Leverett profile, consisting of a shock followed by a rarefaction. Once chemical injection starts, we get a tertiary injection process. We have already presented and discussed the tertiary solution for a surfactant solution injected into a waterflooded reservoir in Figures 7.17 and 7.19. Adding polymer to the injection fluid does not fundamentally change the tertiary solution. There is still a significant oil bank, resulting from increased mobilization of residually trapped oil, that travels rightward as an S-shock that impinges on the trailing S-rarefaction from the secondary waterflooding. As this tertiary S-shock gradually eats away the trailing end of the S-rarefaction, its speed increases when the front saturation decreases, so that it (almost) overtakes the secondary S-shock by the time they reach the producer at the right end of the domain.

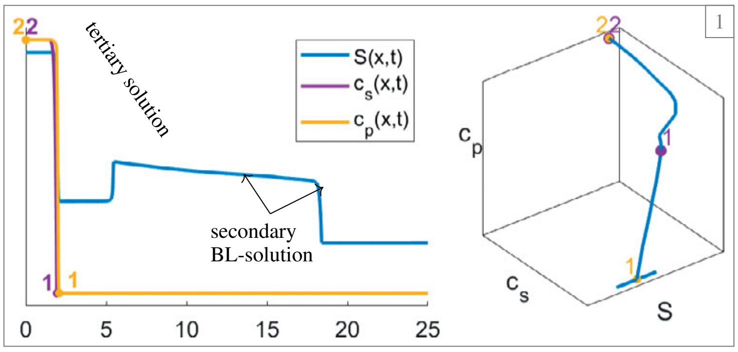
The leading tertiary S-wave (which with time also includes a weak rarefaction) is followed by two trailing c-waves associated with the eigenvalues $\xi^{c_s} = f/(S + a_s'(c_s))$ and $\xi^{c_p} = f/(S + a_p'(c_p))$, where a_s and a_p are the adsorption terms associated with surfactant and polymer, respectively; see Equation (7.14). If there is no adsorption or the adsorption terms are the same (i.e., $h^s_{LR} = h^p_{LR}$ in the notation of (7.17)), the eigenvalues coincide and the two c-waves collapse into a single c-wave, which is found as the upper envelope of the chemical fractional-flow curve. For the specific parameters of this example, the chemical curve is shifted so far to the right, compared with the pure-water curve, that the c-wave is a shock not followed by an S-rarefaction, as was the case in Figures 7.17 and 7.19.

In the general case, $\xi^{c_s} \neq \xi^{c_p}$, and this will cause so-called chromatographic separation between polymer and surfactant. You can see this from the difference in the yellow and magenta lines marked by “1” in the top plot of Figure 7.28, which represent the foot of the c_p- and c_s-waves, respectively. Inaccessible pore space will also influence the degree of separation, because water containing large molecules can only move through a smaller fraction of pore space than water containing surfactant, which implies that a polymer solution will move faster through a bulk volume than a surfactant solution. This separation is clearly evident in the second solution, sampled at time $\frac{2}{3}T$, after chemical injection has ceased. We also see that whereas the trailing edge of the polymer slug is a single rarefaction wave (points 2 to 3), the trailing edge of the surfactant splits into two rarefaction waves, a strong wave from points 2 to 3 and a weak wave from points 4 to 5.

Strategy 2: The recovery with the second, four-slug strategy (Figure 7.29) turns out to be less efficient. As intended, the surfactant slug reduces the interfacial tension between oil and water and reduces the endpoint saturations, but it does not provide sufficient mobility control to push a large amount of oil ahead of the chemical front (see the first solution sample). This happens first when we start injecting polymer in the second half of the chemical injection period (second and third solution samples). As a result, the bank of enhanced oil will consist of two parts and be composed of a number of subwaves that result from the interaction of the surfactant and polymer slugs. Altogether, this means that the production of oil mobilized by the surfactant is significantly delayed, and by the end of the project period, the highly resolved simulations show that the first strategy will have recovered 16.6% more oil. On the positive side, the second strategy requires significantly lower injection pressures.

The simulations presented thus far have been highly resolved. Typical lateral resolution in 3D models is on the order of tens of cells between injectors and producers, and in Figure 7.30 we have thus repeated the simulation of strategy 2 with 12, 25, 50, 100, and 200 cells. Consistent with the discussion in Subsection 7.4.1, the chemical slugs are smeared out to the point where the oil bank is hardly recognizable for the 12 cell simulation. Notice also how far the solution paths deviate from the high-resolution one in state space. Deviations in cumulative oil production nonetheless only range from 4.7% to 0.85%, because we stop the simulations long





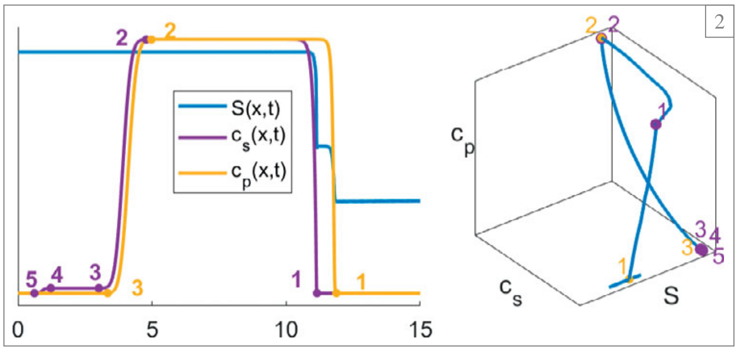


Figure 7.28 Highly resolved numerical solutions for strategy 1 of the slug-injection experiment (5 000 grid cells, 5 000 timesteps). The upper plot shows a color plot of the water saturation in the (x,t)-plane. The colored lines track start and endpoints for the c-waves for surfactant (magenta) and polymer (orange). The two boxes below show the solution in physical space (left) and in state space (right) at two different times, indicated as dashed horizontal lines in the color plot. The tracked points are indicated by colored dots. For the plots in physical space, surfactant and polymer concentrations have been scaled to take values in the unit interval.

