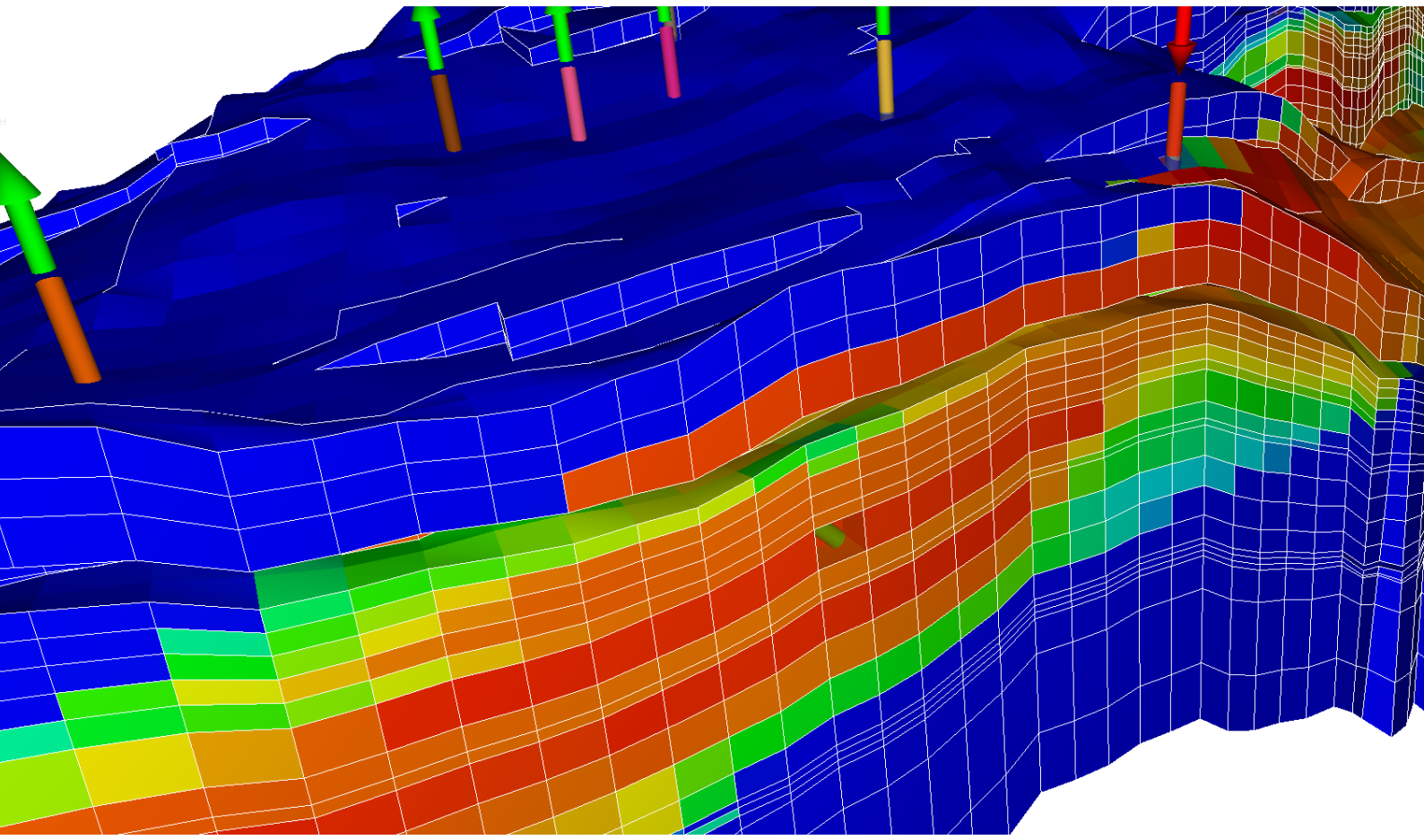
CARL FREDRIK BERG AND PER ARNE SLOTTE

DEPARTMENT OF GEOSCIENCE AND PETROLEUM, NTNU

RESERVOIR SIMULATION
THROUGH OPEN SOURCE SOFTWARE



This book can be found at
https://folk.ntnu.no/carlfrbe/books/resSim.pdf

This book was typeset using LaTeX
LaTeX class: A modified version of tufte-book
Print date: June 15, 2023

Foreword

This book is an introduction to reservoir simulation. In contrast to other books on the same topic, all the software employed in this book are freely available and open source. There are two main reasons for choosing such software. The first is that all students have access the to software, also when they leave the university (or for that matter, a company). This is in contrast to commercial software that require a license. The second reason is the openness of the open source software makes it well suited for research. Commercial software might be a *black box*, where it sometimes is difficult to know exactly which methods have been employed. This is a problem for research which is expected to rigorous. With open source software there are no longer any black box, as the full code base is open for investigation and thereby available for scrutiny. Further, it is possible to change any part of the code to test out new methods and methodology. This enables researchers to test out new ideas and methods in a mature reservoir simulator. Bringing concepts from simplified models, e.g. one dimensional models, into a full reservoir simulator elevates support for the research results. For these reasons, a number of master and PhD theses have already been based on the software presented in this book.

The content of this book started out from the lecture notes of Jon Kleppe, who gave the course Reservoir Simulation at NTNU before me. The shift in software from commercial to open source has changed the content. So too has the shift from Fortran to Python. Still, the main structure is reminiscent of the notes from Kleppe. Thanks to Muhammad Iffan Hannanu for changing the word document files from Kleppe into Latex during a summer project at NTNU.

Trondheim
December 20, 2019
Carl Fredrik Berg

Contents

1
Introduction

> Here had been a treasure of oil that, wisely drilled, would have lasted thirty years: but now the whole field was "on the pump," and hundreds of wells producing so little that it no longer paid to pump them. One sixth of the oil had been saved, and five-sixths had been wasted!
>
> Upton Sinclair, *Oil!*

The main goal of reservoir simulation is to guide management of the subsurface reservoir through future forecasting and prediction, thereby avoiding wasting resources as described in excerpt from the book *Oil!* There is a wide range of subsurface optimization problems that need forecasting: Whether one is working on maximizing the net present value during the oil production from a hydrocarbon reservoir, maximizing fresh water production from an aquifer, or finding the optimal well placement for sequestration of carbon dioxide into a deep subsurface aquifer, reservoir simulation could give predictive input into the optimization. Common for these and similar optimization problems is that we have a porous material, e.g., porous rocks such as sedimentary sandstone or carbonates, sand or soil, and transport of fluids through the porous medium.

The book *Oil!* was the inspiration for the 2007 movie *There Will Be Blood.*

When we say a *reservoir* we usually mean the part of the porous material where the resource collects, e.g., under a dome structure where hydrocarbon collects during their gravity driven upward movement, or a subsurface structure where precipitation is collected during its gravity driven downward movement. Strictly speaking, when monitoring hazardous fluids in soil or predicting sequestration of carbon dioxide, we are not producing a resource, but polluting or storing an unwanted substance. We will still use the term *reservoir*, but now in the meaning the part of the porous material of importance for the process, e.g., the part of the subsurface that will be used for storage. When we talk about reservoirs, we normally consider resources that can be produced by wells. Some shallow resources, such as the tar sands in Alberta, Canada, might be produced by wells, using steam-injection to heat the bitumen, but are more commonly extracted by strip mining. To dis-

Reservoir

tinguish such shallow resources accessible by surface mines from deeper resources that needs to be produced through wells, the shallow resources available through surface mines can be referred to as *deposits,* while deeper resources that needs wells for production can be referred to as *reservoirs.*

The two most important properties of a reservoir is the storage capacity and how easy it is to produce from or inject into the reservoir. The *permeability* of a porous material is a parameter that describes how easy it is for fluids to move through the material. When witnessing a pressure gradient, the fluids in the reservoir will start moving in the direction of the pressure drop. The fluid velocity is described by the *Darcy equation* (see Eq. (3.26)), and is dependent on the viscosity of the fluids and the permeability of the rock material, in addition to the pressure gradient. An extended version of Darcy's equation for systems with more than one fluid phase is additionally dependent on *saturation*, i.e., the fraction of the different fluids filling the pore space (see Eq. (9.3) for the *extended Darcy equation*). A higher permeability (and low viscosity) means that more fluid will be transported at a given pressure gradient. This means that fluids are produced at a higher rate, or that it is easier to inject fluids into the reservoir. A high permeability is therefore almost always preferable. One exception is storage of pollutants, where low permeability hinders the spread of the pollutant, and therefore gives safer storage.

Permeability

Darcy equation

Saturation

Extended Darcy equation

Since reservoirs are permeable, a fraction of the pores must be connected. As we are only interested in the connected part of the pore system, we usually define the *porosity* $\phi = V_c/V_t$ as the fraction of volume of the interconnected pores V_c to the total volume V_t. The spatial extent of the reservoir times the porosity yields the *storage capacity,* which is a principal parameter for the economics of a reservoir.

Porosity

Storage capacity,

During the production or injection phase, reservoir simulation is a continuous activity integrating the stream of data from the reservoir into a reservoir simulation model. Thus the simulation model will be continuously updated, and with it the future predictions will be continuously altered to give the best prediction based on the currently available data of the reservoir.

Reservoir simulators are software designed to model fluid flow in porous media. There is a range of available reservoir simulation software available. The reservoir simulator considered in this text is the software OPM-Flow from the Open Porous Media (OPM) initiative.

Reservoir simulator

See Chap. 2 for description of the different software used in this book, including OPM-Flow, and links to sites where they can be downloaded.

In the context of reservoir modeling, the description of the subsurface, provided as input data to the reservoir simulator, is called a *reservoir simulation model realization*. Since this is a text on reservoir simulation, and not on reservoir modeling in general, we will for the most part simply use the shortened term *reservoir model*. The reservoir model must contains all relevant information about the reservoir, including description of the reservoir geometry (grid

Reservoir simulation model realization

Reservoir model

model), petrophysical properties, and fluid properties.

In order to simulate different production scenarios, the simulator input data must also contain information on wells, well controls, and production constraints. The total simulator input is called a *simulation model*.

Simulation model

Reservoir simulation results are the end result of data gathering and interpretation by many disciplines, including geologists, seismic interpreters and well-log interpretations by petrophysicist. Reservoir simulation therefore ends up as *the* point of contact for technical staff in companies. Analogous, developing reservoir simulation models combines the disciplines of physics, mathematics, chemistry, reservoir engineering, and computer programming. The interdisciplinarity of reservoir simulation can be challenging, as it requires diverse knowledge. It can also be rewarding, as it gives good overview of the workflows leading up to reservoir management decisions.

1.1 Subsurface reservoirs and driving mechanisms

Subsurface reservoirs are typically massive, relatively horizontal layered, structures, divided into fault blocks by faults, as indicated by the outcrop in Fig. 1.1. The reservoir consist of material that is

Figure 1.1: A photo showing layered structures from Kjøllefjord in Finnmark, Norway. The Sami sacrificial site Finnkirka can be seen to the lower right of the image.

both porous and permeable. These structures can be interconnected with the surface, and are then usually filled by water and air. The Netherlands and Israel have been spearheading research into such groundwater flow due to issues with flood control in the Netherlands and problems of inflow of salt water into fresh water reserves connected to the sea in Israel. For oil reservoirs, the pores contain hydrocarbons (oil and/or gas) in addition to brine. The hydrocarbons are hindered in their upward migration by impermeable material, and therefore accumulates. For CO_2 sequestration into an aquifer the reservoir contains CO_2 as a cap under an impermeable material, but will also contain significant amount of CO_2 dissolved in the water.

For groundwater the porous material can be soil and sand, while for deeper aquifers, oil reservoirs and CO_2 storage sites the porous

material is typically sedimentary rocks, such as sandstone and carbonates. Sedimentary rocks have been formed by deposition of sand, clay and organic material. Over time these sediments are pushed downwards and covered by new layers of sediments or other material. When deeper, the sediments can be heated and pressurized, which lead to diagenetic processes changing the structure, e.g., dissolution of specific minerals (e.g., feldspar and mica), overgrowth of other specific minerals (e.g., quartz overgrowth). Diagenetic processes consolidates the material, making the originally unconsolidated material such as loose sand into hard rocks.

As the sediments accumulates over large areas, such as lakes and shores, the sedimentary structures in reservoirs are horizontally extensive. They typically show sheet like structures, called beds. Due to the lateral size, such beds can be interconnected between wells that are kilometers apart. As already mentioned, on top of CO_2 storage sites and hydrocarbon reservoirs is an impermeable bed hindering the upward migration of the CO_2 or hydrocarbons, respectively. Inversely, impermeable beds stops the downward migration of precipitated water and accumulates the fresh water resource in aquifers.

Bed

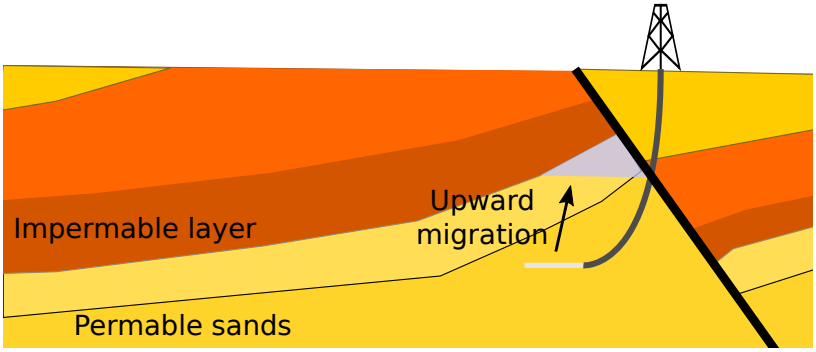


Figure 1.2: Upward migration from the injector and trapping of injected CO_2. The CO_2 is trapped by an impermeable layer and an impermeable fault (indicated as a black line). The perforated part of the injection well is indicated by a light gray color.

Reservoir can also be confined on the sides by faults: Faults are discontinuities in the rock formed from massive movements in the earths crust, which have caused the rock to move in opposite direction on the two sides of the fault. The faults are then abrupt shifts in the elsewhere continuous beds, and these shifts can be large enough to be observed on seismic. Faults have varying degree of permeability, with some faults being practically impermeable, and thereby part of the boundaries of the reservoir. This way they can be part of the trap for upward moving fluids, as shown in Fig. 1.2.

Fault

In carbon dioxide sequestration, fluids are injected into the subsurface reservoir. At the pressure and temperature in the subsurface, the carbon dioxide is often at supercritical conditions, and of a lower density than the displaced brine. This density difference leads to upward migration due to gravity. As the carbon dioxide should be sequestrated for the unforeseeable future, the upward migration needs to hindered, e.g., by injection under an impermeable barrier. Suitable reservoirs for carbon dioxide sequestration are

For typical subsurface temperature and pressure gradients, CO_2 is supercritical when below depths of around 800 m.

therefore often similar to hydrocarbon reservoirs.

The migration of hydrocarbons from source rocks and into reservoirs follows a similar process as the sequestration of CO_2, however, the distance between the source rock and the reservoir is typically much larger than the distance between the CO_2 injection well and the trap for the upward migrating CO_2. Thus, the migration of hydrocarbons cover spatial scales, the *basin scale*, which are much larger than the reservoirs from which hydrocarbons are produced. Sometimes reservoir simulation software tools are used for simulating the process of hydrocarbon accumulation, but reservoir simulation of hydrocarbon resources typically concentrate on the production phase. The accumulation of a resource is also happening over a long time horizon compared to the time horizon for production of the resource. On the flip side of that, for CO_2 sequestration, ensuring safe storage requires simulations on much longer time-scales than the period of CO_2 injection, but then simulations of the time after the injection period. The methods and simulations covered in this book has a stronger focus on the short term, i.e., the production or injection phase.

While sequestration depends on injection into a reservoir, production of fresh water or hydrocarbons depends on fluid extraction from a reservoir. Both fluid injection and fluid extraction starts with drilling wells into the reservoir. If the reservoir pressure is higher than the hydro-static pressure of the fluids inside the well, fluids will start to flow into the well. We then get a flow of fluids from the reservoir, through the well, and up to the surface. This initial fluid extraction driven by the reservoir pressure (and thereby the potential energy of the compressed fluids in the reservoir) is called *primary recovery*. Energy can be added to the reservoir by wells where fluids, e.g., water or gas, are injected into the reservoir. This additional energy can increase recovery from the producing wells. Recovery as a result of injection wells is called *secondary recovery*. For hydrocarbon extraction, where we are only interested in the hydrocarbon portion of the produced fluids, secondary recovery can greatly increase the recovery compared to primary recovery. For this reason, most large oil fields are quickly moved from primary recovery to secondary recovery. Further increase in recovery can be obtained by additives to the injected fluids. Such additives are typically either lowering the surface tension between injected and produced fluid (assuming that the injected and produced fluids are immiscible phases separated by a fluid-fluid interface), or increasing the viscosity of the injected fluid. These two types of changes to the injected fluid typically increase the recovery. All injection methods more advanced than injection of water or gas without additives are called *tertiary recovery* methods.

Primary recovery

Secondary recovery

Tertiary recovery

Already primary recovery involves choices with large impact on the recovery: Where to place the wells? Should one drill horizontal or vertical wells? What drainage rate would yield the optimal production curve? How will the different wells affect each other? With

secondary recovery all these questions get more complex: Where should the injectors be place to optimally drive the resources towards the producers? And at what rate? The complexity is even higher for tertiary methods: Which methods should be used? What is the right amount of additives? When is the optimal onset of the tertiary method? To answer such types of questions we need reservoir simulation, and for reservoir simulation we need a reservoir model.

1.2 What is reservoir modeling?

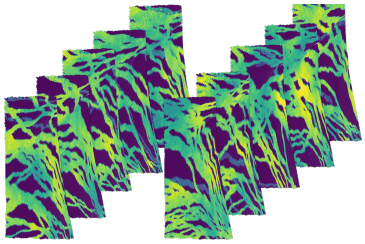


Figure 1.3: An ensemble of reservoir model realizations, showing a spread in static properties arising from uncertainty in the geological model. This ensemble is from the Olympus model created by Fonseca et al. (2018).

Model realizations

The role of reservoir modeling is to summarize as much as possible of our *collective qualitative and quantitative knowledge* about the reservoir in order to give *quantitative answers* to questions related to reservoir development and reservoir management. The product of reservoir modeling is a reservoir model.

Digital representations of the sub-surface are among the products of reservoir modeling, and they are absolutely necessary in order to get quantitative answers through reservoir simulation. To be more specific, these digital representations are called *model realizations*. They exist in the form of geo-model realizations (static) and simulation-model realizations (dynamic), with the latter being what we will simply call reservoir models in the context of reservoir simulation.

Since our knowledge about the true sub-surface is always limited, a single realization can not give the uncertainty span for outcomes that is needed for making good reservoir management decisions. In order to span out the uncertainty, an ensemble of statistically representative realizations is needed, and the actual reservoir model therefore comprise a set of parameterized recipes for creating model realizations. Furthermore, since different models are needed for answering different questions, all models should be based on a common set of concepts and data, stored in a *knowledge database*. This database can be a true knowledge management system, but in practice most companies rely on a mixture of databases (typically for measured data) and more informal storage formats.

Knowledge database

The number of software tools involved in reservoir modeling is extensive, and range from data management tools and asset simulators to specialist simulators for tertiary recovery: Knowledge management systems and database tools are often called *shared earth models*, and serve as the basis upon which ideally all modeling is built. Work-flow managers enables the writing of parameterized recipes for creating model realizations, and exist both internally in various software packages, and as separate work-flow management software. Geo-modeling tools, such as Petrel and RMS, are static model builders, and may contain modules all the way from seismic interpretation to the building of reservoir simulation model grids filled with appropriate properties. Reservoir simulators, such as OPM-Flow, Eclipse, Intersect (IX), Tempest MORE, and tNavigator,

Shared earth models

Work-flow managers

Geo-modeling tools

Reservoir simulators

are used to evaluate the dynamic response of a reservoir to proposed drainage strategies and interventions, while asset simulators, such as Pipe-It, are used for optimizing systems of multiple reservoirs coupled to common top-side equipment and infrastructure.

Asset simulators

1.3 Classification of models

Reservoir models may be classified either based on the purpose of the model, i.e., which questions the model is supposed to answer, or based on the scope of the model, i.e., which sub-surface features are included.

The model purpose can be

- Field development:
  The main questions to be answered in field development are the number of wells and their placement, sub-sea vs. platform, and capacities of sub-sea and top-side equipment such as separators. The main tool for answering these questions is usually a full-field reservoir simulation model. A full field model may contain several reservoirs, with or without pressure communication.

- Asset optimization:
  In a reservoir modeling context, the purpose of asset optimization is to optimize top-side equipment and infrastructure which couple a number of fields, both with regards to which equipment is to be installed and their capacities, and with regards to how the production of each field should be prioritized. It is also of interest to evaluate how different asset-wise production strategies will affect the recovery from each reservoir. Typically a set of coupled, and simplified, full-field reservoir models are used in this context.

- Long term production optimization:
  Short term production optimization, such as rapid alteration of production rates to limit gas production, is mostly affecting the fluid and pressure distribution close to the wells. This is typically on a spatial and time scale much smaller than what is captured by reservoir simulation models. However, by determining targets and limits on the operation of wells and groups of wells, long term production optimization determines the framework in which short term optimization operate. In addition to full field models, more detailed models that cover parts of reservoirs, such as zones, fault-blocks or near-well regions, are typically used in this process.

- Selection of in-fill well targets:
  The process of identifying targets for in-fill wells, which are drilled in order to replace or complement existing wells with declined production, requires input from maps of remaining in-place resources together with reservoir simulation. Full field models can be used in an initial screening phase, while more

detailed models, covering reservoir regions in the vicinity of proposed sites, are used for more detailed target selection and definition.

- Well planning and geo-steering:
  Models that are used in the planning drilling targets and well paths must contain the geology of the overburden in addition to reservoir properties. Geo-steering is the process of utilizing Logging-While-Drilling (LWD) data for guiding the well-path while drilling. Detailed, and continuously updatable, near well models populated with relevant properties related to the logging tools, such as resistivity, are crucial for a successful application of this technology.

- Tertiary recovery:
  Tertiary recovery methods typically influence reservoir flow in ways that are not well described by traditional reservoir scale simulation models. In these cases models with fine-grid representation of geology, and often also special reservoir simulators, are needed in order to evaluate possible effects of applying the tertiary recovery method.

- Data interpretation:
  Reservoir models that cover the volumes influenced by a well test are used both for planning the test and for well test interpretation.

Static vs. dynamic model

The main classification based on scope is the distinction between a *static model*, which comprise the geological features with limited changes during production, and a *dynamic model*, which contain the additional features needed for simulating fluid flow during production. For simulation of the fluid flow in the reservoir we thus need a dynamic model. As already mentioned, we will simply call this dynamic reservoir simulation model the *reservoir model*, while the static model would sometimes be referred to as the *geo-model*.

1.4 What is reservoir simulation?

Reservoir simulation is the use of computers to simulate the fluid flow in a reservoir model by numerically solving equations for fluid flow. Reservoir simulation software are commonly using finite volume methods for conservation of mass, with finite difference methods to solve the flow equations determining flux over the volume element boundaries. Such methods will be treated in detail in this book.

Fluid density and viscosity, and fluid phase behavior, is dependent on pressure, temperature, and composition. Reservoir simulators are distinguished by how this thermodynamic description, conventionally called the PVT-model[1], is handled.

The first distinction is between *thermal* and *isothermal* simulators.

[1] The thermodynamic description is conventionally called the PVT-model. The temperature is often treated as constant in reservoir simulation, while the composition is variable, so the term $pVTx$-model, or even pVx-model, would actually be more descriptive.

Thermal vs. iso-thermal simulation

Although injected fluids can have significantly different temperatures than the reservoir, except for near well region, the temperature changes in reservoirs are often so small that they do not influence fluid and rock properties significantly. In this case we can treat the temperature as constant, and we only need to solve for the conservation of mass. Note that changes in near well properties due to the injection of fluids with significantly different temperature than the reservoir should be taken into account also in this case. For other processes, for instance the injection of steam into heavy oil reservoirs, temperature changes are significant, and must be simulated. This is thermal simulation, where we need to account for the conservation of energy in addition to the conservation of mass.

Compositional vs. black-oil simulators

The second distinction is between *compositional* simulators, which are based on an explicit equation of state, and simulators using table based thermodynamic descriptions, typically the *black oil model*.

Black oil model

The composition of fluids is described in terms of components. A component is a single chemical species, such as water (H_2O), sodium chloride (NaCl), oxygen (O_2), carbon-dioxide (CO_2), different hydrocarbon species (C_mH_n), etc. For real subsurface systems, the composition of the different fluids are never fully described. The number of components also tend to be very large so that including all of them in a simulation is computationally prohibitive. The components used in the simulation is thus always *pseudo components*. A pseudo component description is a mapping from a detailed description using real chemical species into a simplified description with a small number of components. The pseudo components may be defined by lumping real components together. As an example, we might consider the pseudo-component C_{6+} consisting of all hydrocarbon components C_6 and heavier, or the pseudo-component salt, consisting of all salts dissolved in a brine. Real components can also be distributed over pseudo components, such as in the black oil model where the light, and some intermediate, hydrocarbon components are distributed between two pseudo components.

Pseudo components

Dependent on pressure, temperature, and total composition, the fluid components in the pore space may be distributed over several phases in mutual equilibrium. Most reservoir simulators are three-phase simulators, and in a petroleum context these phases are brine, oil, and gas. The vadose zone is the unsaturated zone between the surface and the water table. Thus the pore space in the vadose zone contains two fluid phases; a liquid phase, which is mainly composed of water, and a gas phase with the composition similar to atmospheric air. Below the water table (the pheratic zone) the pore space is fully saturated by a single liquid phase.

The density and viscosity of water rich phases such as brine and ground water are only slightly dependent on the composition and amount of dissolved pollutants, salts, and gases. In these cases the fluid flow can be modeled as a single component flow and dis-

solved components can be tracked as a passive tracer. Typically this is the case with miscible pollutants at small concentrations. We can also track the movement of the pseudo components "injected sea water" and "formation water" in a similar way. If the composition has a significant influence on the liquid properties, such as in the case of sea water intrusion in fresh water aquifers and dissolved CO_2 in brine during CO_2 sequestration, we will need to calculate the fluid properties from the composition.

