11

Compressible Multiphase Flow

The black-oil equations constitute the industry-standard approach to describe compressible three-phase flow. Black-oil models generally have stronger coupling between fluid pressure and the transport of phases/components than the two-phase, incompressible flow models discussed in the previous chapter. For this reason it is common to use a fully coupled solution strategy, in which the whole system of equations is discretized implicitly and all primary unknowns are solved for simultaneously. This chapter introduces you to the underlying physics and describes the various rock-fluid and PVT properties that enter these models. We also explain the basics of how these models are discretized and implemented in MRST. Our implementation will rely heavily on the discrete operators discussed earlier in the book. As a gentle introduction to the numerics of the black-oil equations, we pick up again the simple AD solver from Chapter 7 and extend it to compressible two-phase flow without mass transfer between the phases. The code is applicable to general unstructured grids in 2D and 3D, but it is still a far stretch from a full-fledged reservoir simulator capable of simulating real assets, as discussed in Chapter 12.

11.1 Industry-Standard Simulation

A commercial reservoir simulator is a complex software that involves a wide variety of flow models, computational algorithms, and a lot of subtle tricks to ensure robust behavior. Covering all of these features, as well as the many different scenarios for which a simulator can be used, is well beyond the scope of this chapter. Instead, I will try to provide you with a minimum of insight into the key principles and show how to set up your first black-oil simulation in MRST using the AD-OO framework (this framework is described in more detail in Chapter 12). Let us start by a brief overview before we discuss the different constituents in more detail.

Flow Models and Phase Behavior

Reservoir fluids are complex mixtures of many chemical components that distribute differently among the fluid phases present in the reservoir depending upon temperature and pressure. The composition and characteristics of hydrocarbon fluids vary largely from one reservoir to another, but may also have large variations across reservoir compartments. As we have seen already, the black-oil model groups the chemical components into two

https://doi.org/10.1017/9781108591416.015 Published online by Cambridge University Press

pseudo-components reflecting the phase they belong to at surface conditions. At reservoir conditions, each component can be present in both a liquid oleic and a gaseous phase. We motivate and briefly explain the phase diagrams used to describe the phase distribution of the pseudo-components, but an in-depth discussion of the underlying physics and the many different forms these diagrams can take is outside the scope of the book.

The black-oil equations can also model processes involving other chemical species. In enhanced oil recovery, various forms of chemical and biological substances are added to the injected fluid to increase sweep efficiency and improve local displacement. You can model such systems by adding extra fluid components to the three phases. MRST offers an EOR module with basic models for polymer and surfactant flow; see [30] for more details, as well as a module for simulating solvents. We have already discussed CO_2 storage multiple times. Here, the buoyant and supercritical phase formed when CO_2 is injected sufficiently deep into the subsurface can dissolve into the resident brine. The behavior of the CO_2–brine system can thus be modeled as a hydrocarbon system in which brine plays the role as oil and CO_2 the role as gas. Extensive support has been developed in MRST to describe the long-term migration of buoyant CO_2 in large-scale aquifer systems, in which fully implicit solution methods developed for the black-oil equations play a key role. You can find more details in the thesis by Andersen [19].

Grouping the hydrocarbons into two pseudo-components does not describe all recovery processes with sufficient accuracy. *Compositional simulation* models the reservoir fluids as a system with M components and describes their phase behavior by an equation of state. The basic conservation equations for such systems was given in Section 8.2.2 and typical examples of equations of state were briefly reviewed as part of our discussion of single-phase flow in Section 4.2. As of release 2017b, MRST offers a module for compositional simulation. We will not discuss this at all herein; you can find preliminary details in the thesis by Møyner [212]. Likewise, a new module for geochemistry was also added in the same release, which, in particular, offers non-isothermal aqueous speciation, surface chemistry, redox chemistry, as well as solid and gas phase equilibrium.

Wells and Production Control

In previous chapters, wells have always been completed in a single cell or along straight lines following the axial directions, and have either been controlled by constant pressure or constant fluid rate. Real wells are significantly more complex and generally follow highly deviated trajectories and possibly consist of multiple branches. Many subtleties go into well modeling. In particular, special care is needed in the solution algorithm to handle cross-flow, in which fluids may flow into one part of the wellbore and out of another.

To ensure that each well operates under realistic engineering constraints, reservoir simulators offer a comprehensive set of control options that can be applied to fluid rates and bottom-hole or tubing-head pressures. A typical control strategy consists of setting a target rate and constraints on the pressure the well can sustain before the rate must be reduced. Well control may thus switch depending upon the computed pressures and flow rates once the well fails to meet one of its constraints. In so-called history mode, observed fluid rates at reservoir conditions are set as controls to reproduce production history as closely as possible. Controls also include economic constraints, which force wells to be completely shut in or recompleted if their production falls below a lower economic limit, and gas lift or other mechanisms introduced to help lift the well fluids to the surface when the oil or liquid rates fall below a specified limit. This will strongly affect the bottom-hole pressure in individual wells.

Control strategies can also be imposed on groups of wells, or on the field as a whole. Typical strategies here either involve guide controls, in which total rates are distributed to individual rates according to their production potential, or prioritization schemes that turn wells on in decreasing order of priority. Group and field controls may also involve various restrictions in terms of total pump and lift capacities, reinjection of produced formation gas or water, limitations in surface facilities, etc. Altogether, the resulting control strategies may involve a lot of quite intricate logic. MRST contains data structures for representing surface facilities, but only a limited number of models and control options have so far been implemented in the public version. I therefore consider a more detailed discussion of control strategies to be outside the scope of this book. Examples of third-party implementations with MRST are discussed in two master theses [141, 115].

Many wells also have advanced completions and devices for inflow control. In *multisegment* well models, the well trajectory is discretized into a series of segments that can be linked together into branches. Each branch can, in turn, contain sub-branches. A segment describes how the flow rate between two points (nodes) relates to the pressure drop between the same two points. Nodes can be located at different positions along the wellbore or on opposite sides of devices that restrict the flow, such as chokes and valves. Each node has an associated volume, and mass-conservation equations are solved between the nodes to describe pressure drops and wellbore storage effects. As of release 2017a, MRST contains a general multisegment model that allows branches of segments to be defined in the tubing and in the void space (annulus) between the inner and outer tubing. The branches can be joined at multiple points in a flexible manner to form a general network. Altogether, this enables accurate description of inflow devices and the near-well region, as well as complex cross-flow patterns that may arise in multilateral wells. An in-depth discussion of multisegment well modeling is outside the scope of this book, but as is always the case with MRST, you are free to consult the code to understand the necessary details.

Data Input

To model real assets, a simulator needs comprehensive and flexible input functionality to specify the setup of all ingredients that constitute the simulation model. The industry-standard approach is to define an input language consisting of a number of keywords the user can utilize to construct a model of a computation (i.e., a state machine). The detailed syntax will vary from one simulator to another, but the general setup is conceptually the same. You have already encountered the ECLIPSE input format [270, 33] multiple times throughout the book. Over the years, this software has amassed options for simulating a wide variety of physical processes, and the input language currently has several thousand keywords, many of which have multiple available options. There are also similar file formats that can be used to store simulation results so that these can be reloaded and used to continue a previous simulation.

So far, we have only discussed how ECLIPSE input describes grids and petrophysical properties, and even for this purpose there are a large number of keywords we have not discussed. In general, an ECLIPSE input deck consists of eight different sections, some of which are optional:

`RUNSPEC` – required section, includes a description of the simulation, including parameters such as name of the simulation case, grid dimensions, phases and components present, number of wells, etc.

`GRID` – required section, describes basic grid geometry/topology and petrophysical properties (porosity, permeability, net-to-gross).

`EDIT` – optional section describing user-defined changes that are applied to the grid data *after* these have been processed to modify pore volume, cell centers, transmissibilities, local grid refinements, etc.

`PROPS` – required section describing fluid and rock properties such as relative permeabilities, capillary pressure, and PVT specification as a function of fluid pressures, saturations, and compositions.

`REGIONS` – optional section defining how the computational grid can be split into multiple regions for calculating PVT properties, relative permeabilities and capillary pressures, initial conditions, and fluids in place.

`SOLUTION` – required section specifying how the model is to be initialized: calculated from specified depths of fluid contacts, read from a restart file (output) from a previous run, or specified for each grid cell.

`SUMMARY` – optional section that enables the user to specify the reservoir responses (well curves, field-average pressure, etc.) to be written to a summary file after each time step.

`SCHEDULE` – required section, defines wells and how they are to be operated, specifies time-step selection and tolerances for the solvers, controls the output of quantities defined over the grid (pressures, saturations, fluxes, etc.).

The sections *must come in this prescribed order* in the input file, but the keywords within each section can be given in arbitrary order. Altogether, the input data file can be seen as a kind of scripting language that not only enables you to input data to the simulator, but also provides control of the time dependency and how the simulator will run a prescribed set of time steps.

In the following, we borrow the naming convention from ECLIPSE and refer to the way a simulation is to be run as a *schedule*. In the simplest case, a schedule consists of a number of prescribed time steps and one control per well. However, schedules representing real assets tend to be much more complicated, as can be inferred from the previous discussion of wells.

Let me also make two additional remarks: Since several of the keywords, and in particular those in the `GRID` section, may contain huge amounts of data, the input format offers an `INCLUDE` keyword allowing input data to be separated into multiple files. Moreover, although MRST aims to support the most widely used keywords in the required sections, there are many keywords that are *not* supported. In particular, no effort has been put into supporting the flexible, yet complex, options for user-specified postprocessing of geological models in the `EDIT` section.

The following text will give some details about ECLIPSE input [270, 33], but only when this is prudent to understand how you can set up the various part of a black-oil fluid description in MRST. If you are interested in a more thorough introduction, I recommend the lecture notes by Pettersen [250], which are much to the point, freely available, and complementary to the discussion in my book. An early version of the ECLIPSE 100 User Course [269] can also be found online.

More Advanced Solution Methods

As evidenced by our discussion, discrete flow models found in reservoir simulators are significantly more complex than the incompressible, two-phase flow equations considered in previous chapters. For small models with relatively simple phase behavior and solution trajectories, it is straightforward to linearize the flow equations and solve them with standard linear algebra, much in the same way as we did for single-phase flow in Section 7.2. However, for more complex phase behavior, there are many features complicating the solution process, like the need to switch primary unknowns from one cell to another when phases appear or disappear. Likewise, you may need extra postprocessing steps to record historic solution trajectories to model hysteretic behavior of relative permeabilities and capillary pressure, and so on. As the complexity increases for each of the three constituents of the reservoir model (geology model, fluid model, well model), one really sees the benefit of using automatic differentiation to perform the necessary linearizations.

All these model complications translate to increased nonlinearity in the discretized system, and a simple nonlinear Newton–Raphson solver cannot be expected to always converge unless we instrument it with inexact line-search methods and methods for time-step control, as briefly introduced in Section 10.2.2. Likewise, there is a need for correct scaling of the residual equations, as well as more advanced linear solvers, e.g., a preconditioned Krylov subspace method like the flexible general minimal residual (GMRES) method, combined with specialized preconditioning strategies aimed at lessening the impact of the largely different mathematical characteristics of the various sub-equations. Classic choices of single-stage preconditioners include the incomplete lower/upper (ILU) family or the nested factorization method [24], which is still the default method in ECLIPSE. Another popular choice is the two-stage constrained-pressure residual (CPR) method [304, 305]. Here, the first step computes an approximate solution of a reduced-pressure matrix that is appropriately scaled and selected so that one can use, e.g., a very efficient algebraic multigrid solver [285, 119]. The second step then applies a broadband smoother like ILU to the entire system. Section 12.3.4 briefly outlines such preconditioners and the exact solution strategy used in MRST to obtain robust simulation of real cases. Full details can, as always, be found in the code.

Details of the MRST Implementation

To a large degree, MRST has many of the model complexities discussed thus far implemented in its black-oil type solvers. Our first successful attempt at implementing black-oil simulators relied on discrete operators for discretization and automatic differentiation and used the type of procedural programming discussed so far in the book. As more complex behavior was introduced, the code gradually became more unwieldy, and at a certain stage, we realized that a major restructuring was required to make the code easier to extend and maintain. The result was the object-oriented AD-OO framework [170, 212, 30], which will be discussed in more detail in the next chapter. The key idea of the AD-OO framework is to separate the implementation of physical models (i.e., conservation equations and constitutive relationships), discrete operators, nonlinear solvers and time-stepping, and assembly and solution of the linear system. By using a combination of classes, structures, and functions for generic access, details from within each of these contexts can be hidden from the user/developer and only exposed if really needed. As a result, it is now much simpler to implement generic algorithmic components or extend existing models with more comprehensive flow physics.

This introduces a pronounced divide in MRST between simulators written for incompressible and compressible flow. It also represents a pedagogical challenge. I have solved it by relying on the simplified codes written in a procedural form to introduce key concepts that are also found in the object-oriented framework, albeit implemented in a somewhat different form. Once the basic concepts have been introduced, I switch to using the AD-OO framework when presenting simulation cases with black-oil models. This means, in particular, that the codes used to run black-oil simulations will not be presented at the same level of detail as in previous chapters of the book, but all examples have complete source codes in the `ad-blackoil` or `book` modules.

11.2 Two-Phase Flow without Mass Transfer

For a system consisting of two immiscible phases, each with a fixed chemical composition and no mass transfer between the phases, the basic flow model consists of the conservation equations (8.22) and the standard multiphase extension of Darcy's law, (8.23). This if often referred to as a *dead oil* model, and is representative for reservoirs with relatively thick or residue oil that has lost its light and volatile hydrocarbon components, or for oils containing a constant amount of dissolved gas because pressure is so high that gas cannot appear as a separate phase at reservoir conditions (more about this later).

If we for simplicity neglect capillary pressure and introduce discrete approximations for all derivatives, the discrete flow equations read,

$$\frac{(\boldsymbol{\phi} \boldsymbol{S}_\alpha \boldsymbol{\rho}_\alpha)^{n+1} - (\boldsymbol{\phi} \boldsymbol{S}_\alpha \boldsymbol{\rho}_\alpha)^n}{\Delta t^n} + \texttt{div}(\boldsymbol{\rho v})_\alpha^{n+1} = (\boldsymbol{\rho q})_\alpha^{n+1}, \tag{11.1a}$$

$$\boldsymbol{v}_\alpha^{n+1} = -\frac{\boldsymbol{K} \boldsymbol{k}_{r\alpha}^{n+1}}{\boldsymbol{\mu}_\alpha^{n+1}} \big[\texttt{grad}(\boldsymbol{p}^{n+1}) - g \boldsymbol{\rho}_\alpha^{n+1} \texttt{grad}(z)\big]. \tag{11.1b}$$

There are only two issues that are conceptually different from the corresponding single-phase model (7.2) we studied in Chapter 7: first of all, we now have two phases and must ensure that all saturation-dependent quantities are averaged correctly, as discussed in the previous chapter. Secondly, wells must be treated slightly differently, since the wellbore now may contain a mixture of two fluid phases. We shall put this issue off for now, and return to it later.

A Simple Compressible Two-Phase Simulator

To develop our first simulator, we consider a simple rectangular $200 \times 200 \times 50$ m^3 reservoir, initially filled with oil with oil in the upper 2/3 and by water in the bottom 1/3 of the volume. Water is injected in the lower southwest corner and fluids are produced from the upper northeast corner. To mimic the effect of injection and production wells, we manipulate the flow equations to ensure reasonable inflow/outflow for the perforated cells. The resulting code, called `twoPhaseAD.m`, can be found in the `book` module and follows essentially the same steps as the single-phase code in Section 7.2. Setup of the geological model is essentially the same and we skip the details.

For the fluid model, we assume that the water phase is slightly compressible with quadratic relative permeabilities, whereas the oil phase is lighter, more compressible, and has cubic relative permeabilities

```
[muW, muO ] = deal(1*centi*poise, 5*centi*poise);
[cw,  co  ] = deal(2e-6/psia,     1e-5/psia);
[krW, krO ] = deal(@(S) S.^2,     @(S) S.^3);
```

The reference densities at 200 bar are 1014 kg/m^3 for water and 850 kg/m^3 for oil. Since the capillary pressure is assumed to be zero, we can compute the initial pressure distribution exactly as for single-phase flow on page 206. Transmissibilities and discrete operators are independent of the flow equation and are defined as in Section 7.2.2, except that we now also have to define the upwind operator:

```
upw  = @(flag, x) flag.*x(N(:, 1)) + ~flag.*x(N(:, 2));
```

The setup of the simulation loop follows along the same path as in Section 7.2.4. We start by initializing the unknown variables and introduce indices so that we that later can easily extract the individual variables from the global vector of unknowns,

```
[p, sW]    = initVariablesADI(p0, sW0);
[pIx, sIx] = deal(1:nc, (nc+1):(2*nc));
```

Next, we initialize the necessary iteration variables and create a simulation schedule containing a gradual ramp-up of the time step. Then, the main loop is more or less identical to the one shown on page 210. The only difference is how we compute the discrete equations. We start by computing all cell-based properties

```
% Densities and pore volumes
[rW,rW0,rO,rO0] = deal(rhoW(p), rhoW(p0), rhoO(p), rhoO(p0));
[vol, vol0]     = deal(pv(p), pv(p0));

% Mobility: Relative permeability over constant viscosity
mobW = krW(sW)./muW;
mobO = krO(1-sW)./muO;
```

In the flux terms, we approximate the product of density and mobility at the interface between cells using upwind-mobility weighting. To this end, we first define the difference in phase pressures across each cell interface,

```
dp  = grad(p);
dpW = dp-g*avg(rW).*gradz;
dpO = dp-g*avg(rO).*gradz;
```

which we then use to determine the upstream direction for each phase and pick cell values from the upstream side

```
vW  = -upw(double(dpW) <= 0, rW.*mobW).*T.*dpW;
vO  = -upw(double(dpO) <= 0, rO.*mobO).*T.*dpO;
```

We now have all terms we need to compute the residual equations in each cell,

```
water = (1/dt(n)).*(vol.*rW.*sW - vol0.*rW0.*sW0) + div(vW);
oil   = (1/dt(n)).*(vol.*rO.*(1-sW) - vol0.*rO0.*(1-sW0)) + div(vO);
```

The injector is set to inject 0.1 pore volume of water over a period of 365 days. We model the injection by a source term having constant mass rate equal the total volumetric rate times the surface density of water

```
water(inIx) = water(inIx) - inRate.*rhoWS;
```

For the producer, we just fake plausible behavior; that is, we replace the oil equation by an equation that equates the cell pressure to the specified bottom-hole pressure and set the residual equation for water equal to S_w,

```
oil(outIx) = p(outIx) - outPres;
water(outIx)    = sW(outIx);
```

These conditions will ensure that fluids flow into the perforated cell.

Structure of the Linearized System

As before, the linearized system can be assembled into a global system matrix by concatenating the two ADI vectors,

```
eqs = {oil, water};  eq = combineEquations(eqs{:}); % i.e., cat(eqs{:});
```

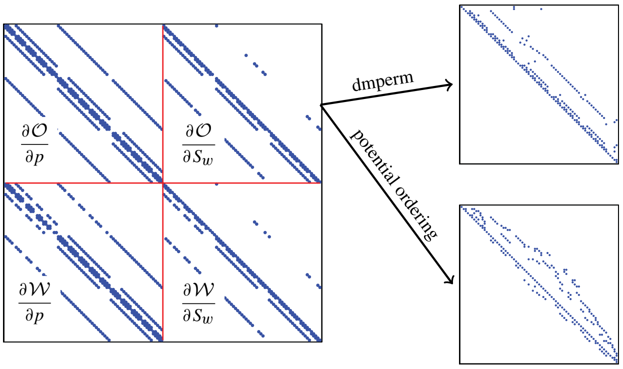


Figure 11.1 Linearized equations for the compressible, two-phase system without mass transfer between phases. The discrete equations have a 2×2 block structure corresponding to the derivatives of the water ($\mathcal{W}$) and oil ($\mathcal{O}$) equations with respect to the two primary variables p and S_w.

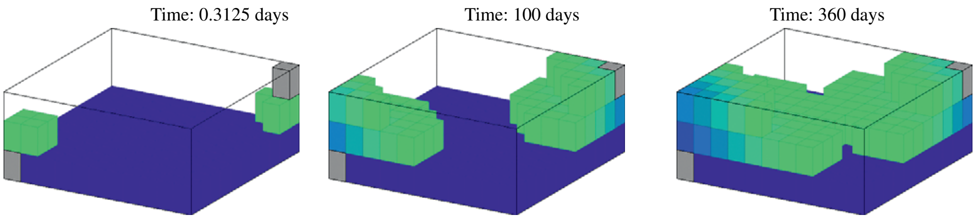


Figure 11.2 Saturation profiles for the simple two-phase problem without mass transfer between the phases. The plots only show cells in which $S_w > 10^{-3}$.

Let us inspect this system in some detail on a small $5 \times 5 \times 3$ grid. A large number of the coefficients in the residual equations are zero in the first iteration steps since the pressure differential induced by the wells has not yet started to cause significant movement of fluids. We therefore run five of the ramp-up steps, bringing us to time five days, before we inspect the linearized system.

In the linearized system in Figure 11.1 we recognize the sparsity pattern of a standard seven-point scheme in matrix blocks that have derivatives with respect to p. Blocks with S_w-derivatives can be reordered to obtain the triangular structure we discussed in Section 10.3.4, which is typical for hyperbolic problems. Indeed, we have already seen that we can rewrite this compressible system as a parabolic pressure equation and a hyperbolic transport equation in the absence of capillary forces.

Figure 11.2 clearly shows that the suggested production strategy might not be optimal. Already in the first time step, water is sucked up from the bottom layer and towards the producer. However, by injecting in the water layer, rather than in the oil zone, we ensure that oil is not pushed down into the water zone. Maybe you can think of a better design to produce this reservoir? You can for instance try to move the injector to the upper southwest corner and repeat the simulation.

We leave the setup of wells and instead investigate what effect compressibility has on the system by comparing solutions in the compressible and the incompressible case. If we reset all the compressibilities to zero and rerun the code, we get the following error message:

```
Warning: Matrix is close to singular or badly scaled.
Results may be inaccurate.
RCOND =  2.977850e-23.
```

This is usually reason to worry, and we will get back to this issue shortly. However, let us first look at the computed results in terms of oil production, pressure in cell perforated by the injector, and field-average pressure. Oil production can be measured from a simple mass-balance calculation. To this end, we need the initial oil in place, which we compute as part of the initialization.

```
ioip  = sum(pv(p0).*(1-sW0).*rhoO(p0));
```

To extract injector pressure, average field pressure, and oil in place at the end of each time step we postprocess the array of structures used to store the computed solutions,

```
t   = arrayfun(@(x) x.time, sol)/day;
ipr = arrayfun(@(x) x.pressure(inIx), sol)/barsa;
fpr = arrayfun(@(x) sum(x.pressure)/G.cells.num, sol)/barsa;
oip = arrayfun(@(x) sum(pv(x.pressure).*(1-x.s).*rhoO(x.pressure)), sol);
```

Notice here the combination of `arrayfun`, which enables us to operate on each element of an array, and anonymous functions used to extract and process the fields belonging to each array element. Figure 11.3 reports these reservoir responses. Both the pressure in the perforated injector cell and the field-average pressure shown in the left plot exhibit

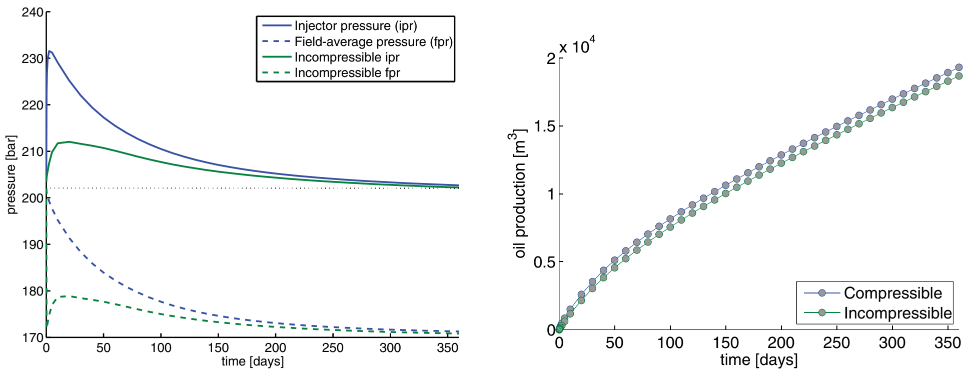


Figure 11.3 Reservoir responses for the first two-phase example without mass transfer. The left plot shows the pressure in the perforated cell and field-average pressure (FPR) and the right plot shows total field oil production (FOPT). (The dotted line indicates initial FPR.)

significantly different behavior in the compressible and incompressible models. In the compressible case, we see that the injector pressure[1] is significantly higher since some of the injected energy is used to compress the oil and expand the pore space rather than only pushing fluids, as in the incompressible case. At the same time, the reduced pressure at the producer will cause the compressible oil to expand. Hence, we also observe a somewhat higher total field-oil production. Looking at the FPR, we see that this is high initially in the compressible case, but decays gradually as more fluids are produced. We also observe the same trend in the incompressible case, but here we have an initial transient period during which the pressure first decreases and then increases slightly before we observe the expected gradual decay. Given that we do not have a real well model, it is somewhat futile to put too much attempt into interpreting this behavior.

