An Introduction to the Numerics of Flow in Porous Media using Matlab

Jørg E. Aarnes, Tore Gimse, and Knut–Andreas Lie

Summary. Even though the art of reservoir simulation has evolved through more than four decades, there is still a substantial research activity that aims toward faster, more robust, and more accurate reservoir simulators. Here we attempt to give the reader an introduction to the mathematics and the numerics behind reservoir simulation. We assume that the reader has a basic mathematical background at the undergraduate level and is acquainted with numerical methods, but no prior knowledge of the mathematics or physics that govern the reservoir flow process is needed. To give the reader an intuitive understanding of the equations that model filtration through porous media, we start with incompressible single-phase flow and move step-by-step to the black-oil model and compressible two-phase flow. For each case, we present a basic numerical scheme in detail, before we discuss a few alternative schemes that reflect trends in state-of-the-art reservoir simulation. Two and three-dimensional test cases are presented and discussed. Finally, for the most basic methods we include simple Matlab codes so that the reader can easily implement and become familiar with the basics of reservoir simulation.

1 Introduction

For nearly half a century, reservoir simulation has been an integrated part of oil-reservoir management. Today, simulations are used to estimate production characteristics, calibrate reservoir parameters, visualise reservoir flow patterns, etc. The main purpose is to provide an information database that can help the oil companies to position and manage wells and well trajectories in order to maximise the oil and gas recovery. Unfortunately, obtaining an accurate prediction of reservoir flow scenarios is a difficult task. One of the reasons is that we can never get a complete and accurate characterization of the rock parameters that influence the flow pattern. And even if we did, we would not be able to run simulations that exploit all available information, since this would require a tremendous amount of computer resources that exceed by far the capabilities of modern multi-processor computers. On the other hand, we do not need, nor do we seek a simultaneous description of the flow scenario on all scales down to the pore scale. For reservoir management it is usually sufficient to describe the general trends in the reservoir flow pattern.

In the early days of the computer, reservoir simulation models were built from two-dimensional slices with 10^2–10^3 grid cells representing the whole reservoir. In contrast, reservoir characterizations today model the porous rock formations by the means of grid-blocks down to the meter scale. This gives three-dimensional models consisting of multi-million cells. Despite an astonishing increase in computer power, and intensive research on computation techniques, commercial reservoir simulators can seldom run simulations directly on geological grid models. Instead, coarse grid models with grid-blocks that are typically ten to hundred times larger are built using some kind of upscaling of the geophysical parameters. How one should perform this upscaling is not trivial. In fact, upscaling has been, and probably still is, one of the most active research areas in the oil industry (see e.g., [7, 14, 15, 32]). This effort reflects that it is a general opinion that, with the ever increasing size and complexity of the geological reservoir models, we cannot run simulations directly on these models in the foreseeable future.

Along with the development of better computers, new and more robust upscaling techniques, and more detailed reservoir characterizations, there has also been an equally significant development in the area of numerical methods. State-of-the-art simulators employ numerical methods that can take advantage of multiple processors, distributed memory workstations, adaptive grid refinement strategies, and iterative techniques with linear complexity. For the simulation, there exists a catalogue of different numerical schemes that all have their pros and cons. With all these techniques available we see a trend where methods are being tuned to a special set of applications, as opposed to traditional methods that were developed for a large class of differential equations. As an example, we mention the recent multiscale numerical methods [4, 5, 12, 19, 20, 25] that are specially suited for differential equations whose solutions may display a multiple scale structure. Although these methods resemble numerical schemes obtained from an upscaling procedure, they are somewhat more rigorous in the sense that they exploit fine-scale information in a mathematically more consistent manner.

It is possible that multiscale methods can help bridge the gap between the size of the geological grid and the size of the simulation grid in reservoir simulation. This type of contribution would certainly take reservoir simulation a big leap forward, and could simplify reservoir management workflow considerably. However, although upscaling is undoubtedly an important part of reservoir simulation technology, neither upscaling nor multiscale techniques will be discussed here. This part of the reservoir simulation framework is discussed separately elsewhere in this book [1]. Here our purpose is to present a self-contained tutorial on reservoir simulation. The main idea is to let the reader become familiar with the mathematics behind porous media flow simulation, and present some basic numerical schemes that can be used to solve the governing partial differential equations. Moreover, to ease the transition from theory to implementation, we supply compact Matlab codes for some of the presented methods applied to Cartesian grids. We hope that this material can give students or researchers about to embark on, for instance, a Master project or a PhD project, a head start.

We assume that the reader has knowledge of mathematics and numerics at the undergraduate level in applied mathematics, but we do not assume any prior knowledge of reservoir simulation. We therefore start by giving a crash course on the physics and mechanics behind reservoir simulation in Section 2. This section presents and explains the role of the various geophysical parameters and indicates how they are obtained. In Section 3 we present the basic mathematical model in its simplest form: the single-phase flow model giving an equation for the fluid pressure. Here we also present some numerical schemes for solving the pressure equation. Then, in Section 4 we move on to multiphase flow simulation. Multiphase flows give rise to a coupled system consisting of a (nearly) elliptic *pressure equation* and a set of (nearly) hyperbolic mass transport equations, so-called *saturation equations.* To balance all terms in these equations properly is a difficult task, and requires quite a bit of bookkeeping. Therefore, to enhance readability, we make some simplifying assumptions before we start to discretise the equations. Section 5 is therefore devoted to immiscible two-phase flow where gas is allowed to be dissolved in the oleic phase. These assumptions give rise to what is known as the *black-oil model.* We then show how to discretise both the pressure equation and the saturation equations separately, and explain how to deal with the coupling between the pressure equation and the saturation equations. Finally, in Section 6, we make some additional assumptions and present a Matlab code for a full two-phase flow simulator. For illustration purposes, the simulator is applied to some two-dimensional test-cases extracted from a strongly heterogeneous reservoir model that was used as a benchmark for upscaling techniques [14].

2 A Crash Course on the Physics and Mechanics behind Reservoir Simulation

The purpose of this section is to briefly summarise some aspects of the art of modelling porous media flow and motivate a more detailed study on some of the related topics. More details can be found in one of the general textbooks describing modelling of flow in porous media, e.g., [6, 11, 16, 21, 28, 30, 34]. For a newcomer in the field, we can in particular recommend the recent book by Chen, Huan, and Ma [13].

Over a period of millions of years, layers of sediments containing organic material built up in the area that is now below the North Sea. A few centimetres every hundred years piled up to hundreds and thousands of meters and made the pressure and temperature increase. Simultaneously, severe geological activity took place. Cracking of continental plates and volcanic activity changed the area from being a relatively smooth, layered plate into a complex structure where previously continuous layers were cut, shifted, or twisted in various directions. As the organic material was compressed, it eventually turned into a number of different hydrocarbons. Gravity separated trapped water and the hydrocarbon components. The lightest hydrocarbons (methane, ethane, etc.) usually escaped quickly, whilst the heavier oils moved slowly towards the surface. At certain cites, however, the geological activity had created and bent layers of low-permeable (or non-permeable) rock, so that the migrating hydrocarbons were trapped. These are today's oil and gas reservoirs in the North Sea, and they are typically found at about 1 000–3 000 meters below the sea bed.

Although we speak about a porous medium, we should remember it is solid rock. However, almost any naturally formed rock contains pores, and the distribution and volume fraction of such pores determine the rock properties, which in turn are the parameters governing the hydrocarbon flow in the reservoir.

2.1 Porosity

The rock *porosity*, usually denoted by ϕ, is the void volume fraction of the medium, that is, $0 \leq \phi < 1$. The porosity usually depends on the pressure; the rock is *compressible*, and the rock compressibility is defined by:

$$c_r = \frac{1}{\phi}\frac{d\phi}{dp}, \tag{1}$$

where p is the overall reservoir pressure. For simplified models, it is customary to neglect the rock compressibility and assume that ϕ only depends on the spatial coordinate. If compressibility cannot be neglected, it is common to use a linearization so that:

$$\phi = \phi_0\big(1 + c_r(p - p_0)\big). \tag{2}$$

For a North Sea reservoir, ϕ is typically in the range 0.1–0.3, and compressibility can be significant, as e.g., witnessed by the subsidence in the Ekofisk area. Since the dimension of the pores is very small compared to any interesting scale for reservoir simulation, one normally assumes that porosity is a piecewise continuous spatial function. However, ongoing research aims to understand better the relation between flow models on pore scale and on reservoir scale.

2.2 Permeability

The (absolute) *permeability*, denoted by $\mathbf{K}$, is a measure of the rock's ability to transmit a single fluid at certain conditions. Since the orientation and interconnection of the pores are essential for flow, the permeability is not necessarily proportional to the porosity, but $\mathbf{K}$ is normally strongly correlated to ϕ.

Rock formations like sandstones tend to have many large or well-connected pores and therefore transmit fluids readily. They are therefore described as permeable. Other formations, like shales, may have smaller, fewer or less interconnected pores and are hence described as impermeable. Although the SI-unit for permeability is m^2, it is commonly represented in Darcy (D), or milli-Darcy (mD). The precise definition of 1D ($\approx 0.987 \cdot 10^{-12}\,\mathrm{m}^2$) involves transmission of a 1cp fluid (see below) through a homogeneous rock at a speed of 1cm/s due to a pressure gradient of 1atm/cm. Translated to reservoir conditions, 1D is a relatively high permeability.

In general, $\mathbf{K}$ is a tensor, which means that the permeability in the different directions depends on the permeability in the other directions. However, by a change of basis, $\mathbf{K}$ may sometimes be diagonalised, and due to the reservoir structure, horizontal and vertical permeability suffices for several models. We say that the medium is isotropic (as opposed to anisotropic) if $\mathbf{K}$ can be represented as a scalar function, e.g., if the horizontal permeability is equal to the vertical permeability. Moreover, due to transitions between different rock formations, the permeability may vary rapidly over several orders of magnitude, local variations in the range 1 mD to 10 D are not unusual in a typical field.

Production (or measurements) may also change the permeability. When temperature and pressure is changed, microfractures may open and significantly change the permeability. Furthermore, since the definition of permeability involves a certain fluid, different fluids will experience different permeability in the same rock sample. Such rock-fluid interactions are discussed below.

2.3 Fluid Properties

The void in the porous medium is assumed to be filled with the different *phases*. The volume fraction occupied by each phase is the *saturation* (s) of that phase. Thus,

$$\sum_{\text{all phases}} s_i = 1. \tag{3}$$

For practical reservoir purposes, usually only three phases are considered; aqueous (w), oleic (o), and gaseous (g) phase. Each phase contains one or more *components*. A hydrocarbon component is a unique chemical species (methan, ethane, propane, etc). Since the number of hydrocarbon components can be quite large, it is common to group components into psuedo-components. Henceforth we will make no distinction between components and pseudo-components.

Due to the varying and extreme conditions in a reservoir, the hydrocarbon composition of the different phases may change throughout a simulation (and may sometimes be difficult to determine uniquely). The mass fraction of component i in phase j is denoted by c_{ij}. In each of the phases, the mass fractions should add up to unity, so that for N different components, we have:

$$\sum_{i=1}^{N} c_{ig} = \sum_{i=1}^{N} c_{io} = \sum_{i=1}^{N} c_{iw} = 1. \tag{4}$$

Next we assign a *density* ρ and a *viscosity* μ to each phase. In general, these are functions of *phase pressure* p_i $(i = g, o, w)$ and the composition of each phase. That is, for gas

$$\rho_g = \rho_g(p_g, c_{1g}, ..., c_{Ng}), \qquad \mu_g = \mu_g(p_g, c_{1g}, ..., c_{Ng}), \tag{5}$$

and similarly for the other phases. These dependencies are most important for the gas phase, and are usually ignored for the water phase.

The compressibility of the phase is defined as for rock compressibility:

$$c_i = \frac{1}{\rho_i}\frac{d\rho_i}{dp_i}, \quad i = g, o, w. \tag{6}$$

Compressibility effects are more important for gas than for oil and water. In simplified models, water compressibility is therefore usually neglected.

Due to interfacial tensions, the phase pressures are different, defining the *capillary pressure*,

$$p_{cij} = p_i - p_j, \tag{7}$$

for $i, j = g, o, w$. Although other dependencies are reported, it is usually assumed that the capillary pressure is a function of the saturations only.

Other parameters of importance are the bubble-point pressures for the various components. At given temperature, the bubble-point pressures signify the pressures where the respective phases start to boil. Below the bubble-point pressures, gas is released and we get transition of the components between the phases. For most realistic models, even if we do not distinguish between all the components, one allows gas to be dissolved in oil. For such models, an important pressure-dependent parameter is the solution gas-oil ratio R_s for the gas dissolved in oil at reservoir conditions. It is also common to introduce so-called formation volume factors (B_w, B_o, B_g) that model the pressure dependent ratio of bulk volumes at reservoir and surface conditions. Thermodynamical behaviour is, however, a very complex topic that is usually significantly simplified or even ignored in reservoir simulation. Thermodynamics will therefore not be discussed any further here.

2.4 Relative Permeabilities

Even though phases do not really mix, for macroscale modelling, we assume that all phases may be present at the same location. Thus, it turns out that the ability of one phase to move depends on the environment at the actual location. That is, the permeability experienced by one phase depends on the saturation of the other phases at that specific location, as well as the phases' interaction with the pore walls. Thus, we introduce a property called *relative* *permeability*, denoted by $k_{ri}, i = g, o, w$, which describes how one phase flows in the presence of the two others. Thus, in general, and by the closure relation (3), we may assume that

$$k_{ri} = k_{ri}(s_g, s_o), \tag{8}$$

where subscript r stands for *relative* and i denotes one of the phases g, o, or w. Thus, the (effective) permeability experienced by phase i is $\mathbf{K}_i = \mathbf{K}k_{ri}$. It is important to note that the relative permeabilities are nonlinear functions of the saturations, so that the sum of the relative permeabilities at a specific location (with a specific composition) is not necessarily equal to one. In general, relative permeabilities may depend on the pore-size distribution, the fluid viscosity, and the interfacial forces between the fluids. These features, which are carefully reviewed by Demond and Roberts [18], are usually ignored. Of greater importance to oil recovery is probably the temperature dependency [29], which may be significant, but very case-related.

Another effect is that due to the adsorption at pore walls and creation of isolated, captured droplets, the relative permeability curves do not extend all over the interval $[0, 1]$. The smallest saturation where a phase is mobile is called the *residual* saturation. The adsorption effects may vary, and this may have important effects. Particularly for simulation of polymer injection, adsorption will occur and have significant impact on the results. The adsorption of a component is usually a nonlinear function, assumed to depend on the rock matrix, and the concentration of the actual component. Microscopic rock-fluid interactions also imply that the rock absolute permeability is not a well-defined property. Liquids obey no-slip boundary conditions, whilst gases may not experience the same effects (Klinkenberg effect).