When simulating petroleum reservoirs, the formation water, that is the brine phase, is usually treated as being immiscible with the hydrocarbon rich phases, and the composition is assumed constant. Thus, the formation water is seen as a phase containing a single pseudo component called water. The oil and gas phase can, on the other hand, exchange components and the composition and saturation of the two phases is determined by thermodynamic equilibrium. In a compositional simulator, this phase equilibrium is calculated using an explicit equation of state and a number of pseudo components closely related to the actual hydrocarbon constituents. Determining the equilibrium state at a given pressure, temperature, and total composition, is called a flash calculation, and in black oil simulators, this equilibrium is inferred from tables. A table lookup is evidently much more efficient than numerically solving a set of equilibrium equations. Since a black oil simulator also work with only two pseudo components, it will use much less computational power than a compositional simulator. The two pseudo components in the black oil model is conventionally called oil and gas. The gas component represents the volatile part of the hydrocarbon mixture, which is in the gas phase at standard conditions, while the oil pseudo component represents the non-volatile part which is in the liquid (oil) phase at standard conditions. In general, both reservoir oil and reservoir gas may contain each component. Sometimes this symmetric formulation is called the wet-gas model, and a distinction is made between this and the dry-gas model, where the gas phase contain no oil component. There is also a dead-oil model, where the components are immiscible[2].

[2] The reader should be warned that the term black oil model is not consistent among authors. In many texts is used for what we here call the dry-gas model, some texts also use the term for the dead-oil model.

The black oil model can be used for simulating any system with two pseudo components

As formulated, the black oil model is used to express the thermodynamic equilibrium between hydrocarbon phases. However, since the model is table based it can be used to simulate almost any system with two pseudo components. We can for example, model CO_2 injection into an aquifer with the black oil model. In this case we have the two pseudo components injected gas (carbon dioxide with additional gas components) and water, and the two phases gas (typically a supercritical mixture of CO_2 with H_2O and additional components in injected gas) and brine (with dissolved CO_2). Since the brine phase usually is denser than the CO_2 rich phase, it is actually common in this context to identify the brine with the oil phase in the black-oil model, and the water component with the oil pseudo component, even though the naming might be confusing.

1.5 *Reservoir simulation workflows*

The use of reservoir simulation correlates with the size and complexity of the reservoir, as the use of reservoir simulation is related to the capital investment and operating costs. Small onshore reservoirs might not use reservoir simulation at all, while large offshore hydrocarbon reservoirs will have a dedicated team of reservoir engineers working solely for a single field. Geologists and reservoir engineers start working on a field right after discovery, trying to assess if the discovery is economically viable to produce. They then need to evaluate the production potential, cost, and risks associated with different possible development plans.

Before any reservoir simulation study, the goals of the study needs to be properly defined. For a small and simple reservoir, produced by pressure depletion, one might not need simulations at all, while for a tertiary driving mechanism the simulation study could end up complex and costly, involving a a number of specialized reservoir models on different scales. Fig. 1.4 shows a flow chart of a typical workflow for reservoir simulations aiming at delivering future prediction of recovery. These predictions are the deliveries

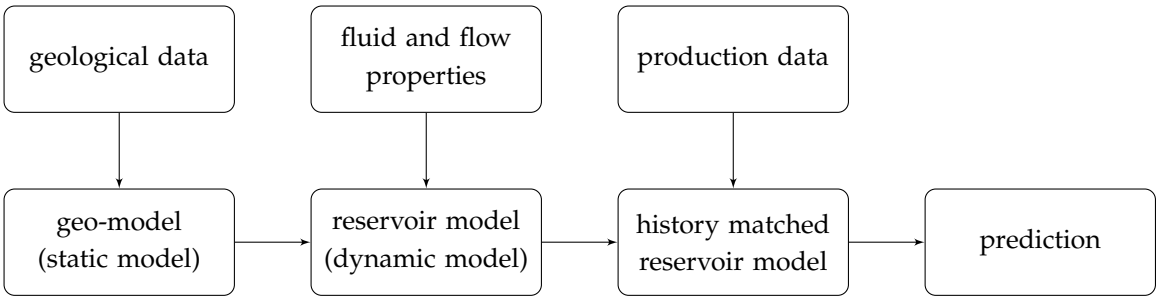


Figure 1.4: Flow chart of a typical reservoir simulation workflow.

from the reservoir simulation work, and they are used as decision support for the management of the reservoir

The first part of any reservoir simulation study is collection of geological data, e.g., seismic data, well log data and core analysis data. Together with analogue data, these geological data must be integrated into the geo-model (the static model). The geo-model contains information on reservoir compartments, and on the distribution of porosity and permeability inside these compartments.

The second part is collection of additional data needed for the dynamic model; fluid samples is analyzed in order to build a PVT model, and special core analysis (SCAL) supply data for the saturation functions (relative permeability and capillary pressure). Saturation functions will be associated with different parts of the reservoir based on the distribution of different geological bodies. One also need to decide on the grid size appropriate for reservoir simulation. The grid size must meet an appropriate balance between computational feasibility and numerical and geological accuracy.

Reservoir simulations are invariably uncertain, and especially so

pre production and in the early stages. Reservoir heterogeneity, the distribution of faults, the shape of relative permeability etc., are all uncertain factors that contributes to the overall uncertainty in the reservoir simulation results. The uncertainty of these parameters can be reduced by additional measurements such as well tests, core analysis and improved seismic. When the production starts, the reservoir model will be improved further by incorporating production data. It is up to the reservoir engineer, in collaboration with the colleagues who have delivered the different data sets, to decide the uncertainty level for the different data sources. The uncertainty in prediction is best represented by an ensemble of model realizations, each giving a different prediction. This is however demanding both in terms of computer- and manpower, so simpler workflows using a single base-case realization plus sensitivities approach is often applied. The tuning of reservoir models in order to be consistent with production data is traditionally known as *history matching*, and the history matched reservoir model is our best approximation for the subsurface behavior build on all available data.

History matching

2
Software

> Cause I'm a 21st century digital boy
> I don't know how to live, but I got a lot of toys
>
> Bad Religion - 21st Century (Digital Boy)

This book will only use open source software. The reservoir simulator used in this book is the open source software OPM-Flow from the Open Porous Media (OPM) initiative. For visualization of reservoir models and simulation results, we will use the open source visualization tool ResInsight, also part of OPM. For scripting we will use the Python programming language, including the `ecl` package. For optimization we will use the Petroleum Field Development Optimization Framework (FieldOpt), which is an open source optimization framework developed at NTNU.

More information about OPM, including the software OPM-Flow and ResInsight, can be found on `https://opm-project.org/`.

The OPM initiative started in 2009 as a collaboration between Equinor and several research institutions (including SINTEF), and the focus of the initiative has been to develop open source software for simulation and visualization of flow and transport in porous media.

2.1 OPM-Flow

OPM-Flow is a fully-implicit black-oil simulator. The simulator is using automatic differentiation, which enable rapid development of new fluid models. The OPM-Flow simulator has extensions to handle polymer and solvent flooding options.

To read more about automatic differentiation, a good introduction is given in (Lie, 2016).

The main operating system for OPM-Flow is a Linux, and there are repository packages for the most common Linux distributions (Ubuntu and CentOS). OPM-Flow can also be run using a Docker container, which is possible in Microsoft Windows. It can also be compiled from source on both Linux and MacOS. Please confirm the OPM website for how to install OPM-Flow: `https://opm-project.org/?page_id=36`.

OPM-Flow is run from the command line, and is therefore easily integrated into sensitivity and optimization workflows. The input data-files for OPM-Flow are similar to input files used for ECLIPSE 100, and many simulation models can be run with both

these reservoir simulation software. However, there are keywords that are not compatible between these two simulators.

The *Flow Documentation Manual* is available from the OPM website `https://opm-project.org/?page_id=955`, and this manual describes all OPM-Flow keywords and their functions.

2.2 Python

Python is a open-source programming language that emphasizes core readability. It is designed to be extensible, and has a wide range of libraries and packages available. It has become very popular in scientific computing, with libraries such as NumPy, SciPy and Matplotlib.

Python is cross-platform, and can be obtained from

`https://www.python.org/`

2.2.1 Naming variables

Note that many of the Python scripts use Hungarian-like notation for prefixes to indicate "what is what":

- `C` for class;
- `t` for objects and structures;
- `h` for handles/pointers;
- `i` for integers;
- `f` for floats;
- `d` for doubles;
- `str` for strings;
- `ch` for chars;
- `b` for booleans;
- `a` for arrays;
- `aa` matrices (i.e., arrays of arrays);
- `aaa` 3D matrices (i.e., arrays of arrays of arrays), and so on.

2.2.2 `ecl` package

The Python package `ecl` is a package for reading and writing the result files from OPM-Flow and other reservoir simulators. that can read/write files compatible with the de-facto market standard formats used defined by the `ECLIPSE` simulator. The file types covered are the restart, init, rft, summary and grid files. Both unified and non-unified and formatted and unformatted files are supported.

This package is a strong tool for pre- and post-processing simulation results from OPM-Flow, e.g. in sensitivity studies and optimization loops. The Python package can be installed form this page:

```
https://github.com/equinor/ecl
```

If you are using pip install, you can install the `ecl` package from the command line:

```
pip install ecl
```

2.3 ResInsight

ResInsight is a open-source visualization software which is part of the OPM project. This software can visualize output from the OPM-Flow simulator. ResInsight is cross-platform, and can be obtained from

```
https://resinsight.org/
```

An important feature in ResInsight is Octave and Python integration. Octave is an open-source software with similar syntax and functionality as Matlab, and can be used to post-process the simulation results from OPM-Flow. The Python integration is a newer add on for using Python to post-process simulation results.

2.4 FieldOpt

FieldOpt is an open source framework for rapid prototyping and testing of optimization procedures such as well placement optimization, reservoir control optimization and inflow-control device optimization. The code is written in C++, and is found as a repository under the Petroleum Cybernetics Group NTNU GitHub page:

```
https://github.com/PetroleumCyberneticsGroup/FieldOpt
```

The GitHub repository also contains a guide for how to compile the software.

The development of FieldOpt is ongoing, and the aim is to contribute to research workflows by providing an interface between optimization methodologies and reservoir simulation software. Several optimization methodologies are implemented in FieldOpt, including compass search, a genetic algorithm and particle swarm optimization.

2.5 Repository

Python codes and example reservoir simulator input files used in this book can be found in the repository

```
https://bitbucket.org/ntnu_petroleum/ressimbook-material
```

The repository can be copied (cloned) by running the command

```
$ git clone https://bitbucket.org/ntnu_petroleum/ressimbook-material.git
```

3
Basic theory for single phase flow

> Nå e me atter på vei
> Gjennom storm, gjennom regn
> Som en Åsgårdsrei
> Og vi går aldri lei
>
> Kvelertak - Kvelertak

In this chapter we will present one-dimensional flow equations for horizontal flow of one single fluid phase, and look at analytical and numerical solutions of pressure as function of position and time. These equations are derived using the continuity equation, Darcy's equation, and compressibility definitions for rock and fluid, assuming constant permeability and viscosity. They are the simplest equations we can have describing transient fluid flow in a porous medium.

To start off, we will first give a short introduction to partial differential equations. The second section contains a note on continuum models, then a brief introduction to Darcy's equation, before we derive the basic equations for flow in porous media. The third section is deriving an analytical expression for one-dimensional single phase flow. This analytical solution will be compared to numerical solutions in the next chapter.

3.1 Partial differential equations

When describing physical processes, including transport in porous media, we commonly describe the problem by a mathematical equation. For porous media flow the potential driving the flow is the pressure gradient, which is one example of a physical quantity that give rise to partial derivatives. The mathematical equations can be solved to give us the state of the system at later times, which introduce partial derivatives with respect to time. Working with partial derivatives is therefore essential for describing and solving physical processes, including subsurface transport processes.

Let $f(x, y, \dots)$ be a function of several variables $x, y, \dots$. As an example, f could be the pressure p, and the variables could be spatial coordinates and time, thus $p(x, y, z, t)$. A *partial derivative* is the derivative with respect to one variable when the other variables

are held constant. As an example, the partial derivative $\partial f/\partial x$ of f with respect to x is the derivative df/dx when all other variables are kept constant. Thus we have

The are many ways to denote the partial derivative, including $\frac{\partial f}{\partial x}$, $\frac{\partial}{\partial x}f$, f_x and $\partial_x f$. We will only use the two first versions.

$$\frac{\partial f}{\partial x} = \left.\frac{df}{dx}\right|_{y,\dots} = \lim_{h\to 0}\frac{f(x+h,y,\dots)-f(x,y,\dots)}{h} \quad , \tag{3.1}$$

where the vertical line in the second term indicates which variables are kept constant.

Partial differential equation

A *partial differential equation* (often referred to by the three letter abbreviation PDE) is an equation that contains partial derivatives. In contrast to *ordinary* differential equations, where the unknown function depends on only one variable, the unknown function in partial differential equations depend on several variables. As an example, consider the partial differential equation

$$\frac{\partial^2 f}{\partial x^2} = \frac{\partial f}{\partial t} \quad . \tag{3.2}$$

Here f is the unknown function, while x and t are the independent variables, thus $f = f(x,t)$ is a function of x and t. It is often beneficial to write the equation on the form $\mathcal{P}f = 0$, where $\mathcal{P}$ is an operator. The corresponding operator $\mathcal{P}$ for Eq. (3.2) would be

$$\mathcal{P} = \frac{\partial^2}{\partial x^2} - \frac{\partial}{\partial t} \quad . \tag{3.3}$$

As already mentioned, a range of physical phenomena, including fluid flow in porous media considered in this book, can be described by partial differential equations. Other physical phenomena described by partial differential equations are the Navier-Stokes equation, the heat equation, Newton's equation of motion etc. In this section on single phase flow in porous media, our unknown function would be the pressure field p, which is dependent on both position and time.

When including gravity, the unknown will not be the pressure field p, but the head field h. See Sec. 3.2.2 for the definition of head and how to include gravity into our flow equations.

There is a range of classes of partial differential equations. The *order* of a partial differential equation is the highest partial derivative in the equation, e.g., Eq. (3.2) is a second order partial differential equation since the highest partial derivative is the partial derivative $\partial^2 f/\partial x^2$ of order two.

Order of a PDE

A partial differential equation is called *linear* if it is a linear combination of the unknown function f and its derivatives. Thus Eq. (3.2) is a linear partial differential equation, while

Linear PDE

$$\frac{\partial f}{\partial t}\frac{\partial^2 f}{\partial x^2} = 0 \tag{3.4}$$

is a non-linear partial differential equation. All second order linear partial differential equation in two variables are therefore on the form

$$A\frac{\partial^2 f}{\partial x^2} + B\frac{\partial^2 f}{\partial x \partial y} + C\frac{\partial^2 f}{\partial y^2} + D\frac{\partial f}{\partial x} + E\frac{\partial f}{\partial y} + Ff + G = 0 \quad . \tag{3.5}$$

There are three basic types of second order linear partial differential equations:

$$\begin{aligned} \text{Parabolic:} \quad & B^2 - 4AC = 0 \\ \text{Hyperbolic:} \quad & B^2 - 4AC > 0 \\ \text{Elliptic:} \quad & B^2 - 4AC < 0 \quad , \end{aligned}$$

where the capital letters are as given in Eq. (3.5). We will encounter two of these basic types in this section; the parabolic type will describe transient single phase fluid flow, while the elliptic type will describe the same equation in steady state.

A partial differential equation that will be recurrent in this book is the following:

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2} = \frac{\partial f}{\partial t} \quad , \tag{3.6}$$

where the function f will be the fluid pressure p. To simplify our notation, we will use the the nabla symbol ∇ to represent the del operator $\nabla = (\partial/\partial x, \partial/\partial y, \partial/\partial z)$. For a general introduction to the mathematical notation, see the mathematical notes in Sec. 17. Taking the dot-product of the del operator with itself, we get

$$\nabla \cdot \nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right) \cdot \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right) = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \tag{3.7}$$

The *dot product*, denoted by a centered dot $\cdot$, of two vectors of equal length is the sum of the products of the corresponding vector entries. Thus, for $\vec{a} = (a_1, a_2, a_3)$ and $\vec{b} = (b_1, b_2, b_3)$ we have $\vec{a} \cdot \vec{b} = \sum_{i=1}^{3} a_i b_i = a_1 b_1 + a_2 b_2 + a_3 b_3$.

Our Eq. (3.6) then simplifies to

$$\nabla^2 f = \frac{\partial f}{\partial t} \quad . \tag{3.8}$$

Treating the multi-dimensional ∇ similar to the single dimensional $\partial/\partial x$ above, this equation is considered as a multi-dimensional parabolic partial differential equation.

3.2 *The diffusivity equation*

In this section we will derive the hydraulic diffusivity equation, which is the fundamental equation for single phase flow in porous media. The starting point for deriving the diffusivity equation is the continuity equation for single phase flow, which is an expression of conservation of mass in a volume element. This will be combined with the Darcy equation to obtain our diffusivity equation.

The driving forces for transport in porous media is viscous, gravity, capillary and diffusion. As we will start off by only considering a single fluid phase with a constant fluid composition, capillary forces and diffusion are not of relevance for us at this moment. We will introduce the effect of gravity in the subsection on the Darcy equation, however, as we will consider horizontal 1D flow in the remainder of this chapter, gravity effects will be excluded too. When we start to consider two phases in Chap. 9 we will introduce capillary forces. Transport by diffusion will not be covered in this book, as this is of minor importance for most subsurface reservoirs due to large length scales. However, diffusion will be discussed in the Sec. 9.4 on numerical diffusion.

3.2.1 *Continuum model*

In-depth treatments for topics in this section can be found in (Whitaker, 1986; Bear, 1988; Torquato, 2001; Bear and Bachmat, 2012; Battiato et al., 2019).

Porous media, such as reservoir rocks, are often extremely complex materials. Modern imaging techniques can characterize the pore structure of small pieces of rock samples to a high level of accuracy. An example of a high-resolution image of a cross-section of a sandstone is shown in Fig. 3.1. We observe that even in such a high resolution image of a small sample the pore-matrix interfaces of this rock is still extremely complex. It is therefore impossible to carry all information for all pore-matrix interfaces for a whole reservoir. This push for describing the flow problems at a larger scale, where the porous material is considered as a continuous effective medium, and where the flow problems can be described by a set of continuous variables and differential equations.

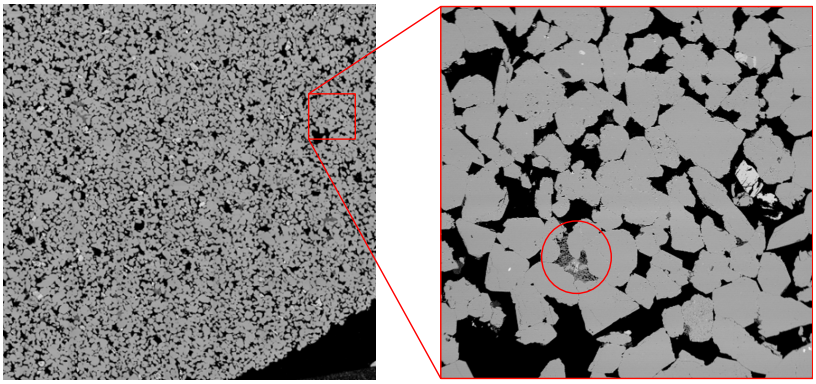


Figure 3.1: An image of the pore structure in a Bentheimer sandstone. The left image shows part of a 1.5 inch cylindrical core sample, while the red box is enlarged in the right image. Some pore filling clay is highlighted by the circle in the right image, and represent pore structure not properly resolved by the current image resolution.

When considering the porous medium as a continuous medium, then any volume element we consider must be large enough so that it still contains something we can consider as a porous medium. If it is too small, then it might contain only part of a single grain or a single pore body, and it is no longer a porous medium. So even the smallest volume elements must contain a large amount of building blocks of our porous material, i.e., a high number of grains or pore bodies.

The continuum scale for a porous medium should not be confused with the continuum scale encountered in other parts of science, and in particular the continuum scale for fluid dynamics. The continuum scale (or macroscopic scale) in fluid dynamics means a scale much larger than the distance between the molecules (considered as the microscopic scale). However, due to the small size of molecules, this is still a very small scale compared to the scale needed for a porous medium continuum scale. When working with porous media, the microscopic scale will be the scale of the individual pore bodies or grains, while the macroscopic scale will be the scale of the effective porous medium consisting of a large number of individual pore bodies. Thus the microscopic scale for fluid dynamics is related to the size of molecules, being nano-meter sized, while the microscopic scale for porous media is on the size of pores, being micro-meter sized, so these two scales are several orders of

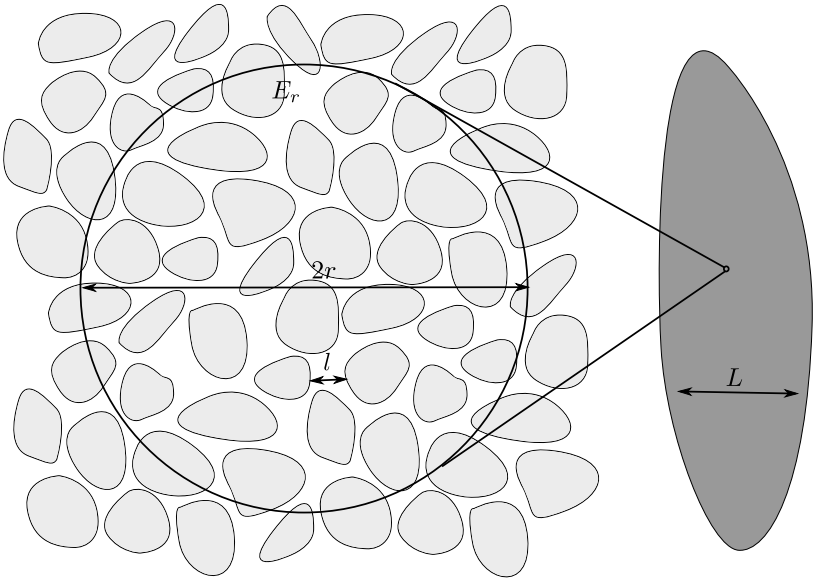


Figure 3.2: A figure showing both the pore-scale and continuum scale of a porous medium.

magnitude different.

When working on the continuum scale, we need an averaging volume to obtain our continuum parameters. For this end we define an *elementary volume* as a subspace $E \subset \mathbb{R}^3$ containing the origin. Since E contains the origin, then for any point x the subspace $E_x = \{x + y \mid y \in E\}$ will contain the point x. By abuse of notation we will use E also for E_x. An example of a widely used elementary volume is the ball E_r of radius r, thus for any x it is the ball of size r centered at x. See Fig. 3.2 for a two-dimensional representation of the elementary volume E_r.

Elementary volume

We will use the elementary volume to define continuum scale properties: A continuum scale porous medium property at every point x will be defined as the effective property inside the elementary volume E. As an example, the porosity at x will be defined as the fraction of the volume of the pore space inside E to the total volume of E. By moving the elementary volume E throughout the space of interest, we assign an effective property value to every point x, thereby obtaining a field of the effective value, e.g., a porosity field.

We observe that the effective properties are dependent on the size of E (but it can be shown that the continuum scale is independent of the shape of E). As an example, if the volume of E goes to zero, then the porosity will be 1 if x is inside the pore space, and 0 if it is inside the matrix. For a zero-sized E the porosity is no longer smooth, but will jump between 0 and 1 depending on whether it is inside or outside the pore space. Thus, for small E the porosity will vary erratically with position. In general small elementary volumes tend to give properties that are rapidly varying. Increasing the size of the elementary volume dampens this variations. This is seen in Fig. 3.3.

A *representative elementary volume* is an elementary volume large

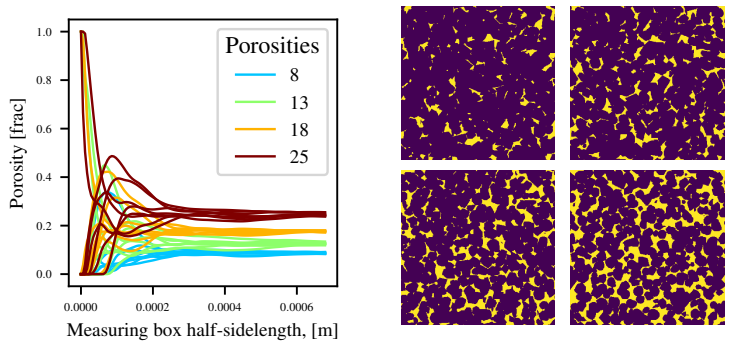


Figure 3.3: A plot showing the evolution of porosity versus elementary volume E_r size r in four models with different porosity. The four images on the right are cross-sections of the four models, with void space in yellow and solid space in purple.

enough for such rapid variations to disappear. For our purpose we seek an elementary volume that is large enough for the effective properties to vary smoothly with position, but still small enough to capture property changes of interest. Unfortunately, there are porous media where such length scales do not exist. Their existence is dependent on the existence of a length scale between the characteristic length on the pore scale and the characteristic scale on the continuum scale. As depicted in Fig. 3.2, we can define a characteristic length on the pore scale l, a continuum scale characteristic length L, and a length scale for the elementary volume r. We thus need that the pore scale characteristic length is much smaller than our averaging length scale r, which again is much smaller than the length scale L for changes in the porous medium on the continuum scale:

For an introduction on how the representative elementary volume can be deduced from the gradient of the effective properties of elementary volumes of different size, please confer (Bear and Bachmat, 2012).

$$l \ll r \ll L \tag{3.9}$$

The existence of a representative elementary volume is the existence of such a scale r. This will give us well defined continuum scale parameters as the effective property from the elementary volume E_r. Except for quite artificial special cases (see, e.g., Howes and Whitaker (1985)), such fields of effective continuum scale parameters are continuous, smooth and differentiable.