Numerical Challenges

Let us return to the numerical difficulties of the problem. Figure 11.4 reports the number of iterations observed along with a plot of how the reciprocal condition number changes with time for the two simulations. The more well-conditioned the system is, the closer the reciprocal condition number should be to unity. All numbers are very small in our case, indicating quite ill-conditioned systems. In particular, during the first four iterations steps, the reciprocal condition numbers for the incompressible case are of the order 10^{-37} to 10^{-24}, which means that this system is effectively singular. Nevertheless, the direct solver manages to compute a solution that can be used to continue the nonlinear iteration toward convergence. This is a well-known problem in reservoir simulation; fully implicit formulations are not suited for solving incompressible flow problems. A common trick to avoid singular matrices is therefore to add a little compressibility to the problem to make it less ill-posed. In our example, we could, for instance, assume incompressible fluids but set the rock compressibility to 10% of the value used in the compressible simulation without visibly affecting the pressure and production curves reported in Figure 11.3.

Also in the compressible case, the condition numbers are not very good. From a pragmatic point of view, you may think that this is not a serious problem since the direct solver in MATLAB has no problem to solve the system. Nevertheless, the problem will get worse if the reservoir has strong heterogeneity. Moreover, when moving to larger systems, we would eventually have to utilize an iterative solver, which in most cases needs a suitable preconditioner to deliver satisfactory convergence, as we will discuss later.

To summarize: This simple example has introduced you to the typical structure of a compressible, multiphase problem. You have seen that the elliptic and hyperbolic characteristics discussed in previous chapters indeed can be observed in the discrete system. We have also demonstrated that there are significant differences in compressible and incompressible systems and that these generally require different solution strategies; hence the divide in MRST between incompressible and compressible solvers. We will return to more numerical

[1] Referring back to the Peaceman well model (4.35) on page 128, this pressure corresponds to p_R and will be lower than the bottom-hole pressure p_{bh} inside the wellbore.

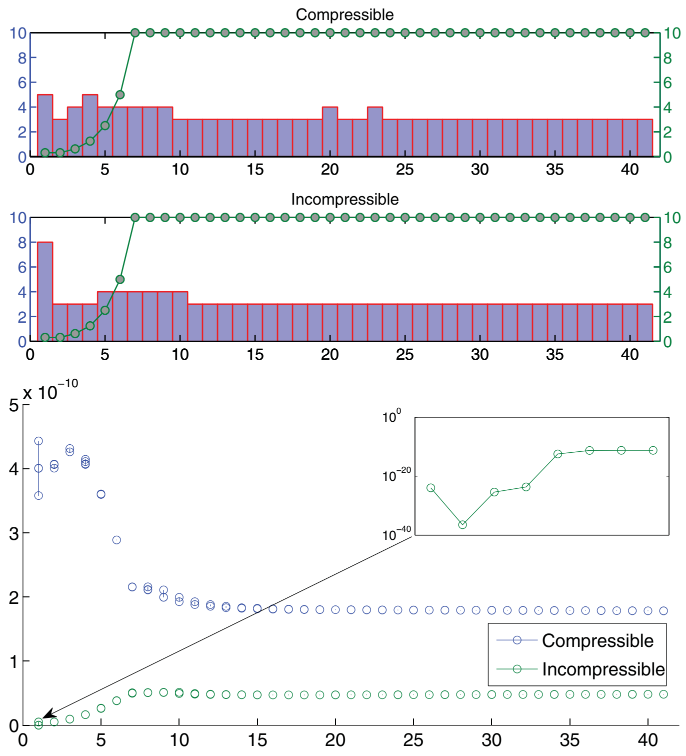


Figure 11.4 Numerical stability for the two-phase problem. In the upper plot, the bars show the number of iterations to convergence for each of the 41 time steps; time-step lengths (in units days) are shown as a line plot. The lower plot shows estimates of the variation in reciprocal condition numbers for the linearized system.

examples later in the chapter, but now it is time for a more in-depth discussion of the fluid and phase behavior of three-phase models.

11.3 Three-Phase Relative Permeabilities

When extending our two-phase simulator to three phases, we have to determine how to model saturation-dependent quantities like capillary pressure and relative permeabilities. In principle, one could imagine that all three phases could flow simultaneously in all parts of the pore volume and that relative permeability and capillary pressure generally were functions of two independent saturations. It is more common to assume that the flow is essentially two-phase, with the third phase acting as a passive background phase that restricts the flow volume and provides resistance to flow but does not contribute actively to the flow. Effectively, this means that the relative permeabilities of gas and water are functions of gas and water saturation, respectively, as we saw in Section 8.1.4. (The same applies to capillary pressure.) A commercial simulator like ECLIPSE offers several different options [271] for combining two-phase models to obtain three-phase relative permeabilities, including both the Stone I [284] and II [283] models. Common for all options is that they rely on interpolation of tabulated input. The various options are classified into families that either use one-dimensional or two-dimensional input. This offers great flexibility, but may also slow down the simulation if you are not careful when you specify the input tables. Fully implicit simulators implemented with the AD-OO framework in the `ad-blackoil` module utilize fluid objects, as discussed in Sections 5.1.1 and 10.1 to evaluate relative permeabilities and capillary pressures as general functions of phase saturations. It is up to you whether these functions should be based on analytic expressions or tabulated input, and the default dependence on saturation only can easily be overridden should you need to include other variable dependencies.

For convenience, the `ad-props` module offers a generic interface for constructing fluid objects based on input in ECLIPSE format. So far, we have only implemented the most common family of input parameters and the default option for interpolating three-phase oil relative permeability from two-phase curves. Nonetheless, it should not be a very difficult exercise to extend this implementation to include other families of input options. The rest of this section describes the default model and its implementation in the `ad-props` module in more detail.

11.3.1 Relative Permeability Models from ECLIPSE 100

The default model for calculating three-phase oil relative permeability in ECLIPSE 100 is based on the same key idea as the Stone I model (8.19)–(8.20) on page 243, but where the Stone I model assumes a nonlinear relationship to interpolate the two-phase curves, the default model employs a simpler linear relationship, which according to [271] avoids problems of negative values and poor conditioning. The model makes the key assumption that water and gas are completely segregated within each cell, giving the fluid distribution illustrated in Figure 11.5. This means that each cell is divided into two zones (here, S_w, S_o, and S_g denote the average saturations):

- in the upper gas zone, only oil and gas flow, and the oil saturation is S_o, the water saturation is S_{wc}, and the gas saturation is $S_g + S_w - S_{wc}$;
- in the lower water zone, only oil and water flow, and the oil saturation is S_o, the water saturation is $S_g + S_w$, and the gas saturation is 0.

Then, the oil relative permeability is interpolated linearly from two-phase relative permeabilities

$$k_{ro}(S_g, S_w) = \frac{S_g\, k_{rog}(S_o)}{S_g + S_w - S_{wc}} + \frac{(S_w - S_{wc})\, k_{row}(S_o)}{S_g + S_w - S_{wc}}, \tag{11.2}$$

with $S_o = 1 - S_w - S_g$. Here, k_{rog} is the oil relative permeability for a system with oil, gas, and connate water, and k_{row} is the oil relative permeability for a system with oil and gas

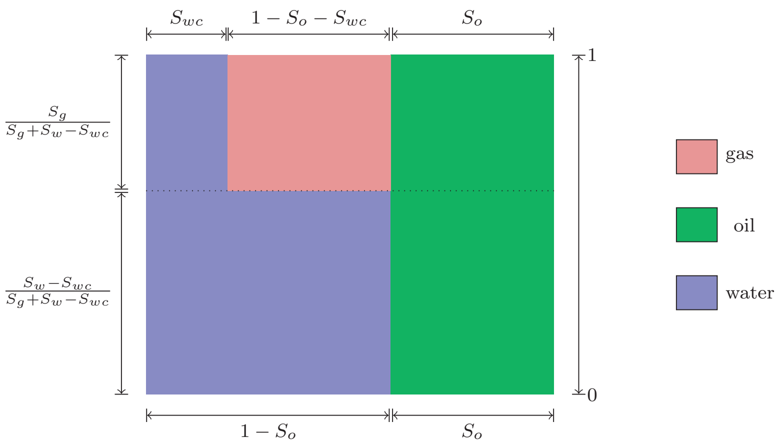

S_{wc}
$1 - S_o - S_{wc}$
S_o
$\frac{S_g}{S_g+S_w-S_{wc}}$
$\frac{S_w-S_{wc}}{S_g+S_w-S_{wc}}$
1
0
$1 - S_o$
S_o
gas
oil
water

Figure 11.5 Fluid distribution inside each cell assumed by the default method for computing three-phase oil relative permeability in MRST; adapted from [271].

only, both tabulated as a function of oil saturation. This model is implemented as default within MRST, too.

To create a fluid object, we must read and process data from the `PROPS` section of ECLIPSE input files. For this, we use the `deckformat` module, which offers general functionality for reading and processing input decks. The most important function is the reader,

```
deck = readEclipseDeck(fn);
```

which reads the input file `fn` and constructs a structure `deck` with one field for each of the sections in the input file, except `EDIT` and `SUMMARY`. In passing, we mention that the `deckformat` module also offers several functions to construct objects for grids, rock properties, and wells, as well as functionality for reading and processing ECLIPSE *output* files. Hence, you can use MRST to visualize output from ECLIPSE simulations, and it is also possible to use existing ECLIPSE runs to initialize simulation cases in MRST.

To understand the input, we must look in more detail at the input format of the `PROPS` section. All data are given as capital letter keywords, followed by numbers and terminated by a single `/`. As an example, let us take an excerpt from the file `BENCH_SPE1.DATA`, which is supplied with MRST and describes input for the SPE 1 benchmark case [241]

```
PROPS    ===============================================================
-------- THE PROPS SECTION DEFINES THE REL. PERMEABILITIES, CAPILLARY
-------- PRESSURES, AND THE PVT PROPERTIES OF THE RESERVOIR FLUIDS
----------------------------------------------------------------------

-- ROCK COMPRESSIBILITY
--    REF. PRES   COMPRESSIBILITY
ROCK
       14.7          3.0E-6          /
```

Here, all lines starting with a double-dash (`--`) are interpreted as comments and the only data contained in this part of the file are the two parameters necessary to describe the compressibility of the rock, i.e., a reference pressure (in units bar) and a compressibility (in units 1/bar). For this particular input file, the relative permeabilities are given by the two keywords `SWOF` and `SGOF`:

```
SWOF
--  SWAT          KRW                 KRO               PCOW
  0.1200000000                   0   1.000000000000    0
  0.1210000000   0.000000011363636   1.000000000000    0
  0.1400000000   0.000000227272727   0.997000000000    0
  0.1700000000   0.000000568181818   0.980000000000    0
  0.2400000000   0.000001363636364   0.700000000000    0
  0.3199999990   0.000002272727261   0.350000004375    0
  0.3700000000   0.000002840909091   0.200000000000    0
  0.4200000000   0.000003409090909   0.090000000000    0
  0.5200000000   0.000004545454545   0.021000000000    0
  0.5700000000   0.000005113636364   0.010000000000    0
  0.6200000000   0.000005681818182   0.001000000000    0
  0.7200000000   0.000006818181818   0.000100000000    0
  0.8200000000   0.000007954545455   0.000000000000    0
  1.0000000000   0.000010000000000                0    0
/
```

The four columns represent water saturation (`SWAT`), relative permeability of water (`KRW`), relative permeability of oil (`KRO`), and oil–water capillary pressure (`PCOW`). After processing, the `deck.PROPS` field will have one subfield containing each of these three data

```
deck.PROPS =
     SWOF: {[14x4 double]}
     SGOF: {[15x4 double]}
          :
     ROCK: [1.0135e+05 4.3511e-10 NaN NaN NaN NaN]
```

The careful reader may observe that each of the `SWOF` and `SGOF` fields represent the input data as a cell containing an array. The reason is that the relative permeability input may contain multiple arrays of different length corresponding to different regions of the reservoir. Each array will then be represented as a cell and the whole function as a cell array.

We can then create a fluid object by calling `initDeckADIFluid` from `ad-props`

```
fluid = initDeckADIFluid(deck);
```

The essential part of `initDeckADIFluid` is a generic loop that traverses the fields of `deck.PROPS` and tries to pass each of them onto a constructor function written specifically for each field

```
props = deck.PROPS;
fns = fieldnames(props);
for k = 1:numel(fns)
    fn   = fns{k};
    if doAssign(fn)
       asgn = str2func(['assign',fn]);
       try
```

```
            fluid = asgn(fluid, props.(fn), reg);
         catch  %#ok
            warning(msgid('Assign:Failed'), ...
                'Could not assign property ''%s''.', fn)
         end
      end
  end
```

The second line extracts a list with the names of all fields present, the fifth line checks the field name against a list of field names that either have not been implemented yet or should for some reason not be processes. The sixth line makes a function pointer by appending the field name to the string "assign," in our case creating function pointers to `assignROCK`, `assignSWOF`, and `assignSGOF`. These functions are then called in the eight line, with the fluid object `fluid` and the appropriate fluid data as input. (The third input argument, `reg`, represents region information.) Let us look at the implementation of rock compressibility. If we for simplicity disregard region information, the model for rock compressibility is set up as follows:

```
function f = assignROCK(f, rock, reg)
  [cR, pRef]  = deal(rock(2),rock(1));
  f.cR = cR;
  f.pvMultR = @(p)(1 + cR.*(p-pRef));
end
```

Although rock compressibility, strictly speaking, is not a *fluid* relationship, it has for historic reasons been bundled with the saturation and pressure-dependent fluid and rock-fluid relationships since it represents a purely pressure-dependent quantity and will enter the flow equations in the same way as the other elements of the fluid object. Notice also that with the generic setup in `initDeckADIFluid`, all you must do to support a new keyword `MYKEY`, is to implement the function `assignMYKEY` and place it in the right directory of the `ad-props` module.

Assigning relative permeabilities is somewhat more involved. If we disregard code lines having to do with the region information (i.e., that the model can use different relative permeability functions in different sub-regions of the grid), the constructor takes the following form

```
function f = assignSWOF(f, swof, reg)
  f.krW   = @(sw, varargin) krW(sw, swof, reg, varargin{:});
  f.krOW  = @(so, varargin) krOW(so, swof, reg, varargin{:});
  f.pcOW  = @(sw, varargin) pcOW(sw, swof, reg, varargin{:});
  swcon   = cellfun(@(x)x(1,1), swof);
  f.sWcon = swcon(1);
end
```

That is, we add three function pointers to the fluid object implementing the relative permeability of water, the relative permeability from an oil–water displacement, and the capillary pressure of between oil and water. Let us take the krOW function as an example,

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

The first line sets up a map that lets us select the correct saturation functions if there are multiple regions. The next line extracts the first and the third column of the input data, i.e., the SWAT and KRO columns. Here, we use cellfun in combination with an anonymous function to extract individual elements in the same way as we used arrayfun to extract simulation results from the sol array of structures in the previous section. The saturation values in the relative permeability tables may not necessarily span the whole unit interval, and in the third line we add a constant extrapolation to ensure that we can use T to extrapolate for all legal saturation values. The last line calls an interpolation function that uses region information to pick the correct relative permeability function from the cell array and then calls an appropriate interpolation function. If T had been a simple array, the last line would have amounted to

```
  F = griddedInterpolant(T(:,1), T(:,2) , 'linear', 'linear');
  v = F(1-so);
```

The functions krW and pcOW are implemented exactly the same way, with the obvious modifications, and so are the functions in assignSGOF.

11.3.2 Evaluating Relative Permeabilities in MRST

By default, fluid objects constructed by initDeckADIFluid only contain functions for evaluating the tabulated two-phase systems found in the input file through functions krW, krOW, krG, and krOG. For convenience, the ad-props module offers the function assignRelPerm that implements the three-phase relative permeability model described in the beginning of the section. You can run assignRelPerm to instrument your fluid object with a generic interface to evaluate relative permeabilities. However, before this function is called, you must have run the necessary routines to assign data tables and functions for interpolating these data to the fluid object f, or alternatively have implemented your own fluid object that offers the same functions krW, krOW, etc. Notice also that if you later update any of the two-phase data in the fluid object, assignRelPerm must be run again to ensure correct evaluation.

The assignRelPerm function inspects the fields that are present in the fluid object and assigns the correct relative-permeability functions accordingly

```
function f = assignRelPerm(f)
  if ~isfield(f, 'krOG')      % two-phase water-oil
    f.relPerm = @(sw, varargin)relPermWO(sw, f, varargin{:});
  elseif ~isfield(f, 'krOW')  % two-phase oil-gas
    f.relPerm = @(sg, varargin)relPermOG(sg, f, varargin{:});
  else                        % three-phase
    f.relPerm = @(sw, sg, varargin)relPermWOG(sw, sg, f, varargin{:});
  end
end
```

The logic is quite simple: If there is no oil–gas data (`krOG`) for a three-phase system available, we must have a water–oil system. If there is no oil–water data (`krOW`) for a three-phase system available, we must have an oil–gas system. Otherwise, all three phases are present and we must set up a full three-phase model. The relative permeabilities for the two-phase water–oil system are set up as follows:

```
function [krW, krO] = relPermWO(sw, f, varargin)
  krW = f.krW(sw, varargin{:});
  if isfield(f, 'krO')
    krO = f.krO(1-sw, varargin{:});
  else
    krO = f.krOW(1-sw, varargin{:});
  end
end
```

Here, the default is to look for a two-phase table called `krO` for the oil relative permeability. If this is not present, the user must have supplied an oil–water table from a three-phase model, and we use this instead. The oil–gas system is treated analogously. Finally, for the three-phase model:

```
function [krW, krO, krG] = relPermWOG(sw, sg, f, varargin)
  swcon = f.sWcon;
  swcon = min(swcon, double(sw)-1e-5);

  d     = (sg+sw-swcon);
  ww    = (sw-swcon)./d;
  krW   = f.krW(sw, varargin{:});

  wg    = 1-ww;
  krG   = f.krG(sg, varargin{:});

  so    = 1-sw-sg;
  krow  = f.krOW(so, varargin{:});
  krog  = f.krOG(so,  varargin{:});
  krO  = wg.*krog + ww.*krow;
```

11.3.3 The SPE 1, SPE 3, and SPE 9 Benchmark Cases

In 1981, Aziz S. Odeh [241] launched an initiative for the independent comparison of the state-of-the-art reservoir simulators for 3D black-oil models. To represent a good cross-section of solution methods used throughout the industry, seven companies were selected, each participating with their own simulator. This type of comparison was well received by both the industry and the academic community. Over the next two decades, the Society of Petroleum Engineering (SPE) published a series of nine additional studies, which are today known as the ten SPE comparative solution projects. In each study, a number of oil companies, software providers, research institutes, universities, and consultants participated on a voluntarily basis, reporting their best attempt to solve a specified problem. Altogether, these ten problems constitute a rich source of test problems that can be used to verify and validate various multiphase models and solution strategies. You have already seen data from the tenth project used multiple times throughout this book.

In this example, we use input decks for the SPE 1 [241], SPE 3 [158], and SPE 9 [159] comparison projects to demonstrate how you can use MRST to set up and compare different fluid models. If you have not already done so, you can download the necessary files by use of MRST's data set manager:

```
mrstDatasetGUI;
```

This will place the data sets on a standard path, so that they can be read and processed using the following set of commands (with the obvious modifications for SPE 3 and SPE 9):

```
mrstModule add deckformat ad-core ad-props
pth  = getDatasetPath('spe1');
fn   = fullfile(pth, 'BENCH_SPE1.DATA');
deck = readEclipseDeck(fn);
deck = convertDeckUnits(deck);
f    = initDeckADIFluid(deck);
f    = assignRelPerm(f);
```

This produces a fluid object with a number of fields (here for SPE 1):

```
     krW: @(sw,varargin)krW(sw,swof,reg,varargin{:})
    krOW: @(so,varargin)krOW(so,swof,reg,varargin{:})
    pcOW: @(sw,varargin)pcOW(sw,swof,reg,varargin{:})
   sWcon: 0.1200
     krG: @(sg,varargin)ireg(Tkrg,sg,varargin{:})
    krOG: @(so,varargin)ireg(Tkro,sgas(so),varargin{:})
    pcOG: @(sg,varargin)ireg(Tpcog,sg,varargin{:})
        :
 relPerm: @(sw,sg,varargin)relPermWOG(sw,sg,f,varargin{:})
```

Here, we recognize relative permeability of water, oil relative permeability of an oil–water system, oil–water capillary pressure, connate water injection, etc. The last function is a

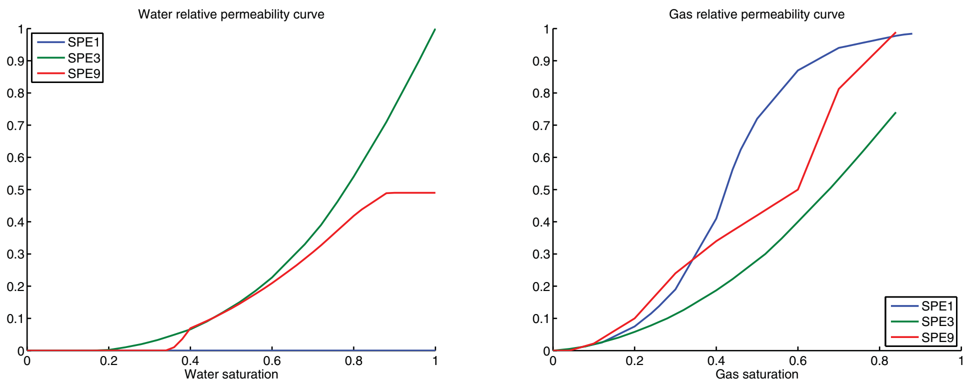


Figure 11.6 Relative permeabilities for water and gas for the fluid models from three of the SPE Comparative Solution Projects.

generic interface for evaluating relative permeabilities of all phases in the three-phase system. Let us look at the two-phase relative permeabilities `krW` and `krG`, which we evaluate by a call on the form

```
s   = linspace(0,1,51)';
krW = f.krW(s);
```

Figure 11.6 shows the resulting curves (extrapolated `krG` values are not plotted for saturations $S_w < S_{wc}$). SPE 1 is represented as a three-phase system in this particular input file, but it is really a two-phase gas–oil problem that describes gas injection into an undersaturated oil, i.e., an oil that can dissolve more gas. (We discuss this sort of PVT behavior in the next section.) The water relative permeability is therefore virtually zero over the whole saturation interval. For the SPE 3 model, both relative permeability curves have monotone derivatives similar to what we have seen in the analytic models discussed in Section 8.1.4. This is not the case for the gas relative permeability of SPE 1, which has an S-shape similar to the fractional flux functions we have seen in two-phase systems. For the SPE 9 model, the relative permeability curves have significant kinks, and this will contribute to increase the nonlinearity of the discretized flow equations.