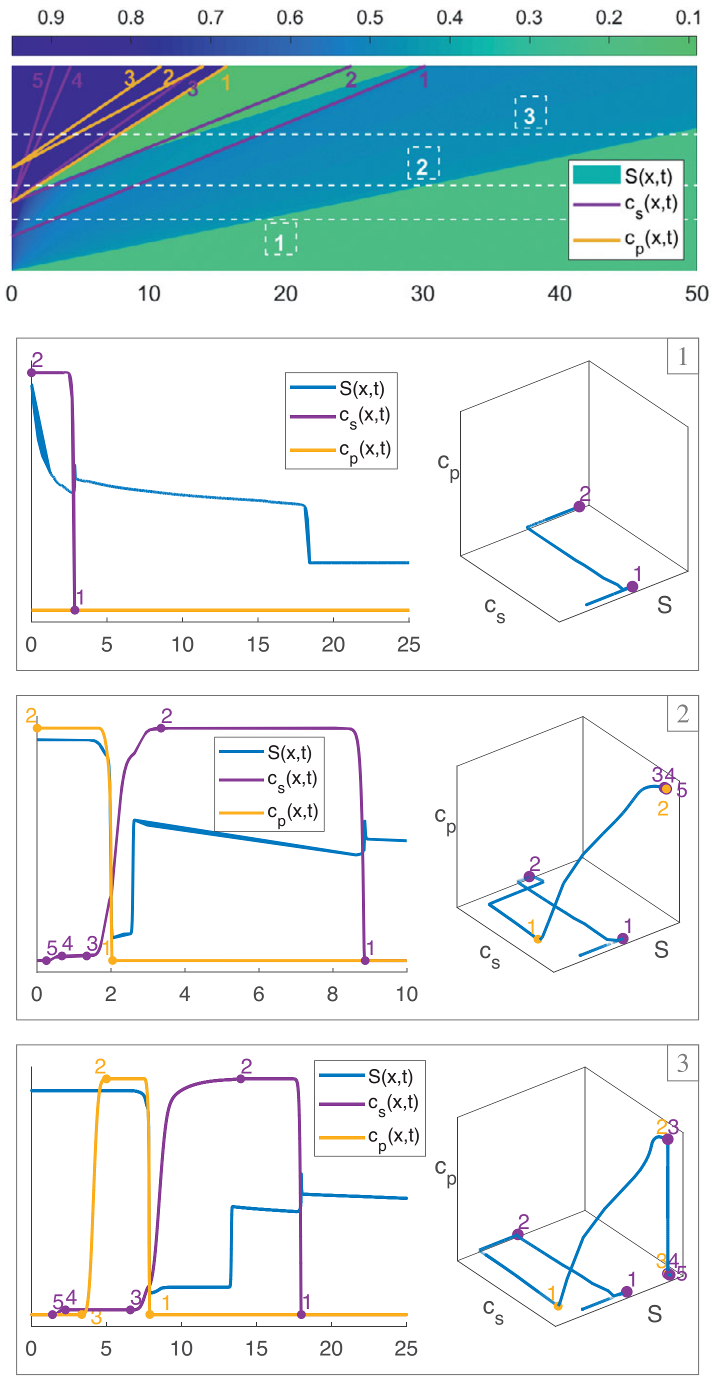


Figure 7.29 Solutions for strategy 2 of the slug-injection experiment shown in (x,t)-space (top) and at three different times in physical space (left column) and in state space (right column). See Figure 7.28 for an explanation of the different plots.

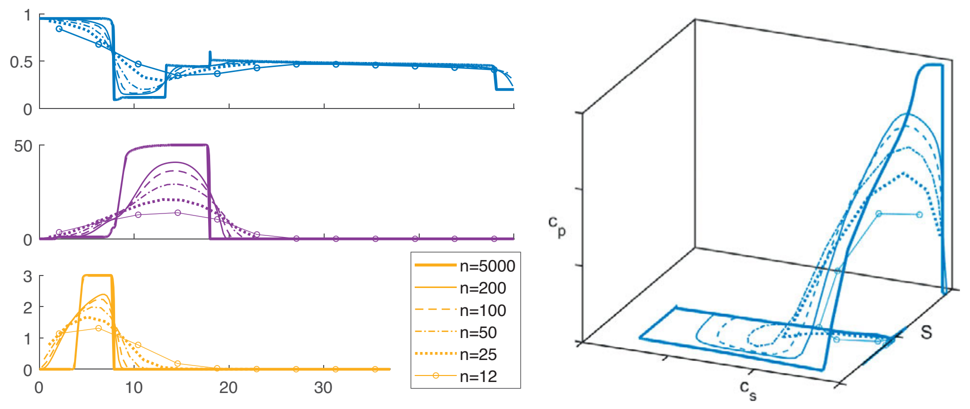


Figure 7.30 Grid-refinement study with n uniform cells for strategy 2; see Figure 7.29. The three coarsest simulations all used 100 timesteps, whereas the finest used 5 000. The left plots show $s(x,t)$, $c_s(x,t)$, and $c_p(x,t)$ for $t = 2T/3$. The right plot shows the same solutions in state space. (Source code: `slugGridRef.m`.)

before the oil bank has reached the producer and the self-sharpening shock of the secondary production profile is reasonably well resolved on a coarse grid.

Improved discretizations: For completeness, we also run scenario 2 with improved temporal and spatial discretizations. Figure 7.31 reports surfactant concentrations computed by the standard upwind method (SPU) and WENO, both with fully implicit (FIM) or AIM temporal discretization. In all simulations, we use 80 equally spaced timesteps. With 25 cells, the estimated Courant numbers vary from approximately 0.2 for the trailing waves to 2.75 for the leading saturation shock. As we have already noted, the smearing increases with decreasing Courant numbers for the explicit part of AIM but decreases for FIM. Hence, we only see a modest effect of replacing FIM by AIM. The effect of using WENO, on the other hand, is pronounced on all four grids. Typically, WENO gives at least as good resolution as SPU gives on a grid that is refined by a factor of 2.