Obtaining measurements of the quantities discussed above, is very difficult. Particularly the relative permeability measurements are costly and troublesome [24]. Recently, the laboratory techniques have made great progress by using computer tomography and nuclear magnetic resonance (NMR) to scan the test cores where the actual phases are being displaced. Although standard experimental procedures exist for measuring two-phase relative permeabilities (systems where only two phases are present), there is still usually a significant uncertainty concerning the relevance of the experimental values found and it is difficult to come up with reliable data to be used in a simulator. This is mainly due to boundary effects. Particularly for three-phase systems, no reliable experimental technique exists. Thus, three-phase relative permeabilities are usually modelled using two-phase measurements, for which several theoretical models have been proposed. Most of them are based on Stone's model [33], where sets of two-phase relative permeabilities are combined to give three-phase data [17]. Recently, a mathematically more plausible model has been proposed [26, 27] for the case with no gravity (or with gravity and no counter-current flow). Contrary to the Stone models, the latter exhibits no elliptic regions for the resulting conservation laws.

The uncertainty regarding relative permeability is, however, modest compared to the uncertainty imposed by the sparsity of absolute permeability data. To measure $\mathbf{K}$, core samples from the rock are used. These may be about 10 cm in diameter, taken from vertical test wells at every 25 cm of depth. Clearly these samples have negligible volume compared to the entire oil reservoir extending over several kilometres. From outcrops and mines it is known that rock permeability may vary widely, indicating that the modelling of permeability for a reservoir based on core samples is a tremendously difficult problem. Some additional information may be gained from seismic measurements, but with the technology available today, only large scale structures may be found by this technique. Therefore, we have very limited information about how the subsurface reservoirs look like. To meet this problem, stochastic methods have been applied extensively in the field of reservoir description; see Haldorsen and MacDonald [23] and the references therein. Moreover, when comparing and adjusting parameters as real-life production starts, permeability models can be tuned to fit the actual production data, and thereby hopefully improve the original models and predict future production better. Such approaches are called *history matching.*

2.5 Production Processes

Initially, a hydrocarbon reservoir is at equilibrium, and contains gas, oil, and water, separated by gravity. This equilibrium has been established over millions of years with gravitational separation and geological and geothermal processes. When a well is drilled through the upper non-permeable layer and penetrates the upper hydrocarbon cap, this equilibrium is immediately disturbed. The reservoir is usually connected to the the well and surface production facilities by a set of valves. If there were no production valves to stop the flow, we would have a “blow out” since the reservoir is usually under a high pressure. As the well is ready to produce, the valves are opened slightly, and hydrocarbons flow out of the reservoir due to over-pressure. This in turn, sets up a flow inside the reservoir and hydrocarbons flow towards the well, which in turn may induce gravitational instabilities. Also the capillary pressures will act as a (minor) driving mechanism, resulting in local perturbations of the situation.

During the above process, perhaps 20 percent of the hydrocarbons present are produced until a new equilibrium is achieved. We call this *primary production* by natural drives. One should note that a sudden drop in pressure also may have numerous other intrinsic effects. Particularly in complex, composite systems this may be the case, as pressure-dependent parameters experience such drops. This may give non-convective transport and phase transfers, as vapour and gaseous hydrocarbons may suddenly condensate.

As pressure drops, less oil and gas is flowing, and eventually the production is no longer economically sustainable. Then the operating company may start *secondary production*, by engineered drives. These are processes based on injecting water or gas into the reservoir. The reason for doing this is twofold; some of the pressure is rebuilt or even increased, and secondly one tries to push out more profitable hydrocarbons with the injected substance. One may perhaps produce another 20 percent of the oil by such processes, and engineered drives are standard procedure at most locations in the North Sea today.

In order to produce even more oil, *Enhanced Oil Recovery* (EOR, or tertiary recovery) techniques may be employed. Among these are heating the reservoir or injection of sophisticated substances like foam, polymers or solvents. Polymers are supposed to change the flow properties of water, and thereby to more efficiently push out oil. Similarly, solvents change the flow properties of the hydrocarbons, for instance by developing miscibility with an injected gas. In some sense, one tries to wash the pore walls for most of the remaining hydrocarbons. The other technique is based on injecting steam, which will heat the rock matrix, and thereby, hopefully, change the flow properties of the hydrocarbons. At present, such EOR techniques are considered too expensive for large scale commercial use, but several studies have been conducted and the mathematical foundations are being carefully investigated, and at smaller scales EOR is being performed.

One should note that the terms primary, secondary, and tertiary are ambiguous. EOR techniques may be applied during primary production, and secondary production may be performed from the first day of production.

3 Incompressible Single-Phase Flow

The simplest possible way to describe the displacements of fluids in a reservoir is by a single-phase model. This model gives an equation for the pressure distribution in the reservoir and is used for many early-stage and simplified flow studies. Single-phase models are used to identify flow directions; identify connections between producers and injectors; in flow-based upscaling; in history matching; and in preliminary model studies.

3.1 Basic Model

Assume that we want to model the filtration of a fluid through a porous medium of some kind. The basic equation describing this process is the continuity equation which states that mass is conserved

$$\frac{\partial(\phi\rho)}{\partial t} + \nabla\cdot(\rho v) = q. \tag{9}$$

Here the source term q models sources and sinks, that is, outflow and inflow per volume at designated well locations.

For low flow velocities v, filtration through porous media is modelled with an empirical relation called Darcy's law after the French engineer Henri Darcy.

Darcy discovered in 1856, through a series of experiments, that the filtration velocity is proportional to a combination of the gradient of the fluid pressure and pull-down effects due to gravity. More precisely, the volumetric flow density v (which we henceforth will refer to as flow velocity) is related to pressure p and gravity forces through the following gradient law:

$$v = -\frac{\mathbf{K}}{\mu}(\nabla p + \rho g \nabla z). \tag{10}$$

Here $\mathbf{K}$ is the permeability, μ is the viscosity, g is the gravitational constant and z is the spatial coordinate in the upward vertical direction. For brevity we shall in the following write $G = -g\nabla z$ for the gravitational pull-down force.

In most real field geological reservoir models, $\mathbf{K}$ is an anisotropic diagonal tensor. Unfortunately, as mentioned in the introduction, commercial reservoir simulators can seldom run simulations directly on these geological models. Effects of small-scale heterogeneous structures are therefore incorporated into simulation models by designing effective permeability tensors that represent impact of small-scale heterogeneous structures on a coarser grid, as discussed in more detail elsewhere in this book [1]. These effective, or upscaled, permeability tensors try to model principal flow directions, which may not be aligned with the spatial coordinate directions, and can therefore be full tensors. Hence, here we assume only that $\mathbf{K}$ is a symmetric and positive definite tensor, whose eigenvalues are bounded uniformly below and above by positive constants.

We note that Darcy's law is analogous to Fourier's law of heat conduction (in which $\mathbf{K}$ is replaced with the heat conductivity tensor) and Ohm's law of electrical conduction (in which $\mathbf{K}$ is the inverse of the electrical resistance). However, whereas there is only one driving force in thermal and electrical conduction, there are two driving forces in porous media flow: gravity and the pressure gradient. Since gravity forces are approximately constant inside a reservoir domain Ω, we need only use the reservoir field pressure as our primary unknown. To solve for the pressure, we combine Darcy's law (10) with the continuity equation (9).

To illustrate, we derive an equation that models flow of a fluid, say, water (w), through a porous medium characterised by a permeability field $\mathbf{K}$ and a corresponding porosity distribution. For simplicity, we assume that the porosity ϕ is constant in time and that the flow can be adequately modelled by assuming incompressibility, i.e., constant density. Then the temporal derivative term in (9) vanishes and we obtain the following elliptic equation for the water pressure:

$$\nabla \cdot v_w = \nabla \cdot \left[-\frac{\mathbf{K}}{\mu_w}(\nabla p_w - \rho_w G) \right] = \frac{q_w}{\rho_w}. \tag{11}$$

To close the model, we must specify boundary conditions. Unless stated otherwise we shall follow common practice and use no-flow boundary conditions.

Hence, on the reservoir boundary $\partial\Omega$ we impose $v_w \cdot n = 0$, where n is the normal vector pointing out of the boundary $\partial\Omega$. This gives us an isolated flow system where no water can enter or exit the reservoir.

In the next subsections we restrict our attention to incompressible flows and present several numerical methods for solving (11). To help the interested reader with the transition from theory to implementation, we also discuss some simple implementations in Matlab for uniform Cartesian grids. We first present a cell-centred finite-volume method, which is sometimes referred to as the *two-point flux-approximation (TPFA) scheme*. Although this scheme is undoubtedly one of the simplest discretisation techniques for elliptic equations, it is still widely used in the oil-industry.

3.2 A Simple Finite-Volume Method

In classical finite-difference methods, partial differential equations (PDEs) are approximated by replacing the partial derivatives with appropriate divided differences between point-values on a discrete set of points in the domain. Finite-volume methods, on the other hand, have a more physical motivation and are derived from conservation of (physical) quantities over cell volumes. Thus, in a finite-volume method the unknown functions are represented in terms of average values over a set of finite-volumes, over which the integrated PDE model is required to hold in an averaged sense.

Although finite-difference and finite-volume methods have fundamentally different interpretation and derivation, the two labels are used interchangeably in the scientific literature. We therefore choose to not make a clear distinction between the two discretisation techniques here. Instead we ask the reader to think of a finite-volume method as a conservative finite-difference scheme that treats the grid cells as control volumes. In fact, there exist several finite-volume and finite-difference schemes of low order, for which the cell-centred values obtained with the finite-difference schemes coincide with cell averages obtained with the corresponding finite-volume schemes. The two representations of the unknown functions will therefore also be used interchangeably henceforth.

To derive a set of finite-volume mass-balance equations for (11), denote by Ω_i a grid cell in Ω and consider the following integral over Ω_i:

$$\int_{\Omega_i} \left(\frac{q_w}{\rho_w} - \nabla \cdot v_w \right) dx = 0. \tag{12}$$

Invoking the divergence theorem, assuming that v_w is sufficiently smooth, equation (12) transforms into the following mass-balance equation:

$$\int_{\partial\Omega_i} v_w \cdot n \, d\nu = \int_{\Omega_i} \frac{q_w}{\rho_w} \, dx. \tag{13}$$

Here n denotes the outward-pointing unit normal on $\partial\Omega_i$. Corresponding finite-volume methods are now obtained by approximating the pressure p_w

with a cell-wise constant function $\mathbf{p}_w = \{p_{w,i}\}$ and estimating $v_w \cdot n$ across cell interfaces $\gamma_{ij} = \partial\Omega_i \cap \partial\Omega_j$ from a set of neighbouring cell pressures.

To formulate the standard two-point flux-approximation (TPFA) finite-volume scheme that is frequently used in reservoir simulation, it is convenient to reformulate equation (11) slightly, so that we get an equation on the form

$$-\nabla \cdot \lambda \nabla u = f, \tag{14}$$

where $\lambda = \mathbf{K}/\mu_w$. To this end, we have two options: we can either introduce a flow potential $u_w = p_w + \rho_w g z$ and express our model as an equation for u_w

$$-\nabla \cdot \lambda \nabla u_w = \frac{q_w}{\rho_w},$$

or we can move the gravity term $\nabla \cdot (\lambda \rho_w G)$ to the right-hand side. Hence, we might as well assume that we want to solve equation (14) for u.

As the name suggests, the TPFA scheme uses two points, the cell-averages u_i and u_j, to approximate the flux $v_{ij} = -\int_{\gamma_{ij}} (\lambda \nabla u) \cdot n \, d\nu$. To be more specific, let us consider a regular hexahedral grid with gridlines aligned with the principal coordinate axes. Moreover, assume that γ_{ij} is an interface between adjacent cells in the x–coordinate direction so that $n_{ij} = (1,0,0)^T$. The gradient ∇u on γ_{ij} in the TPFA method is now replaced with

$$\delta u_{ij} = \frac{2(u_j - u_i)}{\Delta x_i + \Delta x_j}, \tag{15}$$

where Δx_i and Δx_j denote the respective cell dimensions in the x-coordinate direction. Thus, we obtain the following expression for v_{ij}:

$$v_{ij} = \delta u_{ij} \int_{\gamma_{ij}} \lambda d\nu = \frac{2(u_j - u_i)}{\Delta x_i + \Delta x_j} \int_{\gamma_{ij}} \lambda d\nu.$$

However, in most reservoir simulation models, the permeability $\mathbf{K}$ is cell-wise constant, and hence not well-defined at the interfaces. This means that we also have to approximate λ on γ_{ij}. In the TPFA method this is done by taking a distance-weighted harmonic average of the respective directional cell permeabilities, $\lambda_{i,ij} = n_{ij} \cdot \lambda_i n_{ij}$ and $\lambda_{j,ij} = n_{ij} \cdot \lambda_j n_{ij}$. To be precise, the n_{ij}–directional permeability λ_{ij} on γ_{ij} is computed as follows:

$$\lambda_{ij} = (\Delta x_i + \Delta x_j) \left(\frac{\Delta x_i}{\lambda_{i,ij}} + \frac{\Delta x_j}{\lambda_{j,ij}} \right)^{-1},$$

Hence, for orthogonal grids with gridlines aligned with the coordinate axes, one approximates the flux v_{ij} in the TPFA method in the following way:

$$v_{ij} = -|\gamma_{ij}| \lambda_{ij} \delta u_{ij} = 2|\gamma_{ij}| \left(\frac{\Delta x_i}{\lambda_{i,ij}} + \frac{\Delta x_j}{\lambda_{j,ij}} \right)^{-1} (u_i - u_j). \tag{16}$$

Finally, summing over all interfaces to adjacent cells, we get an approximation to $\int_{\partial\Omega_i} v_w \cdot n \, d\nu$, and the associated TPFA method is obtained by requiring the mass balance equation (13) to be fulfilled for each grid cell $\Omega_i \in \Omega$.

In the literature on finite-volume methods it is common to express the flux v_{ij} in a more compact form than we have done in (16). Terms that do not involve the cell potentials u_i are usually gathered into an interface transmissibility t_{ij}. For the current TPFA method the transmissibilities are defined by:

$$t_{ij} = 2|\gamma_{ij}| \left(\frac{\Delta x_i}{\lambda_{i,ij}} + \frac{\Delta x_j}{\lambda_{j,ij}} \right)^{-1}.$$

Thus by inserting the expression for t_{ij} into (16), we see that the TPFA scheme for equation (14), in compact form, seeks a cell-wise constant function $\mathbf{u} = \{u_i\}$ that satisfies the following system of equations:

$$\sum_j t_{ij}(u_i - u_j) = \int_{\Omega_i} f \, dx, \qquad \forall \Omega_i \subset \Omega, \tag{17}$$

This system is clearly symmetric, and a solution is, as for the continuous problem, defined up to an arbitrary constant. The system is made positive definite, and symmetry is preserved, by forcing $u_1 = 0$, for instance. That is, by adding a positive constant to the first diagonal of the matrix $\mathbf{A} = [a_{ik}]$, where:

$$a_{ik} = \begin{cases} \sum_j t_{ij} & \text{if } k = i, \\ -t_{ik} & \text{if } k \neq i, \end{cases}$$

The matrix $\mathbf{A}$ is sparse, consisting of a tridiagonal part corresponding to the x-derivative, and two off-diagonal bands corresponding to the y-derivatives.