Figure 11.7 shows the corresponding relative permeability curves for two-phase oil–water and gas–oil systems and the resulting interpolated three-phase oil relative permeability curve. The plots were generated as follows:

```
[sw, sg] = meshgrid(linspace(0,1,201));  so=1-sw-sg;
[~, krO, ~] = f.relPerm(sw(:),sg(:));
krO = reshape(krO,size(sw)); krO(sw+sg>1)=nan;

[mapx,mapy] = ternaryAxis('names',{'S_w';'S_g';'S_o'});
contourf(mapx(sw,sg,so), mapy(sw,sg,so), krO, 20)
```

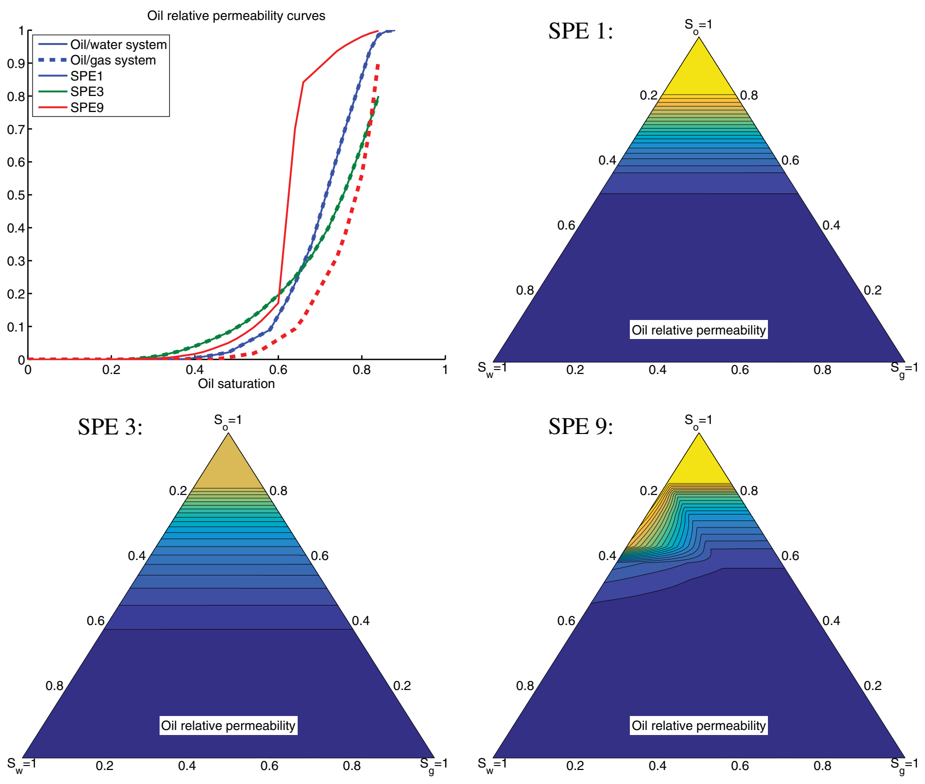


Figure 11.7 Three-phase relative permeabilities of oil for three of the SPE Comparative Solution Projects generated by linear interpolation of two-phase curves from oil–water and oil–gas displacements (upper left plot).

The `krOW` and `krOG` curves representing two-phase displacements coincide for both the SPE 1 and SPE 3 data sets, and hence the interpolated `krO` surface is smooth and monotone along straight lines in phase space. For SPE 9, the two-phase curves are not only significantly different, but the `krOW` curve has a very steep gradient. As a result, we observe kinks and it is also possible to identify straight lines in the phase diagram along which `krO` is non-monotone; see e.g.,

```
sw = linspace(0,.6,21); sg = linspace(0,.4,21);
[~,krO] = f.relPerm(sw',sg');
```

As is warned about in [270], abrupt changes in relative permeability values over small saturation intervals can be the source of significant convergence problems when solving the discretized flow equations and may cause the simulator to slow down. Like ECLIPSE, MRST honors the input data and does not attempt to overcome convergence problems by smoothing kinks or retabulating the input data.

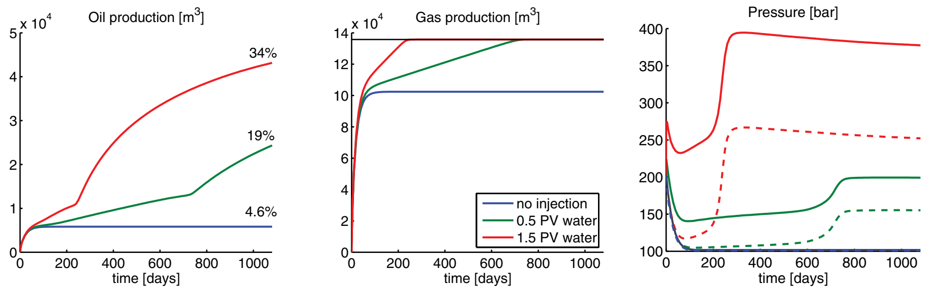


Figure 11.8 Conceptual three-phase simulation with no mass transfer between the phases. The reservoir is initially filled with gas on top, oil in the middle, and water at the bottom. Fluids are produced from the middle layer. The plots show results of simulations with no water injection and with injection of 0.5 and 1.5 pore volumes of water into the bottom layer. The plots to the right show average pressure in the injector cell (solid lines) and FPR (dashed lines).

11.3.4 A Simple Three-Phase Simulator

Having established how to model three-phase relative permeabilities, it is a simple exercise to extend the simple simulator from Section 11.2 to three phases under the assumption of no mass transfer between the phases. This assumption is not very realistic, but is introduced to isolate and illustrate certain aspects of multiphase flow. To exemplify, we consider a setup similar to the one discussed in Section 11.2, with the upper 1/3 initially filled with gas ($S_g = 1 - S_{wc}$), the middle 1/3 filled with oil ($S_o = 1 - S_{wc}$), and the bottom 1/3 is filled with water. Fluids are produced from a cell at the northeast corner of the middle section by setting the pressure in this cell to be 100 bar, which is approximately 100 bar lower than the initial average reservoir pressure. We change the fluid model so that water is twice as viscous as oil, and assume that relative permeabilities follow the SPE 3 fluid model.

When the producer starts depleting the reservoir pressure, the compressed gas will expand so that a large fraction of it is driven out of the reservoir through the oil layer below. Gas is more mobile than oil and will flow past the oil rather than displace it. Hence, only a minor fraction of the oil will be recovered. To increase oil recovery, we inject water into the water zone at the lower southwest corner. Code to simulate this case is given in `threePhaseAD.m` in the `book` module. We run a comparison of no water injection and injection of 0.5 and 1.5 pore volumes of water

```
pvi=0; threePhaseAD; pvi=.5; threePhaseAD; pvi=1.5; threePhaseAD
```

Figure 11.8 reports observed hydrocarbon production and pressure behavior. Without water injection, the reservoir pressure is quickly depleted and reaches a steady state after approximately 100 days, and out of the volumes initially in place, we recover 75%, but only 4.6% of the oil. By injecting 0.5 pore volumes of water, we are able to maintain reservoir pressure and hence get higher oil production in the period until all the gas is depleted and the injected water breaks through in the producer. This happens after approximately 740 days. Towards

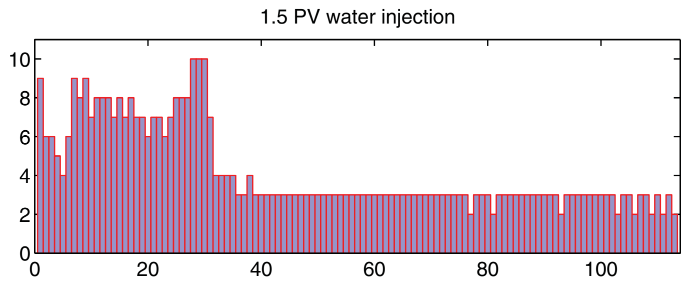


Figure 11.9 Number of nonlinear iterations for the conceptual three-phase simulation.

the end of this period, the pressure increases significantly as the water front approaches the producer and then stabilizes at a plateau. When gas disappears and oil is the most mobile phase, the oil production increases abruptly, but then decays slowly as a result of gradually increasing water cut in the producer. Similar behavior can be observed when injecting 1.5 pore volumes of water over the same time period. However, since injection rates are three times higher, water breakthrough occurs much earlier and we also observe pressures that are much higher than the initial field pressure.

If we try to repeat the simulation with a total injection of two pore volumes, the simulator fails to converge within 15 iterations in the 23rd time step, right before the water front breaks through in the producer. We also observe a similar increase in the number of iterations towards water breakthrough for the other water injection cases; see Figure 11.9. It is well known that Newton's method may experience convergence problems if saturations (or compositions) change too much during a single time step. We have already discussed this issue in Section 10.2.2, where we also introduced a reactive strategy for chopping of the time step.

Computer exercises

11.3.1 Visualize the structure of the linearized three-phase system in the same way as in Figure 11.1. Try to permute the system so that the numbering in the vector of unknowns first runs over variables and then over cells.

11.3.2 What happens if you set water to be incompressible?

11.3.3 Extend the simulator in `threePhaseAD.m` to include the reactive time-step control discussed in Section 10.2.2 and try to extend it so that it also uses changes in saturation as a criterion to chop time steps. Does this enable you to run a simulation with more than two pore volumes injected?

11.4 PVT Behavior of Petroleum Fluids

Our assumptions of no mass transfer between phases and that each fluid phase has constant compressibility, but otherwise constant fluid properties, represent gross oversimplifications.

Pressure and temperature will not only affect the compressibility and viscosity of each fluid phase, but also determine which phases are present and how the different chemical components in the reservoir fluids distribute among the aqueous, oleic, and gaseous phases at reservoir and surface conditions. Hydrocarbon fluids can come in a wide variety of forms, including the bitumen (semisolid rock form of crude oil) of the Athabasca oil province in Canada and heavy (dead) oils containing almost no light hydrocarbon components. We will not discuss such cases in any detail but instead focus on hydrocarbon fluids that have a significant gas component at surface conditions. That is, medium oils containing lighter components that are liberated as free gas at surface conditions; volatile oils that shrink significantly when liberating free gas; rich gases that condense liquid at reduced pressure; wet gases that condensate fluids at lower temperatures; and dry gases that mainly consist of light hydrocarbons and stay in single-phase vapor form.

11.4.1 Phase Diagrams

We have previously discussed equations of state modeling the relationship among pressure, volume (density), and temperature. For reservoir fluids it is more common to use *phase diagrams*, which in this setting are a type of chart showing pressure and temperature conditions at which thermodynamically distinct phases occur and coexist at equilibrium. This section discusses such diagrams for reservoir fluids.

Single-Component Substances

As a starting point, however, let us look at a typical phase diagram for a substance consisting of a single component. To explain such diagrams, we first look at *Gibbs' phase rule* for a system without chemical reactions. This rule states that the number of independent degrees of freedom F that must be specified to determine the intensive state of a system is given by

$$F = 2 + n_c - n_p,$$

where n_c is the number of components and n_p is the number of phases. For a single-component system ($n_c = 1$), we have $F = 3 - n_p$, and hence there can at most be three phases. A state with three phases corresponds to zero degrees of freedom and the phases must therefore coexist in a single point called the *triple point*, at which pressure and temperature are uniquely determined. If two phases coexist, there is one degree of freedom, and pressure can be chosen freely if temperature is fixed, and vice versa. When the substance is in a single-phase state, both pressure and temperature can be chosen independently.

Figure 11.10 illustrates the resulting phase diagram. In the diagram, the three single-phase regions corresponding to solid, liquid, or vapor phase are separated by the *sublimation*, *melting*, and *vaporization* curves. Along these curves, two phases coexist in equilibrium. Across the curves, various properties of the substance (compressibility, viscosity,

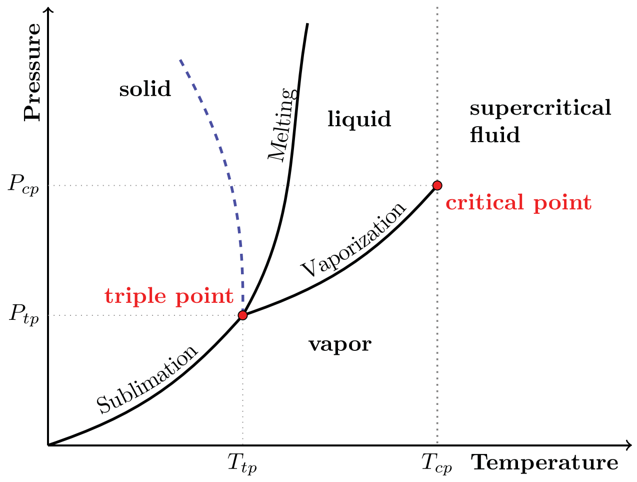


Figure 11.10 Conceptual phase diagram for a single-component substance. The dashed blue line illustrates the melting curve for water, which unlike most other fluids has negative slope.

etc.) change abruptly and have discontinuous derivatives with respect to temperature and pressure. At the triple point, all phases exist in equilibrium. For temperatures below the *critical point* T_{tp}, the substance is in vapor phase if the pressure is below the sublimation curve and in solid state for pressures above. For pressures and temperatures along the sublimation curve, vapor and solid are in equilibrium. For temperatures between the triple point and the critical point ($T_{tp} < T < T_{cp}$), the substance is in vapor state if the pressure is below the vaporization curve, in liquid state if the pressure is above the vaporization curve, and in solid state if the pressure is above the melting curve.

The liquid–vapor curve terminates at the critical point. The critical temperature T_{cp} is the temperature at, and above, which vapor of the substance cannot be liquefied, no matter how much pressure is applied. Critical temperatures depend on the size of the molecule and hence vary a lot for hydrocarbons, from low temperatures for small molecules to high temperatures for large molecules. For water, the critical temperature is 374°C, whereas carbon dioxide has a much lower critical temperature of 31.2°C. Likewise, the critical pressure P_{cp} is the pressure required to liquefy a gas at its critical temperature, or in other words, the pressure above which distinct liquid and gaseous phases do not exist. The critical pressure of water is 217.7 atm and for CO_2 it is 73.0 atm. For temperatures above T_{cp}, the substance is said to be in a *supercritical state*, i.e., in a state in which the liquid and gaseous phases are indistinguishable. This is the state in which CO_2 is usually found when stored in a deep geological formation (approximately below 800 m).

For most substances, the melting curve separating liquid and solid phase has a positive derivative with respect to temperature; the denser the molecules of the substance are brought together, the higher temperature is required to break them apart to form a liquid state. Water is one exception to this rule.

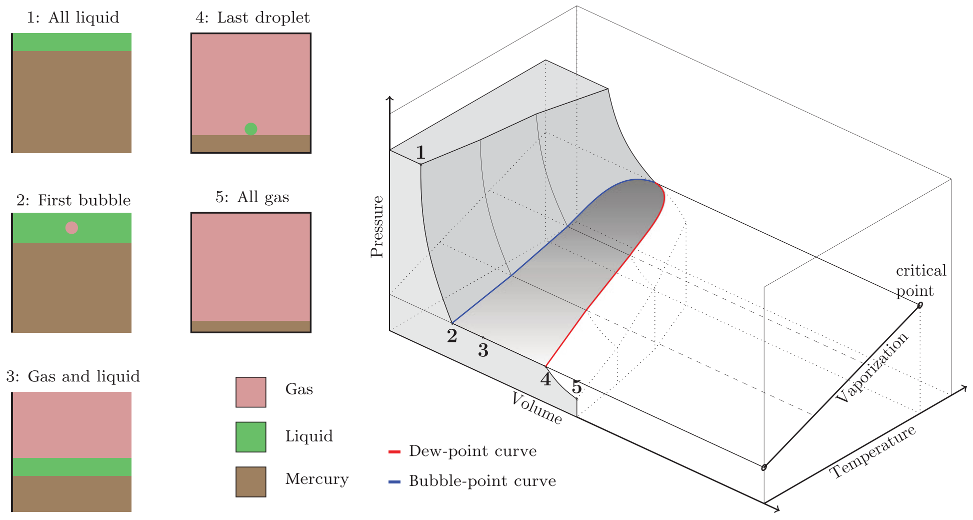


Figure 11.11 PVT behavior of a single-component hydrocarbon substance. The left figures illustrate measurement of pressure-volume behavior in a high-pressure cell at fixed temperature. Mercury is connected to a pump that can inject or extract this fluid to control pressure. Mercury has high surface tension and will hence be non-wetting for most container surfaces like glass. The right plot shows the measurements plotted in a PVT diagram, with the bubble-point and vaporization curves projected onto a pressure-temperature diagram.

For hydrocarbon recovery, we are primarily interested in the *vaporization curve*. To better understand the phase behavior for multicomponent systems, we first briefly explain an experimental setup for determining this curve using a *high-pressure* cell, as illustrated in Figure 11.11. Initially, the cell is filled with liquid hydrocarbon at the top and mercury at the bottom. As mercury is gradually removed, the liquid hydrocarbon expands and the pressure drops. This continues until the first bubble of gas comes out of the solution as we reach the *bubble point* (point 2 in the figure). Beyond this point, the gas phase expands the volume rapidly while keeping the pressure constant (point 3 in the figure). Eventually, we reach the , where the last liquid droplet vaporizes (point 4 in the figure). As more mercury is removed, the gas expands while the pressure decreases. If we repeat the experiments for multiple temperatures, we get the diagram shown in the right plot. By projecting the locus of bubble points, which we henceforth refer to as the *bubble-point curve*, and the locus of dew points, which is called the *dew-point curve*, we obtain the *vaporization curve* in the pressure-temperature diagram.

Binary Substances

Hydrocarbon reservoir fluids are made up of a wide range of chemical substances, and hence their phase diagrams will be very complicated. Gibbs' phase rule obviously gives us little guidance since the number of actual phases occuring in the reservoir is much less

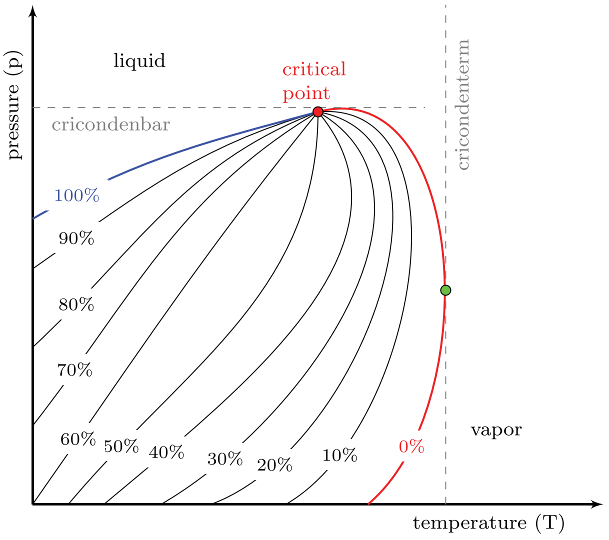


Figure 11.12 A pressure-temperature diagram for a binary system of fixed composition. Bubble-point curve is shown in blue and dew-point curve shown in red. The lines inside the two-phase envelope represent constant percentages of liquid. The diagram is specific to a given chemical composition of the binary mixture and will change if this composition changes, e.g., if components are extracted at different rates.

than the maximum number of phases that may exist. In black-oil models, the hydrocarbon fluids are considered to be a two-component system. (Notice that this distinction between oil and gas components is only relevant when the reservoir is in a state where both can coexist.) For binary substances, the number of degrees of freedom for a two-phase system is $F = 2$, and hence two phases could in principle exist for arbitrary combinations of temperature and pressure, but this is not the case in practice. To describe the reservoir fluids, we utilize a bubble-point/dew-point projection similar to the one we studied for the single-component fluid above. This projection is defined for a constant mole fraction of one of the components, and hence the phase diagram will depend on the composition of the binary mixture. In the projected plot, the bubble-point and dew-point curves no longer coincide along a single vaporization curve. Once the bubble point is reached, the expansion is accompanied by a pressure decrease, since the gaseous phase has a different chemical composition than the oleic phase. The projection hence produces a temperature and pressure envelope, inside which the binary mixture forms two phases as shown in Figure 11.12.

At pressures and temperatures above the bubble-point curve, the binary mixture forms a liquid phase. Likewise, a gaseous phase is formed for all points located below or to the right of the dew-point curve. The bubble-point and dew-point curves meet in the *critical point*, where the phase compositions and all phase properties are identical. Notice that the critical point does not necessary coincide with the maximum temperature of the two-phase envelope. The maximum temperature is called the *cricondentherm* and the maximum pressure is called the *cricondenbar*. The critical point represents a mathematical discontinuity in the system, and the phase behavior is generally difficult to define near this point, meaning that it is not obvious whether a (slight) pressure decline will cause the formation of bubbles of vapor or droplets of liquid.

11.4.2 Reservoir Types and Their Phase Behavior during Recovery

In petroleum literature, the phase diagrams discussed in the previous subsection are sometimes used to classify reservoirs according to the type of phase behavior that takes place during (primary) production. To motivate this classification, and the choice of physical variables and closure relationships to be discussed in the next section, we start by a schematic of a typical production setup for a depleting reservoir as shown in Figure 11.13.

When a well is opened to start production of hydrocarbons, the pressure inside the reservoir will start to decline. Reservoirs are thermally inert, and unless other fluids having significantly different temperature are injected into the reservoir, the temperature of the reservoir fluids will remain (almost) constant at the value given by the geothermal gradient of the area. Inside the reservoir, the extraction of hydrocarbons therefore takes place along a vertical line in the pressure-temperature diagram. Also when fluids are injected to displace the resident hydrocarbons, temperature effects are in many cases so small that the temperature can be considered as constant (near the producing well) in the reservoir model. This means that various types of reservoirs can be classified by the location of the

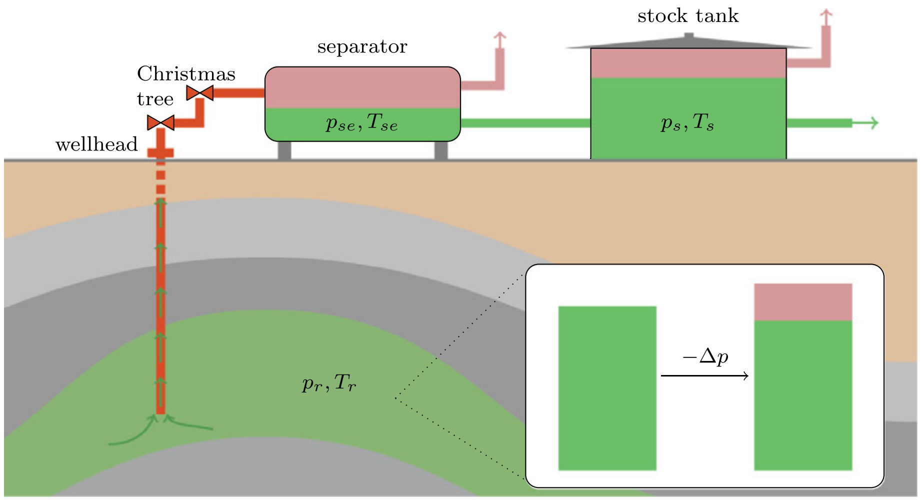


Figure 11.13 Illustration of pressure and temperature changes resulting in liberation and expansion of gas and shrinkage of oil during production of a hydrocarbon reservoir containing both oil and gas. Stock-tank conditions vary but will roughly be 1 atm and 20°C.

initial reservoir pressure and temperature relative to the bubble-point/dew-point envelope. If the temperature of the reservoir is less than the critical temperature ($T_r < T_c$) of the hydrocarbon mixture, the reservoir is said to be an *oil reservoir*. Conversely, if the temperature is larger than the critical temperature ($T_r > T_c$), the reservoir is said to be a *gas reservoir*. In both cases, the reservoir may contain hydrocarbons in both liquid and vapor form, depending on the type of the reservoir.

As the hydrocarbons leave the reservoir and flow to the surface, they undergo a series of changes that reduce the pressure and temperature to surface conditions. These changes generally cause liberation of gas, expansion of the gaseous phase, and shrinkage of the liquid oleic phase. Pressure reductions come in two types: in *flash liberation*, gas comes out of solution as a result of a sudden pressure drop, and then stays confined with the remaining oil. In *differential liberation*, gas is liberated from the oil as pressure is gradually decreased and is continuously removed from the oil. How the reservoir fluids contained in the well stream are brought from reservoir pressure and temperature to surface condition can have a large effect on how much oil and gas is recovered from a given quantity of reservoir fluids. Traditionally, oil has had much higher economic value than gas, which often has been considered as a waste product. It has therefore been highly important to design a production system that maximizes the stock-tank oil yield.

There are many mechanisms that induce pressure reduction: friction and gravity forces acting in the opposite direction of the flow, loss of kinetic energy due to expansion, contraction in the fluid area, flow through constrictions, etc. At the surface, the fluids pass through the *Christmas tree* mounted on top of the well head, whose main purpose is to control the flow of reservoir fluids out of the well. The Christmas tree has isolation valves to stop the fluid flow for maintenance or safety purposes, and flow lines and chokes that control the flow into the downstream production system. Inside the production system, the fluids can continue into one or more pressure vessels (separators) used to separate the produced fluids into oil, gas, and water components, and possibly remove gas and/or water. The oil component may continue to a stock tank for temporary storage, inside which further gas liberation may take place.

We already know that the pressure drop inside the reservoir and the pressure drop from the reservoir to the wellbore both depend on flow rate; this follows from Darcy's law and the Peaceman well model, respectively. Similar relationships between pressure drop and flow rate can be developed for the other parts of the flow system from the bottom hole to the stock tank; that is, pressure loss in the tubing, in the Christmas tree and flow lines, across the choke, and in the separator units. A few examples are presented in Section 11.7.2. These relationships are used by production engineers to optimize the hydrocarbon recovery. For a more detailed discussion, you can consult a textbook on so-called *nodal analysis* [145].