When the number of cells is doubled to 50, Courant numbers for large parts of the trailing chemical waves fall in the range where AIM is effective and we thus observe significant improvements by switching time discretization for both SPU and WENO. On the 100-cell grid, the Courant number exceeds unity in the majority of the cells and the effect of AIM is negligible. With 200 cells, the WENO-FIM scheme breaks down (and produces a singular linearized system) when the Courant number increases well above 20 midway through the polymer injection period. However, if we halve the timestep during this time period, the simulation runs smoothly.

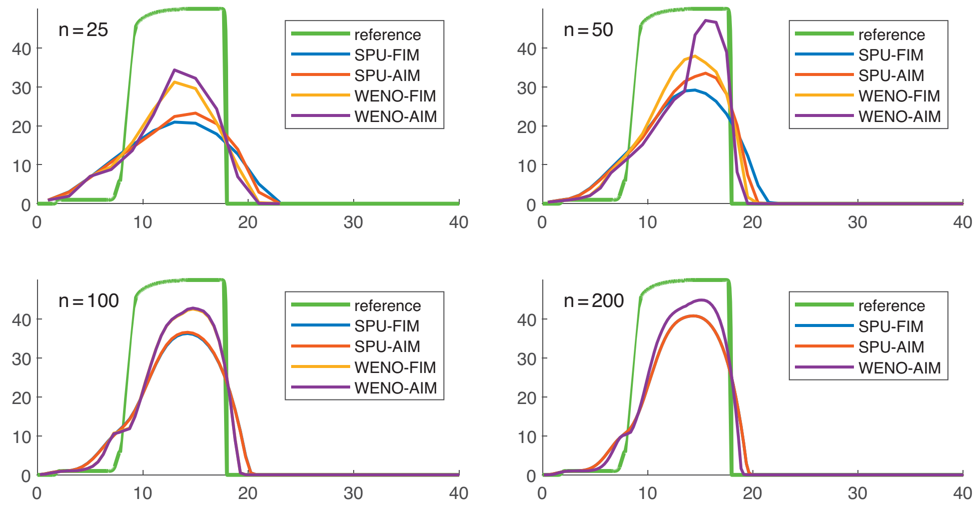


Figure 7.31 Surfactant concentration computed with different spatial and temporal discretizations on a uniform grid with *n* cells, using 80 equally spaced timesteps. (Source code: `slugWENOAIM.m`.)

Altogether, this example demonstrates two important points: (i) AIM seems to be most effective when the Courant numbers of trailing waves straddle unity and (ii) the WENO discretization is not as robust as the SPU scheme and should not be used with very large Courant numbers. What is a large Courant number in this regard may depend on the complexity and nonlinearity of the system you are solving.

7.4.4 Validation against a Commercial Simulator

In the last example, we validate our SP simulator against the commercial simulator ECLIPSE [30, 31]. To this end, we consider two models: a 2D vertical reservoir cross section with a single injector–producer pair and a small 3D sector model.

Vertical cross section: The setup is shown in Figure 7.32 and consists of a large $4\,000 \times 200 \times 125$ m^3 sandbox discretized on a uniform $20 \times 1 \times 5$ Cartesian grid. Initially, the reservoir is at hydrostatic equilibrium and contains all three phases, with a mobile gas cap overlying the mobile oil. Hydrocarbons will be recovered from a producer perforated in the upper two cells in the rightmost column and operating at a constant bottom-hole pressure of 260 bar. The production is supported by an injector perforated in the bottom two cells of the leftmost column, which injects the displacing fluids at constant flow rate of $1\,000$ m^3/day subject to a maximum pressure limit of 800 bar. (You find complete source code in `spValidation2D.m`.)

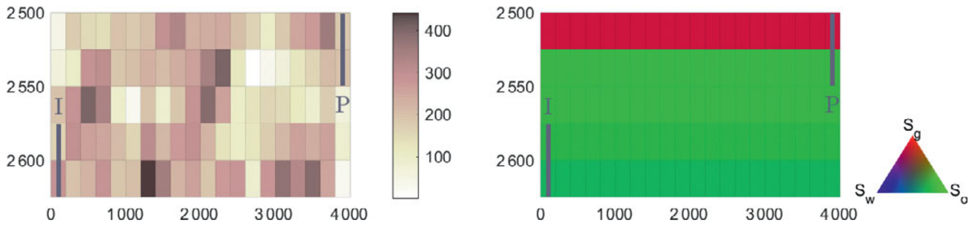


Figure 7.32 Setup of the 2D validation case with horizontal permeability in millidarcies to the left and initial fluid distribution to the right.

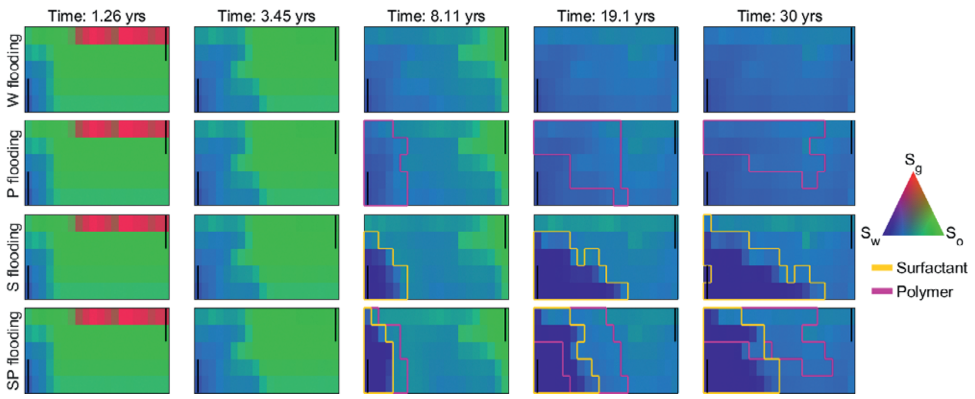


Figure 7.33 Evolution of the phase saturations over the whole simulation period for the 2D validation case. Columns show, from left to right, saturation midway through the water preflush, at the end of the preflush, after the chemical injection period, midway through the water postflush, and at the end of the simulation period. Black lines indicate wells, whereas colored lines outline the polymer/surfactant slugs, measured as $c > 0.05$.