A short Matlab code for the implementation of (17) on a uniform Cartesian grid is given in Listing 1. In a general code, one would typically assemble the coefficient matrix by looping over all the interfaces of each grid block and for each interface, compute the associated transmissibilities and add them into the correct position of the system matrix. Similarly, the interface fluxes would be computed by looping over all grid block interfaces. Such an approach would easily be extensible to other discretisation (e.g., the MPFA method discussed in the next section) and to more complex grids (e.g., unstructured and faulted grids). Our code has instead been designed to be compact and efficient, and therefore relies entirely on the Cartesian grid structure to set up the transmissibilities in the coefficient matrix using Matlab's inherent vectorization.

In the code, `K` is a 3×`Nx`×`Ny`×`Nz` matrix holding the three diagonals of the tensor λ for each grid cell. The coefficient matrix `A` is sparse (use e.g., `spy(A)` to visualise the sparsity pattern) and is therefore generated with Matlab's built-in sparse matrix functions. Understanding the Matlab code should be rather straightforward. The only point worth noting is perhaps that the constant that we use to force $u_1 = 0$ in element `A(1,1)` is taken as the sum of the diagonals in λ. This is done in order to control that this extra equation does not have an adverse effect on the condition number of `A`.

Listing 1. TPFA finite-volume discretisation of $-\nabla(K(x)\nabla u) = q$.

```
function [P,V]=TPFA(Grid,K,q)

% Compute transmissibilities by harmonic averaging.
Nx=Grid.Nx; Ny=Grid.Ny; Nz=Grid.Nz; N=Nx*Ny*Nz;
hx=Grid.hx; hy=Grid.hy; hz=Grid.hz;
L = K.^(−1);
tx = 2*hy*hz/hx; TX = zeros(Nx+1,Ny,Nz);
ty = 2*hx*hz/hy; TY = zeros(Nx,Ny+1,Nz);
tz = 2*hx*hy/hz; TZ = zeros(Nx,Ny,Nz+1);
TX(2:Nx,:,:) = tx./(L(1,1:Nx−1,:,:)+L(1,2:Nx,:,:));
TY(:,2:Ny,:) = ty./(L(2,:,1:Ny−1,:)+L(2,:,2:Ny,:));
TZ(:,:,2:Nz) = tz./(L(3,:,:,1:Nz−1)+L(3,:,:,2:Nz));

% Assemble TPFA discretization matrix.
x1 = reshape(TX(1:Nx,:,:),N,1); x2 = reshape(TX(2:Nx+1,:,:),N,1);
y1 = reshape(TY(:,1:Ny,:),N,1); y2 = reshape(TY(:,2:Ny+1,:),N,1);
z1 = reshape(TZ(:,:,1:Nz),N,1); z2 = reshape(TZ(:,:,2:Nz+1),N,1);
DiagVecs = [−z2,−y2,−x2,x1+x2+y1+y2+z1+z2,−x1,−y1,−z1];
DiagIndx = [−Nx*Ny,−Nx,−1,0,1,Nx,Nx*Ny];
A = spdiags(DiagVecs,DiagIndx,N,N);
A(1,1) = A(1,1)+sum(Grid.K(:,1,1,1));

% Solve linear system and extract interface fluxes.
u = A\q;
P = reshape(u,Nx,Ny,Nz);
V.x = zeros(Nx+1,Ny,Nz);
V.y = zeros(Nx,Ny+1,Nz);
V.z = zeros(Nx,Ny,Nz+1);
V.x(2:Nx,:,:) = (P(1:Nx−1,:,:)−P(2:Nx,:,:)).*TX(2:Nx,:,:);
V.y(:,2:Ny,:) = (P(:,1:Ny−1,:)−P(:,2:Ny,:)).*TY(:,2:Ny,:);
V.z(:,:,2:Nz) = (P(:,:,1:Nz−1)−P(:,:,2:Nz)).*TZ(:,:,2:Nz);
```

Example 1. In the first example we consider a homogeneous and isotropic permeability $K \equiv 1$ for all $x \in \mathbb{R}^2$. We place an injection well at the origin and production wells at the points $(\pm 1, \pm 1)$ and specify no-flow conditions at the boundaries. These boundary conditions give the same flow as if we repeated the five-spot well pattern to infinity in every direction. The flow in the five-spot is symmetric about both the coordinate axes. We can therefore reduce the computational domain to a quarter, and use e.g., the unit box $\Omega = [0,1]^2$. The corresponding problem is called a *quarter-five spot* problem, and is a standard test-case for numerical methods in reservoir simulation.

Figure 1 shows pressure contours. The pressure P is computed by the following lines for a 8×8 grid:

```
>> Grid.Nx=8; Grid.hx=1/Grid.Nx;
>> Grid.Ny=8; Grid.hy=1/Grid.Ny;
>> Grid.Nz=1; Grid.hz=1/Grid.Nz;
>> Grid.K=ones(3,Grid.Nx,Grid.Ny);
>> N=Grid.Nx*Grid.Ny*Grid.Nz; q=zeros(N,1); q([1 N])=[1 −1];
>> P=TPFA(Grid,Grid.K,q);
```

Here the command `P=TPFA(···)` assigns to `P` the first (the local variable `U`) of the four output variables `U, FX, FY`, and `FZ`.

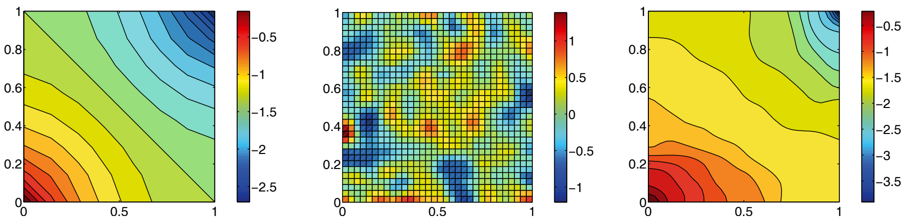


Fig. 1. The left plot shows pressure contours for a homogeneous quarter-five spot. The middle plot shows logarithm of the permeability for the heterogeneous quarter-five spot and the right plot the corresponding pressure distribution. As particles flow in directions of decreasing pressure gradient, the pressure decays from the injector in the lower-left to the producer in the upper-right corner.

Next we increase the number of grid-cells in each direction from eight to 32 and consider a slightly more realistic permeability field obtained from a log-normal distribution.

```
>> Grid.Nx=32; Grid.hx=1/Grid.Nx;
>> Grid.Ny=32; Grid.hy=1/Grid.Ny;
>> Grid.Nz=1; Grid.hz=1/Grid.Nz;
>> Grid.K=exp(5*smooth3(smooth3(randn(3,Grid.Nx,Grid.Ny))));
>> N=Grid.Nx*Grid.Ny*Grid.Nz; q=zeros(N,1); q([1 N])=[1 -1];
>> P=TPFA(Grid,Grid.K,q);
```

The permeability and pressure distribution is plotted in Figure 1.

3.3 Multipoint Flux-Approximation Schemes

The TPFA finite-volume scheme presented above is convergent only if each grid cell is a parallelepiped and

$$n_{ij} \cdot \mathbf{K} n_{ik} = 0, \qquad \forall \Omega_i \subset \Omega, \qquad n_{ij} \neq n_{ik}, \tag{18}$$

where n_{ij} and n_{ik} denote normal vectors into two neighbouring grid cells. A grid consisting of parallelepipeds satisfying (18) is said to be **K**-orthogonal. Orthogonal grids are, for example, **K**-orthogonal with respect to diagonal permeability tensors, but not with respect to full tensor permeabilities. Figure 2 shows a schematic of an orthogonal grid and a **K**-orthogonal grid.

If the TPFA method is used to discretise equation (14) on grids that are not **K**-orthogonal, the scheme will produce different results depending on the orientation of the grid (so-called grid-orientation effects) and will converge to a wrong solution. Despite this shortcoming of the TPFA method, it is still the dominant (and default) method for practical reservoir simulation, owing to its simplicity and computational speed. We present now a class of so-called *multi-point flux-approximation (MPFA) schemes* that aim to amend the shortcomings of the TPFA scheme.

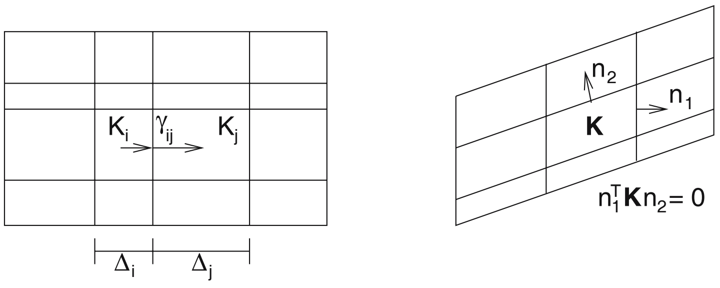


Fig. 2. The grid in the left plot is orthogonal with gridlines aligned with the principal coordinate axes. The grid in the right plot is a **K**-orthogonal grid.

Consider an orthogonal grid and assume that **K** is a constant tensor with nonzero off-diagonal terms. Moreover, for presentational simplicity, let γ_{ij} be an interface between two adjacent grid cells in the x–coordinate direction. Then for a given function u, the corresponding flux across γ_{ij} is given by:

$$\int_{\gamma_{ij}} v_w \cdot n_{ij} \, d\nu = - \int_{\gamma_{ij}} \big(k_{xx}\partial_x u + k_{xy}\partial_y u + k_{xz}\partial_z u\big) \, d\nu.$$

This expression involves derivatives in three orthogonal coordinate directions. Evidently, two point values can only be used to estimate a derivative in one direction. In particular, the two cell averages u_i and u_j can not be used to estimate the derivative of u in the y and z-directions. Hence, the TPFA scheme neglects the flux contribution from $k_{xy}\partial_y u$ and $k_{xz}\partial_z u$.

To obtain consistent interfacial fluxes for grids that are not **K**-orthogonal, one must also estimate partial derivatives in coordinate directions parallel to the interfaces. For this purpose, more than two point values, or cell averages, are needed. This leads to schemes that approximate v_{ij} using multiple cell averages, that is, with a linear expression on the form:

$$v_{ij} = \sum_k t_{ij}^k g_{ij}^k(\mathbf{u}).$$

Here $\{t_{ij}^k\}_k$ are the transmissibilities associated with γ_{ij} and $\{g_{ij}^k(\mathbf{u})\}_k$ are the corresponding multi-point pressure or flow potential dependencies. Thus, we see that MPFA schemes for (14) can be written on the form:

$$\sum_{j,k} t_{ij}^k g_{ij}^k(\mathbf{u}) = \int_{\Omega_i} f \, dx, \qquad \forall \Omega_i \subset \Omega. \tag{19}$$

MPFA schemes can, for instance, be designed by simply estimating each of the partial derivatives $\partial_\xi u$ from neighbouring cell averages. However, most MPFA schemes have a more physical motivation and are derived by imposing certain continuity requirements. We will now outline very briefly one such method,

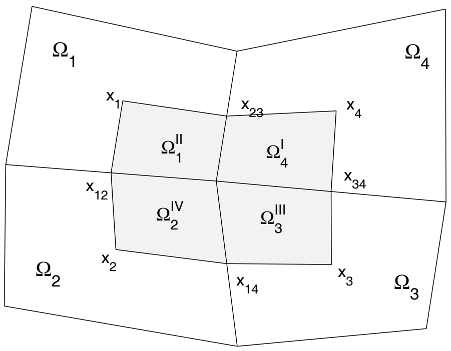


Fig. 3. The shaded region represents the interaction region for the O-method on a two-dimensional quadrilateral grid associated with cells Ω_1, Ω_2, Ω_3, and Ω_4.

called the O-method [2, 3], for irregular, quadrilateral, matching grids in two spatial dimensions.

The O-method is constructed by defining an interaction region (IR) around each corner point in the grid. For a two-dimensional quadrilateral grid, this IR is the area bounded by the lines that connect the cell-centres with the midpoints on the cell interfaces, see Figure 3. Thus, the IR consists of four sub-quadrilaterals ($\Omega_1^{\text{II}}, \Omega_2^{\text{IV}}, \Omega_3^{\text{III}}$, and Ω_4^{I}) from four neighbouring cells (Ω_1, Ω_2, Ω_3, and Ω_4) that share a common corner point. For each IR, define now

$$U_{\text{IR}} = \text{span}\{U_i^{\text{J}} : i = 1, \ldots, 4, \quad \text{J=I}, \ldots, \text{IV}\},$$

where $\{U_i^{\text{J}}\}$ are linear functions on the respective four sub-quadrilaterals. With this definition, U_{IR} has twelve degrees of freedom. Indeed, note that each U_i^{J} can be expressed in the following non-dimensional form

$$U_i^{\text{J}}(x) = u_i + \nabla U_i^{\text{J}} \cdot (x - x_i),$$

where x_i is the cell centre in Ω_i. The cell-centre values u_i thus account for four degrees of freedom and the (constant) gradients ∇U_i^{J} for additional eight.

Next we require that functions in U_{IR} are: (i) continuous at the midpoints of the cell interfaces, and (ii) flux-continuous across the interface segments that lie inside the IR. To obtain a globally coupled system, we first use (i) and (ii) to express the gradients ∇U_i^{J}, and hence also the corresponding fluxes across the IR interface segments, in terms of the unknown cell-centre potentials u_i. This requires solution of a local system of equations. Finally, the cell-centre potentials are determined (up to an arbitrary constant for no-flow boundary conditions) by summing the fluxes across all IR interface segments and requiring that the mass balance equations (13) hold. In this process, transmissibilities are assembled to obtain a globally coupled system for the unknown pressures over the whole domain.

We note that this construction leads to an MPFA scheme where the flux across an interface γ_{ij} depends on the potentials u_j in a total of six neighbouring cells (eighteen in three dimensions). Notice also that the transmissibilities $\{t_{ij}^k\}$ that we obtain when eliminating the IR gradients now account for grid-cell geometries in addition to full-tensor permeabilities.