Gas Reservoirs

All reservoirs in which the initial temperature is higher than the critical temperature are called gas reservoirs. Methane is by far the dominant hydrocarbon component in gas reservoirs. If the initial temperature is higher than the cricondentherm, the hydrocarbon mixture

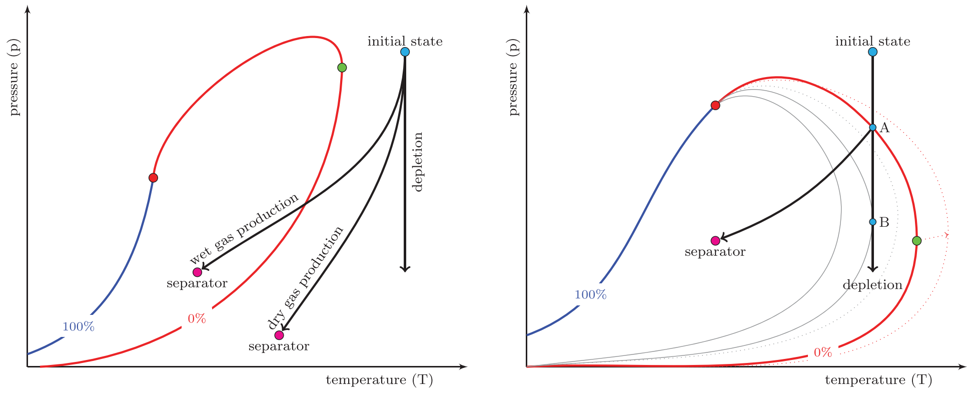


Figure 11.14 Phase diagrams for gas reservoirs. In a single-phase dry/wet gas reservoir (left), the temperature is above the cricondentherm. In a retrograde gas condensate reservoir (right), the temperature is above the critical temperature, but below the cricondentherm. During retrograde depletion below the dew-point curve, the composition of the mixture will change inducing a change in the diagram indicated by the dotted lines.

will be in a single-phase gaseous state and will continue to be so during depletion. As the reservoir is depleted, the composition of the produced fluids do not change, but the hydrocarbon mixture may enter the two-phase envelope on its way to the surface, depending upon the cooling process, as illustrated in the left plot of Figure 11.14.

If a gas reservoir produces liquid hydrocarbons at the surface (condensate), the reservoir is said to be a *condensate gas*, *wet gas*, or *rich gas* reservoir. Conversely, if the temperature–pressure path followed during production does not penetrate the two-phase envelope, all hydrocarbons produced at the surface will be in gas phase and the reservoir is said to be a *dry gas* reservoir. Dry gas reservoirs rarely contain hydrocarbons with carbon number five or higher and typically consist of C_1, C_2, N_2, CO_2, and H_2S. No hydrocarbon separation is needed at the surface since the well stream consists of gas only, but separators/dehydrators may be needed to remove produced/condensed water. Although wet-gas reservoirs mainly consist of methane, they also contain both intermediate hydrocarbons (ethane to hexane) as well as a significant fraction of hydrocarbons with carbon number seven or higher. The condensation of liquid hydrocarbons at the surface is a result of cooling, but whether the well stream is brought to surface conditions in one big flash or through a series of smaller flashes does not have a large effect on the relative amounts of gas and liquid observed at the surface.

Unlike dry/wet gas reservoirs, *dew-point* or *retrograde gas condensate* reservoirs contain a significant fraction of heavier hydrocarbon components (C_{7+}). The right plot in Figure 11.14 shows such a hydrocarbon mixture, which is initially in (an indeterminate) vapor state and will continue to be so as the reservoir is depleted toward point A on the dew-point curve. During this initial decline, the composition of the produced fluid remains constant. Below point A, a liquid oleic phase containing heavier hydrocarbons will condense out of the vapor as a dew or fog and continue to do so until the process reaches point B of maximum liquid percentage. This process is called retrograde, since one normally would see vaporization and not condensation when pressure decays.

The condensed liquid is immobile at low saturations and generally less mobile than the vapor phase. Hence, more light hydrocarbons will be recovered from the reservoir, giving an increased gas–oil ratio at the surface. This changes the composition of the hydrocarbon mixture, increasing the fraction of heavier hydrocarbons, which in turn shifts the two-phase envelope toward the right, as indicated by the dotted lines in the figure. This aggravates the loss of heavier hydrocarbons, which are retained as immobile droplets inside the rock pores. If production is continued, the system will sooner or later reach a point of maximum liquid percentage, below which the condensed liquid will start to vaporize. This improves the mobility of the liquid components and leads to a decrease in the surface gas–oil ratio.

For a dew-point reservoir, the exact design of how the reservoir fluids are brought to surface conditions can have a large impact on the amount of stock-tank oil and gas. This may call for a complex network of separators in which separated vapor or liquid is reinjected into the liquid–vapor mixture to better distribute heavier hydrocarbon components to the stock-tank oil and lighter components to the final gas stream.

In closing, we remark that the distinction between condensate and oil is somewhat artificial and has more to do with how the liquid hydrocarbons are recovered than their chemical composition. Liquid hydrocarbons in the stock tank are called oil if they originate from a liquid oleic state in the reservoir and condensate when they have been condensed from a gaseous state.

Bubble-Point Oil Reservoirs

A reservoir having initial temperature below the cricondentherm is characterized as an oil reservoir. Unlike the dead oil considered in Section 11.2, which either contained a constant amount of dissolved gas or had lost all its components, the oleic phase in bubble-point reservoirs consists of a *live oil* containing dissolved gas, which may be released at lower pressures inside the reservoir and at the surface. This type of reservoir is also called a *solution-gas* reservoir, and will be the topic for most of the simulation examples discussed later in the chapter. Although somewhat artificial and imprecise, it is common to subdivide oil reservoirs into two categories: *black-oil* reservoirs are mainly made up of a variety of hydrocarbons, including heavy and non-volatile components (C_{7+}). *Volatile* reservoirs contain fewer heavy components and have more intermediate hydrocarbon molecules (ethane to hexane) and the oleic phase therefore tends to shrink dramatically when gas is liberated. Black-oil reservoirs tend to have much less shrinkage. The main property that distinguishes the two categories, however, is the presence of vaporized oil in the gaseous phase, which is assumed to be negligible in black-oil reservoirs and significant in volatile reservoirs. This might be a bit confusing since we later will use *black-oil equations* to model both types of reservoirs. The reason is most likely historic. Traditionally, the largest and most profitable reservoirs in the world have predominantly been of black-oil type, and the black-oil equations were developed to model these. Volatile reservoirs are typically found at

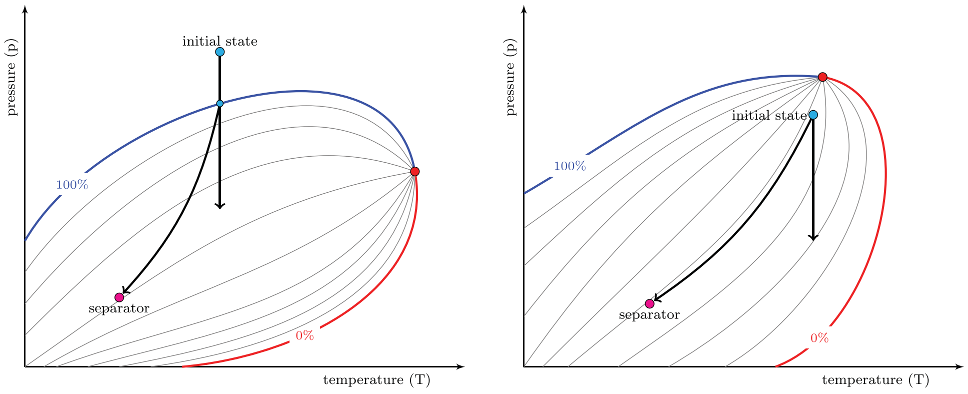


Figure 11.15 Phase diagram for bubble-point oil reservoirs. The left plot shows a black-oil reservoir with undersaturated initial oil and the right plot a volatile oil reservoir.

greater depths, and as exploration goes deeper, more reservoirs of this type have been discovered and entered production. Volatile reservoirs contain less oil per volume, but since the reservoir fluids are more mobile because of lower oil viscosity and higher dissolved-gas content, this type of reservoir can still be more economically attractive than a black-oil reservoir.

If the pressure is above the bubble-point curve as shown in the left plot of Figure 11.15, the hydrocarbon mixture of the oil reservoir is in a liquid state usually referred to as *undersaturated*, since all the available gas has been dissolved in the oleic phase. However, when the first liquid is brought to the surface, it liberates free gas. As production continues, the pressure will decline toward the bubble-point curve. Below the bubble point, gas will be released from the oleic phase as vapor bubbles that gradually form a free gaseous phase. Because the oil cannot hold all the available gas, we say that it is in a *saturated* state. When the gaseous phase exceeds its residual saturation, it will start to flow toward the producer. At the surface, the gas thus consists of both expanded free gas and gas liberated from the oil during production. Oftentimes, surface facilities have to limit the gas rate, which in turn can reduce the oil rate below an economic level unless secondary mechanisms like water injection are applied to maintain the reservoir pressure and push the oleic phase toward the producer (as we saw for the dead oil system in Section 11.3).

The volatile reservoir shown to the right in Figure 11.15 is initially in an saturated state inside the two-phase envelope. The reservoir fluids thus consist of a liquid oleic phase which exists in equilibrium with a gaseous phase containing gas condensate. Because the gaseous phase has a different composition and is lighter than the oleic phase, the gaseous phase will form a gas zone or gas cap overlying the liquid oil zone and has a phase diagram that is largely different from that of the oleic phase. Equilibrium between the two phases implies that the oleic phase must be at its bubble point, whereas the gaseous phase must be at its dew point and may either exhibit retrograde or non-retrograde behavior, as illustrated

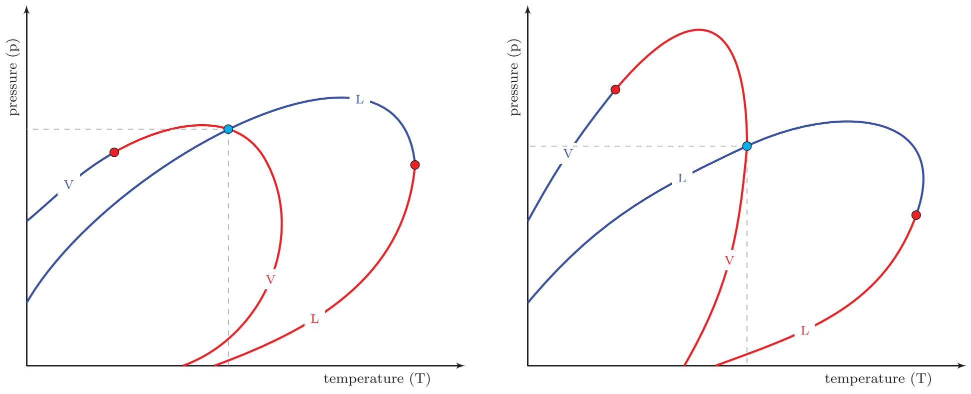


Figure 11.16 Phase diagrams for a reservoir at saturated conditions so that liquid oil is in equilibrium with a gas cap containing vaporized oil. The gaseous phase will exhibit retrograde behavior in the diagram to the left and non-retrograde behavior in the diagram to the right.

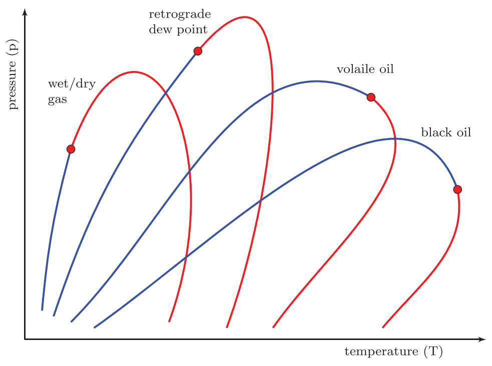


Figure 11.17 Sketch of phase diagrams for different types of hydrocarbon reservoirs.

in Figure 11.16. Notice also that volatile reservoirs are not very different from retrograde gas condensate reservoirs for temperatures close to the critical temperature.

We end our discussion about reservoir classification by presenting two comparisons of the different reservoir types. Figure 11.17 shows a schematic with the different phase diagrams plotted relative to each other. This sketch is by no means accurate, but illustrates how the phase diagrams move to the right when the hydrocarbon mixture contains (a larger fraction of) heavier hydrocarbon components. Table 11.1 reports a number of characteristic properties for different reservoir types.

Table 11.1 *Petroleum fluids and their characteristics at initial conditions or bubble-point pressure (adapted from [177]).*

	Oils			Gases	
Characteristic	Heavy oils and tars	Black oils	Volatile oils	Gas condensates	Wet and dry gases
Mol. weight	210+	70–210	40–70	23–40	<23
Oil color in stock tank	Black	Brown to light green	Greenish to orange	Orange to clear	Clear
Stock-tank °API	5–15	15–45	42–55	45–60	45+
C_7+ fraction (mol%)	>50	35–50	10–30	1–6	0–1
Reservoir temperature °F	90–200	100–200	150–300	150–300	150–300
Saturation pressure [psia]	0–500	300–5,000	3,000–7,500	1,500–9,000	—
Initial B_o [RB/STB]	1.0–1.1	1.1–1.5	1.5–3.0	3.0–20.0	20+
Initial R_s [scf/STB]	0–200	200–900	900–3,500	3,500–30,000	30,000+
Initial R_v [STB/MMscf†]	0	0–10	10–200	50–300	0–50
Max vol% liquid‡	100	100	100	0–45	0

† At bubble-point pressure.
‡ At constant composition expansion of reservoir fluid.

11.4.3 PVT and Fluid Properties in Black-Oil Models

The phase diagrams discussed in the previous section are useful to classify reservoir fluids and visualize the pressure-temperature path from reservoir to surface conditions. They aid the reservoir engineer in deciding the depletion process, designing top-side processing facilities (separators, etc.), and devising appropriate strategies for improved oil recovery (water injection or reinjection of gas to maintain pressure, etc.). Phase diagrams are also useful when setting up a fluid sampling program to determine parameters describing PVT behavior in flow models.

As we have seen already in Section 8.2.3 on page 246, the fluid and PVT description in a black-oil model tries to mimic how the gas and oil pseudo-components at surface conditions partition into the gaseous and oleic phases at reservoir conditions, as illustrated in Figure 11.13 and discussed in detail in the previous section. To this end, the composition and density of the oil and gas components are defined at surface conditions (where they easily can be measured), and then the following three parameters are used to determine the density of the oleic and gaseous phases at reservoir conditions:

B_o – The oil formation-volume factor is defined as the volume of oil in liquid phase that must be recovered from the reservoir to produce one volume unit of stock-tank oil; in other word, $B_o = V_o/V_{os}$. A reservoir volume V with porosity ϕ and oil saturation S_o thus contains $\phi V S_o/B_o$ stock-tank volumes of oil. In SI units, the formation-volume factor is a dimensionless quantity, but in oilfield units it is usually reported in units reservoir barrels per stock-tank barrels [bbl/STB], where 1 bbl equals 0.1589873 m^3. Some authors prefer the reciprocal quantity, $b_o = 1/B_o$, i.e., the factor by which one reservoir volume of oil shrinks (or expands) when brought to surface conditions. MRST uses shrinkage factors rater than formation-volume factors.

B_g – The gas formation-volume factor is defined similarly, as the volume of gaseous phase that must be recovered from the reservoir to produce a unit volume of gas at the surface, i.e., $B_g = V_g/V_{gs}$. It is common to express B_g in units reservoir barrels per standard cubic feet [bbl/scf]. MRST uses the reciprocal expansion factor $b_g = 1/B_g$.

R_s – The solution gas–oil ratio is the volume of gas at standard conditions that dissolves into a unit stock-tank volume of oil at reservoir conditions and is usually reported in units standard cubic feet per stock-tank barrel [scf/STB]. A reservoir volume V with porosity ϕ and oil saturation S_o thus contains $\phi V S_o R_s/B_o$ surface volumes of liberated gas.

This description assumes that the reservoir is in instantaneous phase equilibrium. In addition to these three parameters, we also need the bubble point p_b of the oil, a similar formation-volume factor for water, as well as the dynamic viscosities of the gaseous, oleic, and aqueous phase as function of pressure (and temperature). To model volatile-oil, retrograde dew-point, and wet-gas reservoirs, one also need to model the amount of vaporized oil present in the gaseous phase:

R_v – The vaporized oil–gas ratio (or volatile oil–gas ratio) is defined as the stock-tank volume of condensate oil per unit volume of free gas at surface conditions produced from the gaseous phase at reservoir conditions.

The formation-volume factors and gas–oil and oil–gas ratios all depend on reservoir pressure and temperature and are determined by laboratory experiments (or simulations) reproducing the sequence of differential (and flash) liberations that most likely take place during production of the reservoir. Since reservoirs are almost thermally inert, it is usually sufficient to express these quantities as function of pressure and not temperature.

In the following, we discuss various fluid properties and relationships in some more detail, and in most cases try to mention what are plausible ranges of their values. Readers interested in a more in-depth discussion should consult a textbook on reservoir fluids like e.g., [207, 316, 177] or www.petrowiki.org.

Equation of State: Fluid Densities

Using the quantities defined earlier, we can write the following equations of state that relate the densities of the aqueous, oleic, and gaseous phases (denoted by capital letters W, O, G) at reservoir conditions to the densities of the oil, water, and gas pseudo-components (denoted by o, w, g) at surface conditions

$$\rho_W = \frac{\rho_{ws}}{B_w} = b_w \rho_{ws} \tag{11.3}$$

$$\rho_O = \frac{\rho_{os} + R_s \rho_{gs}}{B_o} = b_o(\rho_{os} + R_s \rho_{gs}) \tag{11.4}$$

$$\rho_G = \frac{\rho_{gs} + R_v \rho_{os}}{B_g} = b_g(\rho_{gs} + R_v \rho_{os}) \tag{11.5}$$

Water density is almost invariant to pressure changes. The oleic phase behaves very differently above and below the bubble point. Above the bubble point, it behaves like a standard fluid in the sense that the density increases with increasing pressure. Below the bubble point, gas dissolution will in most cases dominate compression effects, so that density decreases with increasing pressure as a result of dissolved gas that increases the fraction of lighter hydrocarbon components. The gaseous phase is strongly affected by fluid compression so that density increases with pressure.

API and Specific Gravity

In our discussion of phase behavior, we repeatedly talked about light and heavy hydrocarbon components in the mixture. In petroleum literature, it is common to use the American Petroleum Institute gravity as a measure of how heavy or light a petroleum liquid is. The API gravity is defined as

$$\gamma_{API} = \frac{141.5}{\gamma_\ell} - 131.5, \qquad \gamma_\ell = \frac{\rho_\ell}{\rho_w}, \tag{11.6}$$

where γ_ℓ is the specific gravity of liquid ℓ to water, i.e., the ratio of the density of liquid to the density of water. Bubble-point pressure p_b, oil viscosity μ_o, and the solution gas–oil ratio R_s, all depend strongly on oil gravity. It also has an indirect effect on oil compressibility and oil formation-volume factors, since these depend on R_s.

The API gravity is specified as degrees on a hydrometer (°API) and is defined so that any liquid having API gravity greater than 10° is lighter and floats on water, whereas liquids with values less than 10° are heavier and sink. Using this scale, crude oil is commonly classified as follows:

- light oil: API gravity higher than 31.1°, i.e., less than 870 kg/m^3
- medium oil: API gravity between 22.3 and 31.1°, i.e., 870–920 kg/m^3
- heavy oil: API gravity below 22.3°, i.e., 920–1,000 kg/m^3
- extra heavy oil: API gravity below 10.0°, i.e., greater than 1,000 kg/m^3

Typical API gravity values are also reported in the overview of fluid characteristics for different reservoirs in Table 11.1.

For gas, the specific gravity is measured relative to air and is defined as

$$\gamma_g = \frac{\rho_g}{\rho_{air}} = \frac{M_g}{M_{air}},$$

where M denotes the molar mass and M_{air}= 28.94 kg/kmol. Specific gas gravity is one of the parameters that determine bubble-point pressure, viscosity, and compressibility of gas, as well as the solution gas–oil ratio. Typical values range from 0.55 for dry gas to approximately 1.5 for wet sour gas. Petroleum gases have gravity approximately equal 0.65.

Reservoir water is saline and hence has a specific gravity greater than unity, but has no impact on compressibility, formation-volume factor, and viscosity.

Compressibility and Formation-Volume Factor of Dry Gas

You may recall from Section 4.2 that the equation of state of an ideal gas is given as $pV = nRT$, where n is the number of moles of gas and R is the universal gas constant. The literature is full of equations of state for real gases, but in petroleum engineering it is common to use a so-called *compressibility equation of state* to describe dry gas

$$pV = ZnRT, \qquad Z = \frac{V_{\text{real}}}{V_{\text{ideal}}}. \tag{11.7}$$

The *compressibility factor* or *gas deviation factor* Z measures how much the real gas volume deviates from that of an ideal gas at the same temperature. The gas compressibility factor typically varies between 0.8 and 1.2. For constant temperature and moderate pressures, the gas molecules are close enough to exert an attraction on each other that causes the actual gas volume to be less than that of an ideal gas. As p approaches 0, the molecules move more apart and the gas behaves like an ideal gas. At higher pressures, repulsive forces will tend to force modules more apart and the real gas therefore has a higher volume than an ideal gas. The same type of equation of state can be used also for wet gas and mixtures if Z is replaced by a pseudo function that depends on the composition of the mixture. For volatile oils and retrograde-condensate gases it is more common to use a more advanced equation of state like one of the cubic equations discussed in Section 4.2. For a dry gas, the formation-volume factor can be derived from (11.7),

$$B_g = \frac{p_s}{Z_s T_s}\frac{ZT}{p}, \tag{11.8}$$

where subscript s refers to standard conditions. This varies throughout the world, and can be e.g., 14.65 psia and 60°F or 1 atm and 25°C.

The isothermal compressibility of the gas is defined as the fractional change of volume with respect to pressure for constant temperature,

$$c_g = -\frac{1}{V}\left(\frac{\partial V}{\partial p}\right)_T = -\frac{1}{B_g}\left(\frac{\partial B_g}{\partial p}\right)_T. \tag{11.9}$$

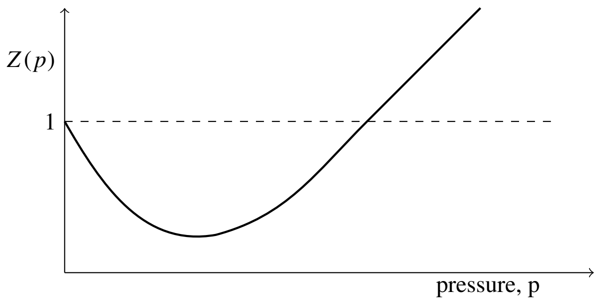


Figure 11.18 Characteristic shape of gas compressibility factor Z as function of pressure for fixed temperature.

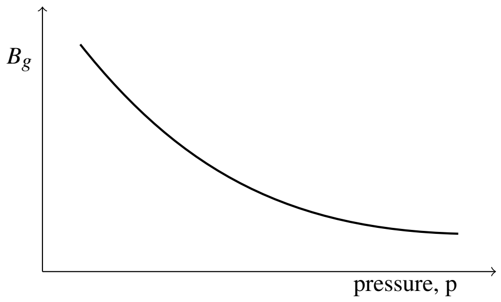


Figure 11.19 Characteristic shape of gas formation-volume factor as function of pressure for fixed temperature and chemical composition.

Inserting the compressible equation of state, it follows that

$$c_g = -\frac{1}{V}\left[\frac{nRT}{p}\left(\frac{\partial Z}{\partial p}\right)_T - \frac{ZnRT}{p^2}\right] = \frac{1}{p} - \frac{1}{Z}\left(\frac{\partial Z}{\partial p}\right)_T.$$

At low pressures, $\partial Z/\partial p$ is negative and hence the compressibility of a real gas is larger than that of an ideal gas. At high pressures, $\partial Z/\partial p$ is positive and hence the compressibility is lower than for an ideal gas.

Gas Viscosity

Dynamic gas viscosities typically range from 0.01 to 0.03 cP and are generally difficult to determine experimentally. It is therefore common to use correlations to determine values. At low pressures, gas molecules are far apart and can easily move past each other. As pressure increases the molecules are forced closer to each other and hence the viscosity of the gas increases. In MRST, gas viscosities are either given as simple explicit formulas (as seen so far) and correlations [316], or as tabulated input read from an ECLIPSE simulation deck (as we will see later).