The production is set up in three stages: First, 1 260 days of water preflush, and then, 1 700 days of chemical injection of polymer with a concentration of 2.0 kg/m^3, surfactant with a concentration of 20 kg/m^3, or a combination of both chemicals. This is followed by 8 000 days of chase water injection. For comparison, we also simulate pure waterflooding. Figure 7.33 reports computed fluid saturations and chemical concentrations throughout the four production scenarios. The water preflush is dominated by production from the gas cap, which is almost fully displaced by the time chemical injection starts (after 3.45 years). Injection of polymer contributes to push more oil toward the producer. If you compare the waterflooding and polymer flooding plots at 8.11 years, you can see that the oil saturation is slightly higher behind the leading water front and that the water saturation is slightly higher behind the trailing polymer front. The surfactant, on the other hand, has a very strong washing effect and reduces the residual water saturation from 0.2 to 0.06 but

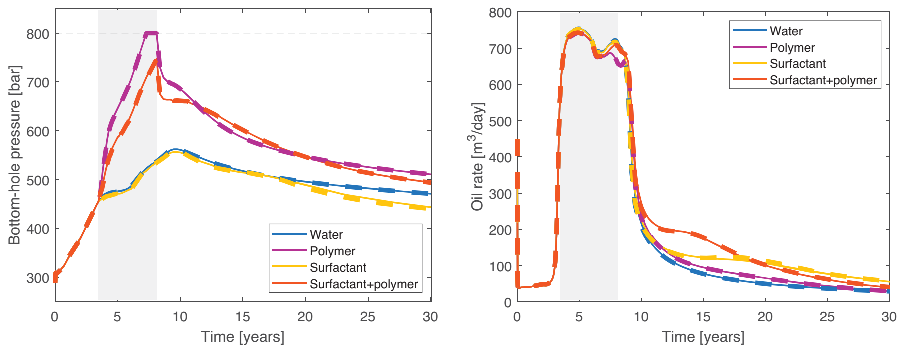


Figure 7.34 Bottom-hole pressure of the injector (left) and oil production rate (right) for the 2D cross-section validation case. Solid lines are simulated by MRST and dashed lines by ECLIPSE. The shaded regions represent the chemical injection period.

also introduces significant gravity segregation so that the displacing fluids move faster toward the producer along the bottom of the reservoir. Including polymer improves the conformance significantly and we end up with a trailing displacement front that is more vertical for SP flooding. (You can also notice a chromatographic separation between the two chemicals.)

Figure 7.34 reports bottom-hole pressures and oil production rates predicted by MRST and by ECLIPSE with the same input data. First of all, there is excellent agreement between the two simulators,[3] which validates our implementation. Let us also compare the four different production scenarios. By injecting polymer, we are able to maintain a slightly higher oil rate after water breakthrough around 8.6 years and up to approximately 25 years. Altogether, this enhances the oil and gas production by 4% and 2%, respectively. However, this comes at the cost of a significantly increased injection pressure in order to overcome the reduced injectivity caused by the more viscous injection fluid. The injector hits its pressure constraint after 7.3 years and is thus not fully able to maintain the prescribed injection rate for the full slug-injection period.

When solvent is injected, we see the opposite effect: Because solvent reduces the interfacial tension between oil and water, it not only mobilizes oil that would otherwise be residually trapped but also increases the mobility of both phases so that a lower injection pressure is sufficient to maintain the prescribed injection rate. This is particularly evident toward the end of the simulation. Altogether, solvent injection gives an enhanced oil and gas production of 14.1% and 7.9%, respectively.

[3] The match cannot generally be expected to be exact due to subtle differences in how the two simulators evaluate oil properties; ECLIPSE 100 interpolates b_o and b_o/μ_o [30], whereas MRST interpolates b_o and μ_o.

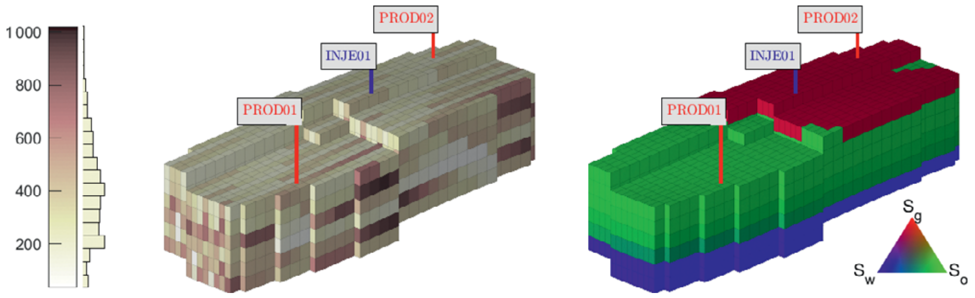


Figure 7.35 Setup of the 3D validation case with horizontal permeability in millidarcies to the left and initial fluid distribution to the right.