3.2.2 *The Darcy equation*

The flow in subsurface reservoirs differs from flows in channels and basins as it takes place inside a porous medium consisting of complex networks of interconnected pores. For flow in porous media there are two essential components: The property of the fluid that flows through the porous medium, and the geometrical structure of pore space of the porous medium itself. The fluid properties are sufficiently described by the viscosity μ and density ρ. The porous medium properties are more complex, but the essentials are summed up by the porosity ϕ and permeability k. Porosity has a clear mathematical definition, already given in the introduction as the fractional volume of the interconnected pore space to the total volume. The definition of the permeability is more complex, as it is a medium property describing the complex geometry of the

pore space. In essence it is a description of how easy, or fast, a fluid flows through a porous medium after factoring out the fluid properties: A medium for which the fluid flow through more easily has a higher permeability.

The first phenomenological description of permeability comes from Henry Darcy. Darcy was working on the water system of Dijon, and considered the flow of water through filter sand (Darcy, 1856). His famous experiment is outlined in Fig. 3.4, and shows flow of volume rate Q through a sand pack. He measured the head of the water, that is the height above a give datum that the water rise in a column (see Fig. 3.4), close to the inlet h_i and outlet h_o of the sand. By changing the angle α of the sand pack, the cross sectional area A and length l of the sand pack, and the volume rate Q of water going through the sand pack, Darcy noted the relationship

$$q = \frac{Q}{A} \propto -\frac{h_o - h_i}{l} \quad , \tag{3.10}$$

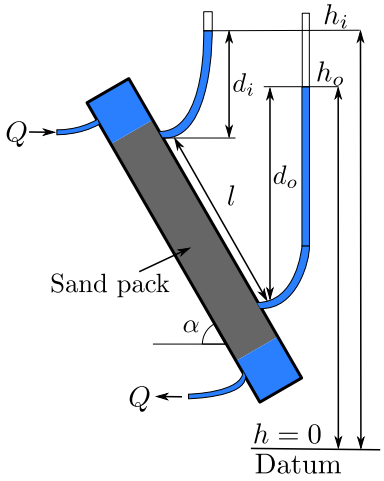


Figure 3.4: A sketch of the experiment conducted by Darcy.

Volumetric fluid flux

where q is the *volumetric fluid flux*. The volumetric flux q is also called the specific discharge. Note that we can define a volumetric flux at any scale, also inside the porous medium. It is important to distinguish between the flow inside the pores, also known as interstitial flow and often denoted by u, and the effective porous medium flow as considered by Darcy. To make the distinction clear we often refer to the volumetric flux as the *Darcy velocity* when it is the effective porous medium volumetric flux we are referring to. This is the volume of fluid flowing Q per area A of porous medium. As the amount of fluid flow Q is measured in meter cubed per second, m^3/s, while area A is given in meter squared, m^2, then the volumetric flux have units meter per second, m/s. The volumetric flux thus has the dimension of a velocity, reflected by the name Darcy velocity. Note that the average interstitial velocity u is higher than the Darcy velocity, at least by a factor $1/\phi$.

Darcy velocity

Darcy investigated water only, thus the viscosity of the fluid was constant. If one also change the viscosity μ of the fluid, then we find the relationship

$$q \propto -\frac{1}{\mu}\frac{h_o - h_i}{l} \quad . \tag{3.11}$$

Note that we are using the head, h, which is the driving force. The pressure at the water-air interface inside the measurement tubes are atmospheric pressure, p_a. Since there is no flow in the measurement tubes for the head, the pressure inside the tubes is governed by hydrostatics, thus the pressure at the measurement point at the sand interface is given by $p_a + \rho g d$, where d is the height between the measurement point and the fluid-air interface in the measurement tube. Assuming a no-flow situation, $Q = 0$, then the pressure difference will be

$$p_o - p_i = (p_a + \rho g d_o) - (p_a + \rho g d_i) = \rho g (d_o - d_i) \quad , \tag{3.12}$$

which is different from zero whenever the angle of the sand pack $\alpha \neq 0$, i.e., whenever the sand pack is not horizontal. Apparently, it

is not the pressure that is driving the fluid flow, as we then have a non-zero pressure gradient but no fluid flow. At no flow, $Q = 0$, the two heads will be equal, $h_i = h_o$, irrespective of the angle α. Darcy only investigated water, considering fluids with different density would have given the proportionality

$$q \propto -\frac{1}{\mu}\frac{\rho g(h_o - h_i)}{l} \quad , \tag{3.13}$$

where ρ is the fluid density and g is the acceleration of gravity. Thus it is ρgh that is the potential driving the fluid flow (and not the pressure).

After including the fluid density and acceleration of gravity, we obtain the proportionality constant we today call the permeability k, which yields Darcy's equation as

$$q = -\frac{k}{\mu}\frac{\rho g(h_o - h_i)}{l} \quad . \tag{3.14}$$

This form of the Darcy equation might be unfamiliar, since it is using the head instead of pressure. It can be rearranged to an equation of pressure, using the correspondence

$$\begin{aligned}\rho g(h_o - h_i) &= \rho g h_o - \rho g h_i \\ &= p_o + \rho g z_o - (p_i + \rho g z_i) = p_o - p_i + \rho g \Delta z \quad ,\end{aligned} \tag{3.15}$$

where $z_i = h_i - d_i$ and $z_o = h_o - d_o$ is the height above datum for the two measurement points, while $\Delta z = z_o - z_i$ is the height difference between the two measurement points. In differential form, we then have the Darcy equation as

$$q = -\frac{k}{\mu}\rho g \nabla h = -\frac{k}{\mu}(\nabla p + \rho g \nabla z) \quad . \tag{3.16}$$

Here ∇z denotes the relative change in elevation in the direction of flow.

It should be noted that in addition to pressure and gravity, also the kinetic energy should be included in the above equation, so that

$$q = -\frac{k}{\mu}(\nabla p + \rho g \nabla z + \nabla E_k) \quad , \tag{3.17}$$

where

$$E_k = \frac{1}{V}\int_V \frac{\rho u^2}{2} dV \quad , \tag{3.18}$$

is the kinetic energy per unit volume, with $\vec{u}$ being the interstitial fluid velocity and V being a representative elementary volume. However, fluid velocities in reservoirs are both quite small and quite constant (except near the wells), so the kinetic energy term is usually neglected.

As reservoirs are typically much more extensive horizontally than vertically, fluid flow is often close to horizontal. It is therefore also quite common to disregard the gravity term $\rho g \nabla z$ in the Darcy equation. However, it is needed for a correct treatment whenever

we are outside the horizontal flow regime, e.g., when treating water flow in soils or CO_2 injection where the initial movement of the CO_2 can be close to vertical. However, when disregarding the gravity term, we obtain the Darcy equation as

$$q = -\frac{k}{\mu}\nabla p \quad . \tag{3.19}$$

Note that we assumed that the total discharge (flow rate) Q was the same in and out of the system, indirectly we then assumed an incompressible fluid. For compressible fluids we can express the Darcy equation in terms of a force potential, Φ:

$$\begin{aligned} q &= -\frac{k\rho}{\mu}\nabla\Phi \\ \Phi &= \int_{p_r}^{p} \frac{1}{\rho(p)} dp + gz \end{aligned} \quad . \tag{3.20}$$

This force potential Φ is also known as a Hubbert potential, as it was introduced by King Hubbert in his treaty on ground-water motion (Hubbert, 1940). Except for an arbitrary constant, the force potential Φ is proportional to the head so that

$$\nabla\Phi = g\nabla h \quad . \tag{3.21}$$

By taking the gradient of the force potential (3.20) and expressing it in terms of ∇p and ∇z we see that the three formulations (in terms of force potential, head, or ∇p and ∇z) are equivalent and Eq. (3.17) is thus also valid for compressible fluids.

The experiment by Darcy is a pseudo one-dimensional experiment, thus the pressure gradient ∇p and the volumetric flux q are both in the same direction, and the permeability k is the proportionality constant for this flow direction. In general, we can consider a three-dimensional system. The pressure field p then give rise to a vector field ∇p, equivalently the head field gives rise to a vector field ∇h. Also the volumetric flux is a vector field $\vec{q}$. We say that the permeability is *isotropic* if it is the same in all directions. If the porous medium is everywhere isotropic, i.e., that the permeability is isotropic everywhere, then the permeability is a scalar field which can be represented by the lower case permeability symbol k.

Isotropic

In general, the permeability can be dependent on the direction of flow. An obvious example would be a porous medium consisting of a bundle of tubes in only one direction; such a medium will have high permeability in the direction of the tubes, while no permeability in all directions perpendicular to the tubes. For the general case the permeability is a tensor of order two, which can be described by a 3×3 matrix:

$$\mathbf{K} = \begin{bmatrix} k_{xx} & k_{xy} & k_{xz} \\ k_{yx} & k_{yy} & k_{yz} \\ k_{zx} & k_{zy} & k_{zz} \end{bmatrix} \quad . \tag{3.22}$$

In general a tensor of order two is a linear operator that produces a vector when applied to a vector. It can be shown that the permeability tensor $\mathbf{K}$ is symmetric (i.e., $k_{ij} = k_{ji}$) and positive definite (i.e., $\vec{x}^T\mathbf{K}\vec{x}$ for every $\vec{x} \neq 0$). See, e.g., Torquato (2001) for a derivation.

As seen in the matrix form above, the permeability matrix might include off-diagonal terms, and the direction of flow will in general

not follow the pressure gradient. Using tensor notation, the three-dimensional version of the Darcy equation will then be

$$\vec{q} = -\frac{\rho g}{\mu}\mathbf{K} \cdot \nabla h \quad . \tag{3.23}$$

General form of the three-dimensional Darcy equation

Disregarding gravity effects, we have the following form of the three-dimensional Darcy equation:

$$\vec{q} = -\frac{1}{\mu}\mathbf{K} \cdot \nabla p \quad . \tag{3.24}$$

If the permeability is isotropic, then the off-diagonal terms are zero, and the diagonal terms are equal, yielding a matrix of the form

$$\mathbf{K} = \begin{bmatrix} k & 0 & 0 \\ 0 & k & 0 \\ 0 & 0 & k \end{bmatrix} \quad . \tag{3.25}$$

Multiplying a vector, e.g., the vector ∇p, which this matrix is equivalent to multiplying by the constant scalar on the diagonal k. Thus for an isotropic medium the three-dimensional form of the Darcy equation reduces to the simpler form (again disregarding gravity):

Three-dimensional Darcy equation for isotropic porous media disregarding gravity

$$\vec{q} = -\frac{k}{\mu}\nabla p \quad . \tag{3.26}$$

For simplicity, we will usually consider isotropic media. The extension to an-isotropic media is fairly straight forward.

3.2.3 *Conservation of mass*

In this subsection we will derive the equation for conservation of mass. The conservation of mass equation is, as the name alludes to, just a mathematical representation of the fact that mass can neither arise nor disappear, thus the change in mass inside a volume element must equal what comes in minus what goes out of that volume element.

Assume a volume element V with surface $\delta V = A$, and let $\vec{n}$ be the unit normal on the surface pointing out of the volume element, as indicated in Fig. 3.5. Since the length of the unit normal vector $\vec{n}$ is one, then the local flux density over the surface is given by the dot product (inner product) $\vec{q} \cdot \vec{n}$, that is, the projection of the volumetric flux (Darcy velocity) vector $\vec{q}$ in the direction of $\vec{n}$, as shown in Fig. 3.5.

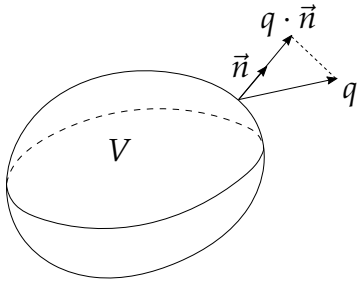


Figure 3.5: A control volume V, as used for mass balance derivations. Here the inner product $\vec{q} \cdot \vec{n}$ gives the flux density, which can be integrated over a surface to give the volume flow.

The volume flow over the full boundary A of the volume element V is given by the integral of the local flux density as

$$\int_A \vec{q} \cdot \vec{n} dA \quad . \tag{3.27}$$

Note that $\vec{q}$ might point either into or out of the volume element. Due to the properties of the dot product, if $\vec{q}$ is pointing inwards, then $\vec{q} \cdot \vec{n}$ is negative since $\vec{n}$ is defined to be pointing outwards. The integral in Eq. (3.27) is then giving the difference between fluid

flowing out and fluid flowing into the volume element. The above equation holds at any scale, thus we could consider the equation either for the interstitial (pore scale) fluid velocity $\vec{u}$ or the effective continuum scale Darcy velocity $\vec{q}$. We will only consider the integral for a representative elementary volume, where $\vec{q}$ is the Darcy velocity, i.e., the effective volumetric flux through the porous material.

As mass is given by volume times density, we see that the amount of mass out minus mass in is given by

$$\int_A \rho \vec{q} \cdot \vec{n} dA \quad , \tag{3.28}$$

where ρ is the fluid density.

Let $\Omega \subset V$ be the pore space inside the volume element V. Then the mass of fluid inside the volume element is given by the integral of fluid density ρ inside the pore space:

$$\int_\Omega \rho dV \quad . \tag{3.29}$$

The above integral can be performed at any scale. We will only operate at a representative elementary volume, where macroscopic properties such as porosity ϕ and the macroscopic fluid density ρ are well defined. With these parameters the fluid mass can be given by

$$\int_V \phi \rho dV \quad . \tag{3.30}$$

In this last equation, ϕ is the porosity, defined as

$$\phi(x) = \frac{\|\Omega_{E_x}\|}{\|E_x\|} \quad , \tag{3.31}$$

where Ω_{E_x} is the pore space inside the elementary volume E_x surrounding the point x.

The pseudo-equation for conservation of mass is

"mass in" − "mass out" = "change in mass"

The difference in mass in and out is given by the negative of Eq. (3.28), while the change in mass is given by the time derivative of Eq. (3.30). We then have the conservation of mass equation

Conservation of mass in a volume element

$$-\int_A \rho \vec{q} \cdot \vec{n} dA = \int_V \frac{\partial}{\partial t}(\phi \rho) dV \quad . \tag{3.32}$$

Applying the divergence theorem, we have

Gauss divergence theorem is found in Sec. 17.3.

$$\int_A \rho \vec{q} \cdot \vec{n} dA = \int_V \nabla \cdot (\rho \vec{q}) dV \quad , \tag{3.33}$$

thus we get

$$\int_V -\nabla \cdot (\rho \vec{q}) dV = \int_V \frac{\partial}{\partial t}(\phi \rho) dV \quad . \tag{3.34}$$

As this equation holds for any volume element (albeit, an element which is large enough to be a representative elementary

Note that in physics it is also common to derive the equation for conservation of mass from the law of mass conservation through the Reynolds transport theorem. To avoid the complications with the material derivative, we have chosen a simpler and hopefully more intuitive approach in this book.

volume), we can use the localization theorem (see Eq. (17.5) in the mathematical notes chapter) to get the general mass conservation equation as:

$$-\nabla \cdot (\rho \vec{q}) = \frac{\partial}{\partial t}(\phi \rho) \quad . \tag{3.35}$$

Equation for conservation of mass.

The volumetric flux is related to the gradient in pore pressure via the Darcy equation, as given by Eq. (3.26):

$$\vec{q} = -\frac{k}{\mu}\nabla p \quad .$$

Darcy equation

Inserting Eq. (3.26) into Eq. (3.35), we get the equation

$$\nabla \cdot \left(\frac{k\rho}{\mu} \nabla p \right) = \frac{\partial}{\partial t}(\phi \rho) \quad . \tag{3.36}$$

General equation for single phase flow

This is the general equation for single phase flow. In this general form of the single phase flow equation, both permeability k, porosity ϕ, fluid density ρ and viscosity μ might be pressure (and temperature) dependent. It can be further extended by including directional permeability $\mathbf{K}$, and by including gravity by using head instead of pressure. Fortunately, for many subsurface problems the change in pressure is small enough to consider the permeability and porosity as constant. If we have an equation of state for fluid density and viscosity, i.e., a description of the fluid density and viscosity as a function of pressure, then we will have a closed system with a single unknown being the pressure.

For special cases we can simplify the single phase flow equation further. Assume a constant (i.e. both pressure independent and spatial constant) permeability k and viscosity μ, then these can be moved outside the derivative in Eq. (3.36), and we get

$$\frac{k}{\mu}\nabla \cdot (\rho \nabla p) = \frac{\partial}{\partial t}(\phi \rho) \quad . \tag{3.37}$$

We may expand the derivatives of the product on both sides of Eq. (3.37), which gives:

$$\frac{k}{\mu}\left(\nabla \rho \cdot \nabla p + \rho \nabla^2 p \right) = \rho \frac{\partial}{\partial t}\phi + \phi \frac{\partial}{\partial t}\rho \quad . \tag{3.38}$$

We will first investigate the left hand side of Eq. (3.38): Compressibility c is a measure of the relative volume change as a response to a pressure change:

compressibility

$$c = -\frac{1}{V}\frac{\partial V}{\partial p}, \tag{3.39}$$

where V is the volume. The liquid compressibility may be used to convert derivatives of density into derivatives of pressure. For a fluid the density $\rho = m/V_f$ is the fraction of mass m divided by the volume of the fluid V_f. Rewriting, we also have $V_f = m/\rho$. We can then derive the fluid compressibility c_f as:

$$c_f = -\frac{1}{V_f}\frac{\partial V_f}{\partial p} = -\frac{\rho}{m}\frac{\partial (m/\rho)}{\partial \rho}\frac{\partial \rho}{\partial p} = -\rho \frac{-1}{\rho^2}\frac{\partial \rho}{\partial p} = \frac{1}{\rho}\frac{\partial \rho}{\partial p}. \tag{3.40}$$

Fluid compressibility: $c_f = \frac{1}{\rho}\frac{\partial \rho}{\partial p}$.

Thus, we see that the first term on the left hand side of Eq. (3.38) is

$$\nabla\rho \cdot \nabla p = \frac{\partial \rho}{\partial p}\nabla p \cdot \nabla p = \rho c_f |\nabla p|^2 \propto c_f \quad . \tag{3.41}$$

Since this term is proportional to the compressibility it may be ignored in the low compressibility limit, e.g., when the fluids are close to in-compressible, such as water and oil. This low compressibility limit assumption would not be appropriate for compressible fluids such as gas.

Low compressibility limit: $c_f \to 0$

The general equation for the left hand side of Eq. (3.38) is

$$\rho\frac{k}{\mu}\left(\nabla^2 p + c_f\left(\nabla p\right)^2\right) \quad . \tag{3.42}$$

As presented earlier, $\nabla^2 = \nabla \cdot \nabla$ is the Laplace operator, which gives the divergence $(\nabla\cdot)$ of the gradient (∇) of a function. Using the low compressibility limit assumption, i.e., assuming $c_f(\nabla p)^2 \ll \nabla^2 p$, then this is simplified to

$$\rho\frac{k}{\mu}\nabla^2 p \quad . \tag{3.43}$$

The right hand side of Eq. (3.38) can be expressed in terms of the time derivative of pressure by applying the chain rule,

$$\begin{aligned}\rho\frac{\partial}{\partial t}\phi + \phi\frac{\partial}{\partial t}\rho &= \rho\frac{\partial \phi}{\partial p}\frac{\partial}{\partial t}p + \phi\frac{\partial \rho}{\partial p}\frac{\partial}{\partial t}p \\ &= \rho\phi\left(\frac{1}{\phi}\frac{\partial \phi}{\partial p} + \frac{1}{\rho}\frac{\partial \rho}{\partial p}\right)\frac{\partial}{\partial t}p\end{aligned} \quad . \tag{3.44}$$

The formation compressibility, c_ϕ, is defined[1] as

Formation compressibility: $c_\phi = \frac{1}{\phi}\frac{\partial \phi}{\partial p}$

$$c_\phi = \frac{1}{\phi}\frac{\partial \phi}{\partial p} \quad , \tag{3.45}$$

and

$$c_t = c_f + c_\phi \quad , \tag{3.46}$$

is called total compressibility. For consolidated sandstone material the rock compressibility is often negligible compared to the fluid compressibility. On the other side, for unconsolidated sands and soils the rock compressibility could be much larger than the fluid compressibility.

Total compressibility: $c_t = c_f + c_\phi$

Using the total compressibility, Eq. (3.44) can be simplified to

$$\rho\frac{\partial}{\partial t}\phi + \phi\frac{\partial}{\partial t}\rho = \rho\phi c_t\frac{\partial}{\partial t}p \quad . \tag{3.47}$$

Equating left (Eq. 3.43) and right (Eq. 3.47) hand side of Eq. (3.38), and then dividing by $\rho\phi c_t$ we get

$$\frac{k}{\mu\phi c_t}\nabla^2 p = \frac{\partial}{\partial t}p \quad . \tag{3.48}$$

This is the (hydraulic) diffusivity equation which is *the* fundamental equation for single phase flow in the low compressibility limit. The

[1] It is left to the reader to show, assuming that the sand grains themselves are incompressible, that the formation compressibility as defined here is also equal to the compressibility of a porous rock sample with total volume V subject to an increased pore pressure: $c_\phi = \frac{1}{V}\frac{\partial V}{\partial p}$.

diffusivity equation is simpler than the general single phase flow equation given by Eq. (3.36). This simplicity came at the expense of the applicability range: The diffusivity equation is assuming the low compressibility limit, thus it is not applicable to, e.g., gases which are highly compressible. A more general form of the diffusivity equation that does not assume a low compressibility limit is

$$\eta \left(\nabla^2 p + c_f \left(\nabla p \right)^2 \right) = \frac{\partial}{\partial t} p \quad . \tag{3.49}$$

The hydraulic diffusivity equation is a multi-dimensional parabolic partial differential equation, and it arises several places in physics; diffusion, heat conduction, electrical conduction etc. The quantity

$$\eta = \frac{k}{\mu \phi c_t} \tag{3.50}$$

Hydraulic diffusivity

is called the (hydraulic) diffusivity. The unit for hydraulic diffusivity η is

$$[\eta] = \frac{[k]}{[\mu][\phi][c_t]} = \frac{\mathrm{m}^2}{(\mathrm{Pa} \cdot \mathrm{s})(\mathrm{Pa}^{-1})} = \frac{\mathrm{m}^2}{\mathrm{s}} \quad , \tag{3.51}$$

where we use that porosity ϕ is dimensionless. The ϕc_t part of the hydraulic diffusivity is called the storativity.

Using the η notation for the diffusivity, we get the following simplified diffusivity equation:

The (hydraulic) diffusivity equation

$$\eta \nabla^2 p = \frac{\partial}{\partial t} p \quad . \tag{3.52}$$

The diffusivity η determines how fast pressure signals move through the reservoir, but it is important to note that the signal moves by a diffusion process, where the actual speed decreases as it spreads. This is very different from a seismic pressure wave which move at constant velocity.

By investigating the different elements of the diffusivity η in Eq. (3.50), we observe that a pressure disturbance moves faster in a high permeable reservoir than in a low permeable reservoir. On the other hand, increased porosity, viscosity or compressibility reduces the speed of the pressure signal.

Given constant boundary conditions, the diffusivity equation will converge to a steady state solution with time. At steady state, the time derivative is zero, giving the equation

$$\nabla^2 p = 0 \quad . \tag{3.53}$$

We observe that this equation is an elliptic linear partial differential equation. This equation is also known as the Laplace equation, named after the french mathematician Pierre-Simon Laplace who formulated and studied this type of equations. The Laplace operator $\Delta = \nabla^2$ is also named after him. Laplace used this operator to study the motion of objects in outer space, and solutions to the Laplace equation are called harmonic functions.

3.3 Analytical 1-dimensional solution

We will now investigate an analytical 1-dimensional solution for single phase flow for a slightly compressible fluid (e.g. water). The underlying partial differential equation is the diffusivity equation given by Eq. (3.52). In one direction, here being the x-direction, we can reduce Eq. (3.52) to its one-dimensional form:

$$\eta \frac{\partial^2 p}{\partial x^2} = \frac{\partial p}{\partial t} \quad . \tag{3.54}$$

1D diffusivity equation

The right hand side of Eq. (3.54) gives the time dependency of the equation. The equation thus describe the transient evolution of the pressure distribution.

When there is no change with time, the system is at steady state. As $\partial p/\partial t = 0$ at steady state, Eq. (3.54) simplifies to

$$\frac{\partial^2 p}{\partial x^2} = 0 \quad , \tag{3.55}$$

1D steady-state equation

at steady state.

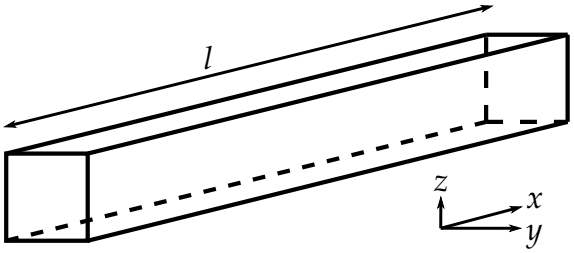


Figure 3.6: The pseudo-1D slab, where the cross-sectional area is constant.

Consider a 1D porous medium of length l and permeability k. Equivalently, we can consider a 3D slab of porous material with constant permeability k and constant cross-sectional area in the $y - z$ direction as depicted in Fig. 3.6. To apply the 1D diffusivity equation, Eq. (3.54), we need the pressure gradient perpendicular to the x-direction to be zero, i.e. $\partial p/\partial y = \partial p/\partial z = 0$. These restrictions hold if they hold initially (i.e. at time zero), we have no-flow boundary conditions on all sides except the end-plates, and we disregard gravity. In the remainder of this subsection we will consider the flow problem as a pure one-dimensional problem, as given by Eq. (3.54).

We want Eq. (3.54) to hold inside a region in the coordinate system given by the variables (x, t). We want our spatial variable x to be bounded, in our case between 0 and l as indicated in Fig. 3.6. The time variable is only bounded on one side, i.e. by the start time t_0. For simplicity, we usually let $t_0 = 0$. The (x, t) region is shown in Fig. 3.7.