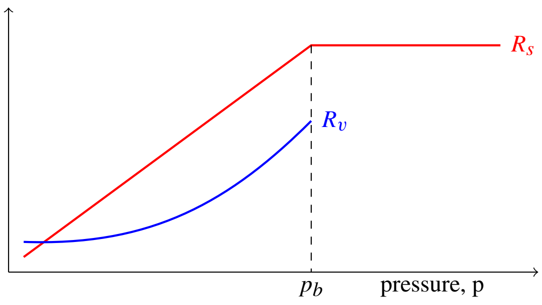


Figure 11.20 Characteristic shape of the solution gas–oil ratio and vaporized oil–gas ratio as functions of pressure for fixed temperature and chemical composition.

Bubble-Point Pressure

Bubble-point pressure is one of the key parameters in the phase behavior of reservoir fluids since it marks the limit between the liquid and two-phase liquid–vapor region for hydrocarbon fluids. In the previous section we outlined briefly how bubble-point pressure can be measured when studying fluid expansion as function of pressure and temperature. The literature is full of correlations that can be used to determine the bubble-point pressure all on the form $p_b = f(T, \gamma_{API}, \gamma_g, R_s)$; see e.g., [177] or www.petrowiki.org for an overview. None of these are implemented in MRST. Instead, the software follows the convention from leading commercial simulators and requires tabulated B and R_s factors, in which p_b is only given implicitly.

Solution Gas–Oil Ratio and Vaporized Oil–Gas Ratio

When discussing dissolved gas, we are in reality referring to light hydrocarbon molecules that appear in gaseous phase at the surface and in liquid oleic phase at reservoir conditions. For a given temperature, the quantity of light molecules present in the oleic phase is limited only by the pressure and the number of light molecules present. For pressures below the bubble point, the oleic phase will release some free gas if the pressure is lowered slightly, and hence we say that the phase is *saturated*. Above the bubble point, the oleic phase is able to dissolve more gas than what is available and is said to be *undersaturated*. Altogether, this means that a plot of the quantity of dissolved gas in the oleic phase as function of pressure increases linearly up to the bubble point and then remains constant; see Figure 11.20. The vaporized oil–gas ratio is only defined for pressures below the bubble point.

Oil Formation-Volume Factor

Most measures of fluid production has traditionally taken place at the surface. It has therefore been customary to introduce the formation volume-factor to relate volumes measured at surface conditions to volumes at reservoir conditions. The oil formation-volume factor is affected by *three* mechanisms: The most important is the liberation of gas when bringing liquid oil from the reservoir to the surface, which will cause a decrease in the liquid volume,

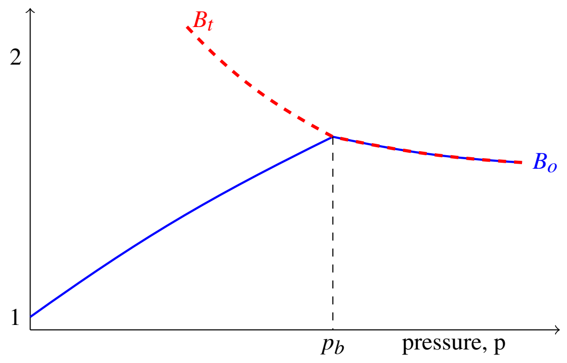


Figure 11.21 Characteristic shape of the oil and total formation-volume factor as functions of pressure for fixed temperature and chemical composition.

so that the B_o factor always assumes values greater or equal 1; the more gas that is dissolved, the larger the B_o factor. At reservoir conditions, increased pressure introduces two additional effects: fluid compression and swelling caused by dissolution of gas. The latter is dominant and below the bubble point, the formation-volume factor therefore increases with increasing pressures. Above the bubble point, there is no more gas to be dissolved, and hence fluid compression causes a gradual decrease in B_o; see Figure 11.21.

The compressibility of oil is defined in the same way as for gas. For undersaturated oil, we therefore have that

$$c_o = -\frac{1}{B_o}\frac{\partial B_o}{\partial p}, \qquad B_o = B_{ob}\exp\big(c_o(p_b - p)\big), \quad p > p_b, \tag{11.10}$$

where B_{ob} is the oil formation-volume factor at the bubble point.

Total Formation-Volume Factor

To better understand the behavior for saturated oil, we introduce the total or two-phase formation-volume factor, defined as the ratio between the total hydrocarbon volume at reservoir conditions and the volume of oil or gas at standard conditions. For simplicity, let us first consider the case of a black oil with a surface oil volume V_o, i.e., a system with dissolved gas but no vaporized oil. At the bubble point, the reservoir fluids inside the reservoir are found as a single oleic phase having a volume $V_o B_{ob}$ and containing $V_o R_{sb}$ volumes of dissolved oil. If we lower the pressure slightly, the system will enter the two-phase region, so that the oleic phase liberates an amount of gas that at the surface would amount to a volume $V_o(R_{sb} - R_s)$. At reservoir conditions, this corresponds to a volume $B_g(R_{sb} - R_s)$ of free gas. Hence, the total formation-volume factor with respect to oil reads,

$$B_{to} = B_o + B_g\big(R_{sb} - R_s\big).$$

Because of the expanding gas, the B_{to} function can have significantly larger values at lower pressures than at the bubble point.

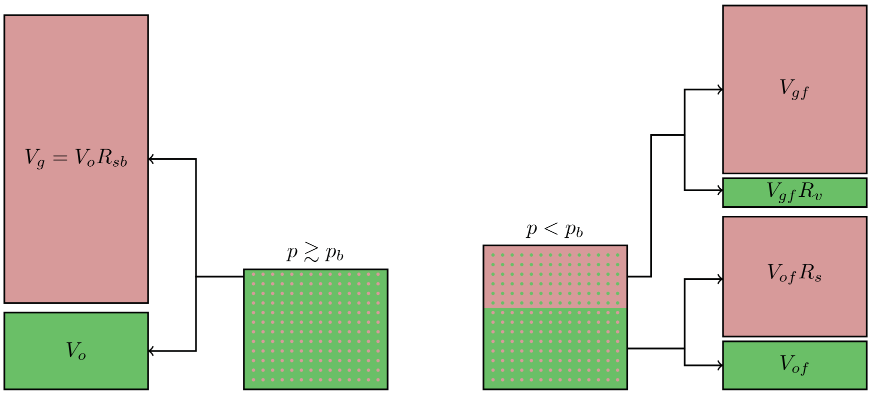


Figure 11.22 Illustration of the volumes involved in a volatile oil model.

For a retrograde gas condensate reservoir, we can start at the dew point and carry out a similar argument to obtain

$$B_{tg} = B_g + B_o\big(R_{vd} - R_v\big),$$

where R_{vd} is the vaporized oil–gas ratio at the dew point.

For volatile oils, things get a bit more complicated as Figure 11.22 illustrates. If no water is present, the pore volume V_ϕ is filled by the oleic and gaseous phase that together contain an amount V_o of oil and V_g of gas. Let V_{of} be the stock-tank volume of free oil and V_{gf} the surface volume of free gas. Then, we have

$$V_\phi = V_{of} B_o + V_{gf} B_g.$$

For pressures below the bubble point, oil and gas will appear in both the oleic and gaseous phase, and hence $V_o = V_{of} + V_{gf} R_v$ and $V_g = V_{gf} + V_{of} R_s$. Combining these two and solving for V_{gf} gives $V_{gf} = (V_g - V_o R_s)/(1 - R_v R_s)$. Inserting into the equation above and simplifying, we have

$$V_\phi = \frac{\big(B_o - R_s B_g\big)V_o + \big(B_g - R_v B_o\big)V_g}{1 - R_s R_v}. \tag{11.11}$$

Just above the bubble point, the hydrocarbon is in liquid phase and hence all gas is resolved in the oil, i.e., $V_g = V_o R_{sb}$. By definition, $V_\phi = B_{to} V_o$. Inserting into the above equation, dividing by V_o, and reorganizing terms, we obtain the following expression for the two-phase formation-volume factor

$$B_{to} = \begin{cases} B_o, & p \geq p_b, \\ \frac{B_o\left(1 - R_{sb} R_v\right) + B_g\left(R_{sb} - R_s\right)}{1 - R_v R_s}, & p < p_b. \end{cases} \tag{11.12}$$

The two-phase formation-volume factor for retrograde gas condensate can be derived similarly as

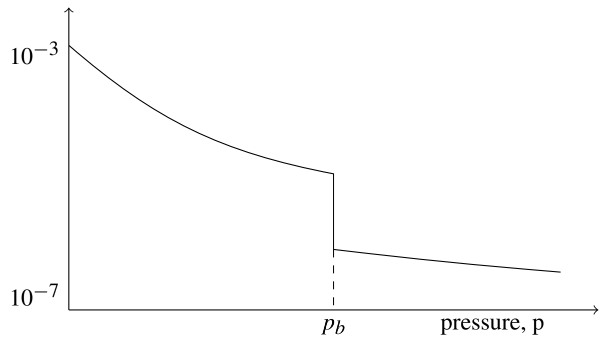


Figure 11.23 Isothermal oil compressibility as function of pressure for fixed temperature and chemical composition.

$$B_{tg} = \begin{cases} B_g, & p \geq p_d, \\ \frac{B_g\left(1-R_{vd}R_s\right)+B_o\left(R_{vd}-R_v\right)}{1-R_vR_s}, & p < p_d. \end{cases} \quad (11.13)$$

Isothermal Oil Compressibility

Isothermal oil compressibility is defined as the relative change in oil volume per change in pressure. It follows from the earlier discussion that oil compressibility has different behavior above and below the bubble point. For an undersaturated oleic phase above the bubble point, the compressibility is defined in the same way as for a gas, i.e., $c_o = (-1/B_o)(\partial B_o/\partial p)_T$, and is a primary drive mechanism. At the bubble point, gas comes out of solution and causes a sharp increase in compressibility. Below the bubble point, the compressibility becomes a much stronger function of pressure as the oil volume now depends on expansion/compression of both liquid oil and dissolved gas

$$c_o = -\frac{1}{B_o}\left(\frac{\partial B_o}{\partial p}\right)_T + \frac{1}{B_o}\frac{B_g - B_oR_v}{1 - R_sR_v}\left(\frac{\partial R_s}{\partial p}\right)_T. \quad (11.14)$$

As a drive mechanism, however, oil compressibility is dominated by the compressibility of gas that has come out of solution.

Oil Viscosity

We have already seen in Section 7.3 that the viscosity of a fluid is affected by both temperature and pressure, with temperature being the most dominant for a fluid of constant composition. Viscosity is also related directly to the type and size of molecules the fluid is made up of. Above the bubble point, the oleic phase has a constant chemical composition and the lower the pressure is, the easier the molecules can move past each other, since they are further apart. Hence, the viscosity increases almost linearly with pressure. Below the bubble point, the oleic phase changes chemical composition as more lighter molecules are removed from the liquid when gas is liberated. The fraction of larger molecules with complex shapes will gradually become larger, causing the viscosity to increase. One order of magnitude

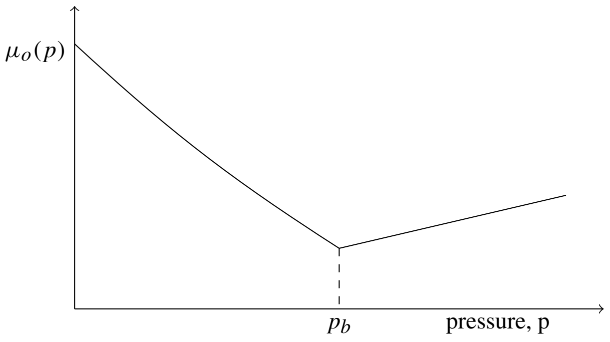


Figure 11.24 Oil viscosity as function of pressure for fixed temperature and chemical composition.

increase in viscosity from bubble point to low reservoir pressure is not uncommon, and this largely reduces the mobility of the remaining oil.

Water Properties

The aqueous phase is usually a brine that contains a number of different dissolved salts and hence has specific density greater than unity. Water may also contain certain amounts of dissolved gas, but this is usually ignored in standard simulation models since the solubility factor in water is orders of magnitude lower than in the liquid oleic phase.

Water viscosity decreases with increasing temperature and increases with increasing pressure, but is generally a very weak function of pressure. Typical values at reservoir conditions range from 0.5 to 1 cP. Water is only slightly compressible at reservoir conditions, with values in the range from 1×10^{-6} to 9×10^{-6} psi^{-1}, which implies that the water formation-volume factor is usually close to unity.

11.5 Phase Behavior in ECLIPSE Input Decks

Use of tabulated data from the `PROPS` section of ECLIPSE input is the primary input mechanism to set up simulation models for the simulators implemented in the `ad-blackoil`, `ad-eor`, and other derived modules. As we saw earlier on page 350, functionality for reading these input files, parsing and processing them to create specific fluid objects is offered by the `deckformat` and `ad-props` modules.

To make the book as self-contained as possible, this section outlines the necessary ECLIPSE keywords and show a few examples of specific phase behavior for the first, third, and ninth SPE benchmark cases. For a more comprehensive discussion of the input format, you should consult the lecture notes of Pettersen [250], or the documentation of ECLIPSE [270, 269] or OPM Flow [33]. The following presentation is inspired by these sources.

Dead Oil

Dead oils are assumed to always stay above the bubble point and hence behave like a standard fluid that is compressed and becomes more viscous at higher pressures. Most dead oils contain a constant amount of dissolved gas that is liberated as the oleic phase is brought to the surface. There are two different ways to describe dead oil in an ECLIPSE input deck:

PVDO – specifies the properties of an undersaturated oil above the bubble point in terms of a table of formation-volume factors and viscosities versus monotonically increasing pressure values. The table is attached to specific (groups of) cells, and the keyword can contain multiple tables if the reservoir contains several zones with distinct oils. The number of tables is therefore given by the NTPVT element of the TABDIMS keyword from the RUNSPEC section. The constant amount of dissolved oil and the associated bubble-point pressure should be specified using the RSCONST keyword. The following example is taken from [250]:

```
PVDO
--   Po    Bo        mu_o
  27.2    1.012     1.160
  81.6    1.004     1.164
 136.0    0.996     1.167
 190.5    0.988     1.172
 245.0    0.9802    1.177
 300.0    0.9724    1.181
 353.0    0.9646    1.185 /
RSCONST
-- Rs    Pbp
  180   230 /
```

Here, the pressures Po and Pbp are given in units bar, the formation-volume factor Bo and the gas–oil ratio Rs are dimensionless, and the viscosity mu_o has unit cP.

PVCDO – describes an oleic phase with constant compressibility by specifying the five constants in the formulas

$$B_o(p) = B_{or} e^{-c_o(p-p_r)}, \qquad (B_o\mu_o)(p) = B_{or}\mu_{or} e^{-(c_o-c_\mu)(p-p_r)}.$$

Here, p_r is reference pressure, B_{or} reference formation-volume factor, c_o compressibility, μ_{or} reference viscosity, and c_μ "viscosibility." In addition, one must specify RCONST and a maximum pressure PMAX. The following example is taken from [269] (with pressures given in units psi):

```
PVCDO
-- P     Bo       Co     mu0    Cmu
  2500  1.260    6E-6   0.5    1E-6 /
RSCONST
-- GOR     Pb
  0.656   2500 /
PMAX
  4500 /
```

Live Oil

Undersaturated oil has a well-defined bubble point that can be determined from the temperature and the oil's chemical composition. Saturated oils, on the other hand, change chemical composition as gas is dissolved or liberated. To model the phase behavior of a live oil, you need to describe both how the bubble point changes with changing composition of the oil, as well as the properties in the undersaturated region for each unique composition. In a black-oil model, the fluid composition of the oleic phase is represented in terms of the solution gas–oil ratio, and all we need to discriminate saturated from undersaturated states is a relationship between R_s and the bubble point p_b. In ECLIPSE input, this is represented by tabulating $p_b(R_s)$. For each value of $p_b(R_s)$, the dead-oil behavior in the undersaturated region is prescribed in the same way as described for dead-oil models. Hence, the input language offers two keywords for describing live oils:

PVTO – consists of NTPVT tables that each give the properties above and below the bubble point. Each table contains from two to NRPVT records that specify PVT data for given R_s values. The records consist of at least four numbers specifying R_s, bubble point pressure p_b, formation-volume factor B_{ob} for saturated oil at the bubble point, and viscosity μ_{ob} at the bubble point. However, each record can also be composed of from two to NPPVT rows of data organized in three columns that describe oil pressure (increasing values), formation-volume factor (decreasing values), and viscosity for undersaturated oil for the specified value of R_s. These extra data *must* be specified for the maximum R_s value inside each table but can optionally be given for any R_s value. We present a specific example of this input format for the SPE 9 model on page 383.

Notice that ECLIPSE interprets the input data by linear interpolation of the reciprocal *values of B_o and $(\mu_o B_o)$ between data points.*

PVCO – specifies properties above and below the bubble point in compressibility form; that is, instead of specifying the undersaturated behavior in terms of a tabular, it is described as for the PVCDO keyword. The data thus consists of NTPV tables that each may have up to NRPVT rows. The following example is from [269]:

```
PMAX
   6000 /
PVCO
--Pb        Rs      Bo         Vo         Co    Cvo
  1214.7    0.137   1.17200    1.97000    1E-5  0
  1414.7    0.195   1.20000    1.55600    1*    0
  1614.7    0.241   1.22100    1.39700    1*    0
  1814.7    0.288   1.24200    1.28000    1*    0
  2214.7    0.375   1.27800    1.09500    1*    0
  2614.7    0.465   1.32000    0.96700    1*    0
  3014.7    0.558   1.36000    0.84800    1*    0
  3414.7    0.661   1.40200    0.76200    1*    0
  3814.7    0.770   1.44700    0.69100    1*    0
/
```

Notice, in particular the use of `1*` to indicate a default value that should be computed by linear interpolation. Since the table only gives one value in the first row, this value should be copied to all the next rows.

Dry Gas

Dry gases are found either below the dew point or (well) to the left of the critical point. Their phase behavior is equivalent to that of dead oils and is described as a tabular of formation-volume factor and viscosity as function of pressure, or by a Z-factor and viscosity as function of pressure:

`PVDG` – specifies `NTPVT` tables that each consists of three columns and up to `NPPVT` rows giving gas pressure (increasing monotonically downwards), gas formation factor, and gas viscosity. A specific example of this input format is presented for the SPE 9 model on the next page.

`PVZG` – specifies properties using Z-factors. The data consists of `NTPV` tables that each consists of two records. The first gives the reference temperature used to convert Z-values into formation-volume factors (see [11.8] on page 373), and the second consists of from 2 up to `NPPVT` rows giving gas pressure, Z-factor, and gas viscosity. Pressures and Z-values should increase monotonically down the table. The lowest pressure value should correspond to the dew point. The gas may contain a constant amount of vaporized oil, specified by the `RVCONST` keyword, which also gives the dew-point pressure.

Wet Gas

Wet or rich gases with vaporized (condensate) oil are similar to live oils in the sense that the phase behavior is different for saturated and undersaturated states. The composition of the wet gas is represented by vaporized oil–gas ratio R_v. To discriminate between saturated and undersaturated states, you must tabulate the maximum amount of vaporized oil as function of pressure, i.e., $R_v(p_g)$. The corresponding keyword is:

`PVTG` – describes properties of wet gas with vaporized oil. The format is similar to the `PVTO` keyword used to describe live oil, and consists of from two to `NRPVT` records that specify PVT data (R_v, B_g, and μ_g) at the saturated state for given p values. Each record can also be composed of from two to `NPPVT` rows of three-column data that specify R_v, B_g, and μ_g for undersaturated states. These extra data *must* be specified for the highest pressure data in each of the `NTPVT` tables.

We shown an example of this input format on page 387 when discussing the PVT behavior of the SPE 3 benchmark.

Miscellaneous Properties

Water properties are described with the keyword `PVTW`, which follows the same format as the `PVCDO` keyword for dead oil. In addition to the PVT behavior, you also need to prescribe the density or specific gravity of each fluid at surface condition, which is defined by the keywords DENSITY or GRAVITY. The PROPS section of the input deck should also describe how the pore volume depends on pressure, either by the ROCK keyword that specifies a reference pressure and rock compressibility, or by the ROCKTAB keyword that tabulates porosity (and permeability) multipliers as function of pressure.

Example: SPE 1 and SPE 9 – Live Oil With Dry Gas

To exemplify live oil and dry gas models, we revisit the SPE 1 and SPE 9 benchmarks discussed on page 355. The SPE 9 benchmark describes a live oil with dry gas specified as follows (the setup for SPE 1 is similar):

```
-- LIVE OIL (WITH DISSOLVED GAS)
PVTO
-- Rs    Pbub     Bo       Vo
 .0       14.7   1.0000    1.20  /
 .165     400.   1.0120    1.17  /
 .335     800.   1.0255    1.14  /
 .500    1200.   1.0380    1.11  /
 .665    1600.   1.0510    1.08  /
 .828    2000.   1.0630    1.06  /
 .985    2400.   1.0750    1.03  /
1.130    2800.   1.0870    1.00  /
1.270    3200.   1.0985     .98  /
1.390    3600.   1.1100     .95  /
1.500    4000.   1.1200     .94
         5000.   1.1189     .94  /
/

PVTW
--Pref  Bw       Comp    Vw     Cv
  3600. 1.0034   1.0E-6  0.96   0.0 /
```

```
-- DRY GAS (NO VAPORIZED OIL)
PVDG
--     Pg        Bg        Vg
      14.7     178.08    .0125
      400.      5.4777   .0130
      800.      2.7392   .0135
     1200.      1.8198   .0140
     1600.      1.3648   .0145
     2000.      1.0957   .0150
     2400.      0.9099   .0155
     2800.      0.7799   .0160
     3200.      0.6871   .0165
     3600.      0.6035   .0170
     4000.      0.5432   .0175  /

DENSITY
--    Oil    Water    Gas
   44.98   63.01   0.0702  /
```

The model is specified in field units, so that densities are given as lb/ft^3 and not kg/m^3 and pressure values are given in unit psi. For the live oil, we see that undersaturated behavior is described for the largest R_s value. We do not repeat the statements required to load these model, but simply assume that we have created a processed deck structure with the necessary input data and created a fluid object f. Once the units have been converted to the SI system and the input data have been processed to construct a fluid object, the phase behavior of the oil is represented by the following functions and constants:

```
    bO: @(po,rs,flag,varargin) bO(po,rs,pvto,flag,reg,varargin{:})
   muO: @(po,rs,flag,varargin) muO(po,rs,pvto,flag,reg,varargin{:})
 rsSat: @(po,varargin) rsSat(po,pvto,reg,varargin{:})
 rhoOS: 720.5105
```

MRST has functions for computing the reciprocal expansion/shrinkage factors rather than the usual formation-volume factors. We also note that R_s is called rsSat rather than Rs. The reason is that dissolved gas will be an unknown in our system of flow equations and hence we change the name of the *parameter* R_s that describes the maximum dissolved gas amount for a given pressure to avoid confusion.

Looking at the functions in more detail, we see that the bO function takes three primary arguments: oil pressure, the amount of dissolved oil, and a flag that indicates whether a given element in `po` and `rs` represents a saturated or an undersaturated state. In the underlying functions, `pvto` is an object that holds the data of the `PVTO` keywords, whereas `reg` represents region information that picks the right table, should the model have multiple oils. The implementation of the `BO`, `bO`, `muO`, and `rSat` functions can be found in the file `assignPVTO.m` and consists of calls to two utility functions that extract region information, which is subsequently used to interpolate the correct data tables. For saturated oil, we use a simple linear interpolation between the data points, whereas the interpolation for undersaturated oil is set up to preserve compressibility and viscosibility if only one undersaturated data curve is given, or to interpolate compressibility and viscosibility if multiple curves are given. I will not go through the interpolation functions in detail; these have been optimized for speed and are only easy to understand if you are a black-belt MATLAB programmer.

Instead, we can plot the various relationships. (You can find the necessary code in the `showSPEfluids` script in the `book` module.) We start by extracting the input data from the `deck` structure:

```
pvto = deck.PROPS.PVTO{1};
rsd  = pvto.key([1:end end]);
pbp  = pvto.data([pvto.pos(1:end-1)' end],1);
Bod  = pvto.data([pvto.pos(1:end-1)' end],2);
muOd = pvto.data([pvto.pos(1:end-1)' end],3);
```

Here, we have only extracted the data points representing saturated states as well as the undersaturated data for the maximum R_s value, which is the minimal data requirement and coincides with what is given in the input file in our case. To simplify subsequent interpolation, the input processing in MRST adds an extra data point inside the undersaturated region for each data point, but we ignore these in our simplified data extraction.