The full SP flooding contains features from both the polymer and the surfactant flooding. First of all, we see an even larger increase in the oil rate after water breakthrough, in particular in the period between years 10 and 18, as a result of the improved flow conformance. The highly viscous polymer solution still reduces injectivity, but the increased mobility caused by the surfactant will to a certain extent compensate for this, so that the pressure increase is less than for pure polymer flooding. The overall result is 10.6% increase in gas recovery and 18.9% increase in oil recovery after 30 years.

Sector model: Our second validation test is taken from [4] and is a 30 $\times$ 20 $\times$ 6 corner-point grid with 2 778 active cells, in which the middle section contains four intersecting vertical faults. The reservoir is produced from two production wells, located on the left and right sides, respectively, each perforating the top layer. These are supported by an injection well at the center of the model that perforates the lower three grid layers. The reservoir contains all three phase fluids in the initial state, and they are all in hydrostatic equilibrium, as shown in Figure 7.35.

To produce the reservoir, we consider the same four types of injection strategies as in the 2D example. The injector is set up with constant injection rate of 2 500 m^3/day and an upper pressure constraint of 290 bar, whereas the producers are set to operate at a constant bottom-hole pressure of 230 bar. The first water injection stage lasts for 560 days and then turns into a chemical injection with polymer solution concentration of 1 kg/m^3 and surfactant concentration of 30 kg/m^3. After 400 days of continuous chemical injection, chase water is injected for another 2 430 days.

Figure 7.36 reports a comparison of well responses simulated by MRST and by ECLIPSE. As in the 2D example, there is excellent agreement for the waterflooding and polymer flooding cases. For cases with surfactant, there are some slight deviations in the bottom-hole pressures, which we believe come from the fact that ECLIPSE uses a kind of operator-splitting method in which the surfactant

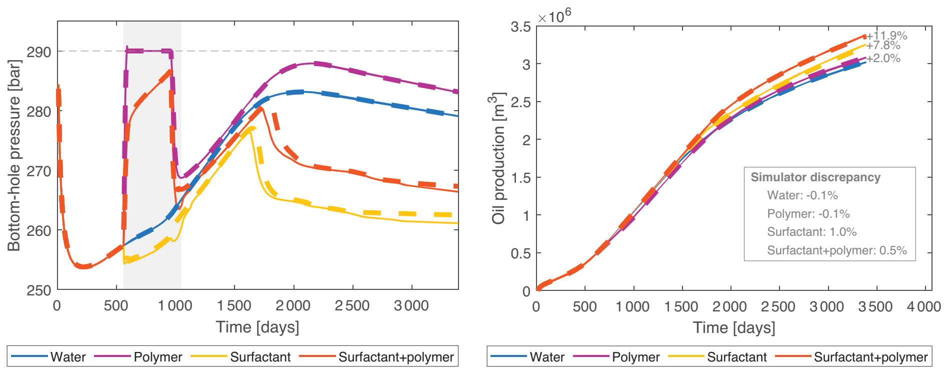
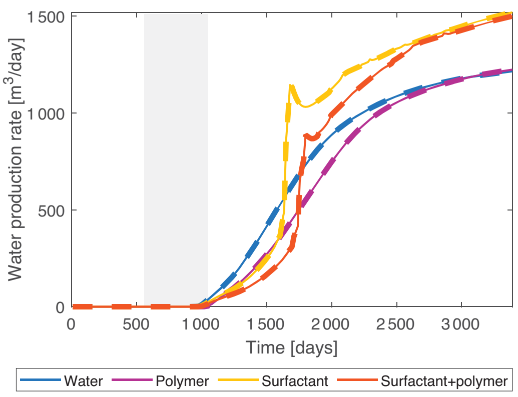


Figure 7.36 Bottom-hole pressure of the injector (left) and cumulative oil production (right) for the 3D cross-section validation case. Solid lines are simulated by MRST and dashed lines by ECLIPSE. The simulator discrepancy in oil production is measured as the difference between MRST and ECLIPSE, normalized by the MRST result.

concentrations are updated implicitly *after* the oil, gas, and water components have been computed [31]. However, the discrepancies in total oil production are nonetheless within 1% for all simulations.

Starting with the waterflooding scenario, we initially see a rapid decay in injection pressure because of gas production, followed by a gradual buildup of pressure until around water breakthrough to compensate for the increase resistance by the multiphase fluids. (The breakthrough appears as a gradual increase in water cut because the water front is very smeared out.) Once sufficient water communication is established between injector and producers, the prescribed injection rate can be maintained with a lower pressure and the injection pressure drops off. When polymer is injected, the pressure increases sharply because of poor injectivity and hits the upper constraint almost immediately, so that we cannot maintain the prescribed injection rate during the injection of the chemical slug. This reduces the efficiency of the polymer injection significantly so that all over oil recovery is only enhanced by 2%.

In the surfactant flooding, we notice that injection pressure decreases when surfactant is injected, as we already observed in the 2D case. More pronounced, however, is the sharp decay that takes place after approximately 1 600 days. Comparing the 3D plots of saturation in Figure 7.37, we see that a wide channel of high saturations has developed between the injector and the second producer for the surfactant flooding. This not only increases the water rate (and lowers the oil rate) but also explains the sharp decay in injection pressure as this water channel offers a path of less resistance to flow for the injected chase water. This effect is also evident in the combined surfactant–polymer flooding but is delayed and weakened because of the improved conformance introduced by the viscous



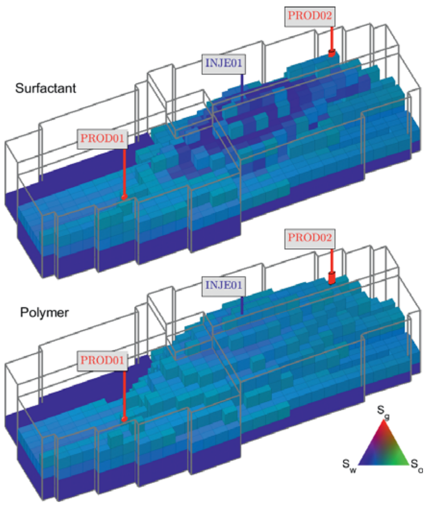


Figure 7.37 Water rates for the second producer (left plot). The plots to the right show phase saturations for surfactant flooding (top) and polymer flooding (bottom) in all cells where the water saturation exceeds 0.55. The plots are sampled after 1 695 days, when the water rate attains a local maximum for surfactant flooding.

polymer mixture. All over, the two surfactant cases enhance the oil recovery by 7.8% and 11.9%, respectively.