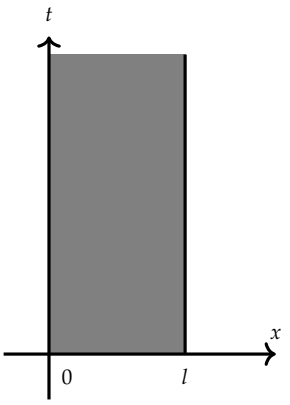


Figure 3.7: Figure indicating the region where Eq. (3.54) holds, i.e. for all $t > 0$ and all $x \in [0, l]$.

To describe our problem, we need an initial condition and boundary conditions in addition to Eq. (3.7). The initial condition is to specify the pressure distribution at a given time, e.g., at time $t = 0$. Thus, our *initial condition* is defined as

$$p(x, 0) = i(x) \mid x \in [0, l] \quad . \tag{3.56}$$

In general, the boundary conditions consist of

$$a_0 \frac{\partial p(0,t)}{\partial x} + b_0 p(0,t) = g_0(t) \tag{3.57}$$

$$a_l \frac{\partial p(l,t)}{\partial x} + b_l p(l,t) = g_l(t) \quad , \tag{3.58}$$

where the first derivative term gives a rate control, while the second term gives a pressure control. A more in-depth presentation of boundary conditions will be given in Sec. 4.5.

For our problem we will consider a boundary condition consisting of a pressure control only, being a constant pressure at the left and right ends; $p(0,t) = p_l$ and $p(l,t) = p_r$ (where the subscripts indicate left and right). Further, we assume as initial condition that the pressure everywhere equals the right side pressure; $p(x,0) = p_r$.

Given our initial conditions and boundary conditions, we now want to solve the general pressure equation given by Eq. (3.54). For this end, we assume that the pressure can be described as a product of two functions; one function dependent only on x and one function dependent only on t:

$$p(x,t) = X(x)T(t) \quad . \tag{3.59}$$

This solution technique is called *separation of variables*, and was first proposed by Joseph Fourier in 1822.

Using the pressure function given by Eq. (3.59) into the partial differential equation Eq. (3.52), we obtain:

$$\begin{aligned} \eta \frac{\partial^2 p}{\partial x^2} &= \frac{\partial p}{\partial t} \\ \eta \frac{\partial^2 XT}{\partial x^2} &= \frac{\partial XT}{\partial t} \\ \eta T \frac{\partial^2 X}{\partial x^2} &= X \frac{\partial T}{\partial t} \\ \frac{X''(x)}{X(x)} &= \frac{T'(t)}{\eta T(t)} \quad . \end{aligned} \tag{3.60}$$

Since the right hand side is dependent on x only, and the left hand side is dependent on t only, both sides must equal a constant C. This gives the following two independent equations:

$$\begin{aligned} X''(x) &= CX(x) \\ T'(t) &= C\eta T(t) \quad . \end{aligned} \tag{3.61}$$

Let us first consider the case of $C = 0$. Then $T'(t) = 0$, so $T(t)$ is a constant. When $X''(x) = 0$, then $X'(x) = E$ for a constant E, and $X(x) = Ex + F$ for E and F constants. As $T(t)$ was also merely a constant, then $X(x)T(t) = Ex + F$, where we have just updated the constants E and F after multiplication with the constant $T(t)$. Thus $Ex + F$ is a solution to Eq. (3.54). For any $C \neq 0$ we can always add the $C = 0$ solution to $X(x)T(t)$ and still have a solution. Thus, in general,

$$X(x)T(t) + Ex + F \quad , \tag{3.62}$$

is a solution to Eq. (3.54) whenever $X(x)T(t)$ is a solution to Eq. (3.54).

Let now $C \neq 0$, then

$$T(t) = De^{C\eta t} \quad , \tag{3.63}$$

would be possible solutions for $T'(t) = T(t)$.

Let us now consider the case of $C > 0$: Then

$$X(x) = Ae^{\sqrt{C}x} + Be^{-\sqrt{C}x} \quad , \tag{3.64}$$

would be possible solutions for $X''(x) = X(x)$. Since $C > 0$, then $T(t) \to \infty$ whenever $t \to \infty$. It is therefore not possible to find a solution that fits our boundary conditions for $C > 0$.

So we are only interested in the solutions when $C < 0$. Then the possible solutions for $X(x)$ and $T(t)$ are given by:

$$\begin{aligned} X(x) &= A\sin\left(\sqrt{-C}x\right) + B\cos\left(\sqrt{-C}x\right) \\ T(t) &= De^{C\eta t} \quad . \end{aligned} \tag{3.65}$$

In Eqs. (3.65) all the capital letters are constants. We are considering the product $X(x)T(t)$, thus we can multiply the constants A and B by D, and thereby remove the D in front of the exponential function. By abuse of notation, we can then assume that the constant $D = 1$, and keep the same notation for the A and B constants.

As explained above, when $X(x)T(t)$ is a solution, so is also

$$\begin{aligned} &X(x)T(t) + Ex + F = \\ &\qquad \left(A\sin\left(\sqrt{-C}x\right) + B\cos\left(\sqrt{-C}x\right)\right)e^{C\eta t} + Ex + F \quad , \end{aligned} \tag{3.66}$$

since the $Ex + F$ part disappears in the derivations $\partial^2/\partial x^2$ and $\partial/\partial t$.

Let us consider Eq. (3.66) with respect to our boundary conditions. We start with the left boundary condition $p(0,t) = p_l$. We have $e^{C\eta t} = 1$ for $t = 0$ and $e^{C\eta t} \to 0$ when $t \to \infty$. Since $x = 0$, both the *sin*-term and x-term disappears. Thus we have $F = p_l - B$ for $t = 0$ and $F = p_l$ for $t \to \infty$, thus $B = 0$.

For the right boundary condition $p(l,\infty) = p_r$ we again have $e^{C\eta t} \to 0$, yielding $El + p_l = p_r$ when $t \to \infty$, so $E = (p_r - p_l)/l$. For $p(l,0) = p_r$ we then see that the *sin*-term must vanish, thus we have $\sqrt{-C} = i\pi/l$ for an integer i. We then see that

$$X(x)T(t) + Ex + F = A\sin\left(\frac{i\pi x}{l}\right)e^{-\frac{i^2\pi^2\eta}{l^2}t} + \frac{p_r - p_l}{l}x + p_l \quad . \tag{3.67}$$

is a solution to the differential equation Eq. (3.54). However, this is a solution for any integer i. Any sum of the $X(x)T(t)$-part of Eq. (3.67) would yield a solution to the differential equation Eq. (3.54). Thus the following expression will also be a solution

$$X(x)T(t) + Ex + F = \sum_i A_i \sin\left(\frac{i\pi x}{l}\right)e^{-\frac{i^2\pi^2\eta}{l^2}t} + \frac{p_r - p_l}{l}x + p_l \quad . \tag{3.68}$$

The last boundary condition is the initial condition that $p(x,0) = p_r$. For $t = 0$ we have $T(t) = 1$, which gives the equation

$$\sum_i A_i \sin\left(\frac{i\pi x}{l}\right) + \frac{p_r - p_l}{l}x + p_l = p_r$$

$$p_r - p_l - \frac{p_r - p_l}{l}x = \sum_i A_i \sin\left(\frac{i\pi x}{l}\right) \quad . \tag{3.69}$$

We can then use Fourier series to find an expression for the factors A_i as:

$$A_i = \frac{2}{l}\int_0^l \left(p_r - p_l - \frac{p_r - p_l}{l}x\right)\sin\left(\frac{i\pi x}{l}\right)dx \quad . \tag{3.70}$$

This gives a solution $p(x,t)$ to the differential equation Eq. (3.54) with the boundary conditions $p(x,0) = p_r$, $p(0,t) = p_l$ and $p(l,t) = p_r$ as

$$p(x,t) = \sum_{i=1}^{\infty} A_i \sin\left(\frac{i\pi x}{l}\right)e^{-\frac{i^2\pi^2}{l^2}\eta t} + \frac{p_r - p_l}{l}x + p_l \quad , \tag{3.71}$$

where the pre-factors A_i are as given by the integral in Eq. (3.70).

Let us now investigate the steady state solution. At steady state we have equation Eq. (3.55). By integration, we then obtain the function for the steady state solution as

$$\begin{aligned} \frac{\partial^2 p}{\partial x^2} &= 0 \\ \frac{\partial p}{\partial x} &= A \\ p(x) &= Ax + B \quad , \end{aligned} \tag{3.72}$$

where A and B are constants. The left boundary condition gives $p(0) = p_l = B$. The right boundary condition gives $p(l) = p_r = Al + p_l$, thus $A = (p_r - p_l)/l$. This gives the following steady state solution:

$$p(x) = (p_r - p_l)\frac{x}{l} + p_l \quad . \tag{3.73}$$

1D steady state single phase equation.

This stead-state solution is plotted in Fig. 3.8.

We see that when $t \to \infty$, then the exponential term in Eq. (3.71) goes to zero. Thus

$$p(x,\infty) = p(x) = (p_r - p_l)x/l + p_l \tag{3.74}$$

equals the steady-state solution. We thus see that our solution converge to the stead-state solution at infinite time.

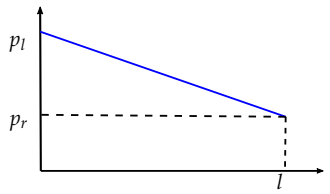


Figure 3.8: Single phase steady state solution for a 1-dimensional system.

If we change the initial pressure distribution $p(x,0) = g(x)$ to any function $g(x)$ different from the constant p_r, then we will get other pre-factors A_i. However, these pre-factors disappear when $t \to \infty$ as the exponential function goes to zero. Thus any initial pressure distribution will converge to the same steady-state solution.

Going back to our original boundary conditions, we said that the initial pressure was equal to the right boundary pressure, i.e.

$g(x) = p(x,0) = p_r$. We then have

$$g(x) - \frac{p_r - p_l}{l}x - p_l = (p_r - p_l)\left(1 - \frac{x}{l}\right) = p_a\left(\frac{x}{l} - 1\right) \quad , \quad (3.75)$$

where $p_a = p_l - p_r$. Using Fourier series, as given in Eqs. (3.70) we obtain the pre-factors

$$\begin{aligned} A_i &= \frac{2p_a}{l}\left(\int_0^l \frac{x}{l}\sin\left(\frac{i\pi x}{l}\right)dx - \int_0^l \sin\left(\frac{i\pi x}{l}\right)dx\right) \\ &= \frac{2p_a}{l}\left(\left[\frac{l}{i^2\pi^2}\sin\left(\frac{i\pi x}{l}\right) - \frac{x}{i\pi}\cos\left(\frac{i\pi x}{l}\right)\right]_0^l + \left[\frac{l}{i\pi}\cos\left(\frac{i\pi x}{l}\right)\right]_0^l\right) \\ &= \frac{2p_a}{l}\left(-\frac{l}{i\pi}(-1)^i + \frac{l}{i\pi}(-1)^i - \frac{l}{i\pi}\right) \\ &= -\frac{2p_a}{i\pi} \quad . \end{aligned} \quad (3.76)$$

Here we are using integration by parts to get $\int x\sin(x)dx = \int cos(x)dx - x\cos(x)$.

Invoking Eq. (3.71) this gives the following solution to the differential equation Eq. (3.54):

$$\begin{aligned} p(x,t) &= \sum_{i=1}^{\infty} -\frac{2p_a}{i\pi}\sin\left(\frac{i\pi x}{l}\right)e^{-\frac{i^2\pi^2}{l^2}\eta t} + \frac{p_r - p_l}{l}x + p_l \\ &= \frac{2}{\pi}(p_r - p_l)\sum_{i=1}^{\infty}\frac{1}{i}\sin\left(\frac{i\pi x}{l}\right)e^{-\frac{i^2\pi^2}{l^2}\eta t} + \frac{p_r - p_l}{l}x + p_l \quad . \end{aligned} \quad (3.77)$$

Analytical solution to the one-dimensional diffusivity equation

This is the analytical solution to the one-dimensional diffusivity equation. We will use this equation repeatedly in the following chapters.

The approximation through a Fourier sum is most challenging around abrupt changes. To investigate how well the Fourier approximation is, we start to investigate the negative of the Fourier series part of Eq. (3.77):

$$p(x,t) = \sum_{i=1}^{\infty}\frac{2p_a}{i\pi}\sin\left(\frac{i\pi x}{l}\right) \quad (3.78)$$

This is the Fourier series of a sawtooth function, which can be implemented in Python for a finite sum as follows:

```
def sawtoothWave(afx,fLength,fPressureAmplitude,inn):
    afSum=0.0
    for ii in range(1,inn+1):
        afSum+=np.sin(ii*math.pi*afx/fLength)/ii
    afPressure=(2.0*fPressureAmplitude/math.pi)*afSum
    return afPressure
```

Note that in Python the indention (i.e. the number of space and tabs at the beginning of a line) determines how the statements are grouped. In this piece of code, all lines after the `def` call have been indented with the same number of leading spaces, and are therefore one group. Note that it is good practice to use either space or tab, and not a mixture, since different operating systems are non-compatible in the interpretation of such characters.

Here `inn` is the number of elements in the sum. The larger the number `inn`, the closer we get to the sawtooth function. We can then calculate the function for different numbers of `inn`:

```
afx=np.arange(-2.0,2.0,0.01)
fLength=1.0
fPressureAmplitude=1.0

plt.plot(afx,sawtoothWave(afx,fLength,fPressureAmplitude,1),label=
    ↪ '1',color='r')
```

The lines starting with `plt.plot` are wrapped here (indicated by a hooked red arrow) for visualization purposes only. You can not break a line in Python, as it then will be interpreted as two statements.

```
plt.plot(afx,sawtoothWave(afx,fLength,fPressureAmplitude,2),label=
    ↪ '2',color='b')
plt.plot(afx,sawtoothWave(afx,fLength,fPressureAmplitude,5),label=
    ↪ '5',color='y')
plt.plot(afx,sawtoothWave(afx,fLength,fPressureAmplitude,10),label
    ↪ ='10',color='g')
plt.plot(afx,sawtoothWave(afx,fLength,fPressureAmplitude,1000),
    ↪ label='1000',color='k')
```

This plots as shown in Fig. 3.9.

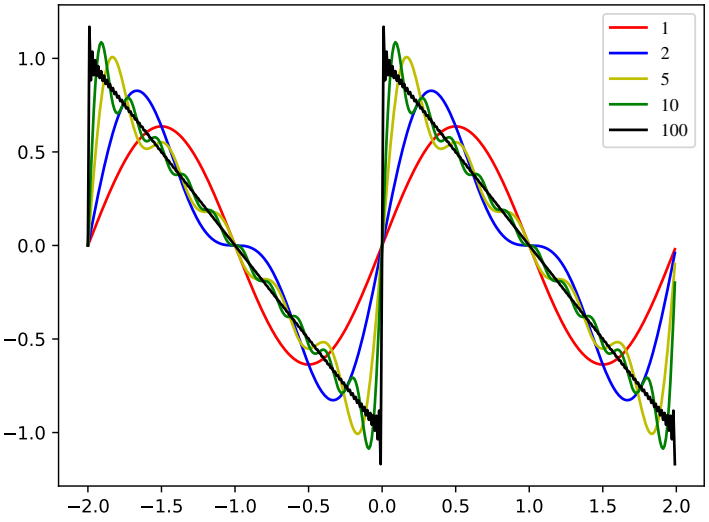


Figure 3.9: Solutions to the sawtooth wave equation for different summation values `inn`. We observe that the function converges to the sawtooth wave shape.

Observe that for a negative amplitude $-p_a$, the sawtooth function will be inverted.

We will now return to the original problem; the solution given by Eq. (3.77). This is an analytical solution of the transient pressure development in the slab. It may be seen from the solution that as time becomes large, the exponential term approaches zero, and the solution becomes the steady state solution.

The Eq. (3.77) can be implemented in Python as:

```
def analytical1DSolution(afx,fLength,fEta,fLeftPressure,
    ↪ fRightPressure,fTime,inn):
    afSum=0.0
    for ii in range(1,inn+1):
        afSum+=np.sin(ii*math.pi*afx/fLength)*np.exp(-ii**2*math.
    ↪ pi**2*fEta*fTime/(fLength**2))/ii
    afPressure=fLeftPressure+(fRightPressure-fLeftPressure)*(afx/
    ↪ fLength+(2.0/math.pi)*afSum)
    return afPressure
```

We see that we need a large enough value `inn` for the solution to converge. We calculate the function for different numbers of `inn`:

```
afx=np.arange(-2.0,2.0,0.01)
fLength=1.0
fEta=1.0
fLeftPressure=2.0
fRightPressure=1.0
```

```
fTime=5E-2

plt.plot(afx,analytical1DSolution(afx,fLength,fEta,fLeftPressure,
    ↪ fRightPressure,fTime,1),label='1',color='r')
plt.plot(afx,analytical1DSolution(afx,fLength,fEta,fLeftPressure,
    ↪ fRightPressure,fTime,2),label='2',color='b')
plt.plot(afx,analytical1DSolution(afx,fLength,fEta,fLeftPressure,
    ↪ fRightPressure,fTime,5),label='5',color='y')
plt.plot(afx,analytical1DSolution(afx,fLength,fEta,fLeftPressure,
    ↪ fRightPressure,fTime,10),label='10',color='g')
plt.plot(afx,analytical1DSolution(afx,fLength,fEta,fLeftPressure,
    ↪ fRightPressure,fTime,100),label='100',color='k',linestyle=
    ↪ 'dashed')
```

This plots as shown in Fig. 3.10. We cannot observe any change to the plot for `inn` > 5, thus the solution seems to converge quite rapidly for the given time step.

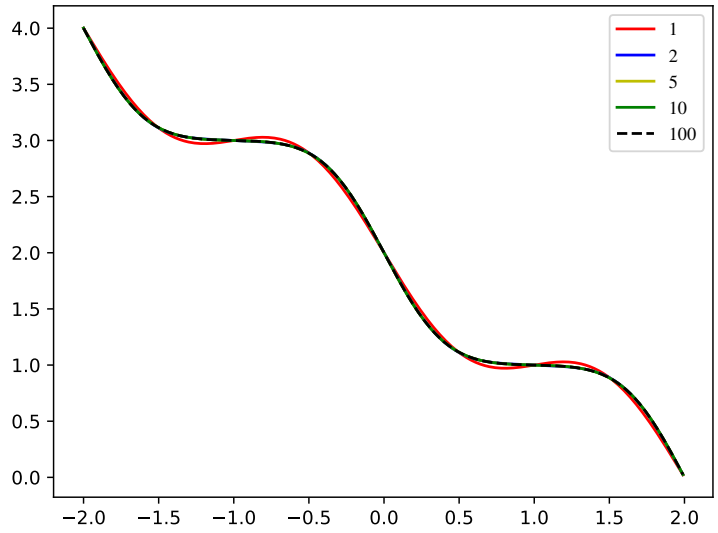


Figure 3.10: Convergence of the analytical solution by increasing the values of `inn`. We observe that the function converges.

We now want to investigate how the solution develops over time, by calculating the analytical solution (with a high enough value `inn`) for different times:

```
fEta=1.0
fLeftPressure=2.0
fRightPressure=1.0
inn=1000


plt.figure()

plt.plot(afx,analytical1DSolution(afx,fLength,fEta,fLeftPressure,
    ↪ fRightPressure,1E-2,inn),label='1E-2',color='r')
plt.plot(afx,analytical1DSolution(afx,fLength,fEta,fLeftPressure,
    ↪ fRightPressure,5E-2,inn),label='5E-2',color='b')
plt.plot(afx,analytical1DSolution(afx,fLength,fEta,fLeftPressure,
    ↪ fRightPressure,1E-1,inn),label='1E-1',color='y')
plt.plot(afx,analytical1DSolution(afx,fLength,fEta,fLeftPressure,
    ↪ fRightPressure,5E-1,inn),label='5E-1',color='g')
```

```
plt.plot(afx,analytical1DSolution(afx,fLength,fEta,fLeftPressure,
    ↪ fRightPressure,1.0,inn),label='1',color='k')
```

This plots as shown in Fig. 3.11. We observe that the analytical solution for different times converge to the steady state solution.

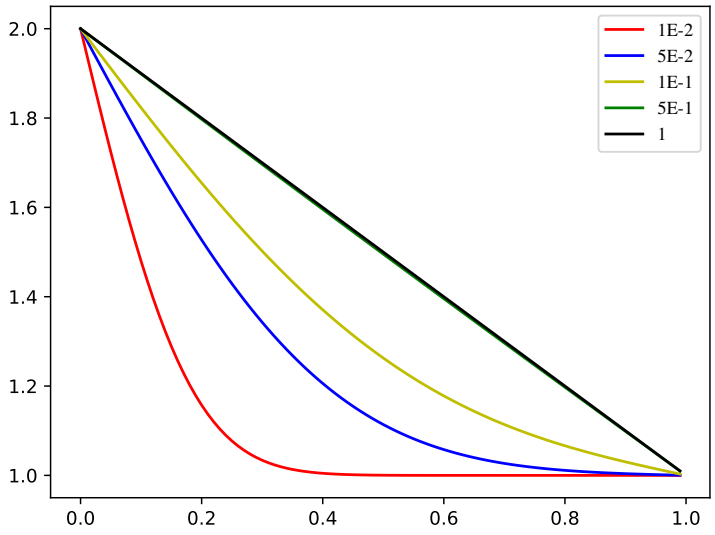


Figure 3.11: The analytical solution for different times. We observe that the pressure converge towards the steady state solution.

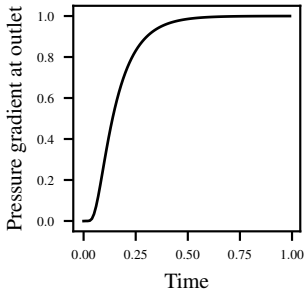


Figure 3.12: The gradient of the pressure curve at the outlet versus time. The pressure gradient can be viewed as a proxy for the outflow on the right side of our model.

Remember from the Darcy equation that the flow rate is proportional to the pressure gradient. Thus the pressure gradient at the outlet would be a proxy for the outflow (production) at the right side. If we plot the pressure gradient at the outlet we get the curve shown in Fig. 3.12. We observe that is takes a while before the pressure pulse reach the right side of our one-dimensional model. After reaching the outlet, the outflow start to rise quickly, before it converge towards a steady state rate.

The time it takes for the effect of the pressure change at the inlet to reach the outlet is dependent on the hydraulic diffusivity (Eq. (3.50)). For most reservoirs the variation in porosity is small compared to the variation in permeability, while the viscosity and compressibility can be estimated from fluid and core samples. The dominating unknown parameter in the hydraulic diffusivity is therefore the permeability. From the time of pressure changes at the outlet of our one-dimensional model one can then estimate the permeability. Analogously, changing the pressure in one well and observing the pressure response in another well can be used to estimate the permeability between the wells in a reservoir. However, the pressure response is also dependent on the geometry of the reservoir.

3.4 Exercises

Exercise 3.1 Create a Python script to calculate the analytical solution for the pressure. The basic data is given in Table 3.1.

a) Plot the pressure function for a set of times that illustrate the transient period.

b) Do a sensitivity study to investigate how the permeability and porosity affects the pressure changes (hence, pick a time when the pressure distribution is in the transient range, and simulate the pressure for a set of different permeability and porosity values around the original values).

	SPE Metric	SI
l	200 m	200 m
k	100 mD	1.0×10^{-13} m^2
μ	1 cP	1×10^{-3} Pa s
ϕ	0.2	0.2
c_t	1×10^{-4} bar^{-1}	1×10^{-9} Pa^{-1}
p_l	200 bar	2×10^7 Pa
p_r	100 bar	1×10^7 Pa

Table 3.1: Data for exercise.

Exercise 3.2

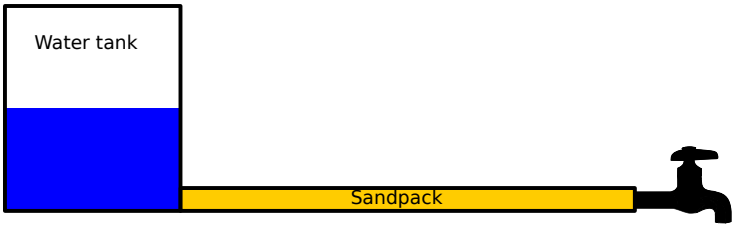


Figure 3.13: A sketch of the water delivery system.

Assume that you have a water delivery system at your cabin, consisting of a tank of water pressurized to 5×10^5 Pa, and a 10 m long sand pack between the pressurized water tank and your faucet. An outline of the system is shown in Fig. 3.13. Assume that the sand pack has a permeability of 1×10^{-12} m^2 and a porosity of 0.3. Other properties are as in Table 3.1.

Calculate the pressure gradient $\partial p/\partial x$ at the water faucet versus time. If we say that we open the faucet at time $t = 0$, and that the water faucet is placed at $x = 10$, what is happening with the pressure gradient

$$\lim_{t \to 0} \frac{\partial p(10)}{\partial x} \tag{3.79}$$

Exercise 3.3

If we use the Darcy equation with the gravity term included to replace the Darcy velocity q in Eq. (3.35), derive the hydraulic diffusivity equation, now with the gravity term included.

4
Finite differences

All I want is the truth,
Just give me some truth

Gimme Some Truth - John Lennon

Analytical solutions for single phase flow are only obtainable for very simple cases, such as the 1-dimensional case considered in the previous chapter. Variations in geometries, rock properties, fluid properties and boundary conditions are in general not possible to solve analytically. Thus, for more realistic reservoir problems we need to solve the equations numerically. The analytical solutions for the simplified cases are still valuable as a comparison for the numerical methods.

In mathematics a finite difference is an expression of the form $f(x+a) - f(x-b)$ for a function f. If we divide this finite difference by $a+b$ we get the finite difference quotient

$$\frac{f(x+a) - f(x-b)}{a+b} \quad . \tag{4.1}$$

Finite differences were introduced by Brook Taylor in his paper *Methodus Incrementorum Directa et Inversa* (Taylor, 1715). They were introduced to study vibrating strings. Taylor also introduced Taylor series, which we will return to later in this chapter.

When both a and b goes to zero, this difference quotient converge to the derivative of f for smooth functions f. Thus difference quotients can be used to approximate derivatives. In contrast to the mathematical term, we will use the term *finite difference* to describe numerical methods which use finite difference quotients to approximate derivatives.