Let us take the SPE 9 model as an example: The input data has eleven data records for R_s values 0, 0.165, 0.335,...,1.5. The last record also contains formation-volume factor and viscosity at pressure 5,000 psi inside the undersaturated region. The three pairs of pressure, B_o, and μ_o values specified for $R_s = 1.5$ define values for compressibility and viscosibility. Assuming that these are constant inside the whole undersaturated region, the input routine computes one additional value at pressure values $p = p_{bp} + 1{,}000$ psi, so that `pvto.data` becomes a 22×3 array. If we need R_s values for *all* data points represented in the `deck` structure, including extra points that have been added in the undersaturated region, we can use the following call:

```
rsd = pvto.key(rldecode((1:numel(pvto.key))',diff(pvto.pos),1));
```

To evaluate the fluid model, we make a mesh of pressure and dissolved gas states and then determine whether each state represents a saturated or undersaturated state by comparing with the interpolated R_s value for the corresponding pressure:

```
[RsMax,pMax] = deal(max(rsd), max(pbp));
[rs,p]     = meshgrid(linspace(10,RsMax-10,M), linspace(0,pMax,N));
Rs         = reshape(f.rsSat(p(:)), N, M);
```

```
isSat      = rs >= Rs;
rs(isSat) = Rs(isSat);
Bo         = reshape( f.BO (p(:), rs(:), isSat(:)), N,M);
muO        = reshape( f.muO(p(:), rs(:), isSat(:)), N,M);
```

Figure 11.25 shows plots of R_s as function of pressure as well as plots of B_o and μ_o as function of pressure and amount of dissolved gas. For comparison, we have overlain the plots for SPE 1 and SPE 9, which describe a gas-injection scenario and a case with

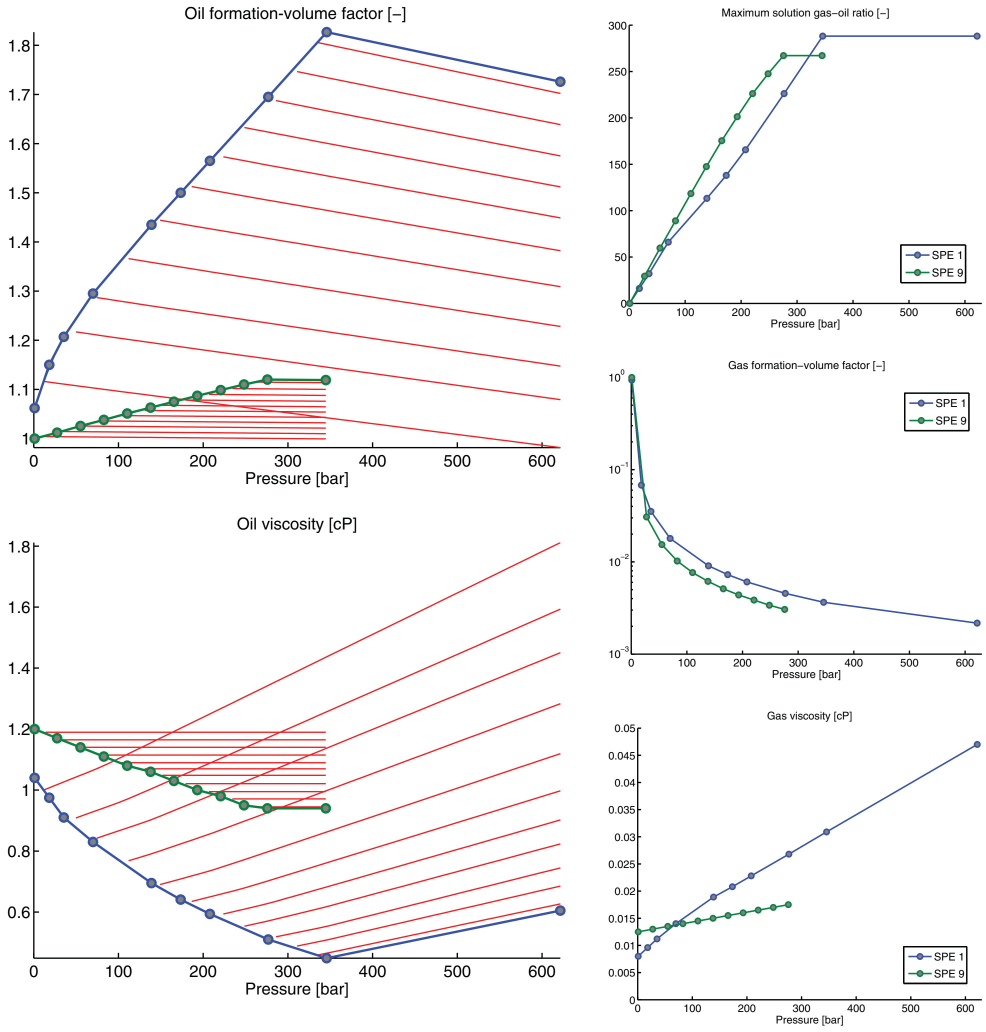


Figure 11.25 Description of phase behavior for the live-oil and dry-gas models in the SPE 1 and SPE 9 benchmarks, shown as blue and green lines, respectively. In the B_o and μ_o plots, the red lines correspond to undersaturated states.

water injection, respectively. There is a pronounced difference in the phase properties of the two models: Whereas the SPE 1 oil shows strong compressibility and viscosibility effects both for the saturated and undersaturated states, the undersaturated SPE 9 oil is almost incompressible and has constant viscosity. In part, this is also indicated by the surface densities of oil and gas, which are 786.5 kg/m^3 and 0.9698 kg/m^3 for SPE 1 and 720.5 kg/m^3 and 1.1245 kg/m^3 for SPE 9. The SPE 1 model has a larger fraction of both lighter and heavier hydrocarbon components and therefore exhibits larger variations in formation-volume factor and viscosity than SPE 9.

For completeness, let us also look at the gaseous and aqueous phases. The behavior of dry gas is represented by the following functions and constants (here with values for SPE 9):

```
     bG: @(pg,varargin) ireg(TbG,pg,varargin{:})
    muG: @(pg,varargin) ireg(TmuG,pg,varargin{:})
  rhoGS: 1.1245
```

The constant arguments `TBG`, `TbG`, and `TmuG` to the functions are data objects containing the tabulated formation-volume factor, its reciprocal expansion factor, and the viscosity from the `PVDG` keyword, respectively. The function name `ireg` is a shorthand for the `interpReg` function we encountered when discussing computation of three-phase relative permeabilities on page 353.

The aqueous phase is represented as a weakly compressible fluid (here for SPE 9),

```
     cW: 1.4504e-10
   muWr: 9.6000e-04
     bW: @(pw,varargin) bW(pw,pvtw,reg,varargin{:})
    muW: @(pw,varargin) muW(pw,pvtw,reg,varargin{:})
  rhoWS: 1.0093e+03
```

To be consistent with ECLIPSE, we use a Taylor expansion to approximate the weakly compressible functions

$$B_w(p) = B_{wr}e^{-c_w(p-p_r)} = B_{wr}e^{-\xi} \approx \frac{B_{wr}}{1+\xi+\frac{1}{2}\xi^2}.$$

At this point, I encourage you to consult the tutorial `fluidInspectionExample` from the `ad-core` module, which introduces a graphical interface for fluid models from the AD-OO framework that enables you to interactively produce many of the same plots as shown here and in Section 11.3. Creating models and launching the GUI only requires three lines of code:

```
[G, rock, fluid, deck] = setupSPE9();              % or setupSPE1(), setupSPE3()
spe9 = selectModelFromDeck(G, rock, fluid, deck);
inspectFluidModel(spe9)
```

Example: SPE 3 – Rich Gas With Retrograde Condensation

The SPE 3 benchmark was designed to compare the ability of compositional simulators to match PVT data for a problem with gas cycling in a rich-gas reservoir with retrograde condensation. The input deck distributed with MRST is a translation of the problem into a black-oil framework, with which one can only hope to describe the pertinent flow physics to a limited extent. The resulting model consists of a dead oil, described with the PVDO keyword, and a rich gas described by the PVTG keyword. Here, we only show excerpts of the specification of the rich gas:

```
-- 'Gas Pressure'  'Gas OGR'   'Gas FVF'    'Gas Visc'
-- Units: psia      stb /Mscf    rb /Mscf      cp
PVTG
      1214.7000   0.0013130      2.2799     0.0149
                          0      2.2815     0.01488/
      1814.7000     0.00353      1.4401     0.01791
                   0.001313      1.4429     0.01782
                          0      1.4445    0.01735 /
      2414.7000     0.01102      1.0438     0.02328
      :
      :
      3449.3322      0.0642      0.7783     0.0403
                     0.0612      0.7777     0.0395
                     0.0567      0.7769     0.03892
                     0.0454      0.7747     0.03748
                     0.0331      0.7723     0.03594
                    0.01102      0.7681     0.03325
                    0.00353      0.7666     0.03236
                   0.001313      0.7662     0.0321
                          0      0.7660     0.03194 /
/
```

The maximum amount of vaporized oil increases as function of pressure, and the input table uses an increasing number of data points within each record, starting with two points for the first pressure value and ending with nine values for the highest pressure value. The data points are quite unevenly scattered in (p_g, R_v) space, as you can see from the plot of R_v and μ_g versus p_g in Figure 11.26. You should therefore be wary of potential numerical inaccuracies in the numerical interpolation. (The `showSPEfluids` script in the `book` module gives the full code used to generate the plots in the figure.)

Looking at the qualitative behavior of the gas properties shown in Figure 11.26, we see that as pressure increases, more oil can be vaporized into the gas and this will increase the viscosity of the gaseous phase. The data points along the vertical lines traverse the phase envelope for fixed pressure, from the blue line corresponding to a saturated state with full vaporization (i.e., maximal supplied R_v value) to the red line representing the dew point, at which the vapor phase contains no vaporized oil ($R_v = 0$). Vaporization of oil also affects the gas formation-volume factor, but since the amount of vaporized oil is so small, the resulting variations in volume across the phase envelope for a fixed pressure are so small that they are only visible when zooming in on the plot. Interestingly, we observe that B_g decreases with increasing vaporization for pressures below 220 bar, but *increases* with increasing vaporization for pressures above 220 bar. However, the density of

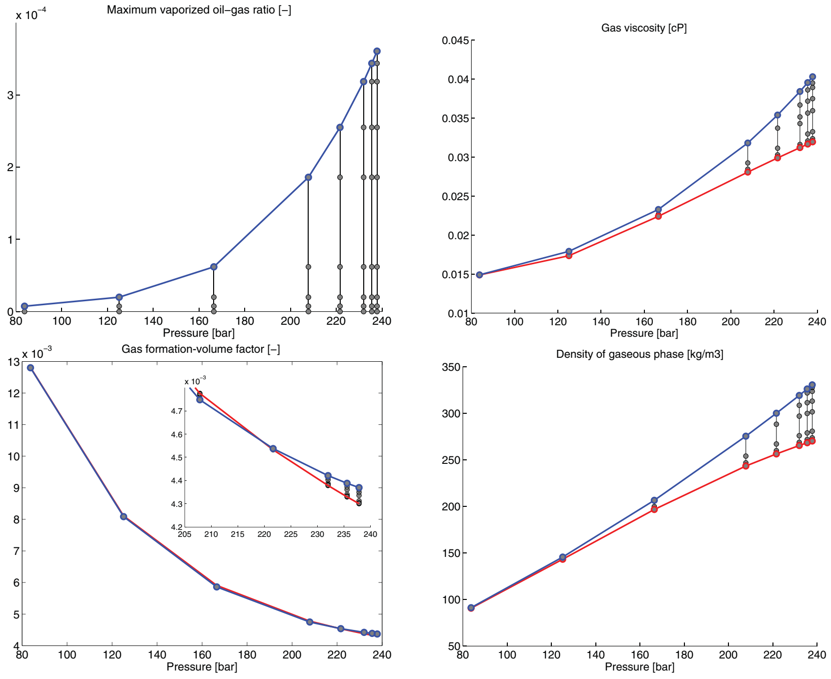


Figure 11.26 Description of phase behavior for the rich-gas system of the SPE 3 benchmark problem. Vertical lines represent transition across the phase envelope for fixed pressure, with the blue lines representing max R_v and red lines $R_v = 0$.

the gaseous phase increases monotonically with increasing oil vaporization, as one would expect physically.

The figure does not report properties of the dead oil, but these behave as expected, with B_o decreasing and μ_o increasing with increasing pressure.

11.6 The Black-Oil Equations

In this section, we discuss the flow equations for black-oil reservoir model, which we presented in Section 8.2.3, in more detail. We go through the equations, component by component, and discuss the various special cases and present how to discretize the various terms using discrete differentiation and averaging operators. As described in Section 11.4, the family of black-oil models assume isothermal conditions inside the reservoir, so that the PVT behavior is modeled as a function of pressure only. The resulting models work best when operating in single-phase hydrocarbon regions far from the critical point, but can also be used in two-phase hydrocarbon regions as long as the variation in composition is relatively small.

11.6.1 The Water Component

The discussion in this chapter has primarily focused on the gas and oil components contained in the gaseous and oleic hydrocarbon phase. Reservoirs will nonetheless always contain aqueous fluids present as connate water within the hydrocarbon-bearing rock layers. The reason is that the reservoir rock was fully or partially saturated with water before hydrocarbons migrated upwards from a deeper source rock. Since most rocks are fully or partially water-wet, the invading hydrocarbons would not be able to fully displace the resident water. Oftentimes, hydrocarbon-bearing rocks will have extensive underlying aquifer that contribute pressure support and fluid inflow that should be accounted for in the reservoir model.

As a rule, aqueous reservoir fluids are saline. The most common dissolved salt is sodium chloride, but the aqueous phase typically contains a wide range of compositions and concentrations of other inorganic chemical species. These species are not accounted for in basic black-oil type models, which simply assume that the resident brine and injected water is a single, chemically inert component that only is present in the aqueous phase. (You can consult Trangenstein and Bell [295] for an alternative model, in which gas is allowed to dissolve into the aqueous phase.) The corresponding flow equations read

$$\partial_t\big(\phi b_w S_w\big) + \nabla\cdot\big(b_w\vec{v}_w\big) - b_w q_w = 0, \tag{11.15}$$

with Darcy's law for the phase velocity

$$\vec{v}_w = -\frac{\mathbf{K}k_{rw}}{\mu_w}(\nabla p_w - g\rho_w\nabla z).$$

These equations are discretized exactly in the same way as (11.1) on page 342, except for the obvious modifications resulting from ρ_w having been replaced by b_w. Let us now look at how you could implement the discrete flow equations in MRST. We have already discussed most of the tools you need. As before, we assume that the fluid properties are collected in a fluid object called `fluid`. For convenience, the `ad-core` module collects all the discrete differentiation and averaging operators in a special structure, which we here will refer to by the short-hand `s`. We start by computing the necessary fluid properties:

```
bW      = fluid.bW(p0);
rhoW    = bW.*fluid.rhoWS;
rhoWf   = s.faceAvg(rhoW);
mobW    = krW./fluid.muW(p0);
pvMultR = fluid.pvMultR(p0);
```

Here, `faceAvg` is the same as the `avg` operator we have used previously, but has been renamed to specify explicitly that this is an averaging operator that should be used to compute average values at cell interfaces. Notice that we evaluate the pressure-dependent properties using oil pressure and not water pressure. In general, there are several ways you can choose the primary unknowns in the black-oil models. Herein, we have chosen to follow the conventions from ECLIPSE and use oil pressure p_o and gas and water saturations, S_w and S_g, as our primary unknowns. (We will get back to the choice of primary variables later in this section.)

The water flux across cell faces will, as before, be computed using single-point upstream-mobility weighting. To this end, we must first compute the water pressure in each grid cell,

```
pW = pO
if isfield(fluid, 'pcOW') && ~isempty(sW)
   pW = pW - fluid.pcOW(sW);
end
```

Then, we can compute the water flux from Darcy's law

```
dpW  = s.Grad(pW) - rhoWf.*gdz;
upcw = (double(dpW)<=0);
vW   = -s.faceUpstr(upcw, mobW).*T.*dpW;
bWvW = s.faceUpstr(upcw, bW).*vW;
```

The function `faceUpstr` implements the upstream operator we previously denoted by `upw`. With this, we have all quantities we need to compute the homogeneous water equation on residual form

```
water = (s.pv/dt).*( pvMult.*bW.*sW - pvMult0.*bW0.*sW0 ) + s.Div(bWvW);
```

Here, quantities ending with `0` refer to variables evaluated at the beginning of the time step and the vector `s.pv` contains pore volumes in each cell at reference pressure. The code lines in this subsection are in essence what is implemented in the flow solvers of the `ad-blackoil` module. However, if you look at the code in e.g., `equationsBlackOil`, you will see that it is more complicated since we in general need to also include (multisegment) well models, source terms, and boundary conditions. Likewise, we must account for various user-defined multipliers that modify transmissibilities and mobilities, enable solution of adjoint equations for computing sensitivities, and store computed results; more about this in the next chapter.

11.6.2 The Oil Component

In the basic black-oil model, the oil component is only present in the liquid oleic phase, and we hence get the following flow equation:

$$\partial_t\big(\phi b_o S_o\big) + \nabla\cdot\big(b_o\vec{v}_o\big) - b_o q_o = 0, \tag{11.16}$$

with Darcy's law for the phase velocity

$$\vec{v}_o = -\frac{\mathbf{K}k_{rw}}{\mu_o}(\nabla p_o - g\rho_o\nabla z).$$

As for the water component, we start by computing the necessary fluid properties. How we compute density, shrinkage factor, and viscosity depends upon whether the oil phase contains dissolved gas or not. If the system contains no dissolved gas, the parameters only depend on oil pressure and can be computed as follows:

```
bO  = fluid.bO(pO);
if isfield(fluid, 'BOxmuO')
   muO = fluid.BOxmuO(pO).*bO;
else
   muO = fluid.muO(pO);
end
rhoO   = bO.*fluid.rhoOS;
```

As pointed out on page 384, the convention in ECLIPSE 100 input is that tabulated formation-volume factors and viscosity for oil should not be interpolated independently, but instead be interpolated as their reciprocal product, $1/(\mu_o B_o)$. MRST signifies input data that require this particular interpolation by a special function `BOxmuO` in the fluid object. If this function is not present, the viscosity and the shrinkage factors are computed independently. If oil also contains dissolved gas, the same properties are evaluated as follows:

```
bO  = fluid.bO(pO,  rs, isSat);
muO = fluid.muO(pO, rs, isSat);
rhoO   = bO.*(rs*fluid.rhoGS + fluid.rhoOS);
```

Here, `rs` is a vector holding the gas–oil ratio and `isSat` is a Boolean vector that tells whether each element of `pO` and `rs` represents a saturated or an undersaturated state. We will get back to how `rs` is computed in Section 11.6.4. With density, shrinkage factor, and viscosity given, we can proceed to compute the oleic phase flux `bOvO` using upstream-mobility weighting as described for the aqueous phase in Section 11.6.1. Altogether, this gives us the homogeneous residual equation

```
oil = (s.pv/dt).*( pvMult.*bO.*sO - pvMult0.*bO0.*sO0 ) + s.Div(bOvO);
```

If oil is allowed to vaporize into the gaseous phase, the conservation equation for the oil component reads

$$\partial_t\big[\phi\big(b_o S_o + b_g R_v S_g\big)\big] + \nabla\cdot\big(b_o\vec{v}_o + b_g R_v\vec{v}_g\big) - \big(b_o q_o + b_g R_v q_g\big) = 0. \qquad (11.17)$$

We therefore need to compute the properties and the phase flux for the gaseous phase before we can compute the homogeneous residual oil equation:

```
oil = (s.pv/dt).*( pvMult.* (bO.* sO  + rv.* bG.* sG) - ...
      pvMult0.*(bO0.*sO0 + rv0.*bG0.*sG0) ) + s.Div(bOvO + rvbGvG);
```

Notice, in particular, that we use single-point upstream-mobility weighting to evaluate the vaporized oil–gas ratio in the term representing flux of oil vaporized in the gas phase `rvbGvG`. This is completely analogous to how we evaluate the shrinkage factors for oil and water at the interface between two neighboring cells.

11.6.3 The Gas Component

The gas component can generally be present both in the gaseous and the liquid oleic phases. The corresponding conservation equation reads

$$\partial_t\big[\phi\big(b_g S_g + b_o R_s S_o\big)\big] + \nabla\cdot\big(b_g\vec{v}_g + b_o R_s\vec{v}_o\big) - \big(b_g q_g + b_o R_s q_o\big) = 0 \qquad (11.18)$$

with Darcy's law for the phase velocity

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

Like for oil, the computation of the gas properties differs for systems with and without vaporized oil:

```
if vapoil
   bG   = fluid.bG(pO, rv, isSat);
   muG  = fluid.muG(pO, rv, isSat);
   rhoG = bG.*(rv*fluid.rhoOS + fluid.rhoGS);
else
   bG   = fluid.bG(pO);
   muG  = fluid.muG(pO);
   rhoG = bG.fluid.rhoGS;
end
```

We also need to compute the phase pressure to determine the phase velocity:

```
pG = pO;
if isfield(fluid, 'pcOG') && ~isempty(sG)
   pG  = pG + fluid.pcOG(sG);
end
```

With these quantities given, we can proceed as above to compute the remaining terms necessary to evaluate the accumulation and flux terms in the homogeneous residual equation

```
gas = (s.pv/dt).*( pvMult.* (bG.* sG  + rs.* bO.* sO) - ...
      pvMult0.*(bG0.*sG0 + rs0.*bO0.*sO0 ) ) + s.Div(bGvG + rsbOvO);
```

11.6.4 Appearance and Disappearance of Phases

With all the details presented thus far, it should be a surmountable task to extend the simple two-phase, dead-oil implementation discussed in Section 11.2 to a similar simplified simulator for black-oil models. In particular, dissolved gas in the liquid oleic phase and/or vaporized oil in the gaseous phase is not too difficult to include as long as each hydrocarbon phase either stays in a saturated state or an unsaturated state throughout the whole simulation. For saturated phases, it is common to use one phase pressure (p_o in MRST) and the water and gas saturation (S_w and S_g) as independent variables since both the solution gas–oil ratio R_s and the vaporized oil–gas ratio R_v can be computed as a function of pressure. If oil is in an undersaturated state, there is no free gas available and the flow problem is essentially a two-phase flow problem (assuming that water is present). However, such cases can also be simulated as a three-phase problem if we set R_s as a primary variable. Likewise, we can choose R_v as a primary variable for cases with undersaturated gas.

Somewhat more intricate details are needed when the hydrocarbon phases switch between saturated and undersaturated states so that phases may appear or disappear during the simulation. As you have seen already, MRST introduces the solution gas–oil ratio `rs` as an extra variable alongside `Sg` for cases with dissolved gas. In addition, we need a status flag in each cell that tells which of the two is the primary unknown. When oil is in a saturated state, `Sg` is the primary variable and `rs` is a secondary variable that can be determined from the pressure. If the computed gas saturation in a cell is zero or falls below, we know that the cell has switched from a saturated to an undersaturated state. In cells containing undersaturated oil, there is no free gas available and we solve for `rs` as a primary variable to track the amount of gas flowing in and out of each cell with the oil. Once a new `rs` value is computed, we need to check whether `rs>fluid.rsSat(p0)`, which means that we have a transition from undersaturated to saturated state and hence need to increase `Sg` to the correct amount of liberated gas.

Switching of variables may unfortunately introduce instabilities in the nonlinear solution process, and in practice we do not allow states to cross from a saturated/undersaturated state and well into the undersaturated/saturated region. We rather stop the update at the interface, i.e., set `rs` to `rsMax=rsSat(p0)` and `Sg=0`, and chop the update to improve robustness of the Newton iteration loop. Cases with transitions between saturated and undersaturated gas are treated analogously. The following code summarizes the logic just explained to choose the primary variable `x` representing gas:

```
% Assumption: rs = rsSat for saturated cells, likewise for rv
% status 1 (oil, no gas): x = rs, sg = 0,     rv = rvMax
% status 2 (gas, no oil): x = rv, sg = 1-sw,  rs = rsMax
% status 3 (oil and gas): x = sg, rs = rsMax, rv = rvMax
watOnly    = sW > 1- sqrt(eps);

if ~vapoil, oilPresent = true;  else, oilPresent = or(sO > 0, watOnly); end
if ~disgas, gasPresent = true;  else, gasPresent = or(sG > 0, watOnly); end

status = oilPresent + 2*gasPresent;
if ~disgas, st1 = false; else, st1 = status==1; end
if ~vapoil, st2 = false; else, st2 = status==2; end
st3 = status == 3;

x = st1.*rs + st2.*rv + st3.*sG;
```

11.7 Well Models

Earlier in the book, we have discussed well models for single-phase flow. Section 4.3.2 introduced the basic inflow-performance relationship, which states that the flow rate in or out from a well is proportional to the pressure difference between the wellbore and the reservoir, $q = J(p_R - p_{wb})$. This section describes well models for multiphase flow in more detail. In particular, we discuss wells with multiple connections and explain how to handle cross flow and introduce a more comprehensive family of models that use a network of 1D flow models to describe flow in horizontal and multilateral wells.