3.4 A Mixed Finite-Element Method

As an alternative to the MPFA schemes, one can use mixed finite-element methods (FEMs) [10]. In mixed FEMs, the fluxes over cell edges are considered as unknowns in addition to the pressures, and are not computed using a (numerical) differentiation as in finite-volume methods. For the mixed FEMs there is little to gain by reformulating (11) into an equation on the form (14). We therefore return to the original formulation and describe how to discretise the following system of differential equations with mixed FEMs:

$$v = -\lambda(\nabla p - \rho G), \qquad \nabla \cdot v = q. \tag{20}$$

As before we impose no-flow boundary conditions on $\partial\Omega$. To derive the mixed formulation, we first define the following Sobolev space

$$H_0^{1,\mathrm{div}}(\Omega) = \{v \in (L^2(\Omega))^d : \ \nabla \cdot v \in L^2(\Omega) \text{ and } v \cdot n = 0 \text{ on } \partial\Omega\}.$$

The mixed formulation of (20) (with no-flow boundary conditions) now reads: find $(p, v) \in L^2(\Omega) \times H_0^{1,\mathrm{div}}(\Omega)$ such that

$$\int_\Omega v \cdot \lambda^{-1} u \, dx - \int_\Omega p \, \nabla \cdot u \, dx = \int_\Omega \rho G \cdot u \, dx, \tag{21}$$

$$\int_\Omega l \, \nabla \cdot v \, dx = \int_\Omega q l \, dx, \tag{22}$$

for all $u \in H_0^{1,\mathrm{div}}(\Omega)$ and $l \in L^2(\Omega)$. We observe again that, since no-flow boundary conditions are imposed, an extra constraint must be added to make (21)–(22) well-posed. A common choice is to use $\int_\Omega p \, dx = 0$.

In mixed FEMs, (21)–(22) are discretised by replacing $L^2(\Omega)$ and $H_0^{1,\mathrm{div}}(\Omega)$ with finite-dimensional subspaces U and V, respectively. For instance, in the Raviart–Thomas mixed FEM [31] of lowest order (for triangular, tetrahedral, or regular parallelepiped grids), $L^2(\Omega)$ is replaced by

$$U = \{p \in L^2(\Omega) : p|_{\Omega_i} \text{ is constant } \forall \Omega_i \in \Omega\}$$

and $H_0^{1,\mathrm{div}}(\Omega)$ is replaced by

$$\begin{aligned} V = \{&v \in H_0^{1,\mathrm{div}}(\Omega) : v|_{\Omega_i} \text{ have linear components } \forall \Omega_i \in \Omega, \\ &(v \cdot n_{ij})|_{\gamma_{ij}} \text{ is constant } \forall \gamma_{ij} \in \Omega, \text{ and } v \cdot n_{ij} \text{ is continuous across } \gamma_{ij}\}. \end{aligned}$$

Here n_{ij} is the unit normal to γ_{ij} pointing from Ω_i to Ω_j. The corresponding Raviart–Thomas mixed FEM thus seeks

$$(p, v) \in U \times V \text{ such that (21)–(22) hold for all } u \in V \text{ and } q \in U. \tag{23}$$

To express (23) as a linear system, observe first that functions in V are, for admissible grids, spanned by base functions $\{\psi_{ij}\}$ that are defined by

$$\psi_{ij} \in \mathcal{P}(\Omega_i)^d \cup \mathcal{P}(\Omega_j)^d \quad \text{and} \quad (\psi_{ij} \cdot n_{kl})|_{\gamma_{kl}} = \begin{cases} 1 & \text{if} \quad \gamma_{kl} = \gamma_{ij}, \\ 0 & \text{else,} \end{cases}$$

where $\mathcal{P}(K)$ is the set of linear functions on K. Similarly

$$U = \text{span}\{\chi_m\} \quad \text{where} \quad \chi_m = \begin{cases} 1 & \text{if} \quad x \in \Omega_m, \\ 0 & \text{else.} \end{cases}$$

Thus, writing $p = \sum_{\Omega_m} p_m \chi_m$ and $v = \sum_{\gamma_{ij}} v_{ij}\psi_{ij}$, allows us to write (23) as a linear system in $\mathbf{p} = \{p_m\}$ and $\mathbf{v} = \{v_{ij}\}$. This system takes the form

$$\begin{bmatrix} \mathbf{B} & -\mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{g} \\ \mathbf{f} \end{bmatrix}. \tag{24}$$

Here $\mathbf{f} = [f_m]$, $\mathbf{g} = [g_{kl}]$, $\mathbf{B} = [b_{ij,kl}]$ and $\mathbf{C} = [c_{m,kl}]$, where:

$$g_{kl} = \Big[\int_\Omega \rho G \cdot \psi_{kl}\, dx\Big], \qquad f_m = \Big[\int_{\Omega_m} f\, dx\Big],$$
$$b_{ij,kl} = \Big[\int_\Omega \psi_{ij} \cdot \lambda^{-1}\psi_{kl}\, dx\Big], \qquad c_{m,kl} = \Big[\int_{\Omega_m} \nabla \cdot \psi_{kl}\, dx\Big].$$

Note that for the current Raviart-Thomas finite elements, we have

$$c_{m,kl} = \begin{cases} 1 & \text{if } m = k, \\ -1 & \text{if } m = l, \\ 0 & \text{otherwise.} \end{cases}$$

The matrix entries $b_{ij,kl}$, on the other hand, depend on the geometry of the grid cells and the form of λ. In other words, the entries depend on whether λ is isotropic or anisotropic, whether λ is cell-wise constant or models subgrid variations in the permeability field.

We provide now a Matlab code implementing the Raviart-Thomas mixed FEM for the first-order system (20) on a regular hexahedral grid in three spatial dimensions. The code is divided into three parts: assembly of the $\mathbf{B}$ block; assembly of the $\mathbf{C}$ block; and a main routine that loads data (permeability and grid), assembles the whole matrix, and solves the system. Again we emphasise compactness and efficiency, by heavily relying on the Cartesian grid and analytical integration of the Raviart-Thomas basis functions to perform a direct assembly of the matrix blocks. For more general grids, one would typically perform an elementwise assembly by looping over all grid blocks and replace the integration over basis functions by some numerical quadrature rule.

Listing 2. Assembly of **B** for the lowest-order Raviart–Thomas elements.

```
function B=GenB(Grid,K)

Nx=Grid.Nx; Ny=Grid.Ny; Nz=Grid.Nz; N=Nx*Ny*Nz;
hx=Grid.hx; hy=Grid.hy; hz=Grid.hz;
L = K.^(-1);
ex=N-Ny*Nz; ey=N-Nx*Nz; ez=N-Nx*Ny;               % Number of edges
tx=hx/(6*hy*hz); ty=hy/(6*hx*hz); tz=hz/(6*hx*hy); % Transmissibilities

X1=zeros(Nx-1,Ny,Nz); X2=zeros(Nx-1,Ny,Nz);       % Preallocate memory
X0=L(1,1:Nx-1,:,:)+L(1,2:Nx,:,:);  x0=2*tx*X0(:); % Main diagonal
X1(2:Nx-1,:,:)=L(1,2:Nx-1,:,:);    x1=tx*X1(:);   % Upper diagonal
X2(1:Nx-2,:,:)=L(1,2:Nx-1,:,:);    x2=tx*X2(:);   % Lower diagonal

Y1=zeros(Nx,Ny-1,Nz); Y2=zeros(Nx,Ny-1,Nz);       % Preallocate memory
Y0=L(2,:,1:Ny-1,:)+L(2,:,2:Ny,:); y0=2*ty*Y0(:);  % Main diagonal
Y1(:,2:Ny-1,:)=L(2,:,2:Ny-1,:);  y1=ty*Y1(:);     % Upper diagonal
Y2(:,1:Ny-2,:)=L(2,:,2:Ny-1,:);  y2=ty*Y2(:);     % Lower diagonal

Lz1=L(3,:,:,1:Nz-1); z1=tz*Lz1(:);                % Upper diagonal
Lz2=L(3,:,:,2:Nz);   z2=tz*Lz2(:);                % Lower diagonal
z0=2*(z1+z2);                                     % Main diagonal

B=[spdiags([x2,x0,x1],[-1,0,1],ex,ex),sparse(ex,ey+ez);...
   sparse(ey,ex),spdiags([y2,y0,y1],[-Nx,0,Nx],ey,ey),sparse(ey,ez);...
   sparse(ez,ex+ey),spdiags([z2,z0,z1],[-Nx*Ny,0,Nx*Ny],ez,ez)];
```

In Listing 2 we show a Matlab function that assembles the **B** matrix. Here λ is assumed to be a diagonal tensor and is represented as a $3\times$`Nx`$\times$`Ny`$\times$`Nz` matrix `K` with `Nx`, `Ny`, and `Nz` denoting the number of grid cells in the x, y, and z direction, respectively. The degrees of freedom at the interfaces (the fluxes) have been numbered in the same way as the grid-blocks; i.e., first in the x-direction, then in the y-direction, and finally in the z-direction. This gives **B** a hepta-diagonal structure as shown in Figure 4, where the three nonzero blocks correspond to the three components of v. Notice, however, that since the grid is K-orthogonal, we can make **B** tridiagonal by starting the numbering of each component of $v = (v_x, v_y, v_z)$ in the corresponding direction; i.e., number v_x as xyz, v_y as yxz, and v_z as zxy. Similar observations can be made for the assembly of **C**, which is given in Listing 3.

Let us now make a few brief remarks with respect to the implementation of the assembly of the matrix blocks **B** and **C**. First, since only a few components of **B** and **C** are nonzero, we store them using a sparse matrix format. The matrices are created in a block-wise manner using Matlab's built in sparse matrix functions. The function `sparse(m,n)` creates an $m \times n$ zero matrix and `spdiags(M,d,m,n)` creates an $m \times n$ sparse matrix with the columns of M on the diagonals specified by d. Finally, statements of the form `M(:)` turn the matrix M into a column vector.

A drawback with the mixed FEM is that it produces an indefinite linear system. These systems are in general harder to solve than the positive definite systems that arise, e.g., from the TPFA and MPFA schemes described in

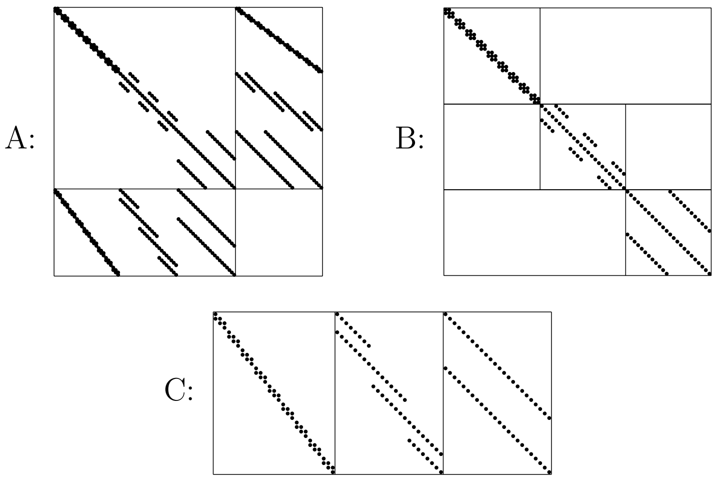


Fig. 4. Sparsity patterns for **A**, **B**, and **C** for a case with $4 \times 3 \times 3$ grid-blocks.

Listing 3. Assembly of **C** for the lowest-order Raviart–Thomas elements.

```
function C=GenC(Grid)

Nx=Grid.Nx; Ny=Grid.Ny; Nz=Grid.Nz;
C=sparse(0,0);                                         % Empty sparse matrix
Nxy=Nx*Ny; N=Nxy*Nz;                                   % Number of grid-points
vx=ones(Nx,1); vy=ones(Nxy,1); vz=ones(N,1);           % Diagonals

for i=1:Ny*Nz                                          % vx-block of C
  Cx=spdiags([vx,-vx],[-1,0]-(i-1)*Nx,N,Nx-1);         % create bidiagonal block
  C=[C,Cx];                                            % append to C
end

for i=1:Nz                                             % vy-block of C
  Cy=spdiags([vy,-vy],[-Nx,0]-(i-1)*Nxy,N,Nxy-Nx);     % create bidiagonal block
  C=[C,Cy];                                            % append to C
end

C = [C, spdiags([vz,-vz],[-Nxy,0],N,N-Nxy)];           % vz-block of C
```

Sections 3.2 and 3.3. In fact, for second-order elliptic equations of the form (11) it is common to use a so-called hybrid formulation. This method leads to a positive definite system where the unknowns correspond to pressures at grid-cell interfaces. The solution to the linear system arising from the mixed FEM can now be obtained from the solution to the hybrid system by performing only local algebraic calculations. This property, which is sometimes referred to as explicit flux representation, allows the inter-cell fluxes that appear in the saturation equation to be expressed as a linear combination of neighbouring values for the pressure.

Explicit flux representation is a feature that also the finite-volume methods enjoy. One of the main advantages is that it allows us to compute fully implicit solutions without computing the fluxes explicitly. Indeed, in the mixed FEM, which does not have an explicit flux representation, one has to solve the full indefinite linear system for the pressure equation alongside the linear system for the saturation equation in order to produce a fully implicit solution. We would like to comment, however, that whether or not a discretisation method for the pressure equation allows an explicit flux representation may mostly be regarded as a minor issue. Indeed, one is always free to use a sequential implicit solution strategy which is often faster and in most cases results in the same, or at least an equally accurate solution.

It is not within our scope here to discuss issues related to solving linear systems that arise from mixed FEM discretisations further. Indeed, for moderately sized problems we can rely on Matlab, and the possibly complex linear algebra involved in solving the sparse system (24) can be hidden in a simple statement of the form `x = A\b`. Readers interested in learning more about mixed FEMs, and how to solve the corresponding linear systems are advised to consult some of the excellent books on the subject, for instance [8, 9, 10].

Example 2. It is now time to consider our first reservoir having more than only a touch of realism. To this end we will simulate two horizontal slices of Model 2 from the 10th SPE Comparative Solution Project [14], which is publicly available on the net. The model dimensions are $1200 \times 2200 \times 170$ (ft) and the reservoir is described by a heterogeneous distribution over a regular Cartesian grid with $60 \times 220 \times 85$ grid-blocks. We pick the top layer, in which the permeability is smooth, and the bottom layer, which is fluvial and characterised by a spaghetti of narrow high-flow channels (see Figure 5). For both layers, the permeabilities range over at least six orders of magnitude. To drive a flow in the two layers, we impose an injection and a production well in the lower-left and upper-right corners, respectively.

Listing 4 contains a simulation routine that loads data, assembles the whole matrix, and solves the system (24) to compute pressure and fluxes (neglecting, for brevity, gravity forces so that $G = 0$) . For the flux, only the interior edges are computed, since the flux normal to the exterior edges is assumed to be zero through the no-flow boundary condition.

In Figure 5 we visualise the flow field using 20 streamlines imposed upon a colour plot of the logarithm of the permeability. For completeness, we have included the Matlab code used to generate each of the plots. Some of the streamlines in the plots seem to appear and disappear at the boundaries. This is a pure plotting artifact obtained when we collocate the staggered edge fluxes at the cell centres in order to use Matlab's `streamline` visualisation routine.

In the next sections we will demonstrate how the models and numerical methods introduced above for single-phase flow can be extended to more general flows involving more than one fluid phase, where each phase possibly contains more than one component.