In this chapter we show how to solve partial differential equations by the finite difference method, where we replace the partial derivatives by finite difference quotients and solve the resulting algebraic system. We will introduce different types of finite difference quotients which are used to replace the partial derivatives. We will also give a brief introduction on discretization of the system; spatially into a grid, and time into discrete time-steps, and we will present boundary conditions for our system.

4.1 Explicit and implicit Euler methods

As an introduction to finite differences, we will start looking at two different Euler methods; the explicit or forward Euler method and

the implicit or backward Euler method. These Euler methods are numerical methods for solving ordinary differential equations.

Explicit versus implicit

The difference between the explicit and implicit method is concerned with which step you are evaluation the state of your system. An explicit method calculate the state of the system at a later step by evaluating the system at the current time step. Contrary, an implicit method calculate the state of the system by solving an equation containing both the current and later state of the system.

Explicit Euler method

Let us start with the explicit Euler method. Assume a function $f(x)$ given by the ordinary differential equation

$$\frac{\partial f}{\partial x} = g(x, f) \quad , \tag{4.2}$$

where g is a function, together with an initial value $f(x_0) = f_0$. While the full curve f initially is unknown, the derivative can be calculated since we know the function $g(x, f)$. The rational behind the explicit Euler method is to approximate the curve by small steps along the tangent line given by the derivative. For a small enough step size, the difference between the tangent line and the curve is small, thus this gives a fair approximation.

The starting point (x_0, f_0) is known, thus the next point along the curve can be approximated as

$$f(x_1) \simeq f(x_0) + f'(x_0)(x_1 - x_0) \quad . \tag{4.3}$$

This will approximate $f(x_1)$. Note that we could have chosen to evaluate the derivative $f'(x)$ at any point between x_0 and x_1: The explicit method is to evaluate it at the current point, i.e., evaluate it at the point x_0. As mentioned, for the implicit method we will evaluate the derivative at the next point x_1.

Unfortunately, when we iterate this process, the error from the previous approximations are propagating through the subsequent approximations:

$$f(x_n) \simeq f(x_{n-1}) + f'(x_{n-1})(x_n - x_{n-1}) \quad . \tag{4.4}$$

In other words, the error in the approximation for $f(x_{n-1})$ will be kept in the approximation for $f(x_n)$. Even though the errors are propagating forward, the explicit Euler method is a fair approximation as long as the step sizes are kept small enough.

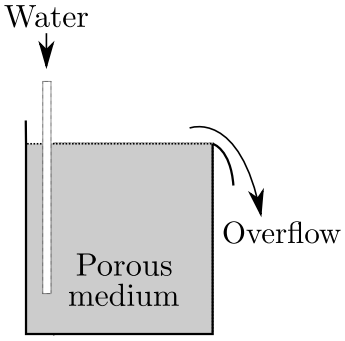


Figure 4.1: A sketch of the example case of water being injected into a porous medium initially filled with oil, and where any excess fluids overflow and leaves the porous medium.

As an example, assume we have a porous medium initially filled with oil, as depicted in Fig. 4.1. Thus the oil saturation s_o at time step zero t_0 is $s_o(t_0) = 1$. In this example we assume that we inject water into the porous medium at a rate Q. The pore volume of the porous medium is V_p. As water and oil are fairly incompressible, we assume that fluids leave the porous medium (e.g. overflows) at the same rate as water is injected. Further, assume that the fraction of oil in the fluid leaving the medium is given by

$$f_o = s_o \quad , \tag{4.5}$$

thus the fraction of oil in the fluid leaving the medium is the same as the fraction of oil inside the porous medium. For this to happen,

we would need full mixing inside the porous medium. Fortunately, we will not deal with the realism of this example.

We can then formulate a differential equation describing the change in oil saturation in the medium from the following mass balance:

$$V_p \frac{\partial s_o}{\partial t} = -Qf_o = -Qs_o \quad . \tag{4.6}$$

The left side of the equation above gives the change in oil volume inside the porous medium per time. As we only inject water, the change in volume of oil inside the porous medium must equal the amount of oil leaving the porous medium, which is given by the right side of the equation. Dividing by V_p on both sides shows that the change in oil saturation is proportional to the amount of oil in the porous medium:

$$\frac{\partial s_o}{\partial t} = -\frac{Q}{V_p} s_o \quad , \tag{4.7}$$

with initial value $s_o(t_0) = s_o(0) = 1$. We immediately recognize this as the Euler function $s_o(t) = e^{-Qt/V_p}$, but for now assume that the analytical solution is unknown so that the equation has to be approximated by a numerical method.

For our numerical scheme, let us use a constant time step $\Delta t = t_{n+1} - t_n$. With an explicit Euler method we evaluate our function at the current time step, thus

$$\begin{aligned} s_o(t_{n+1}) &\simeq s_o(t_n) + s_o'(t_n)\Delta t \\ &= s(t_n) - \frac{Q}{V_p} s_o(t_n)\Delta t = s_o(t_n)\left(1 - \frac{Q\Delta t}{V_p}\right) \quad . \end{aligned} \tag{4.8}$$

Starting with $s_o(0) = 1$ and iterating forward, we get

$$\begin{aligned} s_o(t_n) &= s_o(t_{n-1})\left(1 - \frac{Q\Delta t}{V_p}\right) \\ &= s_o(t_0)\left(1 - \frac{Q\Delta t}{V_p}\right)^n = \left(1 - \frac{Q\Delta t}{V_p}\right)^n \quad . \end{aligned} \tag{4.9}$$

The resulting saturation values when using step size $\Delta t = 1$, $\Delta t = 0.5$ and $\Delta t = 0.1$ are shown in Fig. 4.2, where we have used a value of $Q/V_p = 1$. We observe that we need a small step size to obtain a good approximation. This is a recurring problem with numerical methods; the approximation is dependent on step size.

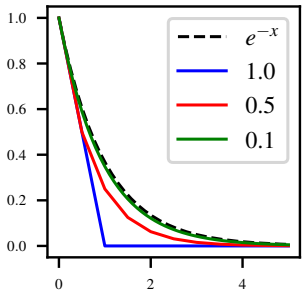


Figure 4.2: Solutions to the explicit Euler method for different step sizes.

As we have seen, in the explicit Euler method we choose to evaluate the function in the previous step. We could as well have evaluated the derivative at any point between t_n and t_{n+1} for the n-th iteration (or between x_n and x_{n+1} if our free variable was x instead of t). The implicit Euler method , on the other hand, is evaluating the function at the next step, thus

$$f(x_{n+1}) \simeq f(x_n) + f'(x_{n+1})\Delta x \quad . \tag{4.10}$$

Implicit Euler method

If we return to our example $\partial s_o/\partial t = -Qs_o/V_p$, but this time use the implicit Euler method, we get

$$s_o(t_{n+1}) \simeq s_o(t_n) - \frac{Q\Delta t}{V_p} s_o(t_{n+1}) \quad . \tag{4.11}$$

We see that we end up with an algebraic equation that has to be solved, with the solution

$$s_o(t_{n+1}) = \frac{s_o(t_n)}{1 + \frac{Q\Delta t}{V_p}} \quad . \tag{4.12}$$

Iterating, we have the algebraic equation

$$s_o(t_{n+1}) = s_o(t_0)\frac{1}{\left(1 + \frac{Q\Delta t}{V_p}\right)^{n+1}} = \left(1 + \frac{Q\Delta t}{V_p}\right)^{-n-1} \quad . \tag{4.13}$$

We have plotted the approximations for step sizes of $\Delta t = 1$, $\Delta t = 0.5$ and $\Delta t = 0.1$ in Fig. 4.3. We observe the resemblance with the explicit method, however the implicit method seems to be working better for the longer step size of $\Delta t = 1$. This is not a coincidence; implicit methods tends to be more stable, as we shall see in the next sections. However, implicit methods are in general more complicated to solve, as we need to solve an algebraic equation. Solving this equation can be computationally demanding. Thus we gain stability and convergence for the cost of using more computational power. Typically, stability and convergence trumps computational power, thus implicit methods are in general the preferred methods.

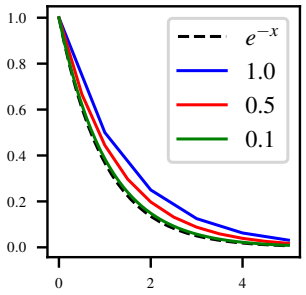


Figure 4.3: Solutions to the explicit Euler method for different step sizes.

As already mentioned several times, we could as well have evaluated the derivative at any point between t_n and t_{n+1} for the n-th iteration. The *Crank-Nicolson method* is evaluating the function at $t_{n-1/2}$, however, with approximating the half-way as an average between t_n and t_{n+1}:

Crank-Nicolson method

$$\begin{aligned} f(x_{n+1/2}) &\simeq f(x_n) + f'(x_{n+1/2})\Delta x \\ &\simeq f(x_n) + \frac{1}{2}\left(f'(x_n) + f'(x_{n+1})\right)\Delta x \quad . \end{aligned} \tag{4.14}$$

Again, we will end up with an algebraic equation, and it can be shown that the solution to this algebraic equation gives

$$s_o(t_n) = s_o(t_0)\left(\frac{1 - \frac{Q\Delta t}{2V_p}}{1 + \frac{Q\Delta t}{2V_p}}\right)^n \quad . \tag{4.15}$$

The resulting approximation for different step lengths are shown in Fig. 4.4. We see that this approximation is superior to the other two approximations, and it can be shown that the error in the Crank-Nicolson method is of a higher order than the explicit and implicit methods.

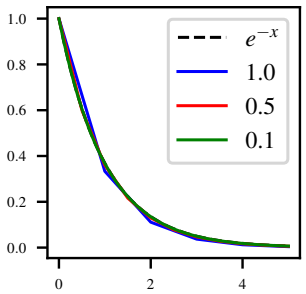


Figure 4.4: Solutions to the Crank-Nicolson Euler method for different step sizes.

As a last example, consider the equation

$$\frac{\partial f}{\partial x} = f \quad . \tag{4.16}$$

This is similar to our equation Eq. (4.7), however, with a switched sign in front of f. Here the solution is e^x.

Using an implicit Euler method and a constant step size $\Delta x = x_{n+1} - x_n$, we get

$$f(x_{n+1}) = f(x_n) + f'(x_{n+1})\Delta x = f(x_n) + f(x_{n+1})\Delta x \quad , \tag{4.17}$$

which by rearranging and iterating gives

$$f(x_{n+1}) = f(x_0)\,(1-\Delta x)^{-n-1} \quad . \tag{4.18}$$

We see from our last equation that a step length $\Delta x = 1$ would give division by zero, thus not a solution. If we choose the step size longer than 1, we see that we divide by a negative number, thus our solution will start to oscillate. These are other known issues with numerical approximations; the numerical solutions can in some cases not exist, they could diverge, or could start to oscillate. While the Crank-Nicolson method could be shown to have a smaller error, it is more prone to such issues, and is therefore seldom used to solve reservoir simulation problems.

4.2 *Difference quotients*

We have now seen how the Euler methods can solve ordinary differential equations. Solving partial differential equations is based on similar ideas: For our finite difference method we need to approximate the derivatives by difference quotients. In the following we will present first and second difference quotients. For this, we will utilize Taylor series.

Let us start by considering the derivatives as limits. The first derivative can be defined as the limit

$$\frac{\partial f(x,y,z,t)}{\partial x} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x,y,z,t) - f(x,y,z,t)}{\Delta x} \quad . \tag{4.19}$$

If we do not go all the way to the limit, but use a finite (but small) value of Δx instead, we obtain a difference quotient. Thus, for a finite (but small) Δx, the expression

The difference quotient is sometimes called the Newton quotient or the Fermat's difference quotient.

$$\frac{f(x+\Delta x) - f(x)}{\Delta x} \quad , \tag{4.20}$$

is a different quotient. In particular, this is a forward difference quotient, while

$$\frac{f(x) - f(x-\Delta x)}{\Delta x} \quad , \tag{4.21}$$

is a backward difference quotient. Without saying so, we used the forward difference quotient to get our explicit Euler method and the backward difference quotient to get the implicit Euler method in the previous section.

For the derivative in Eq. (4.19), we could as well have replaced the forward difference quotient with the backward type, it is just a convention to use the forward type. Do note that for non-smooth functions, the forward and backward definitions can give different results. Since the limit gives the derivative, the difference quotient will be an approximation for the derivative. We see that the approximation gets better the smaller we choose Δx.

In this section we will present several difference quotients and investigate how good of an approximation these difference quotients are. While the development in this section holds for all the

partial derivatives, including the three spacial directions and time as indicated in Eq. (4.19), we will only use the derivative with respect to x in our development to keep the notation simpler.

4.2.1 Taylor series

Taylor series are widely used in deriving methods for numerical solutions. A *Taylor series* is an approximation of a function $f(x)$ around a single point a by an infinite sum of terms that are calculated from the values of the function's derivatives at the given point a:

Taylor series

$$f(a)+\frac{f'(a)}{1!}(x-a)+\frac{f''(a)}{2!}(x-a)^2+\frac{f'''(a)}{3!}(x-a)^3+\dots \quad . \tag{4.22}$$

By Taylor's theorem, we can express the function $f(x)$ as

$$f(x)=f(a)+\frac{f'(a)}{1!}(x-a)+\frac{f''(a)}{2!}(x-a)^2+\frac{f'''(a)}{3!}(x-a)^3+\dots + \frac{f^{(k)}(a)}{k!}(x-a)^k+\frac{f^{(k+1)}(\xi)}{(k+1)!}(x-a)^{k+1} \quad , \tag{4.23}$$

Note that there are conditions on the function f and the distance between x and a for Eq. (4.23) to hold. For some functions there exist points a where the Taylor series does not converge, e.g., $\log(x+1)$ when $a=0$ and $x>1$.

where the last term is the remainder on the Lagrange form, where ξ is a real valued number between x and a. Thus the Taylor series transform a smooth function, i.e., a function that is infinitely differentiable, into a polynomial. This method is based on the fundamental theorem of calculus and integration by parts.

As an example, let us consider the function $f(x)=\sin(x)$. We see that the Taylor series around a point a is

$$\sin(x)=\sin(a)+\frac{\cos(a)}{1!}(x-a)-\frac{\sin(a)}{2!}(x-a)^2 - \frac{\cos(a)}{3!}(x-a)^3+\frac{\sin(a)}{4!}(x-a)^4+\dots \quad . \tag{4.24}$$

If we expand this function around $a=0$, we see that all the terms $\sin(a)=\sin(0)=0$ vanish, and we get the Taylor series

A Taylor series around the value $a=0$ is also called a Maclaurin series.

$$\sin(x)=x-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}+\dots \quad . \tag{4.25}$$

We have plotted the Taylor series for $\sin(x)$ around point $a=0$ for different number of terms in Fig. 4.5. Adding terms in the Taylor series enlarge the distance from a where we have a good approximation of the function. It can be shown that the infinite series converge to $\sin(x)$ for all x.

4.2.2 First-difference quotients

By rewriting (substituting x by $x+\Delta x$ and a by x), we obtain the following equality

$$f(x+\Delta x)=f(x)+\frac{\Delta x}{1!}f'(x)+\frac{\Delta x^2}{2!}f''(x)+\frac{\Delta x^3}{3!}f'''(x)+\dots + \frac{\Delta x^k}{k!}f^{(k)}(x)+\frac{\Delta x^{k+1}}{(k+1)!}f^{(k+1)}(\xi) \quad , \tag{4.26}$$

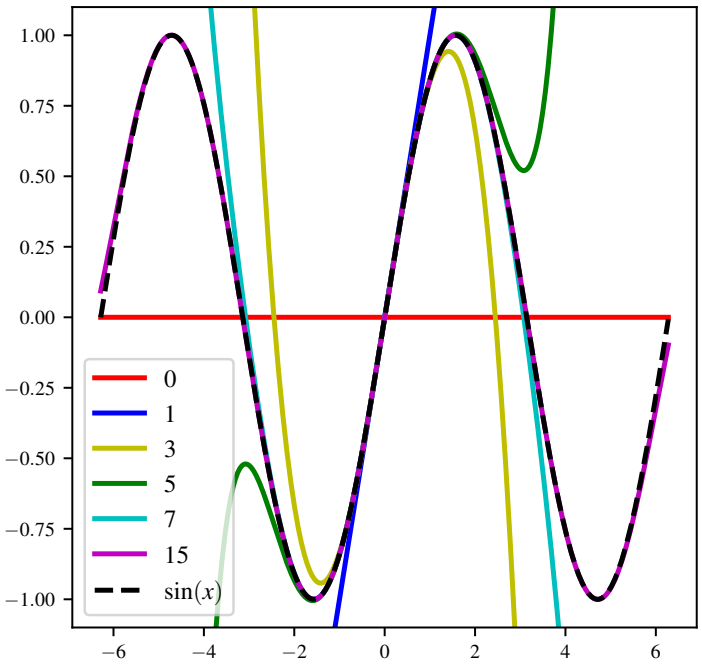


Figure 4.5: Different number of terms in the Taylor series for $\sin(x)$ around point $a = 0$. The highest power in the Taylor series indicate the number of terms, thus $\sin(x) \simeq x - x^3/6$ is said to have 3 terms (however, one of them is zero).

where ξ is a number between x and $x + \Delta x$. Consider the Taylor series

$$f(x + \Delta x) = f(x) + \frac{\Delta x}{1!}\frac{\partial f(x)}{\partial x} + \frac{\Delta x^2}{2!}\frac{\partial^2 f}{\partial x^2}(\xi) \quad . \tag{4.27}$$

If we solve for $\partial f(x)/\partial x$, we obtain

$$\frac{\partial f(x)}{\partial x} = \frac{f(x + \Delta x) - f(x)}{\Delta x} - \frac{\Delta x}{2}\frac{\partial^2 f}{\partial x^2}(\xi) \quad . \tag{4.28}$$

Remember that

$$\frac{f(x + \Delta x) - f(x)}{\Delta x} \quad , \tag{4.29}$$

Forward-difference quotient

is the forward-difference quotient. If $\mathcal{D} = \partial/\partial x$ is the operator for the derivative, while D_+ is the forward-difference operator where

$$D_+(f) = \frac{f(x + \Delta x) - f(x)}{\Delta x} \quad , \tag{4.30}$$

then Eq. (4.28) can be written on operator form as

$$\mathcal{D}(f) = D_+(f) + R_+ \quad , \tag{4.31}$$

where R_+ is the remainder. We see that the remainder $R_+(x)$ is the truncation error in the Taylor series,

$$R_+(x) = -\frac{\Delta x}{2}\frac{\partial^2 f}{\partial x^2}(\xi) \quad . \tag{4.32}$$

As we are numerically solving partial differential equations by finite difference methods, we will call the remainder $R_+(x)$ for a value x the *local discretisation error* at x.

Local discretisation error

From Eq. (4.32) we see that $R_+(x)$ is of first order in Δx. Following the description of the big-oh $\mathcal{O}$ in the mathematical notes in

Sec. 17.5, we then have that $R_+(x) = \mathcal{O}(\Delta x)$, meaning that $R_+(x)$ goes to zero at least as fast as Δx. Replacing the remainder $R_+(x)$ by $\mathcal{O}(\Delta x)$, we see that Eq. (4.28) can be written as

$$\frac{\partial f(x)}{\partial x} = \frac{f(x+\Delta x) - f(x)}{\Delta x} + \mathcal{O}(\Delta x) \quad , \tag{4.33}$$

indicating that the error is of order one in Δx. We say that the forward difference quotient is a first-order approximation of the derivative $\partial f(x)/\partial x$, since the error is of order one.

We will now turn from the forward to the backward approximation. Similarly to the procedure for the forward difference quotient, we can obtain the following Taylor series

$$f(x-\Delta x) = f(x) - \frac{\Delta x}{1!}\frac{\partial f(x)}{\partial x} + \frac{\Delta x^2}{2!}\frac{\partial^2 f}{\partial x^2}(\xi) \quad , \tag{4.34}$$

where ξ now is a number between $x - \Delta x$ and x.

Rearranging Eq. (4.34), we can then approximate the derivative $\partial f/\partial x$ by the backward-difference quotient as

$$\begin{aligned} \frac{\partial f(x)}{\partial x} &= \frac{f(x) - f(x-\Delta x)}{\Delta x} + \frac{\Delta x}{2}\frac{\partial^2 f}{\partial x^2}(\xi) \\ &= \frac{f(x) - f(x-\Delta x)}{\Delta x} + \mathcal{O}(\Delta x) \\ &\simeq \frac{f(x) - f(x-\Delta x)}{\Delta x} \quad . \end{aligned} \tag{4.35}$$

The backward-difference quotient was given as

Backward-difference quotient

$$D_-(f) = \frac{f(x) - f(x-\Delta x)}{\Delta x} \quad . \tag{4.36}$$

In operator form, we then have

$$\mathcal{D}(f) = D_-(f) + R_- \quad , \tag{4.37}$$

where $R_- = \mathcal{O}(\Delta x)$, thus the backward difference quotient is also a first-order approximation of the derivative.

We have now approximated the derivative by either going forward or backward. However, any set of points encapsulating x could be used to approximate the derivative. The last difference quotient we will present is evaluating the function both forward and backward from x. For this end, consider the two Taylor series

$$\begin{aligned} f(x+\Delta x) &= f(x) + \frac{\Delta x}{1!}\frac{\partial f(x)}{\partial x} + \frac{\Delta x^2}{2!}\frac{\partial^2 f(x)}{\partial x^2} + \frac{\Delta x^3}{3!}\frac{\partial^3 f}{\partial x^3}(\xi_f) \\ f(x-\Delta x) &= f(x) - \frac{\Delta x}{1!}\frac{\partial f(x)}{\partial x} + \frac{\Delta x^2}{2!}\frac{\partial^2 f(x)}{\partial x^2} - \frac{\Delta x^3}{3!}\frac{\partial^3 f}{\partial x^3}(\xi_b) \quad , \end{aligned} \tag{4.38}$$

where ξ_f is a number between x and $x + \Delta x$, and ξ_b is a number between $x - \Delta x$ and x (the subscripts indicate f for forward and b for backward). If we subtract the two Taylor series above, we obtain

$$f(x+\Delta x) - f(x-\Delta x) = 2\Delta x\frac{\partial f(x)}{\partial x} + \frac{\Delta x^3}{3!}\frac{\partial^3 f}{\partial x^3}(\xi_f) + \frac{\Delta x^3}{3!}\frac{\partial^3 f}{\partial x^3}(\xi_b) \quad . \tag{4.39}$$

This yields

$$\begin{aligned}\frac{\partial f(x)}{\partial x} &= \frac{f(x+\Delta x)-f(x-\Delta x)}{2\Delta x} - \frac{\Delta x^2}{12}\left(\frac{\partial^3 f}{\partial x^3}(\xi_f) + \frac{\partial^3 f}{\partial x^3}(\xi_b)\right) \\ &= \frac{f(x+\Delta x)-f(x-\Delta x)}{2\Delta x} - \mathcal{O}(\Delta x^2) \\ &\simeq \frac{f(x+\Delta x)-f(x-\Delta x)}{2\Delta x} \quad , \end{aligned} \tag{4.40}$$

where

$$D_0(f) = \frac{f(x+\Delta x)-f(x-\Delta x)}{2\Delta x} \tag{4.41}$$

is called the *centered difference quotient*. We see that $D_0 = 1/2(D_+ + D_-)$. We further observe that the remainder for the centered different quotient is of order two, thus the centered difference quotient is a second-order approximation of the derivative. Thus the centered difference quotient is a higher order approximation than the forward and backward difference quotients. This could indicate that the centered difference is the preferred approximation for the derivative, however, this is not always so – we will see that the preferred approximation depends on the overall problem.

Centered difference quotient

4.2.3 Second-difference quotient

We now want to obtain a difference quotient approximation for the second derivative; a *second-difference quotient*. For this end, consider the forward and backward Taylor series centered around x:

$$\begin{aligned} f(x+\Delta x) &= f(x) + \frac{\Delta x}{1!}\frac{\partial f(x)}{\partial x} + \frac{\Delta x^2}{2!}\frac{\partial^2 f(x)}{\partial x^2} + \frac{\Delta x^3}{3!}\frac{\partial^3 f(x)}{\partial x^3} + \frac{\Delta x^4}{4!}\frac{\partial^4 f}{\partial x^4}(\xi_f) \\ f(x-\Delta x) &= f(x) - \frac{\Delta x}{1!}\frac{\partial f(x)}{\partial x} + \frac{\Delta x^2}{2!}\frac{\partial^2 f(x)}{\partial x^2} - \frac{\Delta x^3}{3!}\frac{\partial^3 f(x)}{\partial x^3} + \frac{\Delta x^4}{4!}\frac{\partial^4 f}{\partial x^4}(\xi_b) \quad . \end{aligned} \tag{4.42}$$

Note that these are the same Taylor series as we used to derive the centered difference quotient (Eq. (4.38)), but to an order four instead of three. If we add the two Taylor series in Eq. (4.42), we obtain

$$f(x+\Delta x) + f(x-\Delta x) = 2f(x) + \Delta x^2 \frac{\partial^2 f(x)}{\partial x^2} + \frac{\Delta x^4}{4!}\left(\frac{\partial^4 f}{\partial x^4}(\xi_f) + \frac{\partial^4 f}{\partial x^4}(\xi_b)\right) \quad . \tag{4.43}$$

Solving for the second derivative, we then get

$$\begin{aligned}\frac{\partial^2 f(x)}{\partial^2 x} &= \frac{f(x+\Delta x)-2f(x)+f(x-\Delta x)}{\Delta x^2} - \frac{\Delta x^2}{4!}\left(\frac{\partial^4 f}{\partial x^4}(\xi_f) + \frac{\partial^4 f}{\partial x^4}(\xi_b)\right) \\ &= \frac{f(x+\Delta x)-2f(x)+f(x-\Delta x)}{\Delta x^2} - \mathcal{O}(\Delta x^2) \\ &\simeq \frac{f(x+\Delta x)-2f(x)+f(x-\Delta x)}{\Delta x^2} \quad , \end{aligned} \tag{4.44}$$

where

$$D_0^2(f) = \frac{f(x+\Delta x)-2f(x)+f(x-\Delta x)}{\Delta x^2} \quad , \tag{4.45}$$

Centered second-difference quotient

is called the *centered second-difference quotient*. Observe that the remainder for the centered second-difference quotient approximation is of order 2 in Δx.