11.7.1 Inflow-Performance Relationships

The inflow-performance relationship for black-oil models is written in terms of volumetric production rates for each fluid phase α at stock-tank conditions

$$q_{\alpha,i} = J_i \lambda_{\alpha,i} \big(p_i - p_{bh} - H_{wi}\big). \tag{11.19}$$

Here, J_i is the productivity index (sometimes called connection transmissibility factor or R-factor) for connection number i, $\lambda_{\alpha,i}$ is phase mobility at the connection, p_i is pressure in the perforated grid block, p_{bh} is bottom-hole pressure, and H_{wi} is pressure drop from the well's datum point to connection i.

The standard Peaceman-type expression for the productivity index of a vertical well penetrating a Cartesian grid block reads

$$J = \frac{\theta K h}{\ln(r_e/r_w) + S}.$$

Here, Kh is effective permeability times net thickness of the connection, r_w is wellbore radius, and r_e is the equivalent radius. We have previously discussed expressions for K and r_e for various special cases in Section 4.3.2. The parameter θ gives the angle the wellbore is connected to the reservoir block, with $\theta = 2\pi$ when the well is perforated in the center of the cell, $\theta = \pi$ when the well is perforated along a cell face, and $\theta = \pi/2$ when the well is perforated along a corner edge; see Figure 11.27. The *skin factor* S accounts for the difference between ideal pressure drawdown predicted by Darcy's law and actual drawdown resulting from formation changes in the near-well region; S also includes the constant $3/4$ from pseudo-steady flow. The typical range of skin factors is $-6 < S <$

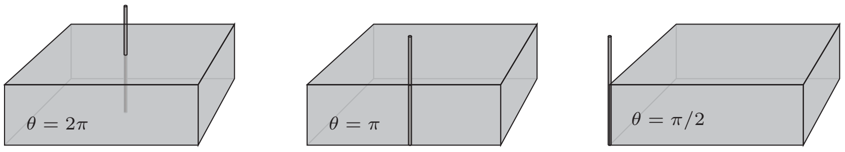


Figure 11.27 The parameter θ measures the angular section of the wellbore that is connected to the reservoir and depends on where the well is placed inside a grid block.

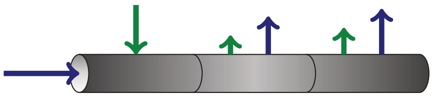


Figure 11.28 Illustration of crossflow in a well; blue and green colors represent different fluid phases.

100. Negative values represent wells stimulated by hydraulic fracturing or other means to improve the permeability in the near-well zone, whereas large positive values represent wells suffering from severe formation damage caused, e.g., by various forms of clogging induced by drilling fluids, fines migration, precipitation of soluble salts, etc.

The mobility $\lambda_{\alpha,i}$ is computed differently for injection and production wells. For *producing connections*, the mobility of a free phase of gas, oil, or water, is given as

$$\lambda_{\alpha,i} = \frac{(b_\alpha k_{r\alpha})_i}{\mu_{\alpha,i}}, \tag{11.20}$$

where the relative permeability $k_{r\alpha}$, the shrinkage/expansion factor b_α, and the phase viscosity μ_α are computed using pressure and saturation values from the grid block perforated by connection number i. If the produced gaseous or oleic phases contain vaporized oil or dissolved gas, respectively, the mobilities account for this as follows

$$\lambda_O = \lambda_o + R_v \lambda_g, \qquad \lambda_G = \lambda_g + R_s \lambda_o. \tag{11.21}$$

For *injecting connections*, MRST computes the mobility as the sum over all phases

$$\lambda_{\alpha,i} = y_\alpha b_\alpha \sum_\beta \Big(\frac{k_{r\beta}}{\mu_\beta}\Big)_i. \tag{11.22}$$

where y_α is the volume fraction of phase α in the mixture *inside the wellbore*. This means that the injectivity is a function of the total mobility of the connected grid block.

Wells connected to multiple grid blocks may sometimes experience *crossflow* as illustrated in Figure 11.28. This means that fluids flow into some connections and out of other connections. Injectivities must therefore be modified so that they represent the mixture of the fluids inside the wellbore. To this end, we first compute inflow from producing connections at reference (surface/stock-tank) conditions and use this to compute the *average* wellbore mixture at reference conditions. Then, we compute the average volumetric mixture at the conditions of each injection connection and use this to define the correct mobility. A similar average wellbore mixture is used to calculate the pressure drop between the well's datum point and each connection.

11.7.2 Multisegment Wells

Many wells are designed so that the reservoir fluids do not flow directly from the sandface and into the wellbore. Oftentimes, the wellbore (production casing) lies inside another casing that has been cemented in place inside the drilled hole. The void *annular* space between

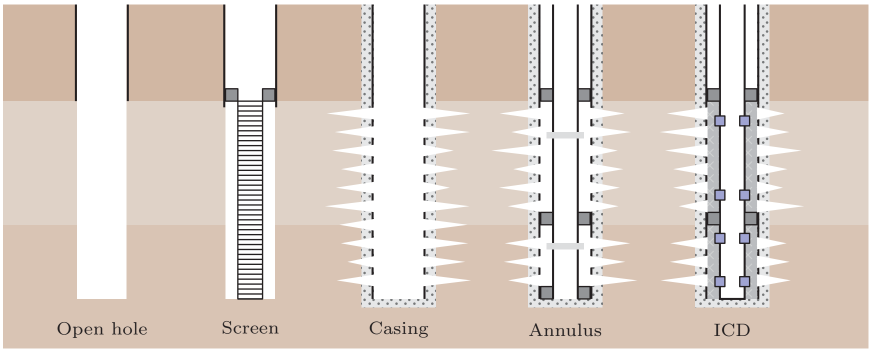


Figure 11.29 Conceptual illustration of different types of well completions.

the casing and the wellbore is limited by sealing packers, which limit the annular space into sections, and sleeves that provide flow paths between the annulus and the wellbore. One can also have various types of screens that aim to reduce/eliminate the inflow of solid particles that would cause erosion inside the wellbore. Simple versions of *sand screens* are made up of ribs, rods, or wire filters, possibly filled with gravel, whereas more modern ones consist of high-permeable, resin-coated gravel packed between two layers of wire-wrap filter media (*gravel pack*). In addition, wells may have various components that can partially or fully choke the inflow to prevent unwanted fluids to enter the wellbore. *Inflow control devices* (ICDs) are passive or autonomous devices installed along the completion interval to optimize inflow along the length of the wellbore. These devices are designed to exert higher back pressure at high flow rates and can be used to equalize inflow from high and low-permeable zones, or delay water or gas breakthrough across selected intervals of horizontal wells. *Inflow control valves* serve the same purpose and are controlled from the surface through electric or hydraulic signals.

To further increase complexity, modern wells typically have directional profiles designed to intersect multiple targets with a single wellbore. Such wells can have multiple production zones, or consist of multiple deviated or horizontal branches (*multilaterals*). To increase the productivity of a well, it is common to install *downhole pumps* that lower the wellbore flowing pressure so that fluids can be drawn from the reservoir at a higher rate. If the downhole pressure is too low, it may be necessary to apply *artificial gas lift*, in which gas is injected downhole to lower the density of the flowing fluids and use gas expansion to help the produced fluids to the surface. Most oil wells require artificial lift at some point to ensure that they keep flowing when their bottom-hole pressure decreases. Wells may also incorporate equipment for downhole compression and separation of fluid phases, which may be used to separate gas (or water) so that it can be reinjected downhole. Downhole processing can also be introduced to reduce the size and weight of surface or seafloor facilities or to overcome bottlenecks in existing surface processing systems.

All these different components will induce pressure drawdown and/or changes in the composition of the flowing fluids. To model this complex behavior, one may use a so-called *multisegment well model*, which can be thought of as a flow network consisting of a set of nodes connected by a set of segments (edges). Each node n has an associated volume and can store fluids, and fluids can flow along each segment connecting two nodes as described by a set of one-dimensional flow equations.

Continuous Flow Equations

The one-dimensional flow equations can either be formulated using phase velocities $\vec{v}_\alpha$ and volume fractions y_α, or component *mixture velocities* $\vec{v}_c^m$ and mass fractions x_c for the gas, oil, and water components. Capillary pressure is negligible inside the wellbore, hence we can write the conservation equations for volumetric quantities as follows:

$$\begin{aligned}\partial_t\big(b_w y_w\big) + \partial_x\big(b_w v_w\big) - b_w q_w &= 0,\\ \partial_t\big(b_o y_o + b_g R_v y_g\big) + \partial_x\big(b_o v_o + b_g R_v v_g\big) - b_o q_o - b_g R_v q_g &= 0,\\ \partial_t\big(b_g y_g + b_o R_s y_o\big) + \partial_x\big(b_g v_g + b_o R_s v_o\big) - b_g q_g - b_o R_s q_o &= 0.\end{aligned} \tag{11.23}$$

The phase velocities do not relate to pressure gradients through Darcy's law, but rather through some nonlinear relation modeling friction, acceleration, inflow devices, valves, and so on. We can write this relationship as

$$\nabla p - g\rho\nabla z - G(v, \rho, \mu) = 0, \tag{11.24}$$

where ρ, v, and μ are the mixture density, velocity, and viscosity, respectively, and G represents the (nonlinear) part of the pressure drop that cannot be explained by a Darcy relationship. The equivalent set of equations on mass form reads,

$$\partial_t\big(x_c\,\rho\big) + \nabla\cdot\vec{v}_c^m - q_c^m = 0, \qquad c \in \{g, o, w\}. \tag{11.25}$$

Here, q_c^m is the mass source term of component c.

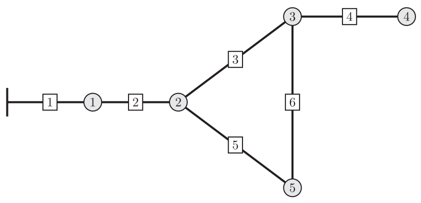
Discrete Well Equations

To define the discrete multisegment well models, we utilize a vector notation analogous to the discrete operators we introduced in Section 4.4.2 for flow equations in the porous medium. Where we previously had two mappings F between cells and their bounding cell faces and C_1, C_2 between faces and adjacent cells, we now have similar mappings S from nodes to all connected segments and N_1, N_2 from segments to the adjacent nodes; see Figure 11.30. Given these mappings, the *discrete gradient operator* reads

$$\texttt{grad}(\boldsymbol{v})[s] = \boldsymbol{p}[N_2(s)] - \boldsymbol{p}[N_1(s)],$$

and similarly the *discrete divergence operator* reads

$$\texttt{div}(\boldsymbol{v})[n] = \sum_{s\in S(n)} \boldsymbol{v}[s]\,\mathbf{1}_{\{n=N_1(s)\}} - \sum_{s\in S(n)} \boldsymbol{v}[s]\,\mathbf{1}_{\{n=N_2(s)\}}.$$



n	S(n)
1	1
1	2
2	2
2	3
2	5
3	3
3	4
3	6
4	4
5	5
5	6

Map: node → segment

s	N_1	N_2
1	0	1
2	1	2
3	2	3
4	3	4
5	2	5
6	5	3

Map: segment → node

$$\mathrm{M} = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & -1 \end{bmatrix}$$

$$\mathrm{N} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & -1 \end{bmatrix}$$

Figure 11.30 Example of well topology and corresponding discrete operators.

These operators can both be represented as sparse matrices, almost like the D matrix. However, the top segment of a well needs special treatment, since the top node corresponds to the reference depth at which the well's bottom-hole pressure is defined. Hence, instead of calculating a pressure drop across the top segment, we impose a boundary condition on the top node corresponding to the correct control of the well. This means that we write the operators as $\texttt{grad}(\boldsymbol{p}) = \boldsymbol{M}\boldsymbol{p}$ and $\texttt{div}(\boldsymbol{v}) = -\boldsymbol{N}^T\boldsymbol{v}$, where $\boldsymbol{M}$ equals $\boldsymbol{N}$ except that the first row is removed. Discrete averaging operators are defined analogously. Given these operators, the discrete flow equations are defined as follows: If we use the mass form, the conservation equation in each node reads

$$\frac{\boldsymbol{V}}{\Delta t}\big[\boldsymbol{x}_c\,\boldsymbol{\rho} - (\boldsymbol{x}_c\,\boldsymbol{\rho})^0\big] + \texttt{div}(\boldsymbol{v}_c^m) - \boldsymbol{q}_c^m = \boldsymbol{0}, \tag{11.26}$$

where superscript 0 refers to values at the previous time step. The equation for the pressure drop reads,

$$\texttt{grad}(\boldsymbol{p}) - g\,\texttt{upw}(\boldsymbol{\rho})\texttt{grad}(z) - G\big(\boldsymbol{v}^m, \texttt{upw}(\boldsymbol{\rho}), \texttt{upw}(\boldsymbol{\mu})\big) = \boldsymbol{0}. \tag{11.27}$$

Here, $\texttt{grad}(z)$ is the discrete gradient of the node depth, i.e., the depth difference between neighboring nodes. Note that (11.27) is not defined for the topmost segment and we need to specify a boundary or control equation at the top to close the coupled system (11.26)–(11.27).

Pressure-Drop Relations

The function G in (11.24) can be used to impose almost any pressure drop. For general network topologies, however, we assume that $\partial G/\partial v^m$ is nonzero. As an example, the pressure drop over a *simple valve* can be modeled as

$$\Delta p = \frac{\rho u_c^2}{2C_v^2}, \tag{11.28}$$

where u_c is the velocity in the constriction and C_v is the discharge coefficient. Another example is the *frictional pressure drop* along the wellbore, which may have a significant effect on the behavior of (long) horizontal wells. In ECLIPSE 100 [271], this pressure loss is given by

$$\Delta p = \frac{2fL}{D}\rho u^2. \tag{11.29}$$

Here, f is the Fanning friction factor, L is the length of the segment, D is the corresponding tubing diameter, ρ is the mixture density, and u is the mixture velocity. For a mixture mass flux v^m or mixture volume flux v, the velocity u is given by

$$u = \frac{v^m}{\rho\pi(D/2)^2} = \frac{v}{\pi(D/2)^2}. \tag{11.30}$$

For annular pressure drop, $(D/2)^2$ is replaced by $(D_o/2)^2 - (D_i/2)^2$, where D_o and D_i are the diameters of the outer and inner tubing, respectively. The Fanning friction factor depends on the Reynolds number $\mathrm{Re} = \rho u D/\mu$ and is calculated as

$$\sqrt{f^{-1}} = -3.6\log_{10}\left[\frac{6.9}{\mathrm{Re}} + \left(\frac{e}{3.7D}\right)^{10/9}\right], \tag{11.31}$$

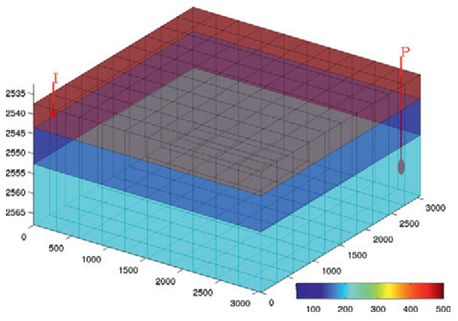
for $\mathrm{Re} > 4{,}000$, and $f = 16/\mathrm{Re}$ in the laminar region for $\mathrm{Re} < 2{,}000$. In between, one interpolates between the corresponding end values for Re equal 2,000 and 4,000. MRST uses the same approach, but also allows use of (11.31) for all velocities. The coefficient e gives roughness of the tubing.

11.8 Black-Oil Simulation with MRST

Altogether, we have discussed a great number of details that must be brought together to make a simulator. In addition, there are still a number of other important topics like preconditioning and details of boundary conditions, time chopping, nonlinear solution strategy, and so on, that I have *not* discussed in detail. As stated in the introduction of the chapter, I consider many of these details to be outside the scope of the book and I hence feel that time is right to switch the presentation to the AD-OO framework, which implements all the necessary ingredients entering industry-standard reservoir simulators. I therefore end the chapter by first briefly outlining how you can use the AD-OO framework to simulate the SPE 1 benchmark model and then discuss some limitations and potential pitfalls that you, as a user of MRST and other reservoir simulators, should be aware of. The next chapter will give a more systematic overview of the AD-OO framework and discuss more simulation examples.

11.8.1 Simulating the SPE 1 Benchmark Case

You have already encountered the fluid model from the SPE 1 benchmark [241] multiple times in this chapter. The benchmark describes a problem with gas injection in a small $10 \times 10 \times 3$ reservoir with a producer and an injector placed in diagonally opposite corners. The porosity is uniform and equals 0.3, whereas the permeability is isotropic with values



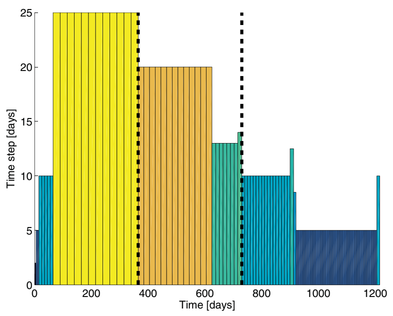


Figure 11.31 Horizontal permeability and well positions for the SPE 1 benchmark case. The injector is perforated in the upper and the producer in the lower layer. The right plot shows the time steps in the simulation schedule.

500, 50, and 200 md in the three layers with thickness 20, 30, and 50 ft; see Figure 11.31. The reservoir is initially undersaturated with constant pressure in each layer and a uniform mixture of water ($S_w = 0.12$) and oil ($S_o = 0.88$) with no initial free gas ($S_g = 0.0$) and a constant solution gas–oil ratio ($R_s \approx 226.2$) throughout the model. Although the problem is really a two-phase gas–oil problem describing gas injection into an undersaturated oil, the input file describes a three-phase problem with almost stationary water. The original problem was posed to study ten years of production, but herein we only report and compare solutions for the first 1,216 days.

We have already discussed the steps to load the ECLIPSE input data and create the necessary objects representing grid, petrophysical and fluid properties in detail earlier in the book. For completeness, however, we include them all here:

```
gravity reset on
deck  = readEclipseDeck(fullfile(getDatasetPath('SPE1'), 'BENCH_SPE1.DATA'));
deck  = convertDeckUnits(deck);
G     = computeGeometry(initEclipseGrid(deck));
rock  = compressRock(initEclipseRock(deck), G.cells.indexMap);
fluid = initDeckADIFluid(deck);
```

In addition, we must set up an initial state. The `SOLUTION` section in the input file specifies how the reservoir should be initialized. Here, we hard-code the fluid equilibrium state for simplicity:

```
[k, k] = ind2sub([prod(G.cartDims(1:2)), G.cartDims(3)], G.cells.indexMap);
p0     = [329.7832774859256 ; ...  % Top layer
          330.2313357125603 ; ...  % Middle layer
          330.9483500720813 ];     % Bottom layer
p0     = convertFrom(p0(k), barsa);
s0     = repmat([ 0.12, 0.88, 0.0 ], [G.cells.num, 1]);
rs0    = repmat( 226.1966570852417 , [G.cells.num, 1]);
rv0    = 0; % dry gas

state  = struct('s', s0, 'rs', rs0, 'rv', rv0, 'pressure', p0);
```

The first statement creates a partition vector containing the layer index, which we use to lookup in the vector of equilibrium pressures values. All the statements just presented are collected in a utility function `setupSPE1` in the `ad-blackoil` module.

The next step is to create an object representing the whole *simulation model* with description of reservoir geometry, petrophysical and fluid parameters, drive mechanisms, and parameters determining the correct flow equations

```
model = selectModelFromDeck(G, rock, fluid, deck);
```

Technically, the `model` object is an instance of a *class* defined using `classdef` and includes both properties and methods that can operate on input parameters and the data contained within the object. Classes can also define events and enumerations, but these are not used in AD-OO. If you are not familiar with the concept of class objects, you can think of `model` as a standard `struct` containing both data members and function pointers. In this particular case, the input data will create a three-phase, black-oil model with dissolved gas but no vaporization of oil. The following is a list of a few of the data members:

```
ThreePhaseBlackOilModel with properties:
                 disgas: 1
                 vapoil: 0
                  fluid: [1x1 struct]
                   rock: [1x1 struct]
                  water: 1
                    gas: 1
                    oil: 1
                gravity: [0 0 9.8066]
          FacilityModel: [1x1 FacilityModel]
              operators: [1x1 struct]
                      G: [1x1 struct]
```

The fields `G`, `rock`, and `fluid` are the standard MRST structures we have used earlier to represent grid, petrophysics, and fluid properties. The `operators` structure contains transmissibilities, pore volume, and discrete operators and is the same structure we referred to as `s` in the previous section. `FacilityModel` is an instance of another class designed to describe models of wells and surface facilities. Last but not least, the object contains five flags telling which of the three phases are present and whether gas is allowed to dissolve in oil and oil is allowed to vaporize into gas.

We have already encountered the flag `vapoil` in the code that computed gas properties on page 392. Likewise, `disgas` and `vapoil` were used to determine the correct primary unknown for the gas phase on page 393. These, and all the other code pieces for computing flow equations on residual form presented in Section 11.6, are excerpts from the function `equationsBlackOil` and a number of helper functions and have been simplified and reorganized for pedagogical purposes. The AD-OO framework never calls `equationsBlackOil` directly, but instead accesses it indirectly through a generic member function called `getEquations`. This way, the parts of the framework that implement the time loop and the nonlinear solver does not need to know any specifics of the model as long as we know that the model exists and that calling `getEquations` will compute the corresponding flow equations on residual form with automatic differentiation activated for all primary unknowns. In addition to the parameters just outlined, the model contains a number of other parameters that control the nonlinear solver, chopping of time steps, and output of results. More details about different model classes, time loops, nonlinear and linear solvers, are given in Chapter 12.

The next thing we must do is create a *simulation schedule*, which essentially consists of a set of time steps and a specification of the drive mechanisms that will be present during each of these time steps. These are described in the SCHEDULE section of the input deck, which for SPE 1 declares two different wells:

```
-- WELL SPECIFICATION DATA
--
--  WELL GROUP LOCATION  BHP   PI
--  NAME NAME    I  J   DEPTH DEFN
WELSPECS
    'P'  'G'    10 10    8400 'OIL'  /
    'I'  'G'     1  1    8335 'GAS'  /
/
```

```
-- COMPLETION SPECIFICATION DATA
--
--  WELL -LOCATION-  OPEN/ SAT CONN  WELL
--  NAME  I  J K1 K2 SHUT  TAB FACT  DIAM
COMPDAT
    'P'  10 10 3  3  'OPEN' 0   -1   0.5  /
    'I'   1  1 1  1  'OPEN' 1   -1   0.5  /
  /
```

The WELSPECS keyword shown to the left introduces the wells, defines their name, the position of the well head, the reference depth for the bottom holes, and the preferred phase. The keyword COMPDAT shown to the right, specifies how each well is completed, i.e., in which cells each well can inject/produce fluids. The next we need to define is how the wells are controlled during the simulation:

```
-- PRODUCTION WELL CONTROLS
--   WELL  OPEN/  CNTL   OIL   WATER   GAS  LIQU   RES    BHP
--   NAME  SHUT   MODE  RATE   RATE  RATE  RATE  RATE
WCONPROD
     'P'  'OPEN' 'ORAT' 20000  4*                        1000 /
  /

-- INJECTION WELL CONTROLS
--   WELL  INJ   OPEN/  CNTL    FLOW
--   NAME  TYPE  SHUT   MODE    RATE
WCONINJE
     'I'  'GAS' 'OPEN' 'RATE' 100000 100000 50000/
  /
```

The specification says that the producer should be controlled by oil rate, whereas the injector should inject gas and be controlled by rate. To turn this input into a data object describing the simulation schedule in a form that MRST understands, we issue the following command:

```
schedule = convertDeckScheduleToMRST(model, deck);
```

which parses the deck structure and creates the following data structure:

```
schedule =                schedule.step =            schedule.control =
    step:[1x1 struct]      control:[120x1 double]        W:[2x1 struct]
 control:[1x1 struct]          val:[120x1 double]       bc:[]
                                                        src:[]
```

The schedule has two data members: a list of steps and a list of controls. The controls specify a set of *targeted*[2] drive mechanisms (wells, boundary conditions, and source terms) that each is unique and will be applied during at least one step. Each step consists of a time-step length (`val`) and the number (`control`) of the corresponding control. In our case, there is only one control consisting of the injection and production wells described previously (see Figure 11.31), which will be active during each of the 120 time steps. The control steps are given by the keyword TSTEP, which can be repeated to specify the full schedule (see Figure 11.31):

```
TSTEP
1.0 2*2.0 2*5.0 5*10.0 11*25.0
  /
TSTEP
   25.0
  /
```

```
-- INITIAL STEP: 1 DAY, MAXIMUM: 6 MONTHS
TUNING
1  182.5  /
1.0 0.5 1.0E-6 /
/
```

The TUNING keyword gives parameters for time-step control, time truncation and convergence control (here: mass-balance error $\leq 10^{-6}$), and control of Newton and linear iterations (not given here). This keyword is ignored by MRST.