7.5 Directions and Suggestions for Future Improvements

This chapter has introduced you to the basic physiochemical mechanisms of EOR by polymer, surfactant, or surfactant–polymer flooding. We have also outlined how to model these processes by appropriate extensions of the standard black-oil equations and discussed our modular implementation, which has been set up so that you easily can utilize the many different discretizations, solvers, and solution algorithms the AD-OO framework of MRST offers.

Our main purpose of developing the `ad-eor` module of MRST is to provide a basic framework and flexible interface that we can build upon when developing novel models and simulation methods that do not exist in any other simulators. We are well aware that the AD-OO framework, with its modular state functions and generic simulator models, is complex and can be difficult to understand for new users. However, we hope that this chapter has made it more accessible and convinced you that it is a powerful tool you can leverage to quickly implement standard models for other well-known mechanisms or develop novel models and computational methods. We welcome and encourage you to participate in the continued research and construction of the `ad-eor` module and MRST in general.

In its current form, the `ad-eor` module of MRST only implements the most rudimentary models for three specific enhanced recovery schemes. Many other chemical and biological agents have been applied for EOR purposes, and researchers have discovered and studied many mechanisms that have yet to be implemented in MRST. We end the chapter by providing a brief overview of mechanisms that have yet to be included in the public version of the `ad-eor` module.

Temperature effects: Temperature affects the properties of both polymers and surfactants. Viscosity, for instance, is usually deeply affected by temperature. Excessively high temperature may even destroy the structure of the polymer and irreversibly reduce the viscosity of the polymer solution. [9, 21, 39]. Temperature also changes the adsorption characteristics of the surfactant at the solid–liquid interface and at the oil–water interface [7, 18], and further affects chemicals' ability to enhance the recovery factor. Therefore, the role of temperature should be emphasized in the case of large temperature changes in the formation.

Mechanical effects: During polymer injection, when the flow rate is high, the polymer not only undergoes the shear-thinning behavior described in this chapter but mechanical degradation may also occur. This phenomenon is generally considered to be an irreversible behavior caused by the breaking of polymer molecular chains [3]. Mechanical degradation can significantly reduce the viscosity of the displacing agent and thus reduce its EOR effect. Accounting for this mechanism is important so that correct design decisions can be made; e.g., by preshearing the polymer mixture according to simulation results before injection to maximize its effectiveness [32].

Wettability alteration: Interfacial phenomena are common in reservoirs where multiphase fluids coexist. Wettability is one of the key parameters to accurately describe the interface phenomenon and plays an important role in surfactant flooding and spontaneous imbibition [43] in fractured reservoirs. In addition, the application of nanomaterials in EOR is attracting more attention. The mechanism of wettability alteration caused by the adsorption of nanomaterials on solid surfaces [42] also needs to be further explored and accurately simulated.

Salinity effects: Like temperature, changes in salt ion type and concentration also affect the properties of polymers and surfactants [40]. For polymers, the viscosity-increasing effect changes with different salt ion concentration; usually the viscosity decreases, but a few special polymers have salt-sensitive self-thickening effects [41]. For surfactants, the presence of salt ions not only impacts the reduction in interfacial tension and the wettability alteration but also plays an important role in the phase behavior and properties of microemulsion.

In the most basic model [31], polymer solution, reservoir brine, and injected water all form pseudophases within the aqueous phase, and the evolution of salt concentration is described with a similar flow equation as for polymer concentration, with an effective water viscosity described using Todd–Longstaff mixing to account for physical dispersion at the front and fingering effects at trailing edges but without adsorption and dead pore space. This model should thus be relatively straightforward to include in the existing `ad-eor` module.

Microemulsions: When the surfactant concentration is greater than the critical micelle concentration, the surfactant will spontaneously aggregate and form micelles. The presence of micelles will cause a part of oil and water to mix thermodynamically and stably, thus forming a new phase state, called the microemulsion phase [1]. The appearance of new phase states will change the original oil–water phase properties (such as density, viscosity, etc.) and will also change the properties between phase states (such as interfacial tension, relative permeability, capillary force, etc.) [12, 25]. Therefore, accurate description and simulation of the microemulsion phase are very important in the simulation of surfactant flooding with medium–high concentrations.

Alkali flooding: In this EOR technique, one injects alkaline chemicals, such as sodium carbonate, that react with acidic oil components to create natural surfactants such as petroleum sulfonate inside the reservoir. Like the water-soluble synthetic surfactant discussed earlier in this chapter, the resulting petroleum sulfonate will mobilize more oil and thus increase the microscopic displacement efficiency by reducing interfacial tension, changing the rock surface wettability, and emulsifying the oil into the injected water. Alkaline chemicals can also have an adverse effect, by reacting with the calcium ions to produce scale and precipitation that may damage the formation. It is thus not suitable for reservoirs with high divalent ion concentration formation water.

Tertiary ASP flooding [35] combines polymer injection with two sources of surfactants, water-soluble synthetic surfactant and alkaline chemicals that form surfactants in situ and contribute to reduce surfactant adsorption. This lowers the required concentration of the synthetic surfactant and makes the injection more sustainable and less detrimental to long-term production. As in SP flooding, the surfactant and polymer can be co-injected, but polymer is often injected as a post-slug to mobilize the oil freed from the rock by the surfactant and provide general mobility control to the flood fronts. In the most basic model [31], the flow equation for the alkali concentration has the same form as for polymer but without the effects of dead pore space and permeability reduction. This model should thus be relatively straightforward to include in the existing `ad-eor` module.