As for the first-difference quotients, there exist several second-difference quotients. However, in this book the centered second-difference quotient is the only one in use, so we will not consider the others.

4.3 Truncation error

As seen from, e.g., the forward-difference operator, Eq. (4.31), when we replace the original operator $\mathcal{D}$ by the difference operator D_+, we get a remainder R_+ representing the error. Thus the differential equation and the corresponding finite difference equation will differ by this remainder. We see that this error is the truncation term from Taylor's theorem, Eq. (4.23). Thus, the differential equation and the corresponding finite difference equation differ by the truncation in the Taylor series. The order of approximation for the different difference quotients introduced in the previous section are given in Tab. 4.1.

Difference quotient	Order of approximation
D_+	1
D_-	1
D_0	2
D_0^2	2

Table 4.1: The order of approximation for the different difference quotients presented in the previous section.

Let us consider the 1D diffusivity equation as given by Eq. (3.54). To simplify notation, we let $\eta = 1$:

$$\frac{\partial^2 p}{\partial x^2} = \frac{\partial p}{\partial t} \quad . \tag{4.46}$$

If we us the centered second-difference quotient D_0^2, Eq. (4.44), for the second derivative, and the first-difference forward-difference quotient D_+, Eq. (4.28), for the time derivative, we obtain the equation

$$\begin{aligned} \frac{p(x+\Delta x,t) - 2p(x,t) + p(x-\Delta x,t)}{\Delta x^2} - \frac{\Delta x^2}{4!}\left(\frac{\partial^4 p}{\partial x^4}(\xi_f) - \frac{\partial^4 p}{\partial x^4}(\xi_b)\right) \\ = \frac{p(x,t+\Delta t) - p(x,t)}{\Delta t} - \frac{\Delta t}{2}\frac{\partial^2 p}{\partial t^2}(\xi_t) \quad . \end{aligned} \tag{4.47}$$

We thus see that if we replace the partial differential equation by the difference quotients as

$$\frac{p(x+\Delta x,t) - 2p(x,t) + p(x-\Delta x,t)}{\Delta x^2} = \frac{p(x,t+\Delta t) - p(x,t)}{\Delta t} \quad , \tag{4.48}$$

we introduce a local discretization error at the point (x,t) of

$$\frac{\Delta x^2}{4!}\left(\frac{\partial^4 p}{\partial x^4}(\xi_f) - \frac{\partial^4 p}{\partial x^4}(\xi_b)\right) - \frac{\Delta t}{2}\frac{\partial^2 p}{\partial t^2}(\xi_t) = \mathcal{O}(\Delta x^2) + \mathcal{O}(\Delta t) \quad . \tag{4.49}$$

This error is a local error for the given point (x,t). For a grid and a set of time steps, we will get a global error being the sum of all such local errors (note that they might cancel out). For a general case it is impractical to calculate all the local errors and sum them up to find the global error.

As the local error is of order $\mathcal{O}(\Delta x^2) + \mathcal{O}(\Delta t)$, also the global error will correlate with $\mathcal{O}(\Delta x^2) + \mathcal{O}(\Delta t)$. Thus, if the grid block sizes and the time steps both go to zero, the error should diminish too. In practice one would investigate the discretization error by conducting grid and time refinements; i.e., one would solve the system for several different grid sizes and time step sizes, and investigate when the solutions converge.

Note that when one is conducting such grid and time refinements, the convergence might not correspond to $\mathcal{O}(\Delta x^2) + \mathcal{O}(\Delta t)$. This is because the step size, e.g., the grid block size Δx, is so large that we might be outside the asymptotic behavior described by the truncation of the Taylor series (such as the case with the function $\log(x+1)$ when $x = 0$ and $\Delta x > 1$).

4.3.1 *Example comparing the difference quotients*

Consider the function $f(x) = x^3$, and assume we want to investigate derivatives of this function at value $x = 1$. For this example we do know the solution, since we know how to take the derivative of this function. We thus have an exact value to compare with, namely $f'(1) = 3$.

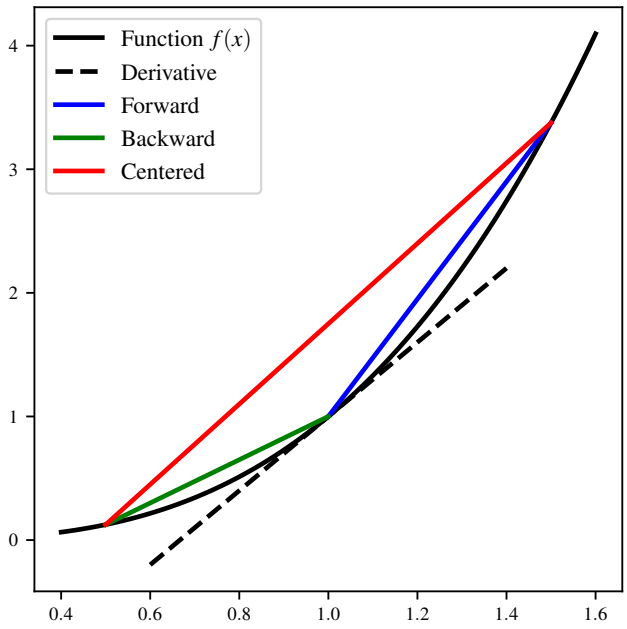


Figure 4.6: Plot of the function $f(x) = x^3$, together with the tangent line at $x = 0$ shown with a dashed line. The three different quotients use a step length of $\Delta x = 0.5$.

We can calculate the forward, backward and centered different quotients for different values of Δx as described in the previous section. For $x = 1$ and $\Delta x = 0.5$ we have illustrated these different quotients in Fig. 4.6. This figure also contains the tangent line at $x = 1$, which we know has a slope of $f'(1) = 3$. We can then compare the three different quotients to the exact value by comparing the three slopes to the slope of the tangent line. From the slopes it is obvious that the centered quotient is closer to the exact derivative given by the dashed line than the forward and backward quotients.

This is as expected, as the centered difference quotient has an error of order $\mathcal{O}(\Delta x^2)$, while the forward and backward difference quotients have errors of order $\mathcal{O}(\Delta x)$.

Δx	Forward	Backward	Centered
0.5	4.75	1.75	3.25
0.1	3.31	2.71	3.01
0.05	3.1525	2.8525	3.0025
0.01	3.0301	2.9701	3.0001
0.005	3.015025	2.985025	3.000025
0.001	3.003001	2.997001	3.000001

Table 4.2: Approximation to $f(x) = x^3$ at $x = 1.0$ for different methods and different step lengths Δx.

The derivative approximations for a set of different Δx values are shown in Tab. 4.2. The convergence of these three different quotients are plotted in Fig. 4.7. It is clear from both the table and the plot that the centered difference quotient converges much faster to the analytical value of 3.0.

Let us also consider the error, given as the absolute value of the difference between the analytical value and the difference quotient value $E(\Delta x) = \|D(f) - \mathcal{D}(f)\| = \|R\|$ (note the difference between the capital D representing the finite difference and the curly $\mathcal{D}$ representing the derivative). From the derivations above we see that the error $E(\Delta x) \propto \Delta x$ for the forward and backward methods, while the error is $E(\Delta x) \propto (\Delta x)^2$ for the centered method. Taking the logarithm on both sides of the proportionality symbol, we see that $\log(E) \propto \log(\Delta x)$ for the forward and backward methods. This should give a slope of 1 when plotting the error versus Δx on a log-log-plot. We also see that $\log(E) \propto \log(\Delta x^2) = 2\log(\Delta x)$ for the centered difference, which will give a slope of 2 when plotting the error versus Δx on a log-log-plot. In the right figure in Fig. 4.7 we can clearly see that the forward and backward methods has a slope of 1, while the centered difference has a slope of 2, as expected.

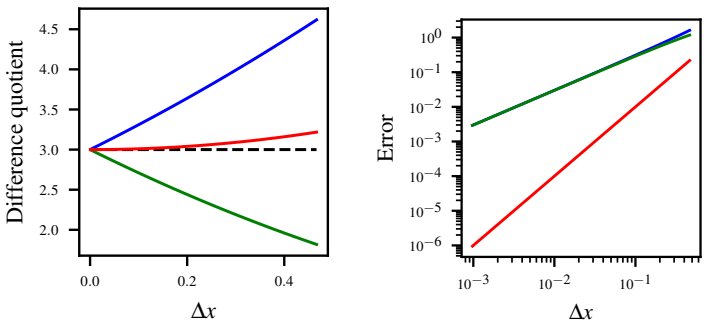


Figure 4.7: Plots showing how the different types of difference quotients converge towards the exact solution. The left plot shows the absolute values, while the right plot is a logarithmic plot of the error values. The color coding for the lines is the same as in Fig. 4.6, with blue for forward, green for backward, and red for centered difference quotient.

Note that the error for the forward- and backward-difference quotient curves in Fig. 4.7 are slightly different, however they seem to converge for small values of Δx. The difference is due to the difference in truncation error $R_+ \neq R_-$. As $\xi_f \in [x, x + \Delta x]$ and $\xi_b \in [x - \Delta x, x]$, we have $\|\xi_f - \xi_b\| < 2\Delta x$. So when $\Delta x \to 0$, then

$\|\xi_f - \xi_b\| < 2\Delta x \to 0$, and further $\|\partial f/\partial x(\xi_f) - \partial f/\partial x(\xi_b)\| \to 0$ for a smooth function f. Since

$$\|R_+ - R_-\| = \left\| \frac{\Delta x}{2} \left(\frac{\partial f}{\partial x}(\xi_f) - \frac{\partial f}{\partial x}(\xi_b) \right) \right\| = \frac{\Delta x}{2} \left\| \frac{\partial f}{\partial x}(\xi_f) - \frac{\partial f}{\partial x}(\xi_b) \right\| \quad , \tag{4.50}$$

we see that the difference between the forward and backward error converge to zero faster than the errors themselves (of order $\mathcal{O}(\Delta x)$) converge to zero. Thus the errors as a function of Δx converge for the forward- and backward-difference quotients. This is clearly observed in Fig. 4.7.

4.4 *Discretization*

In numerical methods we use finite differences to approximate our functions at discrete points in the solution space. The diffusivity equation has both a spatial and a time dimension, and the solution space for the one-dimensional diffusivity equation is indicated in Fig. 3.7. To pave the ground for a numerical solution, which we will return to in the next chapter, we need to discretize the (x, t)-space to obtain our finite differences. The simplest discretization is to divide the space into grid points with a constant separation in each direction, i.e., use constant step sizes Δt and Δx. Such a discretization is indicated in Fig. 4.8.

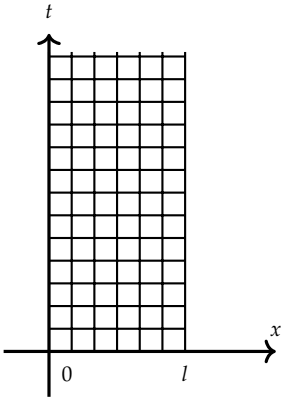


Figure 4.8: This figure indicates a possible set of discrete points in the (x, t) space than can be used for the numerical approximation of the partial differential equation, e.g. Eq. (3.54).

The discretization of the spatial coordinates needs to conform to the underlying geometry and properties of the structure we are working with. We will return to this spatial discretization below. For the time discretization we have more freedom, as we only need to conform to discretizing the space $[t_0, \infty]$. The simplest discretization is to keep constant time steps of size Δt. It is however normal to adapt time step sizes to ensure convergence.

For our 1-dimensional case the spatial x-coordinate must be subdivided into a number of discrete distances. Then, the pressure in each block can be solved for numerically for each time step. For our simple one dimensional slab visualized in Fig. 3.6 we assumed constant properties for the material (which gives a constant hydraulic diffusivity constant η). For such a case we can divide the x-direction into n grid blocks of equal size Δx, as indicated in Fig. 4.9.

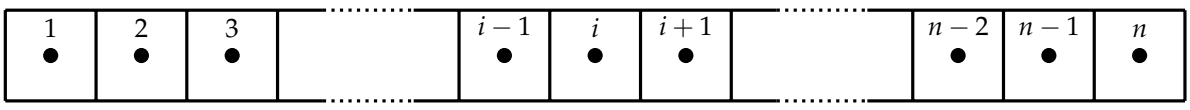


Figure 4.9: Figure indicating the grid cells for the pseudo-1D slab.

The grid blocks are assigned indices, i, referring to an effective point of each block representing the average properties of this block: The effective geological properties of the grid cell includes porosity and permeability, while state variables include pressure (and later also saturation). Grid boundaries are associated with properties such as flow rates.

If our properties, and thereby η, is varying, our discretization would need to capture this property variation, which could preclude having a constant grid block size. We will return to different types of grids in Chap. 7, however, we will introduce two main types of grid systems used for finite-difference schemes already here. Following Settari and Aziz (1972) these grid types will be referred to as block-centered and point-distributed.

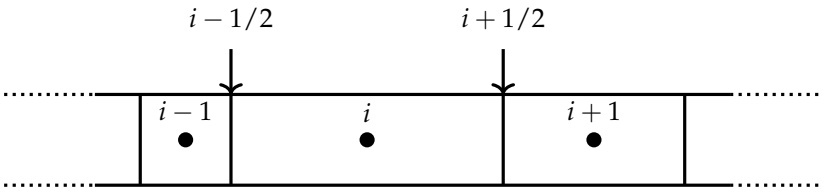


Figure 4.10: Figure indicating the grid cell boundaries and centers for a block-centered grid with varying grid sizes.

In a *block-centered grid*, the location of the grid boundaries $\{x_{i+1/2}\}$ are defined while the grid cell centers are calculated. The grid cell centers are calculated as

Block-centered grid

$$x_i = \frac{x_{i-1/2} + x_{i+1/2}}{2} \quad . \tag{4.51}$$

For block-centered grids, the grid cell centers are all midway in the grid cell, thereby the name. An example of a block-centered grid is shown in Fig. 4.10.

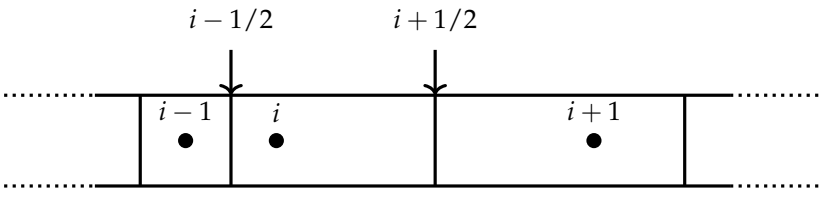


Figure 4.11: Figure indicating the grid cell boundaries and centers for a point-distributed grid with varying length between the cell centers.

In a *point-distributed grid* the location of all the grid centers are defined. The grid boundaries are calculated as

Point-distributed grid

$$x_{i+1/2} = \frac{x_i + x_{i+1}}{2} \quad . \tag{4.52}$$

For point-distributed grids the grid cell centers are not necessarily in the middle of the grid cells, as indicated in Fig. 4.11. Note that for a grid with equal size cells, the block-centered and point-distributed grids are equal, e.g., the grid depicted in Fig. 4.9.

The block-centered grids are more easily adapted to the distribution of reservoir properties, e.g., rock properties such as porosity and permeability from a geo-model, as we can start by assigning the cell boundaries at the boundaries between different rock properties. It might also look more natural to have the grid cell properties evaluated at the center of the grid cell. As it is more complicated to represent irregular grids with the point-distributed grid model, you might need a higher number of grid-cells to be able to represent the same underlying geometry. Thus the point-distributed model is less

adaptable to represent a given geo-model. However, it turns out that the point-distributed grids give a consistent finite-difference operator for situations where the grid sizes vary. It might therefore give convergence with fewer grid cells. For this reason we will mostly consider point-distributed grids in this book, despite the fact that they are less versatile for representing the distribution of properties.

A comparison of the truncation error for the block-centered and point-distributed grids can be found in Chapter 3.5 in Aziz and Settari (1979).

Let us return to the general form of the diffusivity equation, Eq. (3.36), where we allow for heterogeneous permeability and varying density and viscosity. Assume that the density and viscosity is kept constant, while the permeability might be variable, then we need to solve the partial derivative

$$\frac{\partial}{\partial x}\left(a(x)\frac{\partial}{\partial x}f(x)\right) \quad , \tag{4.53}$$

where $a(x) = k(x)$ gives the permeability and $f(x) = p(x)$ gives the pressure. We can also generalize the grid, allowing for the Δx to be different at the two sides, say $\Delta x_l \neq \Delta x_r$ (as is the case for the point-distributed grid). Such a situation is illustrated in Fig. 4.12.

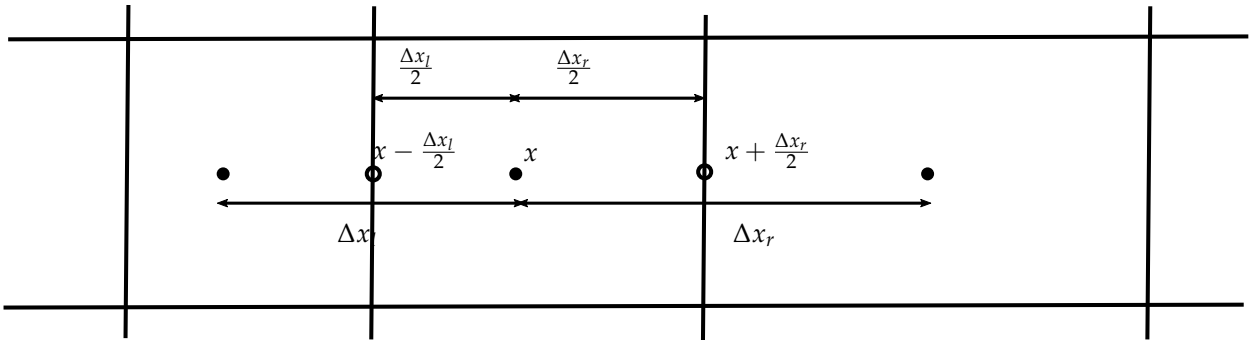


Figure 4.12: Figure indicating the grid cells for a point-distributed grid with varying grid block sizes.

Using the centered difference quotient, as given by Eq. (4.40), with step size $\Delta x_l/2$ and $\Delta x_r/2$, for the derivative $\partial f/\partial x$, we approximate the partial differentials as

$$\begin{aligned} a\frac{\partial f}{\partial x}\left(x - \frac{\Delta x_l}{2}\right) &\simeq a\left(x - \frac{\Delta x_l}{2}\right)\frac{f(x) - f(x - \Delta x_l)}{\Delta x_l} \\ a\frac{\partial f}{\partial x}\left(x + \frac{\Delta x_r}{2}\right) &\simeq a\left(x + \frac{\Delta x_r}{2}\right)\frac{f(x + \Delta x_r) - f(x)}{\Delta x_r} \quad . \end{aligned} \tag{4.54}$$

We then express the second-difference quotient as

Second-difference quotient for varying block sizes.

$$\begin{aligned} \frac{\partial}{\partial x}\left(a(x)\frac{\partial}{\partial x}f(x)\right) &\simeq \frac{a\frac{\partial f}{\partial x}\left(x + \frac{\Delta x_r}{2}\right) - a\frac{\partial f}{\partial x}\left(x - \frac{\Delta x_l}{2}\right)}{\left(x + \frac{\Delta x_r}{2}\right) - \left(x - \frac{\Delta x_l}{2}\right)} \\ &\simeq \frac{a\left(x + \frac{\Delta x_r}{2}\right)\frac{f(x+\Delta x_r) - f(x)}{\Delta x_r} - a\left(x - \frac{\Delta x_l}{2}\right)\frac{f(x) - f(x-\Delta x_l)}{\Delta x_l}}{\frac{\Delta x_r + \Delta x_l}{2}} \quad . \end{aligned} \tag{4.55}$$

One can show that this approximation is of order $\mathcal{O}(\Delta x^2)$, where $\Delta x = \max(\Delta x_l, \Delta x_r)$.

Assuming a constant grid cell size, Δx, we get the following expression for the second-difference quotient

Second-difference quotient for constant block sizes, but varying properties.

$$\frac{\partial}{\partial x}\left(a(x)\frac{\partial}{\partial x}f(x)\right) \simeq \frac{a\left(x+\frac{\Delta x}{2}\right)\left(f(x+\Delta x)-f(x)\right)-a\left(x-\frac{\Delta x}{2}\right)\left(f(x)-f(x-\Delta x)\right)}{(\Delta x)^2} \,. \tag{4.56}$$

4.5 Boundary conditions

In a reservoir simulation model the most important boundaries consist of two types: wells and the outer boundaries of the reservoir. We need to specify the boundary conditions for both of these. Note that we might have additional boundaries, e.g., faults, aquifers etc., which might have more complex boundary conditions.

Basically, we have two types of boundary conditions; Dirichlet boundary conditions specify the value of the function at the boundary of our domain, while Neumann boundary conditions specify the value of the derivative (or gradient) at the boundary. As we are trying to solve for the pressure, a Dirichlet boundary condition then means to specify the pressure at the boundary, while a Neumann boundary condition would then be to specify the flow rate at the boundary (as the Darcy equation equates the pressure gradient with flow rate). Additionally, we could have a mix of these two boundary conditions (Robin boundary condition).

4.5.1 Pressure boundary conditions

Let Γ be the boundary of our porous medium (reservoir) Ω. With pressure boundary conditions, which are Dirichlet boundary conditions, pressure is specified at the boundaries, thus $p(\vec{x})$ must be specified for all $\vec{x} \in \Gamma$.

Our 1-dimensional system considered in the previous chapter had Dirichlet boundary conditions given by a constant pressure at each end of the system, namely $p(0,t) = p_l$ and $p(l,t) = p_r$.

Figure 4.13: Figure indicating the pressure points at the boundaries of the pseudo-1D slab.

For the discretized 1-D slab considered before, we define the index of the left side as $1/2$, and the index of the right side as $n + 1/2$, as indicated in Fig. 4.13. The reason for the indices $1/2$ and $n + 1/2$ is that the boundary conditions are applied to the ends of the first and the last blocks, respectively, not internally in the block. Thus, the boundary conditions cannot directly be substituted into the difference equation. However, we can use Taylor series to

derive equations for the end blocks. For the first block we obtain a forward difference as

$$\begin{aligned} p_2 &= p(x_2) = p(x_1 + \Delta x) \\ &= p(x_1) + \frac{\Delta x}{1!}\frac{\partial p}{\partial x}(x_1) + \frac{(\Delta x)^2}{2!}\frac{\partial^2 p}{\partial x^2}(x_1) + \frac{\Delta x^3}{3!}\frac{\partial^3 p}{\partial x^3}(\xi_2) \quad . \end{aligned} \tag{4.57}$$

Using the distance $\Delta x/2$ we get a backward difference as

$$\begin{aligned} p_l &= p(x_{1/2}) = p\left(x_1 - \frac{\Delta x}{2}\right) \\ &= p(x_1) + \frac{(-\frac{\Delta x}{2})}{1!}\frac{\partial p}{\partial x}(x_1) + \frac{(-\frac{\Delta x}{2})^2}{2!}\frac{\partial^2 p}{\partial x^2}(x_1) + \frac{(-\frac{\Delta x}{2})^3}{3!}\frac{\partial^3 p}{\partial x^3}(\xi_{1/2}) \quad . \end{aligned} \tag{4.58}$$

By adding two times Eq. (4.58) to Eq. (4.57), we obtain a second-difference quotient as:

Second-difference quotient for the left Dirichlet boundary condition

$$\frac{\partial^2 p}{\partial x^2}(x_1) = \frac{p_2 - 3p_1 + 2p_l}{\frac{3}{4}(\Delta x)^2} + \mathcal{O}(\Delta x) \quad . \tag{4.59}$$

A disadvantage of this formulation is that the residual term is only of first order, i.e. proportional to Δx.

Similarly, the following expression can be obtained for the right hand side boundary condition:

Second-difference quotient for the right Dirichlet boundary condition

$$\frac{\partial^2 p}{\partial x^2}(x_n) = \frac{2p_r - 3p_n + p_{n-1}}{\frac{3}{4}(\Delta x)^2} + \mathcal{O}(\Delta x) \quad . \tag{4.60}$$

4.5.2 Flow rate boundary condition

Flow rate boundary conditions, which are Neumann boundary conditions, is to specify the flux over the boundary Γ of the porous medium. Thus $\vec{q}(\vec{x}) \cdot \vec{n} = g(\vec{x})$ for a function g defined for all points $\vec{x} \in \Gamma$. In practice, we specify the flow rate q into or out of an end face of the grid system. For a grid where the x-direction is perpendicular to the boundary, one can define a Neumann boundary condition as $\partial p/\partial x(x) = g(x)$. A no-flow boundary condition amount to setting the flow rate to zero at the boundary. The no-flow boundary condition is common at outer boundaries of the reservoir, between non-communicating layers, and across sealing faults in the reservoir.

For our one-dimensional system considered before, this would amount to specify the flux $q(0,t) = q_l$ and $q(l,t) = q_r$ at the two ends. From Darcy's law we can express the Darcy velocity as:

$$q_l = -\frac{k}{\mu}\frac{\partial p}{\partial x}(x_{1/2}) \quad , \tag{4.61}$$

where we use the simplified notation $x_{1/2} = x_1 - \Delta x/2$.