Listing 4. Mixed finite-element discretisation of $v = -K\nabla p$, $\nabla \cdot v = q$ with lowest-order Raviart–Thomas elements applied to the top layer of the SPE-10 test case [14].

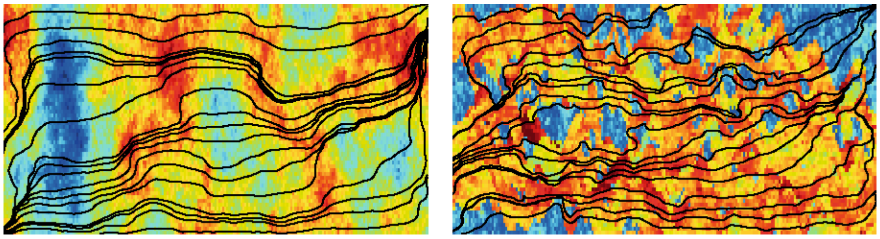
```
Grid.Nx=60; Grid.hx=20*.3048;                    % Dimension in x-direction
Grid.Ny=220; Grid.hy=10*.3048;                   % Dimension in y-direction
Grid.Nz=1;  Grid.hz= 2*.3048;                    % Dimension in z-direction
Nx=Grid.Nx; Ny=Grid.Ny; Nz=Grid.Nz;              % Local variables
N=Nx*Ny*Nz;                                      % Total number of grid points

Ex=(Nx-1)*Ny*Nz;                                 % Number of edges in x-direction
Ey=Nx*(Ny-1)*Nz;                                 % Number of edges in y-direction
Ez=Nx*Ny*(Nz-1);                                 % Number of edges in z-direction
E=Ex+Ey+Ez;                                      % Total number of edges in grid

q=zeros(E+N,1);                                  % Right-hand side
q(E+1)=1;                                        % Injection in block (1,1,1)
q(E+N)=-1;                                       % Production in block (Nx,Ny,Nz)
load Udata; Grid.K=KU(:,1:Nx,1:Ny,1:Nz);         % Load and extract permeability

B=GenB(Grid,Grid.K);                             % Compute B-block of matrix
C=GenC(Grid);                                    % Compute C-block of matrix
A=[B,C';-C,sparse(N,N)];                         % Assemble matrix
A(E+1,E+1)=A(E+1,E+1)+1;
x=A\q;                                           % Solve linear system

v=x(1:E);                                        % Extract velocities
vx=reshape(v(1:Ex),Nx-1,Ny,Nz);                  %   x-component
vy=reshape(v(Ex+1:E-Ez),Nx,Ny-1,Nz);             %   y-component
vz=reshape(v(E-Ez+1:E),Nx,Ny,Nz-1);              %   z-component
p=reshape(x(E+1:E+N),Nx,Ny,Nz);                  % Extract pressure
```



```
% Collocate velocities at cell centre
U = [zeros(1,Ny); vx; zeros(1,Ny)]; U=0.5*(U(1:end-1,:)+U(2:end,:));
V = [zeros(Nx,1), vy, zeros(Nx,1)]; V=0.5*(V(:,1:end-1)+V(:,2:end));

% Make grid and sampling points along diagonal
[Y,X]=meshgrid([1:Ny]*hy-0.5*hy,[1:Nx]*hx-0.5*hx);
sy = linspace(0.5*hy, (Ny-0.5)*hy, 20);
sx = (Nx-0.5)*hx/((Ny-0.5)*hy)*( (Ny-0.5)*hy - sy );
pcolor(Y,X,log10(squeeze(K(1,:,:)))); shading flat;

% Trace forward to producer and backward to injector
hp=streamline(Y,X,V,U,sy,sx); hn=streamline(Y,X,-V,-U,sy,sx);
set([hp, hn],'Color','k','LineWidth',1.5);
axis equal; axis tight; axis off;
```

Fig. 5. Streamlines and logarithm of permeability for two-dimensional simulation of the top and bottom layer of the SPE-10 test case. The source code for making the plots is included.

4 Multiphase and Multicomponent Flows

As for single-phase flow, the fundamental model describing the flow of a multiphase, multicomponent fluid is the conservation (or continuity) equations for each component ℓ:

$$\frac{\partial}{\partial t}\Big(\phi \sum_{\alpha} c_{\ell\alpha}\rho_\alpha s_\alpha\Big) + \nabla\cdot\Big(\sum_{\alpha} c_{\ell\alpha}\rho_\alpha v_\alpha\Big) = \sum_{\alpha} c_{\ell\alpha} q_\alpha, \qquad \alpha = w, o, g. \quad (25)$$

Here we recall that $c_{\ell\alpha}$ is mass fraction of component ℓ in phase α, ρ_α is the density of phase α, v_α is phase velocity, and q_α is phase source. As for single-phase flow, the phase velocities must be modelled. This is usually done by extending Darcy's law to relate the phase velocities to the phase pressures p_α

$$v_\alpha = -\mathbf{K}\frac{k_{r\alpha}}{\mu_\alpha}(\nabla p_\alpha - \rho_\alpha G). \qquad (26)$$

In the multiphase version of Darcy's law (as opposed to the single-phase version (10)) we have used the relative permeabilities $k_{r\alpha}$ (see Section 2) to account for the reduced permeability of each phase due to the presence of the other phases.

4.1 Black-Oil Models

A large class of models that are widely used in porous media flow simulations are the black-oil models. The name refers to the assumption that the hydrocarbons may be described as two components: a heavy hydrocarbon component called "oil" and a light hydrocarbon component called "gas". The two components can be partially or completely dissolved in each other depending on the pressure and the temperature, forming either one or two phases (liquid and gaseous). In general black-oil models, the hydrocarbon components are also allowed to be dissolved in the water phase and the water component may be dissolved in the two hydrocarbon phases. The hydrocarbon fluid composition, however, remains constant for all times. The alternative, where the hydrocarbons are modelled using more than two components and hydrocarbon are allowed to change composition, is called a *compositional* model.

We will henceforth assume three phases and three components (gas, oil, water). By looking at our model (25)–(26), we see that we therefore so far have introduced 27 unknown physical quantities: nine mass fractions $c_{\ell\alpha}$ and three of each of the following quantities ρ_α, s_α, v_α, p_α, μ_α, and $k_{r\alpha}$. The corresponding numbers of equations are: three continuity equations (25), an algebraic relation for the saturations (3), three algebraic relations for the mass fractions (4), and Darcy's law (26) for each phase. This gives us only ten equations. Thus, to make a complete model of multiphase, multicomponent flow one must add extra closure relations.

In Section 2, we saw that the relative permeabilities are usually assumed to be known functions of the phase saturations. These functions are normally obtained from physical experiments, small-scale numerical simulations, or simply from known rock-type properties. The capillary pressure curves are also assumed to depend solely on the phase saturations, and can be obtained in a similar fashion. One usually employs oil-water and gas-oil capillary pressures:

$$p_{cow} = p_o - p_w, \qquad p_{cgo} = p_g - p_o.$$

The densities and viscosities are obtained from lab experiments and are related to the phase pressures. Summing up, this gives us a total of eleven closure relations. Finally, one can introduce six algebraic relations for $c_{\ell g}/c_{\ell o}$ and $c_{\ell g}/c_{\ell w}$ in the form of PVT models.

In the next section we consider the special case of immiscible flow. For more advanced models, we refer the reader to one of the general textbooks for reservoir simulation [6, 11, 16, 21, 28, 30, 13, 34].

5 Immiscible Two-Phase Flow

In this section we will consider the flow of two phases, one water phase w and one hydrocarbon phase. The water phase will consist of pure water, whereas the hydrocarbon phase generally is a two-component fluid consisting of dissolved gas and a residual (or black) oil. These assumptions translate to the following definitions for the mass fractions

$$\begin{aligned} c_{ww} &= 1, & c_{ow} &= 0, & c_{gw} &= 0, \\ c_{wo} &= 0, & c_{oo} &= \frac{m_o}{m_o + m_g}, & c_{go} &= \frac{m_g}{m_o + m_g}, \\ c_{wg} &= 0, & c_{og} &= 0, & c_{gg} &= 0. \end{aligned}$$

Here m_o and m_g are the masses of oil and gas, respectively. By adding the continuity equations for the oil and gas components, we obtain a continuity equation for the water-phase (w) and one for the hydrocarbon phase (o). Since $c_{oo} + c_{go} = 1$, both of these continuity equations have the following form:

$$\frac{\partial(\phi \rho_\alpha s_\alpha)}{\partial t} + \nabla \cdot (\rho_\alpha v_\alpha) = q_\alpha, \tag{27}$$

Expanding space and time derivatives, and dividing by the phase densities, we obtain an alternative formulation of the continuity equations (27):

$$\frac{\partial \phi}{\partial t} s_\alpha + \phi \frac{\partial s_\alpha}{\partial t} + \phi \frac{s_\alpha}{\rho_\alpha} \frac{\partial \rho_\alpha}{\partial t} + \nabla \cdot v_\alpha + \frac{v_\alpha \cdot \nabla \rho_\alpha}{\rho_\alpha} = \frac{q_\alpha}{\rho_\alpha}, \tag{28}$$

In the following we will rewrite the two continuity equations into a more tractable system of equations consisting of a pressure equation (as introduced for the single-phase model) and a saturation or fluid-transport equation.

5.1 The Pressure Equation

To derive the pressure equation, we introduce first, for brevity, the mobility of phase α: $\lambda_\alpha = k_{r\alpha}/\mu_\alpha$. Summing continuity equations (28) for the oil and water phases, letting $q = q_w/\rho_w + q_o/\rho_o$, and using the fact that $s_w + s_o = 1$, we deduce

$$\nabla \cdot (v_w + v_o) + \frac{\partial \phi}{\partial t} + \phi \frac{s_w}{\rho_w}\frac{\partial \rho_w}{\partial t} + \phi \frac{s_o}{\rho_o}\frac{\partial \rho_o}{\partial t} + \frac{v_w \cdot \nabla \rho_w}{\rho_w} + \frac{v_o \cdot \nabla \rho_o}{\rho_o} = q. \quad (29)$$

Introducing the *rock compressibility* defined by (1) and the *phase compressibilities* defined by (6), and inserting the expression for the Darcy velocities, we obtain

$$\begin{aligned}
&- \nabla \cdot \big[\mathbf{K}\lambda_w(\nabla p_w - \rho_w G) + \mathbf{K}\lambda_o(\nabla p_o - \rho_o G)\big] + c_r \phi \frac{\partial p}{\partial t} \\
&\qquad - c_w \big[\nabla p_w \cdot \mathbf{K}\lambda_w(\nabla p_w - \rho_w G) - \phi s_w \frac{\partial p_w}{\partial t}\big] \\
&\qquad\qquad - c_o \big[\nabla p_o \cdot \mathbf{K}\lambda_o(\nabla p_o - \rho_o G) - \phi s_o \frac{\partial p_o}{\partial t}\big] = q. \quad (30)
\end{aligned}$$

In this equation we have three pressures: oil pressure p_o, water pressure p_w, and total pressure p. If we now treat, say, p_o as the primary variable, replace p_w with $p_o - p_{cow}$ (assuming the compressibilities are known) and express the total pressure as a function of p_o, equation (30) becomes a parabolic equation that can be solved for the oil-phase pressure p_o.

5.2 A Pressure Equation for Incompressible Immiscible Flow

Designing numerical methods for the pressure equation (30) that correctly account for flow dynamics and properly balance the spatial and temporal derivatives can be a very difficult task due to the many prominent scales that occur in porous media permeability. Since the temporal derivative terms are relatively small, equation (30) is nearly elliptic and can usually be discretised with numerical methods that are well suited for elliptic differential equations such as the ones described in Sections 3.2–3.4. The purpose of the current presentation is to help the reader become familiar with the basic flow equations, and to explain in detail some possible numerical methods for discretising the corresponding differential equations. We therefore make some simplifying assumptions before we consider numerical discretisation techniques.

To enhance readability, we henceforth assume that the rock and the two fluid phases are incompressible, i.e., $c_r = c_w = c_o = 0$. Equation (30) then reduces to

$$v = -\big[\mathbf{K}\lambda_w(\nabla p_w - \rho_w G) + \mathbf{K}\lambda_o(\nabla p_o - \rho_o G)\big], \qquad \nabla \cdot v = q.$$

In this equation, there are two unknown phase pressures, p_o and p_w. To eliminate one of them, it is common to introduce the capillary pressure

$p_{cow} = p_o - p_w$, which is assumed to be a function of water saturation s_w. Unfortunately, this leads to a rather strong coupling between the pressure equation and the saturation equation. We will therefore follow another approach; see [11] for more details. Instead of using the phase pressures p_w and p_o, we introduce a new *global pressure* p. The global pressure is defined to contain saturation-dependent pressure terms, thereby giving a better decoupling of the pressure and saturation equations . To this end, we first assume that the capillary pressure p_{cow} is a monotone function of the water saturation s_w and then define the global pressure as $p = p_o - p_c$, where the saturation-dependent complementary pressure p_c is defined by

$$p_c(s_w) = \int_1^{s_w} f_w(\xi) \frac{\partial p_{cow}}{\partial s_w}(\xi) d\xi. \tag{31}$$

Here the *fractional-flow* function $f_w = \lambda_w/(\lambda_w + \lambda_o)$ measures the water fraction of the total flow. Since $\nabla p_c = f_w \nabla p_{cow}$, we are able to express the total velocity $v = v_w + v_o$ as a function of the global pressure p:

$$v = -\mathbf{K}(\lambda_w + \lambda_o)\nabla p + \mathbf{K}(\lambda_w \rho_w + \lambda_o \rho_o)G. \tag{32}$$

Finally, introducing the total mobility $\lambda = \lambda_w + \lambda_o$ we obtain the following elliptic equation for the global pressure p:

$$-\nabla \cdot \big[\mathbf{K}\lambda\nabla p - \mathbf{K}(\lambda_w \rho_w + \lambda_o \rho_o)G\big] = q \tag{33}$$

To make the pressure equation complete we need to prescribe some boundary conditions. The default is to impose no-flow boundary conditions, but if the reservoir is connected to e.g., an aquifer, it might be possible to determine an approximate pressure distribution on the reservoir boundary.