Compositional effects: To study the action mechanism of various chemical agents in more detail, it may be necessary to study crude oil components separately. Likewise, chemical agents can also be used alongside of gases, such as carbon dioxide, nitrogen, oil field associated gas, etc. In all of these cases, compositional EOR models are necessary to simulate the interaction between each components and the resulting changes in fluid properties [13, 28]. For more details on compositional simulation, we refer to Chapter 8.

Acknowledgement. We thank Olav Møyner and Xavier Raynaud for useful discussions and comments on the chapter and invaluable help with the implementation of the `ad-eor` module.

References

[1] S. Abbott. *Surfactant Science: Principles and Practice*, volume 1. 2016. URL www.stevenabbott.co.uk/practical-surfactants/the-book.php

[2] S. M. Ali and S. Thomas. The promise and problems of enhanced oil recovery methods. *Journal of Canadian Petroleum Technology*, 35(7), 57–63, 1996. doi: 10.2118/96-07-07.

[3] B. Al-Shakry, T. Skauge, B. Shaker Shiran, and A. Skauge. Impact of mechanical degradation on polymer injectivity in porous media. *Polymers*, 10(7):742, 2018. doi: 10.3390/polym10070742.

[4] K. Bao, K.-A. Lie, O. Møyner, and M. Liu. Fully implicit simulation of polymer flooding with MRST. *Computational Geosciences*, 21(5–6):1219–1244, 2017. doi: 10.1007/s10596-017-9624-5.

[5] D. Baxendale, K. Bao, M. Blatt, et al. *Open Porous Media: Flow Documentation Manual*, 2019–10 Rev-1 edition, 2019. URL http://opm-project.org

[6] P. Bedrikovetsky. *Mathematical Theory of Oil and Gas Recovery: With Applications to Ex-USSR Oil and Gas Fields*, volume 4 of *Petroleum Engineering and Development Studies*. Kluwer Academic Publishers, Dordrecht, the Netherlands 1993.

[7] A. F. Belhaj, K. A. Elraies, S. M. Mahmood, N. N. Zulkifli, S. Akbari, and O. S. Hussien. The effect of surfactant concentration, salinity, temperature, and ph on surfactant adsorption for chemical enhanced oil recovery: a review. *Journal of Petroleum Exploration and Production Technology*, 10(1):125–137, 2020. doi: 10.1007/s13202-019-0685-y.

[8] G. Cheraghian, S. S. K. Nezhad, M. Kamari, M. Hemmati, M. Masihi, and S. Bazgir. Adsorption polymer on reservoir rock and role of the nanoparticles, clay and SiO_2. *International Nano Letters*, 4(3):114, 2014. doi: 10.1007/s40089-014-0114-7.

[9] B. Choi, M. S. Jeong, and K. S. Lee. Temperature-dependent viscosity model of HPAM polymer through high-temperature reservoirs. *Polymer Degradation and Stability*, 110:225–231, 2014. doi: 10.1016/j.polymdegradstab.2014.09.006.

[10] M. A. Christie and M. J. Blunt. Tenth SPE comparative solution project: A comparison of upscaling techniques. *SPE Reservoir Evaluation & Engineering*, 4:308–

317, 2001. doi: 10.2118/72469-PA. URL www.spe.org/web/csp/datasets/set02.htm.

[11] C. Dai and F. Zhao. *Oilfield Chemistry*. Springer, Singapore, 2019. doi: 10.1007/978-981-13-2950-0.

[12] M. Delshad, G. A. Pope, and K. Sepehrnoori. A compositional simulator for modeling surfactant enhanced aquifer remediation, 1 formulation. *Journal of Contaminant Hydrology*, 23(4):303–327, 1996. doi: 10.1016/0169-7722(95)00106-9.

[13] C. Han, M. Delshad, K. Sepehrnoori, and G. A. Pope. A fully implicit, parallel, compositional chemical flooding simulator. In *SPE Annual Technical Conference and Exhibition, 9–12 October, Dallas, Texas*. Society of Petroleum Engineers, 2005. doi: 10.2118/97217-MS.

[14] S. T. Hilden, H. M. Nilsen, and X. Raynaud. Study of the well-posedness of models for the inaccessible pore volume in polymer flooding. *Transport in Porous Media*, 114(1):65–86, 2016. doi: 10.1007/s11242-016-0725-8.

[15] E. L. Isaacson. Global solution of a Riemann problem for a non-strictly hyperbolic system of conservation laws arising in enhanced oil recovery. Unpublished. Technical report issued by the Enhanced Oil Recovery Institute, University of Wyoming. 1989.

[16] T. Johansen and R. Winther. The solution of the Riemann problem for a hyperbolic system of conservation laws modeling polymer flooding. *SIAM Journal on Mathematical Analysis*, 19(3):541–566, 1988. doi: 10.1137/0519039.

[17] T. Johansen and R. Winther. The Riemann problem for multicomponent polymer flooding. *SIAM Journal on Mathematical Analysis*, 20(4):908–929, 1989. doi: 10.1137/0520061.

[18] W. Karnanda, M. S. Benzagouta, A. AlQuraishi, and M. M. Amro. Effect of temperature, pressure, salinity, and surfactant concentration on IFT for surfactant flooding optimization. *Arabian Journal of Geosciences*, 6(9):3535–3544, 2013. doi: 10.1007/s12517-012-0605-7.

[19] M. Khatibi, N. Potokin, and R. W. Time. Experimental investigation of effect of salts on rheological properties of non-Newtonian fluids. *Transactions of the Nordic Rheology Society*, 24:117–126, 2016.

[20] L. W. Lake. *Enhanced Oil Recovery*. Prentice Hall, Old Tappan, NJ, 1989.