As we did for the pressure boundary condition, we will again apply a Taylor series expansion around x_1, but this time we will let

the derivative of the pressure be the function:

$$\frac{\partial p}{\partial x}\left(x_1+\frac{\Delta x}{2}\right)=\frac{\partial p}{\partial x}(x_1)+\frac{(\frac{\Delta x}{2})}{1!}\frac{\partial^2 p}{\partial x^2}(x_1)+\frac{(\frac{\Delta x}{2})^2}{2!}\frac{\partial^3 p}{\partial x^3}(\xi_2)$$
$$\frac{\partial p}{\partial x}(x_{1/2})=\frac{\partial p}{\partial x}\left(x_1-\frac{\Delta x}{2}\right)=\frac{\partial p}{\partial x}(x_1)+\frac{(-\frac{\Delta x}{2})}{1!}\frac{\partial^2 p}{\partial x^2}(x_1)+\frac{(-\frac{\Delta x}{2})^2}{2!}\frac{\partial^3 p}{\partial x^3}(\xi_l) \tag{4.62}$$

Subtracting the second expression from the first, then solving for the second derivative, we obtain the following approximation:

$$\frac{\partial^2 p}{\partial x^2}(x_1)=\frac{\frac{\partial p}{\partial x}(x_1+\frac{\Delta x}{2})-\frac{\partial p}{\partial x}(x_{1/2})}{\Delta x}+\mathcal{O}(\Delta x) \tag{4.63}$$

We now want to replace the pressure derivatives. For the first pressure derivative we apply a central difference quotient:

$$\frac{\partial p}{\partial x}\left(x_1+\frac{\Delta x}{2}\right)=\frac{p(x_2)-p(x_1)}{\Delta x}+\mathcal{O}(\Delta x^2) \quad . \tag{4.64}$$

The second pressure derivative can be replaced using Eq. (4.61), yielding the following expression for the left Neumann boundary condition:

Second-difference quotient for the left Neumann boundary condition

$$\frac{\partial^2 p}{\partial x^2}(x_1)=\frac{p_2-p_1}{(\Delta x)^2}+q_l\frac{\mu}{\Delta x k}+\mathcal{O}(\Delta x) \quad . \tag{4.65}$$

Similarly, a constant rate at the right hand side, q_r , would yield the following expression:

Second-difference quotient for the right Neumann boundary condition

$$\frac{\partial^2 p}{\partial x^2}(x_n)=\frac{p_n-p_{n-1}}{(\Delta x)^2}+q_r\frac{\mu}{\Delta x k}+\mathcal{O}(\Delta x) \quad . \tag{4.66}$$

As for the Dirichlet boundary condition, also the Neumann boundary boundary condition gives a local discretisation error of order one, in contrast to the centered second-difference quotient of order two.

4.5.3 Mixed boundary condition

For reservoir models it is common to have both types of boundary conditions; Dirichlet and Neumann. At the outer boundaries of the reservoir it is common with a no-flow boundary condition (a Neumann boundary condition), while the wells can be either pressure controlled (a Dirichlet boundary condition), which is normal for producers, or the wells can be rate controlled (a Neumann boundary condition), which is normal for injectors.

It is also possible to have a mix of the two boundary conditions, in mathematics called *Robin boundary conditions*;

$$\left(a\nabla p\cdot\vec{n}+bp\right)(\vec{x})=g(\vec{x}) \quad , \tag{4.67}$$

where $\vec{x}\in\Gamma$ is on the boundary. Such mixed boundary conditions do occur in reservoir simulation models, e.g., at leaking faults.

4.6 Exercises

Exercise 4.1 Consider the function $f(x) = x^2$ at point $x = 2$.

a) Calculate the forward, backward and centered difference quotient for $f(x)$ at $x = 2.0$ for different step lengths Δx. Plot how the difference quotients converge towards the analytical solution.

b) Calculate the centered second-difference quotient for $f(x)$ at $x = 2.0$ for different step lengths Δx.

Exercise 4.2 Consider the function $f(x) = \sin(x)$ at point $x = 1.0$.

a) Calculate the forward, backward and centered difference quotient for $f(x)$ at $x = 1.0$ for different step lengths Δx.

b) Plot the error $E = \|D(f)(x) - f'(x)\|$ for the different difference quotients for $x = 1.0$, e.g. $E_+ = \|D_+(\sin(x))(1.0) - \cos(1.0)\|$ for the forward difference. Try to fit the error with a polynomial. What order is the fitted polynomial for the different methods?

c) Repeat the tasks above for the difference quotient given by

$$D_3(f) = \frac{1}{6\Delta x}\left[2f(x+\Delta x) + 3f(x) - 6f(x-\Delta x) + f(x-2\Delta x)\right] \quad .$$

What order of accuracy does the difference quotient D_3 give?

5
Numerical methods for single-phase flow

> Do not worry about your difficulties in mathematics, I can assure you mine are still greater
>
> Albert Einstein

In this chapter we will return to the 1-dimensional single phase case considered in Chap. 3. The diffusivity equation, Eq. (3.52), which was solved analytically in Chap. 3, will be solved numerically in this chapter. We will use finite difference approximations for the two derivative terms $\partial^2 p/\partial x^2$ and $\partial p/\partial t$, as presented in the previous chapter.

5.1 Finite difference approximations

In this section we will conduct a finite difference approximations by using the difference quotients introduced in the previous chapter to the second order spacial derivative $\partial^2 p/\partial x^2$ and the first order time derivative $\partial p/\partial t$.

In the previous chapters we used lower case letters for the pressure field, p. In the following we will use upper case letter P to denote the pressure variables at the different points in a numerical solution. Thus lower case p is the actual solution, while upper case P is the numerical approximation. Further, we will use an *index system* where we indicate the spatial grid index by a subscript, and the time level by a superscript. Thus P_i^t is the numerical pressure in the spatial point represented by the grid cell with index i and at time level t. Thus, if we have constant grid size and time steps, then $P_i^t = P(i\Delta x, t\Delta t)$ when Δx is the spatial grid size and Δt is a constant time step. Note that we do not necessarily have constant grid cell size and constant time steps, we will still index cells and time steps by indices. Note that we are a bit sloppy with the time step notation, sometimes t represents the time step index, while other times it represents the actual time. In the latter case, $P_i^t = P(i\Delta x, t)$.

Lower case denote the actual solution, while upper case denote the numerical approximation.

Index system

For a three-dimensional system we have $P_{i,j,k}^t = P(i\Delta x, j\Delta y, k\Delta z, t\Delta t)$, where Δx, Δy and Δz are constant grid cell sizes in the three spatial directions.

Consider the 1D diffusivity equation as given by Eq. (3.54):

$$\eta \frac{\partial^2 p}{\partial x^2} = \frac{\partial p}{\partial t} \quad . \tag{5.1}$$

We will now consider different difference quotients for the derivatives in this equation.

Using the centered second-difference quotient, Eq. (4.44), for the second derivative, we have

$$\frac{\partial^2 p}{\partial x^2}(x,t) \simeq \frac{p(x+\Delta x,t) - 2p(x,t) + p(x-\Delta x,t)}{\Delta x^2} \quad , \tag{5.2}$$

where the truncation error is proportional to $(\Delta x)^2$, thus of order $\mathcal{O}((\Delta x)^2)$. For the numerical solution, employing the index system where subscript is grid index number and superscript is time level, we get:

Second-difference quotient in space

$$\left(\frac{\partial^2 P}{\partial x^2}\right)_i^t = \frac{P_{i+1}^t - 2P_i^t + P_{i-1}^t}{\Delta x^2} \tag{5.3}$$

Any time level could be used in the equation above; at time levels $t + \Delta t$ and $t + \frac{\Delta t}{2}$ we get the following:

$$\left(\frac{\partial^2 P}{\partial x^2}\right)_i^{t+\Delta t} = \frac{P_{i+1}^{t+\Delta t} - 2P_i^{t+\Delta t} + P_{i-1}^{t+\Delta t}}{\Delta x^2} \tag{5.4}$$

$$\left(\frac{\partial^2 P}{\partial x^2}\right)_i^{t+\frac{\Delta t}{2}} = \frac{P_{i+1}^{t+\frac{\Delta t}{2}} - 2P_i^{t+\frac{\Delta t}{2}} + P_{i-1}^{t+\frac{\Delta t}{2}}}{\Delta x^2} \tag{5.5}$$

These shifted time levels will be used for different solution techniques and to represent the boundary conditions.

At a constant position, x, we can use the forward difference quotient Eq. (4.29) for the time to get the approximation:

$$\frac{\partial p}{\partial t}(x,t) \simeq \frac{p(x,t+\Delta t) - p(x,t)}{\Delta t} \quad , \tag{5.6}$$

where the error term is proportional to Δt, i.e., of order $\mathcal{O}(\Delta t)$. Employing the index system, we get:

Forward difference quotient in time

$$\left(\frac{\partial P}{\partial t}\right)_i^t = \frac{P_i^{t+\Delta t} - P_i^t}{\Delta t} \quad . \tag{5.7}$$

Similarly, expanding backward in time as given by Eq. (4.36), we get:

$$\left(\frac{\partial P}{\partial t}\right)_i^t = \frac{P_i^t - P_i^{t-\Delta t}}{\Delta t} \quad . \tag{5.8}$$

Shifting the time step, when then have:

Backward difference quotient in time

$$\left(\frac{\partial P}{\partial t}\right)_i^{t+\Delta t} = \frac{P_i^{t+\Delta t} - P_i^t}{\Delta t} \quad . \tag{5.9}$$

Observe that the right hand side of the backward difference quotient shifted in time as given by Eq. (5.9) is identical to the right hand side of the forward difference quotient given by Eq. (5.7).

Finally, we can also consider the centered difference quotient given by Eq. (4.40), where the step side is $\Delta t/2$:

$$\frac{\partial p}{\partial t}(x,t) \simeq \frac{p(x,t+\frac{\Delta t}{2}) - p(x,t-\frac{\Delta t}{2})}{\Delta t} \quad . \tag{5.10}$$

Shifting the time step by $\Delta t/2$, we obtain:

$$\frac{\partial p}{\partial t}(x, t + \frac{\Delta t}{2}) \simeq \frac{p(x, t + \Delta t) - p(x, t)}{\Delta t} \quad . \tag{5.11}$$

With the index system we then get:

Centered difference quotient in time

$$\left(\frac{\partial P}{\partial t}\right)_i^{t+\frac{\Delta t}{2}} = \frac{P_i^{t+\Delta t} - P_i^t}{\Delta t} \tag{5.12}$$

Again, we see that the right hand side is equal to Eq. (5.7) and Eq. (5.9). Thus all the three different difference quotients, forward, backward and centered, yields the same right hand side after an appropriate shift to the time level where the derivative is evaluated.

All of these derived equations will be used in the following.

5.2 Numerical solutions

In this section we will derive a discretized reformulation of the diffusivity equation, together with appropriate boundary conditions and initial conditions.

The 1D diffusivity equation, as given by Eq. (3.54);

$$\eta \frac{\partial^2 p}{\partial x^2} = \frac{\partial p}{\partial t} \quad , \tag{5.13}$$

can be reformulated into our numerical index system as

$$\eta \left(\frac{\partial^2 P}{\partial x^2}\right)_i^t = \left(\frac{\partial P}{\partial t}\right)_i^t \quad . \tag{5.14}$$

With Dirichlet (pressure) boundary conditions, using our index system, we have

$$\begin{aligned} P_{i=1/2}^t &= p_l \\ P_{i=n+1/2}^t &= p_r \end{aligned} \quad , \tag{5.15}$$

for all time steps t. The reason we here use indices $i = 1/2$ and $n + 1/2$ is that the boundary conditions are applied to the ends of the first and the last grid blocks, respectively. Transferring the left side Dirichlet boundary condition, Eq. (4.59), to our index system, we obtain

$$\left(\frac{\partial^2 P}{\partial x^2}\right)_1^t = \frac{P_2^t - 3P_1^t + 2p_l}{\frac{3}{4}(\Delta x)^2} \quad . \tag{5.16}$$

Similarly, we transfer the right side Dirichlet boundary condition, Eq. (4.60), to our index system as:

$$\left(\frac{\partial^2 P}{\partial x^2}\right)_n^t = \frac{2p_r - 3P_n^t + P_{n-1}^t}{\frac{3}{4}(\Delta x)^2} \quad . \tag{5.17}$$

We can obtain similar equations for Neumann boundary conditions.

If the system is initialized with a pressure equal the right side boundary condition pressure, then initial condition (initial pressures) for our 1D system may be specified as:

$$P_i^0 = p_r \quad , \tag{5.18}$$

for all grid indices i. For non-horizontal systems, the system is usually initialized by calculating the hydro-static pressures from a reference pressure at a given depth and fluid densities.

Having derived numerical representations for the differential equations, and having initial and boundary conditions, we can solve for the pressures. In the equations above we used the time t for the difference quotients. By substitution, we can as well assign a time level of $t + \Delta t$ or $t + \frac{\Delta t}{2}$ with equal generality. In the following we will see that using different time levels will lead to different numerical formulations. We will start with using the time level t, which is giving an explicit formulation.

5.2.1 *Explicit formulation*

In this subsection we will obtain a set of difference equation that can be solved explicitly for the pressures. For this end, consider Eq. (5.14):

$$\eta \left(\frac{\partial^2 P}{\partial x^2} \right)_i^t = \left(\frac{\partial P}{\partial t} \right)_i^t \quad . \tag{5.19}$$

For the second order spatial derivative on the left side we will use the centered second-difference quotient as given by Eq. (4.44), formulated with our numerical index system as Eq. (5.3). The forward difference quotient given by Eq. (4.29) for the time derivative is given in our numerical index system by Eq. (5.7). Combining Eq. (5.3) and Eq. (5.7) we get

$$\eta \frac{P_{i+1}^t - 2P_i^t + P_{i-1}^t}{\Delta x^2} = \frac{P_i^{t+\Delta t} - P_i^t}{\Delta t} \quad . \tag{5.20}$$

Note that the errors involved in this numerical form of the diffusivity equation are of order $\mathcal{O}(\Delta t)$ and $\mathcal{O}(\Delta x^2)$. Also note that there is only one pressure at time $t + \Delta t$, namely $P_i^{t+\Delta t}$. Solving for this pressure $P_i^{t+\Delta t}$, we get an explicit formulation for the pressure as:

$$P_i^{t+\Delta t} = P_i^t + \left(\frac{\Delta t}{\Delta x^2} \right) \eta \left(P_{i+1}^t - 2P_i^t + P_{i-1}^t \right) \quad . \tag{5.21}$$

This equation holds for all the internal grid cells.

With the boundary conditions given by Eq. (5.16) and Eq. (5.17), we then have the following system of equations:

$$\begin{aligned}
P_1^{t+\Delta t} &= P_1^t + \frac{4}{3} \left(\frac{\Delta t}{\Delta x^2} \right) \eta (P_2^t - 3P_1^t + 2p_l) \\
P_i^{t+\Delta t} &= P_i^t + \left(\frac{\Delta t}{\Delta x^2} \right) \eta (P_{i+1}^t - 2P_i^t + P_{i-1}^t) \quad \forall i \in \{2, \ldots, n-1\} \\
P_n^{t+\Delta t} &= P_n^t + \frac{4}{3} \left(\frac{\Delta t}{\Delta x^2} \right) \eta (2p_r - 3P_n^t + P_{n-1}^t) \quad .
\end{aligned} \tag{5.22}$$

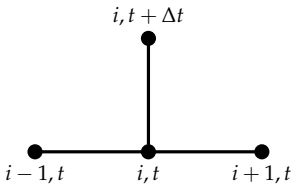


Figure 5.1: The stencil for the explicit method as given by Eq. (5.20), indicating the input and output data for a single time step. As seen from Eq. (5.21), you need the pressures at points $i-1, i$ and $i+1$ at time-step t to calculate the pressure of point i at time-step $t + \Delta t$.

To simplify we can define $\alpha = \eta \Delta t / \Delta x^2$. We see that given the pressures at a given time step, then Eqs. (5.22) gives us the pressures at the next time step. Starting with the pressures given by the initial condition, we can then calculate pressures at all subsequent time steps using Eqs. (5.22) iteratively.

A single time step can be implemented in Python as:

```
def explicitTimeStep(afPressure,fAlpha,fLeftPressure,
    ↪ fRightPressure):
  afPressureNextTimestep=afPressure[:]
  afPressureNextTimestep[0]=afPressure[0]+(4.0/3.0)*fAlpha*(
    ↪ afPressure[1]-3.0*afPressure[0]+2.0*fLeftPressure)
  afPressureNextTimestep[-1]=afPressure[-1]+(4.0/3.0)*fAlpha*(2.0*
    ↪ fRightPressure-3.0*afPressure[-1]+afPressure[-2])
  afPressureNextTimestep[1:-1]=afPressure[1:-1]+fAlpha*(afPressure
    ↪ [2:]-2.0*afPressure[1:-1]+afPressure[:-2])
  return afPressureNextTimestep
```

Here `afPressure` is the vector (array) of pressures $\vec{P} = [P_1, P_2, \ldots, P_n]^T$. The precursor `af` indicates that this is an array of floats (decimal-numbers). Similarly `falpha` is the float value α. Finally, `fLeftPressure` and `fRightPressure` are the float values of the boundary conditions p_l and p_r, respectively.

The T indicates that we take the transpose of the vector, i.e.,

$$[P_1, P_2, \ldots, P_n]^T = \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_n \end{bmatrix} .$$

We use the same constants as before:

```
# Initialization of parameters
iNumberCells=100 # number of grid cells
fDeltat=1.0E-5 # time step size [seconds]
fEta=1.0 # hydraulic diffusivity [meter squared per second]
fModelLength=1.0 # meters
fInitialPressure=1.0 # pascal
fLeftPressure=2.0 # pascal
fRightPressure=1.0 # pascal

fDeltax=fModelLength/iNumberCells
fAlpha=fEta*fDeltat/fDeltax**2
afPressure=np.zeros(iNumberCells)
afPressure[:]=fInitialPressure
```

The last lines shows that to initialize the model, we define the pressure field (at time step zero) to be equal the right side pressure.

We now iterate over 30000 time steps, all of length $\Delta t = 1.0 \times 10^{-5}$ s as given by the float variable `fDeltat` above:

```
#Time loop
fTime=0.0
iNumberLoops=30000
fMaxtime=fDeltat*iNumberLoops
while(fTime<fMaxtime):
  # Increase the time by one time-step
  fTime+=fDeltat
  # Calculate the pressures at the next time step from
  # the current pressures
  afPressure=explicitTimeStep(afPressure,fAlpha,fLeftPressure,
    ↪ fRightPressure)
```

Plotting the pressure at each thousandth time-step gives the plot shown in Fig. 5.2. We see again that the pressure converges

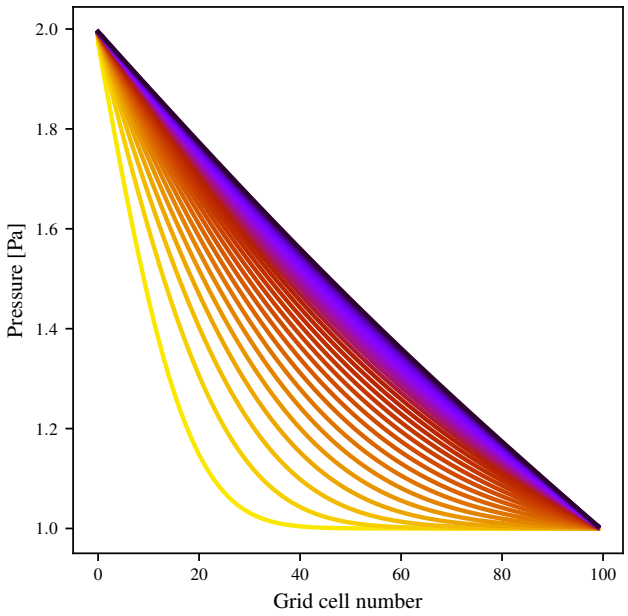


Figure 5.2: Solution to the explicit calculation of pressure p. We have plotted one line for each 1000th time-step.

to the steady-state solution, given by the straight line between the pressures at the end of the 1D system.

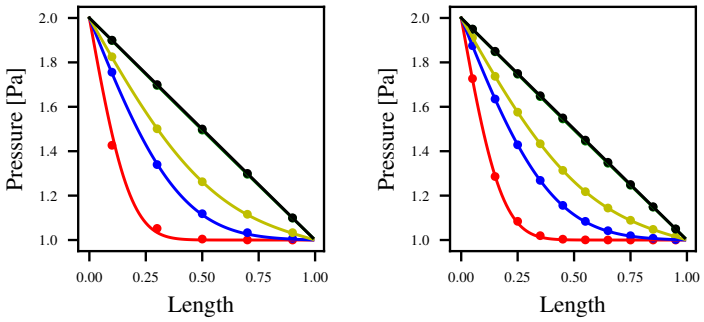


Figure 5.3: Solution to the explicit calculation of pressure P compared to the analytical solution p. The left plot shows the explicit solution when using 5 grid cells, while the right plot use 10 grid cells.

To test how the solution behaves at larger gird block sizes, we calculated the pressure using significantly fewer grid blocks, namely 5 and 10. The results are shown in Fig. 5.3, where the points represents the grid cell values, while the lines represent the analytical solution. We observe that at 5 grid cells we have significant discrepancies for the early time step between the numerical and analytical solution we derived in Chap. 3. Already at 10 grid cells the solution seems to be fair for all plotted time steps. Note that the first pressure point is at length $\Delta x/2$. Remember that the solution is also dependent on the time step length Δt. As we are using relatively small time steps in this solution, the error is likely caused by the spatial grid being too coarse.

5.2.2 *Implicit formulation*

The implicit formulation differs from the explicit formulation in that the time level of the pressure approximation is at time $t + \Delta t$, with the time derivative approximation a backward-difference quotient. Consider Eq. (5.14) evaluated at time $t + \Delta t$:

$$\eta \left(\frac{\partial^2 P}{\partial x^2} \right)_i^{t+\Delta t} = \left(\frac{\partial P}{\partial t} \right)_i^{t+\Delta t} \quad . \tag{5.23}$$

We use the centered second-difference quotient given by Eq. (4.44) for the spacial derivative, now evaluated at time $t + \Delta t$, as given by Eq. (5.3). For the time derivative we use the backward difference quotient given by Eq. (4.36), now evaluated at time $t + \Delta t$ as in Eq. (5.9). Inserting these equations into Eq. (5.23), we get

$$\eta \frac{P_{i+1}^{t+\Delta t} - 2P_i^{t+\Delta t} + P_{i-1}^{t+\Delta t}}{\Delta x^2} = \frac{P_i^{t+\Delta t} - P_i^t}{\Delta t} \quad . \tag{5.24}$$

Including our boundary conditions, we then get the following set of equations:

$$\begin{aligned}
\eta \frac{P_2^{t+\Delta t} - 3P_1^{t+\Delta t} + 2p_l}{\frac{3}{4}\Delta x^2} &= \frac{P_1^{t+\Delta t} - P_1^t}{\Delta t} \\
\eta \frac{P_{i+1}^{t+\Delta t} - 2P_i^{t+\Delta t} + P_{i-1}^{t+\Delta t}}{\Delta x^2} &= \frac{P_i^{t+\Delta t} - P_i^t}{\Delta t} \quad \forall i \in \{2, \ldots, n-1\} \\
\eta \frac{2p_r - 3P_n^{t+\Delta t} + P_{n-1}^{t+\Delta t}}{\frac{3}{4}\Delta x^2} &= \frac{P_n^{t+\Delta t} - P_n^t}{\Delta t} \quad .
\end{aligned} \tag{5.25}$$

Using the same constant as in the explicit case, $\alpha = \eta \Delta t / \Delta x^2$, the equations might be rearranged into the following forms:

$$\begin{aligned}
P_1^t &= (4\alpha + 1)P_1^{t+\Delta t} - \frac{4}{3}\alpha P_2^{t+\Delta t} - \frac{8}{3}\alpha p_l \\
P_i^t &= -\alpha P_{i+1}^{t+\Delta t} + (2\alpha + 1)P_i^{t+\Delta t} - \alpha P_{i-1}^{t+\Delta t} \quad \forall i \in \{2, \ldots, n-1\} \\
P_n^t &= (4\alpha + 1)P_n^{t+\Delta t} - \frac{4}{3}\alpha P_{n-1}^{t+\Delta t} - \frac{8}{3}\alpha p_r \quad .
\end{aligned} \tag{5.26}$$

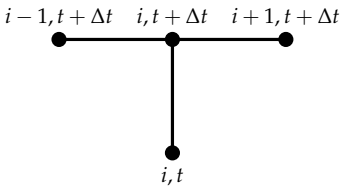


Figure 5.4: The stencil for the implicit method, Eq. (5.24), indicating the variables involved in a single time-step.

Now we have a set of n equations with n unknowns, which must be solved simultaneously. These equations can be written on matrix form as follows:

$$\vec{P}^t = \mathbf{A}\vec{P}^{t+\Delta t} - \vec{e} \quad . \tag{5.27}$$

With this notation, the pressure vector is $\vec{P}^t = [P_1^t, P_2^t, \ldots, P_n^t]$, with the obvious shift in time for $\vec{P}^{t+\Delta t}$, while the matrix $\mathbf{A}$ and vector $\vec{e}$ are given by

$$\mathbf{A} = \begin{bmatrix}
4\alpha+1 & -4\alpha/3 & 0 & 0 & \cdots & 0 & 0 & 0 \\
-\alpha & 2\alpha+1 & -\alpha & 0 & \cdots & 0 & 0 & 0 \\
0 & -\alpha & 2\alpha+1 & -\alpha & \cdots & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & 0 & \cdots & 2\alpha+1 & -\alpha & 0 \\
0 & 0 & 0 & 0 & \cdots & -\alpha & 2\alpha+1 & -\alpha \\
0 & 0 & 0 & 0 & \cdots & 0 & -4\alpha/3 & 4\alpha+1
\end{bmatrix} , \tag{5.28}$$

and

$$\vec{e} = \begin{bmatrix} \frac{8}{3}\alpha p_l \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \frac{8}{3}\alpha p_r \end{bmatrix} \quad . \tag{5.29}$$

To solve for the pressure $\vec{P}^{t+\Delta t}$, we then have to invert the matrix $\mathbf{A}$ to obtain

$$\vec{P}^{t+\Delta t} = \mathbf{A}^{-1}\vec{P}^{t} + \mathbf{A}^{-1}\vec{e} = \mathbf{A}^{-1}\left(\vec{P}^{t} + \vec{e}\right) \quad . \tag{5.30}$$

There are many methods for inverting a matrix, including the Gaussian elimination method. Our matrix $\mathbf{A}$ is what is called a *sparse matrix*, meaning a matrix where most elements are zero. There is a range of fast matrix inversion methods for sparse matrices, and several are implemented in Python. As it is more complicated to set up a sparse matrix inversion, we will just consider inversion of general matrices in this book.