5.3 The Saturation Equation

The pressure equation gives an equation for our first primary unknown p. To derive a complete model, we must also derive equations for the phase saturations s_w and s_o using the continuity equations of each phase. However, since $s_w + s_o = 1$, we need only one saturation equation and it is common practice to pick s_w as the second primary unknown. To connect the continuity equation for water to the pressure equation (33), we need to derive an expression for the phase velocity v_w in terms of the global and complementary pressures p and p_c. To this end, we will use what is called the total velocity formulation and express v_w in terms of the total velocity v and some additional terms that only depend on the saturation s_w. From Darcy's law (26) it follows that

$$\mathbf{K}\lambda_o\lambda_w\nabla p_{cow} = \lambda_o v_w - \lambda_w v_o + \mathbf{K}\lambda_o\lambda_w(\rho_o - \rho_w)G. \tag{34}$$

Inserting $v_o = v - v_w$ into this equation and dividing by λ, we obtain

$$v_w = f_w\big[v + \mathbf{K}\lambda_o \nabla p_{cow} + \mathbf{K}\lambda_o(\rho_w - \rho_o)G\big],$$

where we have used the fractional flow function introduced above. Finally, expanding $\nabla p_{cow} = \frac{\partial p_{cow}}{\partial s_w}\nabla s_w$ and invoking the incompressibility assumption, we arrive at an equation for the fluid transport having the following form:

$$\phi\frac{\partial s_w}{\partial t} + \nabla\cdot\Big(f_w(s_w)\big[v + d(s_w,\nabla s_w) + g(s_w)\big]\Big) = \frac{q_w}{\rho_w}. \tag{35}$$

To make a complete description of the flow, the continuity equation must be equipped with boundary conditions, e.g., no-flow conditions, and initial conditions $s_w(x,0) = s_w^0(x)$. Henceforth we will drop the subscript w.

Equation (35) is called the *saturation equation* and is generally a parabolic equation. However, on a reservoir scale, the terms $f(s)v$ and $f(s)g(s)$ representing viscous and gravity forces, respectively, usually dominate the term $f(s)d(s,\nabla s)$ representing capillary forces. The saturation equation will therefore usually have a strong hyperbolic nature and will require other discretisation techniques than those introduced for the almost-elliptic pressure equation (33). Examples of appropriate discretisation techniques will be presented below, but first we discuss how to solve the coupled system consisting of (35) and (33) or (30).

5.4 Solution Strategies for the Coupled System

The equations (33) and (35) derived in the previous section are called the fractional-flow model for immiscible two-phase flow. The model consists of an elliptic pressure equation (33) (or the more general parabolic equation (30), which has a similar elliptic nature) and a saturation or fluid transport equation (35) that has a certain hyperbolic nature. The equations are nonlinearly coupled. The coupling is primarily through the saturation-dependent mobilities λ_α in the pressure equation and through the pressure-dependent velocities v_α in the saturation equation. However, the equations are also coupled through other terms that depend on pressure or saturation, e.g., viscosities and capillary and complementary pressures.

A natural strategy for solving the coupled system is to make an implicit discretisation for each equation and simultaneously solve for the two primary unknowns p and s_w. This strategy is usually referred to as *fully implicit* solution and is a common solution method in industry due to its robustness. However, fully implicit solution is computationally expensive, since we need to solve a large nonlinear system of equations through some iterative procedure like e.g., the Newton–Raphson method. On the other hand, more efficient methods can be developed by using operator splitting to decouple the two equations. In this sequential approach, each equation is solved separately and one can therefore use very different methods to discretise the two fundamentally different equations. As an example, we mention the IMPES (implicit pressure, explicit saturation) method, which used to be quite popular in the industry. Currently, a more popular choice is to use a method called the adaptive implicit method (AIM), in which some grid blocks are solved fully implicitly while the others are treated with a sequential splitting method (IMPES). The method therefore gives robustness in problematic areas with large changes in pressure and saturations (like near a well bore), while at the same time giving high computational efficiency away from problem regions.

In the current paper, we will only present sequential splitting methods. For the global pressure and total velocity formulation (33) and (35) of incompressible and immiscible two-phase flow, a sequential splitting method can be designed as follows: First, the saturation distribution from the previous time step (or initial data) is used to compute the saturation-dependent coefficients in (33), before the equation is solved for global pressure and total velocity. Then, the total velocity v is kept constant as a parameter in (35), while the saturation is advanced in time. Next, the new saturation values are used to update the saturation-dependent coefficients in (33), and the pressure equation is solved again, and so on. Consequently, we can develop numerical schemes for (33) and (35) without being concerned about the nonlinear coupling between the two.

Whereas the advantage of a sequential method is its efficiency (and potential spatial accuracy), the disadvantage is the splitting errors introduced by decoupling the equations. In certain (gas-dominated) cases, splitting errors may lead to unphysical flow predictions unless inordinately small splitting steps are used. To remedy this potential pitfall, one can introduce an extra loop for each time step and iterate a few times (until convergence) between solving the pressure and saturation, before moving on the next time step. The corresponding method is sometimes called sequential implicit, indicating its intermediary nature. We will now use the sequential splitting method to develop and present a full simulator for our simple two-phase model. For presentational simplicity, the sequential implicit technique will not be employed, but can easily be included whenever necessary.

5.5 Discretising the Pressure Equation

To discretise the pressure equation (33) with a finite-volume method, terms corresponding to gravity must be moved over to the right-hand side. The single-phase formulation (13) is hence replaced with

$$\begin{aligned} -\int_{\partial\Omega_i} \mathbf{K}\lambda(s_w^k)\nabla p^{k+1}\,d\nu &= \\ &\int_{\Omega_i} q\,dx - \int_{\partial\Omega_i} \mathbf{K}\Big(\lambda_w(s_w^k)\rho_w + \lambda_o(s_w^k)\rho_o\Big)G\cdot n\,d\nu, \qquad (36) \end{aligned}$$

where the superscript k denotes the time step. The integrals over the cell boundaries are computed with a TPFA scheme or an MPFA scheme, as presented for the single-phase flow in Sections 3.2 and 3.3.

For the mixed FEM, we need only to replace equations (21)–(22) with

$$\int_\Omega \Big(\mathbf{K}\lambda(s_w^k)\Big)^{-1} v^{k+1}\cdot u\,dx - \int_\Omega p^{k+1}\,\nabla\cdot u\,dx = \int_\Omega \Big(\rho_w f_w(s_w^k) + \rho_o f_o(s_w^k)\Big) G\cdot u\,dx, \tag{37}$$

$$\int_\Omega q\,\nabla\cdot v^{k+1}\,dx = \int_\Omega fq\,dx. \tag{38}$$

Thus, the pressure equation modelling two-phase flow may be discretised with the same methods that were used to discretise the single-phase pressure equation in Section 3. The main difference is that for two-phase flow the pressure is a dynamic function of saturation, and must therefore be solved repeatedly throughout a simulation.

5.6 Discretising the Saturation Equation

Traditionally, algorithms for solving the pressure equation (33) has accounted for the majority of the computational time in reservoir simulations. However, pressure solvers have improved a lot during the last years as a result of improved numerical linear algebra (e.g., multigrid and other iterative methods). It is therefore likely that the design of robust numerical schemes that balance viscous, gravity, and capillary forces in the saturation equations correctly, may be an equally challenging part of reservoir simulation in the future. The saturation equations are, for example, typically advection dominated on a reservoir scale, in particular in high-flow regions. This implies that propagating interfaces between oil and water, commonly referred to as saturation fronts, can be very sharp. Numerical diffusion may therefore dominate the capillary forces if one does not use a scheme with high resolution that allows accurate tracing of the saturation fronts.

Most commercial reservoir simulators use an implicit or semi-implicit time discretisation to evolve saturation profiles in time, while a conservative finite-volume method is used to resolve the spatial derivatives. Consider a cell Ω_i with edges γ_{ij} and associated normal vectors n_{ij} pointing out of Ω_i. Using the θ-rule for temporal discretisation, a finite-volume scheme takes the following general form

$$\frac{\phi_i}{\Delta t}\left(s_i^{k+1} - s_i^k\right) + \frac{1}{|\Omega_i|}\sum_{j\neq i}\left[\theta F_{ij}(s^{k+1}) + (1-\theta)F_{ij}(s^k)\right] = \frac{q_i(s_i^k)}{\rho}. \tag{39}$$

Here ϕ_i is the porosity in Ω_i, q_i denotes the source term, Δt is the time step, and s_i^k is the cell-average of the water saturation at time $t = t_k$,

$$s_i^k = \frac{1}{|\Omega_i|}\int_{\Omega_i} s(x,t_k)\,dx.$$

Finally, F_{ij} is a numerical approximation of the flux over edge γ_{ij},

$$F_{ij}(s) \approx \int_{\gamma_{ij}} f_w(s)_{ij} \big[v_{ij} + d_{ij}(s) + g_{ij}(s) \big] \cdot n_{ij} \, d\nu. \tag{40}$$

Here $f_w(s)_{ij}$ denotes the fractional-flow function associated with γ_{ij}, v_{ij} is the Darcy flux, d_{ij} the diffusive flux, and g_{ij} the gravitational flux across the edge. Different finite-volume schemes are now defined by the quadrature rule used for the edge integrals in (40) and by the way the integrand is evaluated. For a first-order scheme, it is common to use upstream weighting for the fractional flow, e.g.,

$$f_w(s)_{ij} = \begin{cases} f_w(s_i) & \text{if } v \cdot n_{ij} \geq 0, \\ f_w(s_j) & \text{if } v \cdot n_{ij} < 0, \end{cases} \tag{41}$$

and so on. If we now choose $\theta = 0$ in the temporal discretisation, we end up with a first-order, explicit scheme. Such a scheme is quite accurate but imposes stability restrictions on the time step in the form of a CFL condition. The time-step restrictions may be quite severe due to the near-singular nature of the forcing velocity field near wells and the possible presence of cells with arbitrarily small porosity. We will come back to this discussion in Section 6.1. To get rid of the time-step restriction, it is customary to use implicit schemes for which $\theta > 0$; for instance, $\theta = 1.0$ gives a fully implicit scheme. Unfortunately, this introduces other difficulties like excessive numerical diffusion for large time steps and the need for a fast solver for the corresponding nonlinear system of equations; see Section 6.2.

If, for instance, the impact of capillary forces is important, methods with high-order spatial resolution may be more appropriate to prevent the numerical diffusion from dominating the capillary diffusion. However, since most reservoir simulation studies are conducted on quite coarse grids, the use of high-resolution schemes, e.g., as discussed elsewhere in this book [22], has yet to become widespread in the reservoir engineering community. The primary benefits with the low-order finite-volume schemes are that they are quite robust and easy to implement.

6 A Simulator for Incompressible Two-Phase Flow

In this section we present a full simulator for a sequential IMPES formulation of an incompressible and immiscible two-phase flow system. We provide a Matlab code designed for simulating water injection into a reservoir filled with oil. However, to keep the presentation simple, and the implementation code short, we disregard gravity and neglect capillary forces (i.e., $\nabla p_{cow} = 0$). Under these assumptions, equations (33) and (35) reduce to the following system of equations:

Listing 5. TPFA finite-volume discretisation of $-\nabla(K\lambda(s)\nabla u) = q$.

```
function [P,V]=Pres(Grid,S,Fluid,q)

% Compute K*lambda(S)
[Mw,Mo]=RelPerm(S,Fluid);
Mt=Mw+Mo;
KM = reshape([Mt,Mt,Mt]',3,Grid.Nx,Grid.Ny,Grid.Nz).*Grid.K;

% Compute pressure and extract fluxes
[P,V]=TPFA(Grid,KM,q);
```

$$-\nabla \cdot \mathbf{K}\lambda(s)\nabla p = q, \tag{42}$$

$$\phi\frac{\partial s}{\partial t} + \nabla \cdot \big(f(s)v\big) = \frac{q_w}{\rho_w}. \tag{43}$$

Since we only inject water and produce whatever reaches our producers, the source term for the saturation equation becomes

$$\frac{q_w}{\rho_w} = \max(q,0) + f(s)\min(q,0).$$

To close the model, we must also supply expressions for the saturation-dependent quantities. Here we use simple analytical expressions:

$$\lambda_w(s) = \frac{(s^*)^2}{\mu_w}, \qquad \lambda_o(s) = \frac{(1-s^*)^2}{\mu_o}, \qquad s^* = \frac{s - s_{wc}}{1 - s_{or} - s_{wc}}.$$

Here s_{or} is the irreducible oil saturation, i.e., the lowest oil saturation that can be achieved by displacing oil by water, and s_{wc} is the connate water saturation, i.e., the saturation of water trapped in the pores of the rock during formation of the rock; see Section 2.

Listing 5 shows a Matlab code for solving (42) with the TPFA scheme. Observe that the only difference between this code and the single-phase code in Listing 1 is that the input `K`, which corresponds to $\mathbf{K}\lambda$, now depends on saturation. This dependence involves the viscosities μ_w and μ_o, the irreducible oil saturation s_{or}, and the connate water saturation s_{wc}. In a similar manner, the mixed finite-element method introduced in Section 3.4 can easily be extended to two-phase flows. In fact, the only point we need to change is the assembly of $\mathbf{B}$ in (24). In other words, the code in Listing 2 can be extended to two-phase flows similarly by letting `K` represent $\mathbf{K}\lambda$ as in Listing 5. The corresponding relative permeability module is given in Listing 6.

For the saturation equation, we use a finite-volume scheme on the form

$$s_i^{n+1} = s_i^n + (\delta_x^t)_i\Big(\max(q_i,0) - \sum_j f(s^m)_{ij}v_{ij} + f(s_i^m)\min(q_i,0)\Big),$$

where $(\delta_x^t)_i = \Delta t/(\phi_i|\Omega_i|)$. By specifying for time level m in the evaluation of the fractional flow functions, we obtain either an explicit ($m = n$) or a fully

Listing 6. Relative permeabilities of oil and water.

```
function [Mw,Mo,dMw,dMo]=RelPerm(s,Fluid)

S = (s−Fluid.swc)/(1−Fluid.swc−Fluid.sor);   % Rescale saturations
Mw = S.^2/Fluid.vw;                          % Water mobility
Mo =(1−S).^2/Fluid.vo;                       % Oil mobility

if (nargout==4)
  dMw = 2∗S/Fluid.vw/(1−Fluid.swc−Fluid.sor);
  dMo = −2∗(1−S)/Fluid.vo/(1−Fluid.swc−Fluid.sor);
end
```

Listing 7. Matrix assembly in upwind finite-volume discretisation of (43).

```
function A=GenA(Grid,V,q)

Nx=Grid.Nx; Ny=Grid.Ny; Nz=Grid.Nz; N=Nx∗Ny∗Nz;
N=Nx∗Ny∗Nz;                                       % number of unknowns
fp=min(q,0);                                      % production

XN=min(V.x,0); x1=reshape(XN(1:Nx,:,:),N,1);      % separate flux into
YN=min(V.y,0); y1=reshape(YN(:,1:Ny,:),N,1);      % − flow in positive coordinate
ZN=min(V.z,0); z1=reshape(ZN(:,:,1:Nz),N,1);      %   direction (XP,YP,ZP)
XP=max(V.x,0); x2=reshape(XP(2:Nx+1,:,:),N,1);    % − flow in negative coordinate
YP=max(V.y,0); y2=reshape(YP(:,2:Ny+1,:),N,1);    %   direction (XN,YN,ZN)
ZP=max(V.z,0); z2=reshape(ZP(:,:,2:Nz+1),N,1);    %

DiagVecs=[z2,y2,x2,fp+x1−x2+y1−y2+z1−z2,−x1,−y1,−z1]; % diagonal vectors
DiagIndx=[−Nx∗Ny,−Nx,−1,0,1,Nx,Nx∗Ny];            % diagonal index
A=spdiags(DiagVecs,DiagIndx,N,N);               % matrix with upwind FV stencil
```

implicit ($m = n+1$) scheme. Here v_{ij} is the total flux (for oil and water) over the edge γ_{ij} between two adjacent cells Ω_i and Ω_j, and f_{ij} is the fractional flow function at γ_{ij}, which we evaluate using the upwind discretisation in (41). Written in compact form, the two schemes read

$$S^{n+1} = S^n + (\delta_x^t)^T \big(\mathbf{A} f(S^m) + Q^+\big), \qquad m = n, n+1, \tag{44}$$

where S^n is the vector of cell-saturations s_i^n and $f(S^m)$ is the vector of fractional flow values $f(s_i^m)$. Furthermore, Q^+ and Q^- denote the positive an negative parts, respectively, of the vector representing q and $\mathbf{A}$ is a matrix implementing $[f(s)Q^- - \nabla \cdot (f(s)v)]$ on a cell-by-cell basis. A Matlab code for the assembly of $\mathbf{A}$, given a vector of saturations S, is presented in Listing 7.