[21] E. A. Lange and C. Huh. A polymer thermal decomposition model and its application in chemical EOR process simulation. In *SPE/DOE Improved Oil Recovery Symposium*. Society of Petroleum Engineers, 1994. doi: 10.2118/27822-MS.

[22] K.-A. Lie. *An Introduction to Reservoir Simulation Using MATLAB/GNU Octave: User Guide for the MATLAB Reservoir Simulation Toolbox (MRST)*. Cambridge University Press, Cambridge, UK, 2019. doi: 10.1017/9781108591416.

[23] K.-A. Lie, T. S. Mykkeltvedt, and O. Møyner. A fully implicit WENO scheme on stratigraphic and unstructured polyhedral grids. *Computational Geosciences*, (24):405–423, 2020. doi: 10.1007/s10596-019-9829-x.

[24] T. S. Mykkeltvedt, X. Raynaud, and K.-A. Lie. Fully implicit higher-order schemes applied to polymer flooding. *Computational Geosciences*, 21(5–6):1245–1266, 2017. doi: 10.1007/s10596-017-9676-6.

[25] L. Patacchini, R. De Loubens, A. Moncorge, and A. Trouillaud. Four-fluid-phase, fully implicit simulation of surfactant flooding. *SPE Reservoir Evaluation & Engineering*, 17(2):271–285, 2014. doi: 10.2118/161630-PA.

[26] Ø. Pettersen. Basics of reservoir simulation with the eclipse reservoir simulator. Lecture notes, Department of Mathematics, University of Bergen, 2006. URL www.mj-oystein.no/index_htm_files/ResSimNotes.pdf

[27] G. A. Pope. The application of fractional flow theory to enhanced oil recovery. *Society of Petroleum Engineers Journal*, 20(3):191–205, 1980. doi: 10.2118/7660-PA.

[28] G. A. Pope and R. C. Nelson. A chemical flooding compositional simulator. *Society of Petroleum Engineers Journal*, 18(5):339–354, 1978. doi: 10.2118/6725-PA.

[29] W. Pu, F. Jiang, B. Wei, Y. Tang, and Y. He. Influences of structure and multi-intermolecular forces on rheological and oil displacement properties of polymer solutions in the presence of Ca^{2+}/Mg^{2+}. *RSC Advances*, 7(8):4430–4436, 2017. doi: 10.1039/C6RA25132C.

[30] Schlumberger. *ECLIPSE: Reference Manual*. Schlumberger, 2014.1 edition, 2014.

[31] Schlumberger. *ECLIPSE: Technical Description*. Schlumberger, 2014.1 edition, 2014.

[32] R. S. Seright, J. M. Maerker, and G. Holzwarth. Mechanical degradation of polyacrylamides induced by flow through porous-media. In *Abstracts of Papers of the American Chemical Society*, volume 182, pp. 17. American Chemical Society, Washington, DC, 1981.

[33] J. J. Sheng. *Modern Chemical Enhanced Oil Recovery: Theory and Practice*. Gulf Professional Publishing, Boston, 2010. doi: 10.1016/C2009-0-20241-8.

[34] J. J. Sheng. *Enhanced Oil Recovery Field Case Studies*. Gulf Professional Publishing, Boston, 2013. doi:10.1016/C2010-0-67974-0.

[35] J. J. Sheng. A comprehensive review of alkaline–surfactant–polymer (ASP) flooding. *Asia-Pacific Journal of Chemical Engineering*, 9(4):471–489, 2014. doi: 10.1002/apj.1824.

[36] K. S. Sorbie. *Polymer-Improved Oil Recovery*. Springer Science+Business Media, New York, 2013. doi: 10.1007/978-94-011-3044-8.

[37] G. W. Thomas and D. H. Thurnau. Reservoir simulation using an adaptive implicit method. *Society of Petroleum Engineers Journal*, 23(5):759–768, 1983. doi: 10.2118/10120-PA.

[38] M. R. Todd and W. J. Longstaff. The development, testing, and application of a numerical simulator for predicting miscible flood performance. *Journal of Petroleum Technology*, 24(7):874–882, 1972. doi: 10.2118/3484-PA.

[39] E. Vermolen, M. J. Van Haasterecht, S. K. Masalmeh, M. J. Faber, D. M. Boersma, and M. A. Gruenenfelder. Pushing the envelope for polymer flooding towards high-temperature and high-salinity reservoirs with polyacrylamide based ter-polymers. In *SPE Middle East Oil and Gas Show and Conference*. Society of Petroleum Engineers, 2011. doi: 10.2118/141497-MS.

[40] Z. Wu, T. Cheng, J. Yu, and H. Yang. Effect of viscosity and interfacial tension of surfactant–polymer flooding on oil recovery in high-temperature and high-salinity reservoirs. *Journal of Petroleum Exploration and Production Technology*, 4(1):9–16, 2014. doi: 10.1007/s13202-013-0078-6.

[41] Q. You, K. Wang, Y. Tang, G. Zhao, Y. Liu, M. Zhao, Y. Li, and C. Dai. Study of a novel self-thickening polymer for improved oil recovery. *Industrial & Engineering Chemistry Research*, 54(40):9667–9674, 2015. doi: 10.1021/acs.iecr.5b01675.

[42] H. Zhang, A. Nikolov, and D. Wasan. Enhanced oil recovery (EOR) using nanoparticle dispersions: underlying mechanism and imbibition experiments. *Energy & Fuels*, 28(5):3002–3009, 2014. doi: 10.1021/ef500272r.

[43] X. Zhou, N. R. Morrow, and S. Ma. Interrelationship of wettability, initial water saturation, aging time, and oil recovery by spontaneous imbibition and waterflooding. In *SPE/DOE Improved Oil Recovery Symposium*. Society of Petroleum Engineers, 1996. doi: 10.2118/35436-MS.