To run the schedule, we must select a nonlinear solver, which is implemented as a class in the AD-OO framework:

```
nls = NonLinearSolver('useLinesearch', true);
```

The NonLinearSolver class implements an abstract framework for nonlinear solvers, which uses the Newton linearization (or some other mechanism like a fix-point step) provided by the model to iterate towards a solution. It is capable of time-step selection and chopping based on convergence rates and can be extended via subclassing with alternative linear solvers and time-step classes. Convergence is handled by the model. The NonLinearSolver simply responds based on what the model reports in terms of convergence to ensure some level of encapsulation. In the current example, we initialize the solver class to use line-search, as discussed in Section 10.2.2. The excerpts from the `nls` object

```
NonLinearSolver with properties:
               maxIterations: 25
             maxTimestepCuts: 6
                LinearSolver: [1x1 BackslashSolverAD]
            timeStepSelector: [1x1 SimpleTimeStepSelector]
               useLinesearch: 1
                     verbose: 0
                            :
```

show that the nonlinear solver will at most use 25 iterations, be allowed to cut the time step six times before giving up. The solver will use the BackslashSolverAD linear solver and SimpleTimeStepSelector to select time steps, use line-search as specified, run in non-verbose mode so that no details about convergence are reported to screen, and so on.

[2] This is an important distinction: the schedule may for instance request that a given well is operated according to a prescribed rate (rate control). However, if this rate cannot be achieved without violating physical limits on the bottom-hole pressure, the simulation will switch to using a bottom-hole control instead.

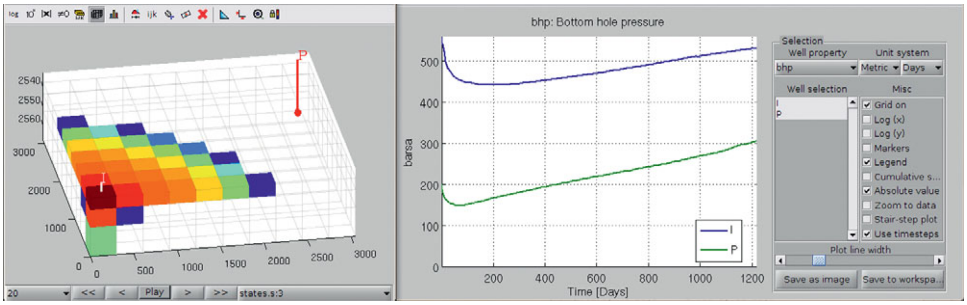


Figure 11.32 Graphical user interfaces for plotting reservoir data and well responses. The left plot shows the saturation of free gas after 20 time steps. The right plot shows the variation in bottom-hole pressure in the injector and producer throughout the simulation.

We are now ready to run the full simulation. We specify that we wish output of well solutions, reservoir states, and a report with details about time steps, convergence, etc.:

```
[wellSols, states, report] = ...
     simulateScheduleAD(state, model, schedule, 'nonlinearsolver', nls);
```

```
Solving timestep 001/120:                     -> 1 Day
Solving timestep 002/120: 1 Day               -> 3 Days
:                        :                               :
Solving timestep 120/120: 3 Years, 111 Days -> 3 Years, 121 Days
*** Simulation complete. Solved 120 control steps in 103 Seconds .. ***
```

To inspect the solutions, we issue the following commands, which will bring up the two graphical user interfaces shown in Figure 11.32. First, the plot of well responses

```
plotWellSols(wellSols, report.ReservoirTime)
```

and then the plot of reservoir states

```
figure;
plotToolbar(G, states)
plotWell(G, schedule.control(1).W,'radius',.5)
axis tight, view(-10, 60)
```

Let us study the dynamics of the reservoir in some detail. Figure 11.33 shows how the bottom-hole pressure in the injector and the solution gas–oil ratio inside the reservoir evolve as functions of time. Before production starts, the reservoir is in an undersaturated state, shown as a green marker in the plots of `rs` as function of `p0`. Initially, the injector needs a relatively high bottom-hole pressure (bhp) to force the prescribed gas rate into the reservoir. Because the reservoir is undersaturated, the injected gas will dissolve in the oil phase, so that the viscosity decreases. A lower bhp is thus required to maintain the injection rate. As the injection continues, cells near the well will gradually become saturated so that free gas is liberated and starts accumulating at the top of the reservoir. Similarly, some free

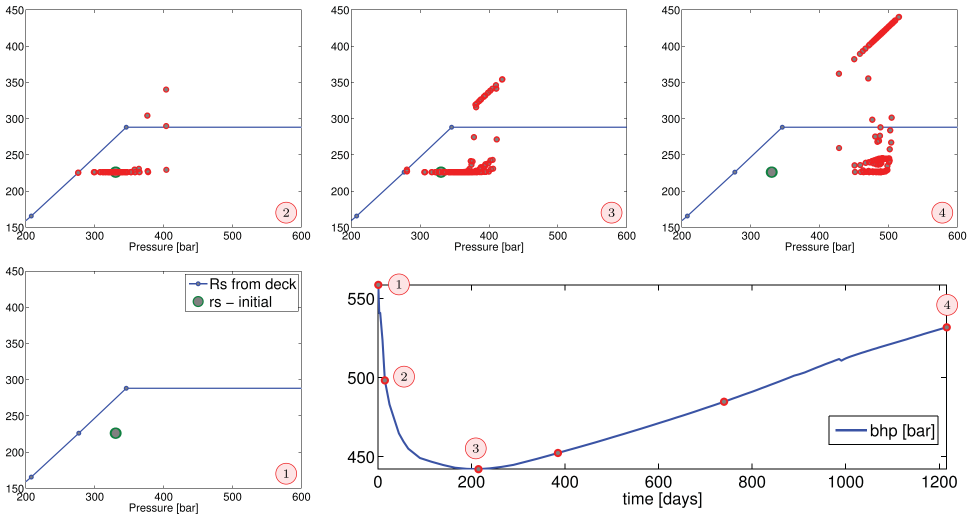


Figure 11.33 Reservoir dynamics of the SPE 1 case during the first three years. The lower-right plot shows the bottom-hole pressure of the injector as function of time. The other plots show the solution gas–oil ratio as function of pressure for all cells in the reservoir at four different instances in time.

gas will be liberated near the producer because of the pressure drawdown. As more gas is injected into the reservoir, more cells transition from an undersaturated to a saturated state, giving an increasing plume of free gas that expands out from the injector along the top of the reservoir (see the left plot in Figure 11.32). The total injection rate is higher than the production rate, and hence the pressure in the reservoir will increase, and after approximately 200 days, we see that the injector bhp starts to increase again to maintain the specified rate.

11.8.2 Comparison against a Commercial Simulator

To build confidence in our simulator, we can verify the simulation in the previous subsection against ECLIPSE. To compare the two, we first extract surface rates and bottom-hole pressures from MRST as plain arrays:

```
[qWs, qOs, qGs, bhp] = wellSolToVector(wellSols);
```

We also extract time steps and determine indices for injector and producer:

```
T    = convertTo(cumsum(schedule.step.val), year);
inj  = find([wellSols{1}.sign] == 1);
prod = find([wellSols{1}.sign] == -1);
```

Output from ECLIPSE is usually stored in so-called *summary/restart files*, which can be used to reinitialize a simulation so that it can be continued. The `deckformat` module in MRST offers useful functionality for reading and interpreting such (binary) output files (see

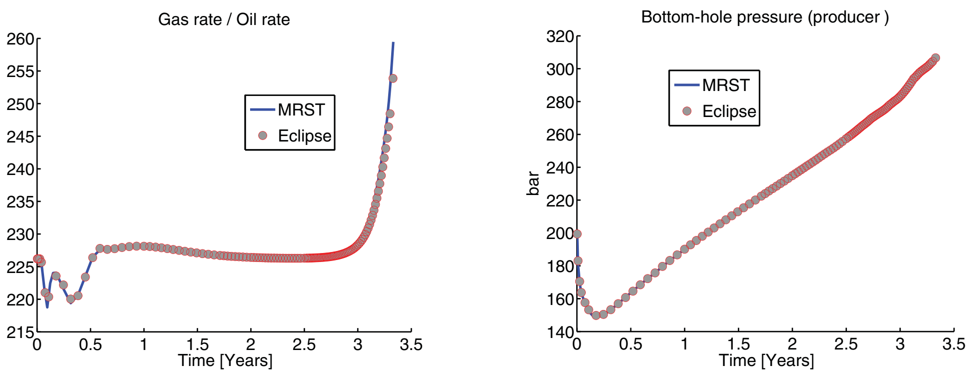


Figure 11.34 Comparison of well responses for the SPE 1 benchmark simulated by MRST and ECLIPSE 100.

`readEclipseSummaryFmt` and `readEclipseSummaryUnFmt`). To simplify the comparison for the SPE 1 benchmark, the data set supplied with MRST includes a `*.mat` file with selected well responses obtained by simulating the exact same input deck with ECLIPSE 100. These data are stored as a MATLAB structure, which also contains a few functions for extracting the raw data. We load the summary

```
load SPE1_smry
ind = 2:size(smry.data,2);
Te  =  smry.get(':+:+:+:+', 'YEARS', ind);
```

Here, we have ignored the first entry, which contains zero values. Figure 11.34 shows a comparison of producer gas–oil ratio and bottom-hole pressure in the producer. These are extracted as follows:

```
eGOR  = convertFrom(smry.get('PRODUCER', 'WGOR', ind), 1000*ft^3/stb)';
eBHPp = convertFrom(smry.get('PRODUCER', 'WBHP', ind), psia)';
eBHPi = convertFrom(smry.get('INJECTOR', 'WBHP', ind), psia)';

mGOR  = qGs(:,prod)./qOs(:,prod);
mBHPp = bhp(:,prod);
mBHPi = bhp(:,inj);
```

As we see from the figure, there is (almost) perfect match between MRST and ECLIPSE, which is what we should expect, given that the two simulators implement (almost) exactly the same numerical methods.

11.8.3 Limitations and Potential Pitfalls

Numerical flow models based on the black-oil equations and their like are complex creatures and should be treated cautiously to ensure that the results they produce make sense.

A commercial simulator usually has a large number of precautions built into it, so that it on one hand will refuse to simulate unphysical situations, but on the other hand manages to give an answer even though the user gives incomplete or inconsistent data. (You may also recall our discussion of time-step chops implemented to ensure convergence when faced with severe nonlinearity, which potentially *could* be the result of an inadequate model or incorrect parameters.) Being aware of limitations and potential pitfalls is important, in particular when using a prototyping tool like MRST that has very few precautions built into it, but also when using mature commercial tools, whose alluring robustness may sometimes be deceptive. In this section, I will try to review a few potential issues and comment on possible precautions.

When Can Black-Oil Models be Applied?

Any model is no better than its assumptions, and it is important to know what the limitations of a model are so that it is not used for the wrong purposes. Black-oil models can, in principle, be applied to simulate all the five types of reservoirs outlined in Section 11.4.2, as long as these are produced by depletion or waterflooding. As a general rule of thumb, you should be more precautious when applying black-oil models to gas injection processes that involve significant changes in the composition of the oil and gas pseudo-components during the displacement. Vaporization and retrograde condensation in rich gases are examples of such processes, and here you should be careful to check the validity of black-oil models against more comprehensive compositional models. To ensure that the black-oil description is sufficiently accurate, the recovery process should satisfy the following characteristics [269]:

- The recovery process should be isothermal.
- The path taken by the reservoir fluids should be far from the critical point.
- The amount of liberated gas or condensate dropout should be a small part of the hydrocarbon in place.
- The remaining hydrocarbon composition should not change significantly when gas is liberated or vaporized condensate drops out.

If these conditions are not satisfied, you should consider using a compositional simulator.

Consistent PVT Parameters

PVT parameters used in black-oil models were originally derived from laboratory analysis of (recombined) reservoir fluids, but in recent years it has become more common to derive the necessary pressure-dependent parameters by simulating differential liberation experiments by use of compositional models with appropriate equations of state. Regardless of their source, the derived PVT model must satisfy certain physical properties; see [73]. First of all, the total compressibility of the system must be non-negative, i.e.,

$$c_t = S_w c_w + S_o c_o + S_g c_g + c_r \geq 0,$$

where c_r is the rock compressibility and c_α is the isothermal compressibility of phase α,

$$c_w = -\frac{B_w'}{B_w}, \qquad c_o = -\frac{B_o'}{B_o} + \frac{(B_g - B_o R_v)R_s'}{B_o(1 - R_s R_v)},$$
$$c_g = -\frac{B_g'}{B_g} + \frac{(B_o - B_g R_s)R_v'}{B_g(1 - R_s R_v)}.$$

Since this property has to be valid for all saturations, it follows that each of the isothermal compressibilities must also be nonnegative, which gives the following three consistency conditions:

$$\begin{aligned} B_w' &\leq 0, \\ \big(1 - R_s R_v\big)B_o' &\leq \big(B_g - R_v B_o\big)R_s', \\ \big(1 - R_s R_v\big)B_g' &\leq \big(B_o - R_s B_b\big)R_v'. \end{aligned} \tag{11.32}$$

To ensure that the partial volumes in (11.11) are also positive, the following additional conditions must be satisfied:

$$R_s R_v < 1, \qquad B_w > 0, \qquad B_g - R_v B_o > 0, \qquad B_o - R_s B_g > 0. \tag{11.33}$$

As we have seen in Section 11.4, B_o, R_s, and R_v are normally increasing functions with positive derivatives, whereas B_g and B_w are decaying functions with negative derivatives.

Use of Tabulated Data

Use of tabulated data, as discussed earlier in Section 11.5, is very common in industry-grade simulation models. This means that when the simulator requires values between data points, these need to be interpolated. Likewise, values requested outside the range of tabulated data need to be extrapolated. Algorithmic choices of how data are interpolated and extrapolated are specific to the individual simulator, but can have a significant impact on both the accuracy and stability of the simulator, and the user should therefore be aware of what methods are used.

The default choice in ECLIPSE is to use linear interpolation and zeroth order extrapolation for rock-fluid properties, so that the simulator picks the value of the first or last tabulated point rather than using trend interpolation based on the closest points. This means that *derivatives* of tabulated quantities are piecewise constants and effectively are computed using finite-differences between tabulated points. PVT properties are also interpolated within their range of definition, but properties like R_s and R_v may also be extrapolated. In MRST, we have chosen to follow the interpolation conventions from ECLIPSE in the `ad-props` module to be able to produce simulation results that are as identical as possible when using the same discretizations and solvers. This raises several issues you should be aware of:

- First of all, you should make sure that the data points cover the value range of the simulation. For PVT properties, this usually means that data should be defined for higher pressures than what are initially found in the reservoir, since pressure can increase during the simulation as a result of fluid injection. In ECLIPSE 100, you can use the keyword `EXTRAPMS` to force the simulator to issue a warning whenever it extrapolates outside of tabulated data. At the time of writing, such a safeguard is not yet implemented in MRST.

- To reduce numerical errors from the numerical interpolation, make sure that data points are sufficiently dense and distributed so that interpolation errors do not vary too much in magnitude throughout the parameter range.
- Try to avoid discontinuities, strong kinks, and other abrupt changes in tabulated properties unless these are dictated by physics. (See, e.g., the relative permeabilities for SPE 9 in Figures 11.6 and 11.7.) If necessary, introduce extra data points near strong nonlinearities and local extremal points to avoid introducing unphysical oscillations and too-large values for derivatives. This will generally improve the convergence of the nonlinear solver and reduce the number of (unnecessary) time-step chops.
- On the other hand, introducing too many points will make table lookup more expensive and contribute to slow down your simulation. For MRST, you should be particularly beware of multiple regions. Property evaluation is vectorized over all cells to make it as efficient as possible, but with multiple relperm or PVT regions, each region must be processed sequentially in a `for` loop, and this can add a significant computational overhead if the number of regions is large.
- In certain cases, ECLIPSE interpolates functions of the input data or functional *combinations* of multiple properties, rather than the tabulated values. The primary example is the PVT data for oil. Here, ECLIPSE 100 interpolates reciprocals of B_o and $\mu_o B_o$, whereas ECLIPSE 300 interpolates reciprocals of B_o, but interpolates μ_o directly. Using different interpolations may have a significant effect on simulation results.

To give you an idea of how things can go wrong, let us revisit two of the SPE benchmark models, starting with SPE 3. To sample the model, we make a 101×21 regular mesh of (p, R_v) values covering the rectangular domain $[p_0, p_1] \times [0, R_v(p_1)]$ for $p_0 = 80$ and $p_1 = 280$ bar. In the input data, properties are only defined for pressures in the interval $[83.75, 237.82]$ bar, and hence we need to use both interpolation and extrapolation to compute values. Figure 11.35 reports the computed gas viscosities for values toward the upper end of the pressure interval. Here, the left plot clearly shows that the viscosity is not necessarily a monotone function of pressure for fixed R_v. More important, the right plot shows that we get incorrect behavior with highly unrealistic viscosities for R_v values less than $R_v(p)$ once the pressure exceeds maximum data values by more than a few percent.

Figure 11.36 shows that the PVT model in SPE 1 P violates the consistency condition $B_o - R_s B_g > 0$ in (11.33) for high pressure values, as observed previously by Coats [73]. By convention, the formation-volume factor B_o of oil is computed by interpolating the shrinkage factor b_o rather than the tabulated B_o values. In the tabulated data, b_o declines with increasing R_s values and will hence cross zero for sufficiently high values. This leads to blowup and negative values for B_o. This behavior is only observed for (p, R_s) values that are larger than what is to be expected in practical computations, but it is still interesting to note this artifact.

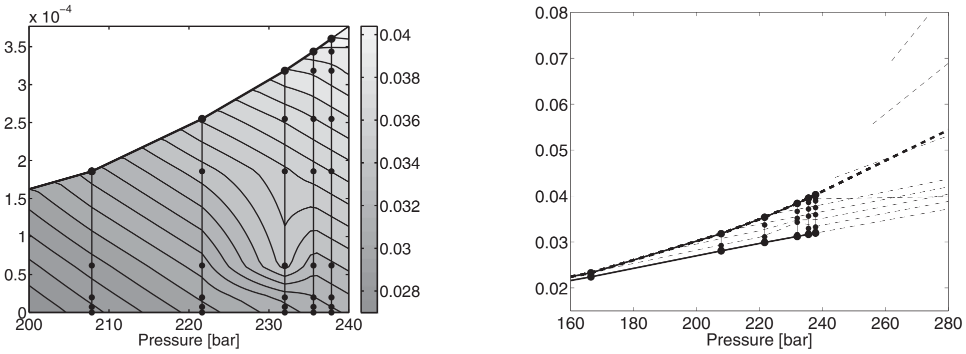


Figure 11.35 Interpolated and extrapolated gas viscosity for the SPE 3 fluid model. The left plot shows contours of μ_g over parts of the (p, R_v) domain, with data points shown as filled circles. The right plot shows μ_g as function of pressure only, with dashed lines corresponding to constant R_v values.

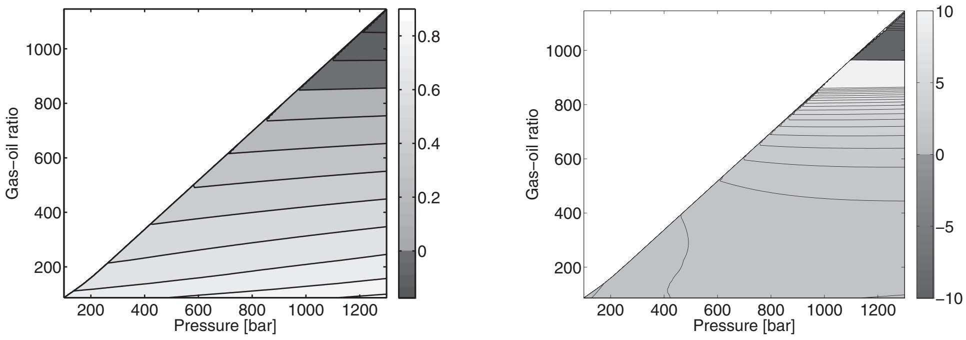


Figure 11.36 Inconsistency in the PVT data of the SPE 1 benchmark giving negative partial volumes. The left plot shows the interpolated shrinkage factor b_o, whereas the right plot shows the resulting values of $B_o - R_s B_g$. To emphasize the blowup and the negative values, the color scale has been cropped to $[-10, 10]$, but the actual values are up to two orders larger.

Initialization and Equilibration

In all examples studied so far in the book, the initial saturation has either been a constant, or the fluid phases have been uniformly segregated throughout the reservoir. In a real reservoir, the initial fluid distribution will usually be more complicated, as we saw for the heterogeneous capillary fringe in Section 10.3.7. Setting the correct initial state is essential to obtain accurate estimates of fluids in place and predict how the reservoir will behave under production. ECLIPSE input offers two ways of setting up the initial state:

- you can either input individual cell values of the initial state using the keywords `PRESSURE`, `SWAT`, `SGAS`, and `RS` or `PBUB` in an ECLIPSE input file, or
- you can calculate the initial state equilibration, i.e., determine it on the basis of hydrostatic equilibrium.

At the time of writing, MRST's support of initialization is somewhat unfinished. All cells in a grid can be assigned specific pressure and saturation values given in a deck by the function `initStateDeck(model,deck)`. The function also performs a full equilibration initialization, proceeding in two steps

```
regions = getInitializationRegionsDeck(model, deck);
state0 = initStateBlackOilAD(model, regions);
```

The first call extracts the different subregions from the `REGIONS` section of the input deck, whereas the second call computes vertical equilibrium inside each region. This part of the software will be unified in the near future.

Let us discuss this initialization in some more detail. In the previous chapter we saw how variations in rock properties can cause variations in capillary pressures. Observed fluid contacts (gas–oil and oil–water) generally show vertical variation and will not always be planar. To determine the free fluid contacts, we must know the phase pressures inside the reservoir. The level of free gas is then defined as the depth at which the gas and oil pressures coincide, i.e., the depth at which the oil–gas capillary pressure is zero. The level of free water is defined analogously.

Typical input data to an equilibration calculation consist of pressure at a datum depth, gas–oil and oil–water contacts, capillary pressure at each contact, and gas–oil ratio (R_s) or bubble-point pressure (p_b) as a function of depth. Given these data, it is, in principle, a straightforward task to determine the fluid distribution, which generally will consist of five different zones:

1. A gas cap, in which gas is the only continuous phase ($S_g = 1 - S_{wi}$)
$$\frac{dp_g}{dz} = g\rho_g, \quad p_o = p_g - P_{cog}(S_g), \quad p_w = p_o - P_{cow}(S_{wi}).$$
2. An oil–gas transition zone, in which both gas and oil are continuous:
$$\frac{dp_g}{dz} = g\rho_g, \quad \frac{dp_o}{dz} = g\rho_o, \quad p_w = p_o - P_{cow}(S_{wi}).$$
The saturations are $S_w = S_{wi}$, $S_g = P_{cog}^{-1}(p_g - p_o)$, and $S_o = 1 - S_g - S_{wi}$.
3. An oil zone, in which oil is the only continuous phase ($S_o = 1 - S_{wi}$):
$$\frac{dp_o}{dz} = g\rho_o, \quad p_g = p_o + P_{cog}(0), \quad p_w = p_o - P_{cow}(S_{wi})$$
4. An oil–water transition zone, in which both oil and water are continuous:
$$\frac{dp_o}{dz} = g\rho_o, \quad \frac{dp_w}{dz} = g\rho_w, \quad p_g = p_o + P_{cog}(0).$$
The saturations are $S_g = 0$, $S_w = P_{cow}^{-1}(p_o - p_w)$, and $S_o = 1 - S_w$.
5. A water zone, in which only water is present:
$$\frac{dp_w}{dz} = g\rho_w, \quad p_o = p_w + P_{cow}(1), \quad p_g = p_o - P_{cog}(0).$$

In practice, however, the equilibration procedure is complicated by the fact that the fluid contacts may vary between different compartments in the reservoir and that capillary pressure curves may vary from one cell to the next. Notice also that numerical artifacts and inaccuracies may arise if the grid is not aligned with the fluid contacts; see e.g., [251].

Once your model is initialized, it is a good precaution to run a simulation with no external drive mechanisms applied to see if the reservoir contains any movable fluids, i.e., check whether the initial phase and pressure distributions remain invariant in time.