6.1 An Explicit Solver

The explicit solver is obtained by using $m = n$ in (44). Explicit schemes are only stable provided that the time step Δt satisfies a stability condition (a so-called CFL condition). For the homogeneous transport equation (with $q \equiv 0$), the standard CFL condition for the first-order upwind scheme reads

$$\max_{s\in(0,1)} |f'(s)| \max_i \, (\delta_x^t)_i \sum_j |v_{ij}| \le 2(1 - s_{wc} - s_{or}), \tag{45}$$

where $(\delta_x^t)_i = \Delta t/(\phi_i|\Omega_i|)$. For the inhomogeneous equation, we can derive a stability condition on Δt using a more heuristic argument. Physically, we require that $s_{wc} \le s_i^{n+1} \le 1 - s_{or}$. This implies that the discretisation parameters $|\Omega_i|$ and Δt must satisfy the following dynamic conditions:

$$s_{wc} \le s_i^n + (\delta_x^t)_i \Big(\max(q_i, 0) - \sum_j f(s^n)_{ij} v_{ij} + f(s_i^n) \min(q_i, 0) \Big) \le 1 - s_{or}.$$

This condition is saturation dependent and must therefore be enforced on every saturation timestep, and will generally give a time-varying time-step Δt^n. However, it is possible to derive an alternative stability condition that gives a new bound on the timestep only when the velocity field has been recomputed. To this end, note that since we have assumed incompressible flow, and employed a mass conservative scheme to solve the pressure equation, the interface fluxes v_{ij} satisfy the following mass balance property:

$$v_i^{\text{in}} = \max(q_i, 0) - \sum_j \min(v_{ij}, 0) = -\min(q_i, 0) + \sum_j \max(v_{ij}, 0) = v_i^{\text{out}}.$$

Thus, since $0 \le f(s) \le 1$ it follows that

$$-f(s_i^n) v_i^{\text{in}} \le \max(q_i, 0) - \sum_j f(s^n)_{ij} v_{ij} + f(s_i^n) \min(q_i, 0) \le (1 - f(s_i^n)) v_i^{\text{in}}.$$

This implies that the general saturation dependent stability condition holds in Ω_i if the following inequality is satisfied:

$$\max\left(\frac{f(s_i^n) - 0}{s_i^n - s_{wc}}, \frac{1 - f(s_i^n)}{1 - s_{or} - s_i^n} \right) (\delta_x^t)_i v_i^{\text{in}} \le 1. \tag{46}$$

Finally, to remove the saturation dependence from (46), we invoke the mean value theorem and deduce that (46) holds whenever

$$\Delta t \le \frac{\phi|\Omega_i|}{v_i^{\text{in}} \max\{f'(s)\}_{0\le s\le 1}}. \tag{47}$$

This condition is saturation dependent only through the velocities v_{ij} and the timestep Δt need therefore only be modified each time we compute a new solution to the pressure equation.

The fully explicit upwind method incorporating the stability condition (47) is given in Listing 8. A few points in the implementation are worth noting. First of all, instead of coding the finite-volume stencil explicitly, we represent it as a matrix-vector product to make full use of Matlab's inherent efficiency for matrix-vector operations and vectorisation of loops. Second, in the derivation

Listing 8. Explicit upwind finite-volume discretisation of (43).

```
function S=Upstream(Grid,S,Fluid,V,q,T)

Nx=Grid.Nx; Ny=Grid.Ny; Nz=Grid.Nz;                    % number of grid points
N=Nx*Ny*Nz;                                             % number of unknowns
pv = Grid.V(:).*Grid.por(:);                            % pore volume=cell volume*porosity

fi =max(q,0);                                             % inflow from wells
XP=max(V.x,0); XN=min(V.x,0);                             % influx and outflux, x-faces
YP=max(V.y,0); YN=min(V.y,0);                             % influx and outflux, y-faces
ZP=max(V.z,0); ZN=min(V.z,0);                             % influx and outflux, z-faces

Vi = XP(1:Nx,:,:)+YP(:,1:Ny,:)+ZP(:,:,1:Nz)-...           % total flux into
    XN(2:Nx+1,:,:)-YN(:,2:Ny+1,:)-ZN(:,:,2:Nz+1);        %   each gridblock
pm = min(pv./(Vi(:)+fi));                                 % estimate of influx
cfl = ((1-Fluid.swc-Fluid.sor)/3)*pm;                     % CFL restriction
Nts = ceil(T/cfl);                                        % number of local time steps
dtx = (T/Nts)./pv;                                        % local time steps

A=GenA(Grid,V,q);                                         % system matrix
A=spdiags(dtx,0,N,N)*A;                                   % A * dt/|Omega_i|
fi =max(q,0).*dtx;                                        % injection

for t=1:Nts
 [mw,mo]=RelPerm(S,Fluid);                                % compute mobilities
 fw = mw./(mw+mo);                                        % compute fractional flow
 S = S+(A*fw+fi);                                         % update saturation
end
```

of the saturation solver we have tacitly assumed that the porosities are all nonzero. This means that porosity fields generally must be preprocessed and values below a certain threshold replaced by a small nonzero value. Finally, to get an exact value for $\max\{f'(s)\}_{0\leq s\leq 1}$ is a bit cumbersome, but numerical approximations are readily available. Here we have expanded $f'(s)$ as follows:

$$\frac{\partial f}{\partial s} = \frac{\partial f}{\partial s^*}\frac{\partial s^*}{\partial s} = \frac{1}{1-s_{wc}-s_{or}}\frac{\partial f}{\partial s^*},$$

and computed a rough approximation to the upper bound for $\partial f/\partial s^*$ that corresponds to the current choice of viscosities.

Example 3. Let us revisit the quarter-five spot from Example 1, but now assume that the reservoir is initially filled with pure oil. To produce the oil in the upper-right corner, we inject water in the lower left. We assume unit porosity, unit viscosities for both phases, and set $s_{or} = s_{wc} = 0$. In this nondimensional model, it takes one time unit to inject one pore-volume water, i.e., the time unit corresponds to the number of injected pore volumes of water.

Listing 9 contains a sequential splitting code for simulating the water injection problem up to water-breakthrough in the production well with the explicit upwind scheme. (For the homogeneous quarter-five spot it is known that breakthrough occurs when approximately 0.7 pore-volumes of water have been injected). The algorithm is quite simple. First, we set up the grid, the

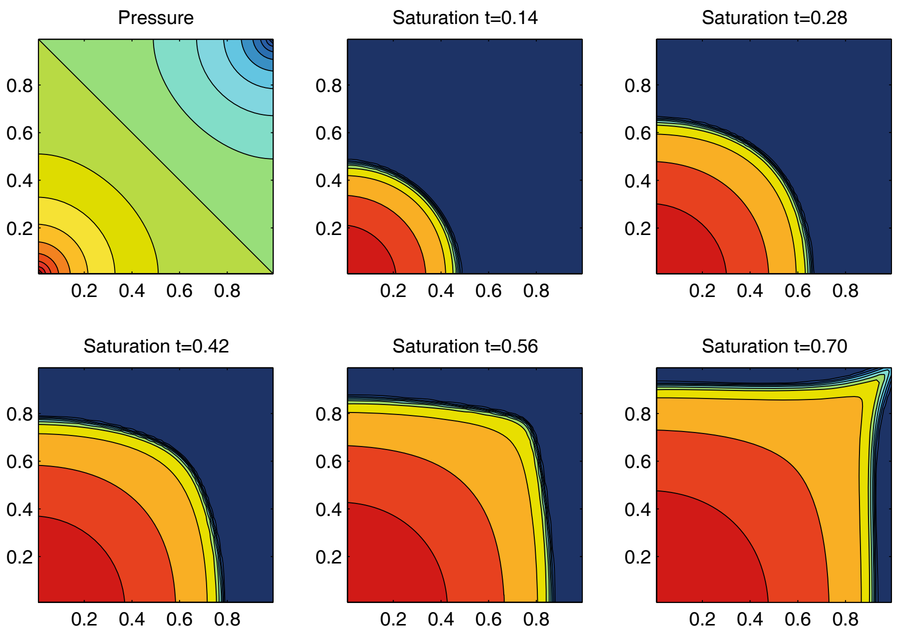


Fig. 6. Pressure and saturation profiles for the homogeneous quarter-five spot.

Listing 9. Two-phase, immiscible, incompressible simulator for a homogeneous quarter-five spot.

```
Grid.Nx=64; Dx=1; Grid.hx = Dx/Grid.Nx;              % Dimension in x−direction
Grid.Ny=64; Dy=1; Grid.hy = Dy/Grid.Ny;              % Dimension in y−direction
Grid.Nz=1; Dz=1; Grid.hz = Dz/Grid.Nz;               % Dimension in z−direction
N=Grid.Nx∗Grid.Ny;                                   % Total number of grid blocks
Grid.V=Grid.hx∗Grid.hy∗Grid.hz;                      % Cell volumes
Grid.K=ones(3,Grid.Nx,Grid.Ny,Grid.Nz);              % Unit permeability
Grid.por =ones(Grid.Nx,Grid.Ny,Grid.Nz);             % Unit porosity
Q=zeros(N,1); Q([1 N])=[1 −1];                       % Production/injection

Fluid.vw=1.0; Fluid.vo=1.0;                          % Viscosities
Fluid.swc=0.0; Fluid.sor=0.0;                        % Irreducible saturations

S=zeros(N,1);                                        % Initial saturation
nt = 25; dt = 0.7/nt;                                % Time steps
for t=1:nt
  [P,V]=Pres(Grid,S,Fluid,Q);                        % pressure solver
  S=Upstream(Grid,S,Fluid,V,Q,dt);                   % saturation solver

  % plot filled contours at the midpoints of the grid cells
  contourf(linspace(Grid.hx/2,Dx−Grid.hx/2,Grid.Nx),...
           linspace(Grid.hy/2,Dy−Grid.hy/2,Grid.Ny),...
           reshape(S,Grid.Nx,Grid.Ny),11,'k');
  axis square; caxis([0 1]);                         % equal axes and color
  drawnow;                                           % force update of plot
end
```

fluid properties and generate the initial saturation distribution. Then, the solution is advanced in time by repeating the following two steps: (i) solve the pressure equation and compute edge velocities; (ii) using the fixed edge velocities, solve the fluid transport equation a time step Δt.

In Figure 6 we have plotted the initial pressure profile and the saturation profiles at five equally-spaced time levels as computed on a uniform 64×64 grid. The saturation profile consists of a leading shock-wave, in which water immediately displaces a fraction of the in situ oil. For a unit viscosity ratio, the post-shock saturation value equals $\sqrt{2}/2$. Behind the front, the water saturation is increasing monotonically towards the injector, meaning that more oil is gradually displaced as more water is injected. Close to the injector, the level curves are almost circular, corresponding to the circular symmetry in pressure at the injector. As more water is injected, the leading water front develops a finger extending toward and finally breaking through to the producer.

6.2 An Implicit Solver

Implicit schemes are unconditionally stable in the sense that there is no CFL condition on the size of the time step. On the other hand, implicit discretisation of (43) gives rise to *nonlinear* systems of equations. Such systems are normally solved with a Newton or a Newton–Raphson iterative method. These methods typically give second-order convergence, but it is well known that they can be very sensitive to the initial guess. It is therefore common to use an alternative technique to compute an initial approximation for each time step, and then continue with a Newton or Newton–Raphson method.

We shall employ a Newton–Raphson method to solve the implicit system. However, since we do not want to mix too many different methods, we assume that the initial approximation for each Newton–Raphson iteration (which will be the saturation field from the previous time step) is sufficiently close to the solution S^{n+1}. In practice, this means that one often has to start with small time steps after each update of the velocity field. In the code that we present in Listing 11 the timesteps are selected in a dynamic fashion in order to ensure convergence. That is, if the Newton–Raphson iteration has not converged in ten iterations, then the time step is simply decreased by a factor two so that the original timestep is split into two sub-timesteps. If the Newton–Raphson still does not converge in less than ten iterations, the original time step is split into four sub-timesteps, and so on. In commercial software, the maximum timestep is controlled by using an estimate of the time discretisation error. Here we have chosen the maximum timestep by convenience to be five days.

To derive the Newton–Raphson method for the implicit upwind discretisation, consider the following homogeneous equation (see (44)):

$$0 \equiv G(S^{n+1}) = S^{n+1} - S^n - (\delta_x^t)^T \big[\mathbf{A} f(S^{n+1}) + Q^+\big]. \tag{48}$$

Listing 10. Implicit upwind finite-volume discretisation of (43).

```
function S=NewtRaph(Grid,S,Fluid,V,q,T);

N = Grid.Nx*Grid.Ny*Grid.Nz;                            % number of unknowns
A = GenA(Grid,V,q);                                     % system matrix

conv=0; IT=0; S00=S;
while conv==0;
 dt = T/2^IT;                                           % timestep
 dtx = dt./(Grid.V(:)*Grid.por(:));                     % timestep / pore volume
 fi  = max(q,0).*dtx;                                   % injection
 B=spdiags(dtx,0,N,N)*A;

 I=0;
 while I<2^IT;                                          % loop over sub−timesteps
  S0=S; dsn=1; it=0; I=I+1;

  while dsn>1e−3 & it<10;                               % I T E R A T I O N
   [Mw,Mo,dMw,dMo]=RelPerm(S,Fluid);                    % mobilities and derivatives
   df=dMw./(Mw + Mo)−Mw./(Mw+Mo).^2.*(dMw+dMo); % df_w/ds
   dG=speye(N)−B*spdiags(df,0,N,N);                     % G'(S)

   fw = Mw./(Mw+Mo);                                    % fractional flow
   G = S−S0−(B*fw+fi);                                  % G(s)
   ds = −dG\G;                                          % increment ds
   S = S+ds;                                            % update S
   dsn = norm(ds);                                      % norm of increment
   it  = it+1;                                          % number of N−R iterations
  end

  if dsn>1e−3; I=2^IT; S=S00; end                       % check for convergence
 end

 if dsn<1e−3; conv=1;                                   % check for convergence
 else IT=IT+1; end                                      % if not converged, decrease
end                                                     %   timestep by factor 2
```

Assume now that we have obtained an approximation $\tilde{S}$ to the true fix-point S^{n+1}. Then, by a Taylor expansion, we have

$$0 = G(S^{n+1}) \approx G(\tilde{S}) + G'(\tilde{S})\big(S^{n+1} - \tilde{S}\big),$$

and a new, and hopefully better, approximation to S^{n+1} is obtained by $\tilde{S}+d\tilde{S}$, where $d\tilde{S}$ satisfies $-G'(\tilde{S})d\tilde{S} = G(\tilde{S})$. For the implicit scheme (48) we have

$$G'(S) = I - (\delta_x^t)^T \mathbf{A} f'(S),$$

where $f'(S) = [f'(s_i)]_i$. The code for the fully implicit upwind discretisation is given in Listing 10.

Example 4. In Example 3 we have seen the typical behaviour of a two-phase oil-water system. Let us now consider a reservoir with a much higher degree of realism by revisiting the two-dimensional SPE-10 models from Example 2. We consider an incompressible oil-water system, for which $s_{wc} = s_{or} = 0.2$, $\mu_w = 0.3$ cp, and $\mu_o = 3.0$ cp. The reservoir is initially filled with oil, meaning

Listing 11. Two-phase, immiscible, incompressible simulator for the top layer of the SPE-10 model.

```
Grid.Nx=60; Grid.hx=20*.3048;                         % Dimension in x-direction
Grid.Ny=220; Grid.hy=10*.3048;                        % Dimension in y-direction
Grid.Nz=1;  Grid.hz=2*.3048;                          % Dimension in z-direction
N=Grid.Nx*Grid.Ny*Grid.Nz;                            % Number of grid celles
Grid.V=Grid.hx*Grid.hy*Grid.hz;                       % Volume of each cells
Fluid.vw=3e-4; Fluid.vo=3e-3;                         % Viscosities
Fluid.swc=0.2; Fluid.sor=0.2;                         % Irreducible saturations
St = 5;                                               % Maximum saturation time step
Pt = 100;                                             % Pressure time step
ND = 2000;                                            % Number of days in simulation

Q=zeros(Grid.Nx,Grid.Ny,1);                           % Source term for injection
IR=795*(Grid.Nx*Grid.Ny/ (60*220*85));                %  and production. Total
Q(1,1,:)=IR; Q(Grid.Nx,Grid.Ny,:)=-IR; Q=Q(:);        %  rate scaled to one layer

load Udata; Grid.K=KU(:,1:Grid.Nx,1:Grid.Ny,1);       % Permeability in layer 1
Por=pU(1:Grid.Nx,1:Grid.Ny,1);                        % Preprocessed porosity in layer 1
Grid.por=max(Por(:),1e-3);

S=Fluid.swc*ones(N,1);                                % Initial saturation
Pc=[0; 1]; Tt=0;                                      % For production curves
for tp=1:ND/Pt;
  [P,V]=Pres(Grid,S,Fluid,Q);                         % Pressure solver
  for ts=1:Pt/St;
    S=NewtRaph(Grid,S,Fluid,V,Q,St);                  % Implicit saturation solver
    subplot('position',[0.05 .1 .4 .8]);              % Make left subplot
    pcolor(reshape(S,Grid.Nx,Grid.Ny,Grid.Nz)');      % Plot saturation
    shading flat; caxis([Fluid.swc 1-Fluid.sor]);     %

    [Mw,Mo]=RelPerm(S(N),Fluid); Mt=Mw+Mo;            % Mobilities in well-block
    Tt=[Tt,(tp-1)*Pt+ts*St];                          % Compute simulation time
    Pc=[Pc,[Mw/Mt; Mo/Mt]];                           % Append production data
    subplot('position',[0.55 .1 .4 .8]);              % Make right subplot
    plot(Tt,Pc(1,:),Tt,Pc(2,:));                      % Plot production data
    axis([0,ND,-0.05,1.05]);                          % Set correct axis
    legend('Water_cut','Oil_cut');                    % Set legend
    drawnow;                                          % Force update of plot
  end
end
```

that the initial water saturation is equal the connate water saturation $s(x, 0) \equiv s_{wc}$. The porosity is strongly correlated with the horizontal permeability and contains about 2.5% zero values. To avoid division by zero in these cells, we simply replace the zero values by a certain minimal nonzero value.

Listing 11 contains a simulation routine that loads data and computes the flow using sequential splitting. Since both the porosity and permeability has a strongly heterogeneous structure spanning several orders of magnitude it is not practical to use the explicit scheme due to the inordinately large number of time steps enforced by the stability condition. Instead we use an implicit scheme—the Newton–Raphson method presented in Listing 10.

Figures 7 and 8 show saturation and production profiles during the simulation for the top and bottom layers in the SPE-10 model. In the figures, oil and water cuts refer to the fractional flows of oil and water, respectively, into

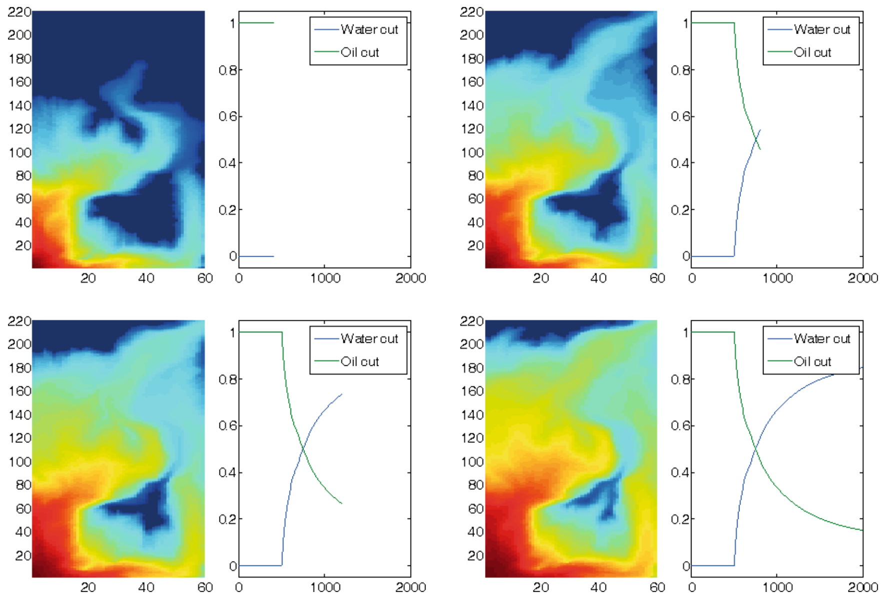


Fig. 7. Saturation and production profiles after 400, 800, 1200, and 2000 days of production for the Tarbert formation in the top layer of the SPE-10 model.

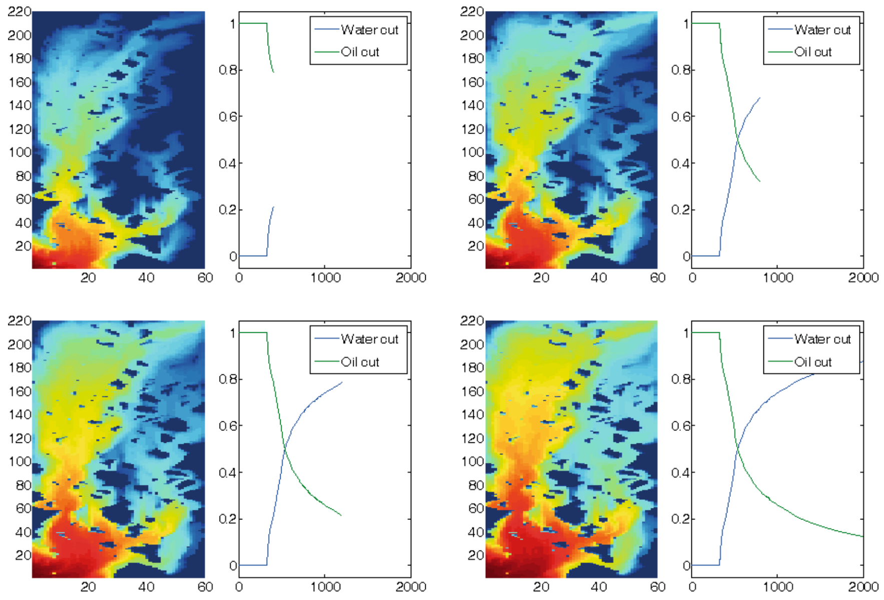


Fig. 8. Saturation and production profiles after 400, 800, 1200, and 2000 days of production for the Upper Ness formation in the bottom layer of the SPE-10 model.

the production well. Here we clearly see that the spaghetti of channelling systems that we have in the Upper Ness formation leads to a very different flow scenario from what we observed in the top layer, which has a quite smooth permeability distribution compared to the bottom layer.

Acknowledgement. This research was funded in part by the Research Council of Norway under grant no. 158908/I30.

References

1. J.E. Aarnes, V. Kippe, K.-A. Lie, and A.B. Rustad. Modelling of multiscale structures in flow simulations for petroleum reservoirs. In *this book*.
2. I. Aavatsmark, T. Barkve, Ø. Bøe, and T. Mannseth. Discretization on unstructured grids for inhomogeneous, anisotropic media. part i: Derivation of the methods. *Siam J. Sci. Comp.*, 19(5):1700–1716, 1998.
3. I. Aavatsmark, T. Barkve, Ø. Bøe, and T. Mannseth. Discretization on unstructured grids for inhomogeneous, anisotropic media. part ii: Discussion and numerical results. *Siam J. Sci. Comp.*, 19(5):1717–1736, 1998.
4. T. Arbogast. An overview of subgrid upscaling for elliptic problems in mixed form. In Z. Chen, R. Glowinski, and K. Li, editors, *Current trends in scientific computing*, Contemporary Mathematics, pages 21–32. AMS, 2003.
5. T. Arbogast. Analysis of a two-scale, locally conservative subgrid upscaling for elliptic problems. *SIAM J. Numer. Anal.*, 42(2):576–598, 2004.
6. K. Aziz and A. Settari. *Petroleum reservoir simulation.* Elsevier, London and New York, 1979.
7. J.W. Barker and S. Thibeau. A critical review of the use of pseudorelative permeabilities for upscaling. *SPE Reservoir Eng.*, 12(2):138–143, 1997.
8. D. Braess. *Finite elements: Theory fast solvers and applications in solid mechanics.* Cambridge University Press, Cambridge, 1997.
9. S.C. Brenner and L.R. Scott. *The mathematical theory of finite element methods.* Springer–Verlag, New York, 1994.
10. F. Brezzi and M. Fortin. *Mixed and hybrid finite element methods.* Computational Mathematics. Springer–Verlag, New York, 1991.
11. G. Chavent and J. Jaffre. *Mathematical models and finite elements for reservoir simulation.* North Holland, 1982.
12. Z. Chen and T.Y. Hou. A mixed multiscale finite element method for elliptic problems with oscillating coefficients. *Math. Comp.*, 72:541–576, 2003.
13. Zhangxin Chen, Guanren Huan, and Yuanle Ma. *Computational methods for multiphase flows in porous media.* Computational Science & Engineering. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2006.
14. M. A. Christie and M. J. Blunt. Tenth SPE comparative solution project: A comparison of upscaling techniques. *SPE Reservoir Eval. Eng.*, 4(4):308–317, 2001. url: www.spe.org/csp.
15. M.A. Christie. Upscaling for reservoir simulation. *JPT J. Pet. Tech.*, 48:1004–1010, 1996.
16. L.P. Dake. *Fundamentals of reservoir engineering.* Elsevier, Amsterdam, 1978.

17. M. Delshad and G.A Pope. Comparison of the three-phase relative permeability models. *Transp. Porous Media*, 4:59–83, 1979.
18. A.H. Demond and P.V. Roberts. An examination of relative permeability relations for two-phase flow in porous media. *Water Res. Bull.*, 23:617–628, 1987.
19. B. Engquist and W. E. The heterogeneous multi-scale methods. *Comm. Math. Sci.*, 1:87–133, 2003.
20. B. Engquist and W. E. Multiscale modeling and computation. *Notices Amer. Math. Soc.*, 50(9):1062–1070, 2003.
21. R.E. Ewing. *The mathematics of reservoir simulation.* SIAM, 1983.
22. T.R. Hagen, M.O. Henriksen, J.M. Hjelmervik, and K.-A. Lie. How to solve systems of conservation laws numerically using the graphics processor as a high-performance computational engine. In *this book*.
23. H.H. Haldorsen and C.J. MacDonald. Stochastic modeling of underground reservoir facies (SMURF). In *SPE Annual Technical Conference and Exhibition, 27-30 September, Dallas, Texas*, SPE 16751, 1987.
24. M. Honarpour and S.M. Mahmood. Relative-permeability measurements: An overview. *J. Petr. Tech*, 40:963–966, 1988.
25. T.Y. Hou and X-H. Wu. A multiscale finite element method for elliptic problems in composite materials and porous media. *J. Comput. Phys.*, 134:169–189, 1997.
26. R. Juanes and T.W. Patzek. Relative permeabilities for strictly hyperbolic models of three-phase flow in porous media. *Tranp. Porous Media*, 57(2):125–152, 2004.
27. R. Juanes and T.W. Patzek. Three-phase displacement theory: an improved description of relative permeabilities. *SPE J.*, 9(3):302–313, 2004.
28. L. Lake. *Enhanced oil recovery.* Prentice Hall, Inglewood Cliffs, NJ, 1989.
29. B.B. Maini and T. Okazawa. Effects of temperature on heavy oil-water relative permeability. *J. Can. Petr. Tech*, 26:33–41, 1987.
30. D.W. Peaceman. *Fundamentals of numerical reservoir simulation.* Elsevier, Amsterdam, 1977.
31. P.A. Raviart and J.M. Thomas. A mixed finite element method for second order elliptic equations. In I. Galligani and E. Magenes, editors, *Mathematical Aspects of Finite Element Methods*, pages 292–315. Springer–Verlag, Berlin – Heidelberg – New York, 1977.
32. P. Renard and G. de Marsily. Calculating equivalent permeability. *Adv. Water Resour.*, 20:253–278, 1997.
33. H.L. Stone. Estimation of three-phase relative permeability and residual oil. *J. Can. Petr. Tech*, 12:53–61, 1973.
34. S.M. Skjæveland and Jon Kleppe, editors. *SPOR Monograph, Recent advances in improved oil recovery methods for North Sea sandstone reservoirs.* Norwegian Petroleum Directorate, Norway, 1992.