Sparse matrix

The python package `scipy.sparse.linalg` contains several sparse matrix inversion options.

Setting up the matrix $\mathbf{A}$ and inverting it to $\mathbf{A}^{-1}$ can be implemented in Python as follows:

Classical literature on reservoir simulation, e.g., (Aziz and Settari, 1979) and (Peaceman, 1977), use significant space on solution methods for inverting matrices. Today all computing languages, including Python, have libraries for efficient matrix inversion. As we will just use existing packages, we will not treat the topic of matrix inversion in much detail.

```
def createImplicitMatrix(fAlpha,iNumberCells):
  #Create matrix
  aafMatrix=np.zeros([iNumberCells,iNumberCells])
  #First row
  aafMatrix[0,0]=4.0*fAlpha+1
  aafMatrix[0,1]=-4.0*fAlpha/3.0
  #Middle rows
  for ii in np.arange(1,iNumberCells-1):
    aafMatrix[ii,ii-1]=-fAlpha
    aafMatrix[ii,ii]=2.0*fAlpha+1
    aafMatrix[ii,ii+1]=-fAlpha
  #Last row
  aafMatrix[-1,-2]=-4.0*fAlpha/3.0
  aafMatrix[-1,-1]=4.0*fAlpha+1
  #Invert matrix
  aafMatrixInv=np.linalg.inv(aafMatrix)
  return aafMatrixInv
```

Here we use the `Numpy`-package `numpy.linalg.inv` for the inversion. This package is not very efficient for large sparse matrices. Thus for larger grids, such as those found in full field reservoir simulations models, one would need to use a more efficient method for inverting the matrix.

The vector $\vec{e}$ can be defined as:

```
def createImplicitVector(fAlpha,iNumberCells,fLeftPressure,
    ↪ fRightPressure):
  afVector=np.zeros([iNumberCells])
  afVector[0]=8.0*fLeftPressure*fAlpha/3.0
  afVector[-1]=8.0*fRightPressure*fAlpha/3.0
  return afVector
```

Having defined the vector $\vec{e}$ and calculated the inverted matrix $\mathbf{A}^{-1}$, we can calculate $\vec{P}^{t+\Delta t}$ in Eq. (5.30) as:

```
def implicitTimeStep(afPressure,aafMatrixInv,afVector):
  return np.dot(aafMatrixInv,afPressure+afVector)
```

Here the function `np.dot` calculates the matrix multiplication.

We use the same constants as before, and iterate over 30000 time steps, all of length $\Delta t = 1.0 \times 10^{-5}$ s:

```
#Time loop
fTime=0.0
iNumberLoops=30000
fMaxtime=fDeltat*iNumberLoops
while(fTime<fMaxtime):
  fTime+=fDeltat
  afPressure=implicitTimeStep(afPressure,aafMatrixInv,afVector)
  iPltCount+=1
  if iPltCount>iPltLim:
    iPltCount=0
    plt.plot(afPressure,color=cmap(1-fTime/fMaxtime))
```

Plotting the pressure at each thousand time step (`iPltLim=1000`) gives the plot shown in Fig. 5.5. We see again that the pressure converges to the steady-state solution.

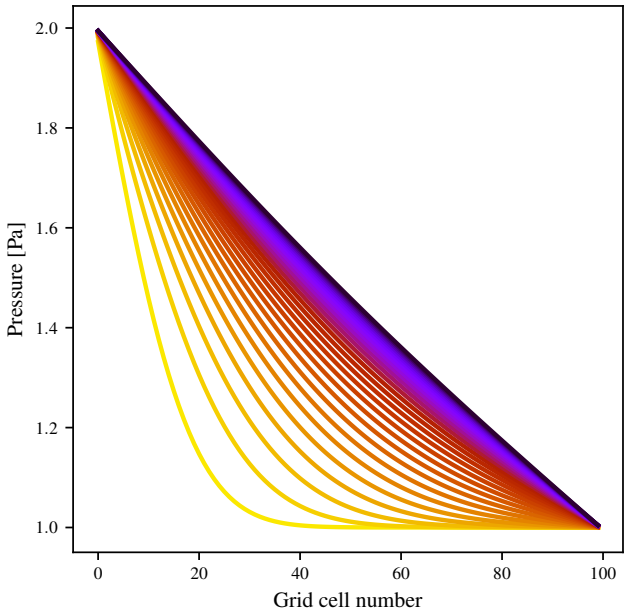


Figure 5.5: Solution to the implicit calculation of pressure. We have plotted one line for each 1000 time-step.

As in the explicit case, to test the implicit solution, we calculated the pressure using significantly fewer grid blocks, namely 5 and 10. The results are shown in Fig. 5.6, where the numerical solution is given by the points, while the analytical solution is given by the lines. As with the explicit case, we observe that we have significant discrepancies for the early time step for the case with 5 grid cells, while 10 grid cells seems to give fair solutions for all plotted time steps.

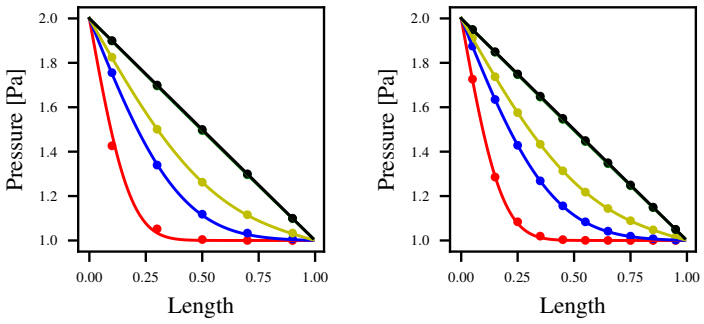


Figure 5.6: Implicit calculations of pressure P compared to the analytical solution p. The left plot shows implicit solutions when using 5 grid cells, while the right plot use 10 grid cells.

Comparing Figs. 5.6 and 5.3, we observe that the explicit and implicit formulations gives similar fits for the 5 grid cell case. The implicit calculation involves inverting a matrix, an operation that could have a significant computational cost. In our simple example, this inversion is conducted only once, thus the extra computational cost is small. If time steps are changing, this involves changing the α in the matrix $\mathbf{A}$, and thereby the matrix inversion needs to be conducted anew. Thus varying time steps could lead to a significant increase in computational cost for implicit versus explicit calculations. In our example the increased computational cost of the implicit method does not seem to have improved our results notably over the explicit case. However, the implicit case is more stable, as we will see in Sec. 5.3.

We also observe that the biggest discrepancies between the numerical and analytical solutions are close to the inlet, where the second derivative is at its largest (where you have the largest change in slope of the curve). The boundary condition had an error term of order $\mathcal{O}(\Delta x)$, compared to an error term of order $\mathcal{O}(\Delta x^2)$ for the internal cells. This could explain the large discrepancy for the first point. The discrepancy for the second point could also be related to the boundary condition, or it might be the result of poor approximation by the different quotient for such large values and rapid changes in the second derivative.

5.2.3 *Crank-Nicholson formulation*

The *Crank-Nicholson formulation* is another formulation for solving the diffusivity equation. As it is less used, we will not treat this formulation in as much detail as the explicit and implicit ones.

As seen from Eqs. (5.5) and (5.12), we also have the possibility of writing the equation at a time level half-way between t (explicit) and $t + \Delta t$ (implicit):

$$\eta \left(\frac{\partial^2 P}{\partial x^2} \right)_i^{t+\frac{\Delta t}{2}} = \left(\frac{\partial P}{\partial t} \right)_i^{t+\frac{\Delta t}{2}} . \tag{5.31}$$

By inserting Eqs. (5.5) and (5.12) into Eq. (5.31), we obtain the

difference equation for block i as:

$$\eta \frac{P_{i+1}^{t+\frac{\Delta t}{2}} - 2P_i^{t+\frac{\Delta t}{2}} + P_{i-1}^{t+\frac{\Delta t}{2}}}{\Delta x^2} = \frac{P_i^{t+\Delta t} - P_i^t}{\Delta t} \quad . \tag{5.32}$$

Here i is assumed to be an internal grid cell, and the boundary conditions needs to be treated similar to the explicit and implicit formulations above.

Since the pressures are defined at time levels t and $t + \Delta t$, and not at $t + \frac{\Delta t}{2}$, we cannot solve this equation as it is. Therefore, we rewrite the left side as the average of the explicit and implicit formulations:

$$\frac{\eta}{2}\left[\frac{P_{i+1}^{t+\Delta t} - 2P_i^{t+\Delta t} + P_{i-1}^{t+\Delta t}}{\Delta x^2} + \frac{P_{i+1}^{t} - 2P_i^{t} + P_{i-1}^{t}}{\Delta x^2}\right] = \frac{P_i^{t+\Delta t} - P_i^t}{\Delta t} \tag{5.33}$$

This yields n linear equations with n unknowns, and this set of linear equations must be solved simultaneously (as in the implicit case). Thus, the Crank-Nicholson formulation is an implicit method.

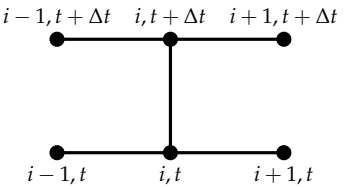


Figure 5.7: The stencil for the Crank-Nicholson formulation, indicating the input and output data for a single time step.

5.3 Stability

In Figs. 5.2 and 5.5 we identified errors at the early time steps, however, later time steps gave good approximations. We say that a numerical formulation is *stable* if an error at one point does not propagate into larger errors at later steps in the computation. In this section we will investigate the stability of numerical methods.

Stable numerical formulation

If p_i^t is the exact solution, while P_i^t is the numerical solution, we will find conditions for stability, i.e., conditions for the difference between the exact and numerical solution;

$$E_i^t = p_i^t - P_i^t \quad , \tag{5.34}$$

to stays bounded when $t \to \infty$. Finding requirements for a stable formulation involves conditions on the relation between time steps and grid resolution. In the following we will therefore assume that time steps Δt and grid sizes Δx are constant values.

There exist several methods for investigating the stability, however we will only use the von Neumann method. We will also restrict our investigation to the explicit and implicit formulations presented above, while the Crank-Nicholson formulation can be investigated similarly and is left as an exercise.

Several methods in addition to the von Neumann stability method are discussed in (Mitchell and Griffiths, 1980).

The starting point for the von Neumann method is to consider the difference between the exact and numerical solution E^{t_0} at a given time step t_0. Similar to what we did with the analytical solution, we assume that the function E^{t_0} can be written as a Fourier series of the form

$$E^{t_0}(x) = \sum_{j=1}^{\infty} \gamma_j e^{i\beta_j x} \quad , \tag{5.35}$$

where i is the imaginary unit $i = \sqrt{-1}$, and γ_j and β_j are constants.

When we found the analytical solution, Eq. (3.77), for the diffusivity equation, we used the sin-cosine form of the Fourier series. Here we use the exponential form instead, employing that $\exp(i\theta) = \cos(\theta) + i\sin(\theta)$.

To investigate the propagation of the error E^{t_0} for time steps $t > t_0$, we want to find a function $\epsilon(x,t)$ that is a solution to our numerical formulation, and where $\epsilon(x,t_0) = E^{t_0}(x)$. Let such a solution be written as

$$\epsilon(x,t) = \sum_{j=1}^{\infty} \psi_j(t) e^{i\beta_j x} \quad , \tag{5.36}$$

where $t \geq t_0$. Here we assumed that the function ϵ could be split into a sum where each summand consists of a product of two functions; one function dependent only on the spacial coordinate x and one function dependent only on the time t. To simplify our derivations and notation, we will continue considering only a single element of the sum:

$$\epsilon(x,t) = \psi(t) e^{i\beta x} \quad . \tag{5.37}$$

We observe that the original error component E^{t_0} will be bounded in time if

$$\left| \frac{\psi(t+\Delta t)}{\psi(t)} \right| \leq 1 \quad . \tag{5.38}$$

This last inequality is called the *von Neumann's criterion for stability*. We will deduce this criterion for the explicit and implicit formulations in the following subsections.

von Neumann's criterion for stability

5.3.1 Stability analysis for the explicit formulation

Following Eq. (5.20), we might write the explicit difference equation using the index system as:

$$\eta \frac{P_{i+1}^t - 2P_i^t + P_{i-1}^t}{\Delta x^2} = \frac{P_i^{t+\Delta t} - P_i^t}{\Delta t} \quad . \tag{5.39}$$

Assume that P_i^t is a solution to the underlying differential equation, Eq. (3.54), and that E^{t_0} is the difference between the exact and numerical solution at time step t_0 as given by Eq. (5.34). Further, assume a function ϵ as given by Eq. (5.36), and that this has a numerical solution written as ϵ_i^t with our numerical index system, where ϵ_i^t for $t > t_0$ satisfy the explicit numerical scheme. Let $\tilde{P}_i^t$ be the numerical solution for $t > t_0$ which is set equal to the exact solution at t_0, i.e., $\tilde{P}_i^{t_0} = p_i^{t_0}$. This solution is also propagated forward using the numerical scheme:

Here we are a bit sloppy with our notation, using ϵ for both the underlying error and the numerical representation of this error. The exact and numerical representation can be distinguished by the use of the index notation for the numerical representation, while there are no indices when we add an error to the exact solution.

$$\eta \frac{\tilde{P}_{i+1}^t - 2\tilde{P}_i^t + \tilde{P}_{i-1}^t}{\Delta x^2} = \frac{\tilde{P}_i^{t+\Delta t} - \tilde{P}_i^t}{\Delta t} \quad . \tag{5.40}$$

This is then the numerical solution for $t \geq t_0$ where we have removed the error at time-step t_0.

Then, as $P_i^t = \tilde{P}_i^t + \epsilon_i^t = p_i^{t_0} + \epsilon_i^t$ for $t = t_0$, the numerical scheme for P_i^t can be written as

$$\eta \frac{(\tilde{P}_{i+1}^t + \epsilon_{i+1}^t) - 2(\tilde{P}_i^t + \epsilon_i^t) + (\tilde{P}_{i-1}^t + \epsilon_{i-1}^t)}{\Delta x^2} = \frac{(\tilde{P}_i^{t+\Delta t} + \epsilon_i^{t+\Delta t}) - (\tilde{P}_i^t + \epsilon_i^t)}{\Delta t} \quad , \tag{5.41}$$

where $\epsilon^{t_0} = E^{t_0}$. Thus, we have an error $\epsilon^{t_0} = E^{t_0}$ at time-step t_0, and we now want to investigate how this error propagates. Combining the two equations above, we then have:

$$\eta \frac{\epsilon_{i+1}^t - 2\epsilon_i^t + \epsilon_{i-1}^t}{\Delta x^2} = \frac{\epsilon_i^{t+\Delta t} - \epsilon_i^t}{\Delta t} \quad . \tag{5.42}$$

So also the error in time-step t_0, ϵ^{t_0}, propagates forward by the same numerical scheme.

Writing ϵ_i^t on the form given by Eq. (5.36), using only a single element of the sum, we have

$$\epsilon_i^t = \epsilon(x_i, t) = \psi(t) e^{i\beta x_i} \mid t \geq t_0 \quad . \tag{5.43}$$

With this notation we then have

$$\begin{aligned}
\epsilon_i^t &= \epsilon(x_i, t) = \psi(t) e^{i\beta x_i} \\
\epsilon_{i+1}^t &= \epsilon(x_i + \Delta x, t) = \psi(t) e^{i\beta(x_i + \Delta x)} \\
\epsilon_{i-1}^t &= \epsilon(x_i - \Delta x, t) = \psi(t) e^{i\beta(x_i - \Delta x)} \\
\epsilon_i^{t+\Delta t} &= \epsilon(x_i, t + \Delta t) = \psi(t + \Delta t) e^{i\beta x_i} \quad .
\end{aligned} \tag{5.44}$$

Using this notation into Eq. (5.42), we get

$$\begin{aligned}
\frac{\eta \Delta t}{(\Delta x)^2} \left(\psi(t) e^{i\beta(x_i + \Delta x)} - 2\psi(t) e^{i\beta x_i} + \psi(t) e^{i\beta(x_i - \Delta x)} \right) &= \psi(t + \Delta t) e^{i\beta x_i} - \psi(t) e^{i\beta x_i} \\
\alpha \psi(t) \left(e^{i\beta x_i} e^{i\beta \Delta x} - 2e^{i\beta x_i} + e^{i\beta x_i} e^{-i\beta \Delta x} \right) + \psi(t) e^{i\beta x_i} &= \psi(t + \Delta t) e^{i\beta x_i} \\
\psi(t) e^{i\beta x_i} \left(1 + \alpha \left(e^{i\beta \Delta x} - 2 + e^{-i\beta \Delta x} \right) \right) &= \psi(t + \Delta t) e^{i\beta x_i} \\
\psi(t) \left(1 - 4\alpha \sin^2 \left(\frac{\beta \Delta x}{2} \right) \right) &= \psi(t + \Delta t) \quad .
\end{aligned} \tag{5.45}$$

Here we employ that:

$$\begin{aligned}
i \sin(\theta) &= \frac{1}{2} \left(e^{i\theta} - e^{-i\theta} \right) \\
-\sin^2(\theta) &= \frac{1}{4} \left(e^{2i\theta} - 2e^{i\theta - i\theta} + e^{-2i\theta} \right) \\
-\sin^2(\theta) &= \frac{1}{4} \left(e^{2i\theta} - 2 + e^{-2i\theta} \right)
\end{aligned}$$

We then obtain the following expression:

$$\frac{\psi(t + \Delta t)}{\psi(t)} = 1 - 4\alpha \sin^2 \left(\frac{\beta \Delta x}{2} \right) \quad . \tag{5.46}$$

The ratio $\frac{\psi(t+\Delta t)}{\psi(t)}$ may be interpreted as the ratio of increase in error during the time interval Δt . The von Neumann criterion for stability, as given by Eq. (5.38) is that the ratio of increase in error must be smaller or equal to 1, i.e.

Strictly speaking the ratio of error $|\psi(t + \Delta t)/\psi(t)|$ must be smaller than $1 - \mathcal{O}(\Delta t)$ to allow for exponentially growing solutions.

$$\left| \frac{\psi(t + \Delta t)}{\psi(t)} \right| \leq 1 \quad . \tag{5.47}$$

On the other hand, if this ratio is larger than one, the solution becomes unstable.

As our explicit formulation gave Eq. (5.46), we get the following von Neumann criterion for stability:

The von Neumann criterion for stability for the explicit method

$$\left| 1 - 4\alpha \sin^2 \left(\frac{\beta \Delta x}{2} \right) \right| \leq 1 \quad . \tag{5.48}$$

Writing out this absolute value inequality, we have

$$-1 \leq 1 - 4\alpha \sin^2 \left(\frac{\beta \Delta x}{2} \right) \leq 1 \quad . \tag{5.49}$$

Since Δx and Δt are positive values, the right inequality always holds as $0 \leq \sin^2(\beta \Delta x/2) \leq 1$. By multiplying by -1 and using $0 \leq \sin^2(\beta \Delta x/2) \leq 1$) the left inequality can be reformulated as

$$\begin{aligned} 4\alpha - 1 &\leq 1 \\ \alpha &\leq \frac{1}{2} \\ \frac{\eta \Delta t}{(\Delta x)^2} &\leq \frac{1}{2} \\ \Delta t &\leq \frac{(\Delta x)^2}{2\eta} \quad . \end{aligned} \tag{5.50}$$

This is then the von Neumann stability condition for the explicit method.

When given a grid size Δx, we thus obtain an upper limit for our time steps in the explicit solution when fulfilling the von Neumann stability criterion. In other words, whenever Eq. (5.50) holds our explicit formulation will be stable.

5.3.2 *Stability analysis for the implicit formulation*

The implicit form of the difference equation using the index system was given by Eq. (5.24) as

$$\eta \frac{P_{i+1}^{t+\Delta t} - 2P_i^{t+\Delta t} + P_{i-1}^{t+\Delta t}}{\Delta x^2} = \frac{P_i^{t+\Delta t} - P_i^t}{\Delta t} \quad . \tag{5.51}$$

Following a similar procedure as in the explicit case, we obtain the following equation for the perturbation error term ϵ:

$$\eta \frac{\epsilon_{i+1}^{t+\Delta t} - 2\epsilon_i^{t+\Delta t} + \epsilon_{i-1}^{t+\Delta t}}{\Delta x^2} = \frac{\epsilon_i^{t+\Delta t} - \epsilon_i^t}{\Delta t} \quad . \tag{5.52}$$

Filling in equality's similar to Eq. (5.44), and assuming that $\epsilon(x,t) = \psi(t)\exp(i\beta x)$, we have:

$$\begin{aligned} \alpha\psi(t+\Delta t)\left(e^{i\beta x_i}e^{i\beta\Delta x} - 2e^{i\beta x_i} + e^{i\beta x_i}e^{-i\beta\Delta x}\right) &= \psi(t+\Delta t)e^{i\beta x_i} - \psi(t)e^{i\beta x_i} \\ \psi(t+\Delta t)e^{i\beta x_i}\left(-1 + \alpha\left(e^{i\beta\Delta x} - 2 + e^{-i\beta\Delta x}\right)\right) &= -\psi(t)e^{i\beta x_i} \\ \psi(t+\Delta t)\left(-1 - 4\alpha\sin^2\left(\frac{\beta\Delta x}{2}\right)\right) &= -\psi(t) \\ \frac{\psi(t+\Delta t)}{\psi(t)} &= \frac{1}{1 + 4\alpha\sin^2\left(\frac{\beta\Delta x}{2}\right)} \quad . \end{aligned} \tag{5.53}$$

The condition for von Neumann stability then becomes:

The von Neumann criterion for stability for the implicit method

$$\left| \frac{1}{1 + 4\alpha\sin^2\left(\frac{\beta\Delta x}{2}\right)} \right| \leq 1 \tag{5.54}$$

which is always true, since the denominator is always greater than 1. Thus there are no upper limit on the time-step Δt for the implicit formulation.

Application of the von Neumann stability analysis to the Crank-Nicholson formulation, shows that it is also unconditionally stable, just as the implicit case. As a final note, the von Neumann method is strictly speaking just valid for pure initial value problems with periodic initial data. However, it gives a necessary condition for stability regardless of the initial data.

5.4 Comparison of methods

The explicit method avoids matrix inversion, as the pressures are obtained explicitly from pressures at the previous time step. Therefore the explicit formulation is simpler than the implicit formulation, and accordingly need less computational power. As we have to store the matrix, the implicit method also needs more memory. This can be significant for larger grids, however, this is usually not a big issue with modern day memory capacity. As the matrix inversion is conducted only once in our simple 1D case, the difference in computational power is negligible. Larger and more complex grids and varying time-steps will give a significant difference in computational time for the two methods. This difference is more pronounced for multi-phase cases, which we will treat later in this book, as we then need to update the matrix in every time-step, and therefore also invert the matrix in every time-step.

Both the explicit and the implicit method use the second-difference quotient, Eq. (4.44), to approximate the second derivative, thus they both have a discretization error of order $\mathcal{O}(\Delta x^2)$. For the time derivative the explicit case use the forward-difference, while the implicit method use the backward-difference. Both the forward and backward-difference have a truncation error of order $\mathcal{O}(\Delta t)$, hence the truncation error is of the same order for the two methods. Thus, both discretization errors are of the same order for the two methods.

The explicit formulation becomes unstable for large time steps, as indicated by the von Neumann stability criterion, Eq. (5.50):

$$\Delta t \leq \frac{(\Delta x)^2}{2\eta} \quad . \tag{5.55}$$

We observe that the upper boundary for the time step size is limited by both grid block size and properties of the rock and fluid. It is the grid block with the smallest value of $(\Delta x)^2/(2\eta)$ that determines the limiting time step size. For grids with varying grid block size, this limitation may be severe. This is one reason why the explicit method is seldom used for realistic grids of reservoirs.

From Eq. (5.54) we observe that the von Neumann stability criterion applied to the implicit formulation indicate that the implicit method is unconditionally stable, i.e., the method is stable for all time step sizes. The additional computational cost per time step for the implicit method compared to the explicit method is in practice compensated for by the possibility of using larger time steps. How-

ever, as larger time steps lead to larger numerical errors, one always have to check that the numerical discretization errors are within acceptable limits.

The Crank-Nicholson formulation has less discretization error than the two other methods, as the central approximation of the time derivative, Eq. (4.40), has a truncation error of order $\mathcal{O}(\Delta t^2)$, compared to a truncation error of order $\mathcal{O}(\Delta t)$ for the explicit and implicit cases. Applying a von Neumann stability analysis to the Crank-Nicholson formulation, similar to what was done for the other two formulations above, yields that the Crank-Nicholson method is unconditionally stable, just as the method using the implicit formulation. Thus the Crank-Nicholson method has a smaller order for its discretization error than the other two methods presented above, and at the same time it is unconditionally stable. This could have indicated that the Crank-Nicholson method is a preferred method. Unfortunately, the Crank-Nicholson method can give oscillating (albeit decaying) results for the solved pressure when $\Delta t > (\Delta x)^2/(2\eta)$. It thus have a similar time step range as the explicit method, but at a higher computational cost. It is therefore seldom used.

5.5 Exercises

Exercise 5.1 Create a Python script to calculate both the explicit and implicit solution for the pressure. The basic data is given in Table 5.1.

a) Plot the pressure function for the same set of times as used in Exercise 3.1. Compare the numerical solutions to the analytical solutions.

b) Calculate the von Neumann stability criterion for the explicit solution. Investigate numerical solutions around the stability criterion. When does your simulations break down? Compare the solutions (when you get an explicit solution) to the implicit method. How does the implicit method behave for time steps larger than the time step limit given by the stability analysis for the explicit case?

	SPE Metric	SI
l	200 m	200 m
k	100 mD	1.0×10^{-13} m^2
μ	1 cP	1×10^{-3} Pa s
ϕ	0.2	0.2
c_t	1×10^{-4} bar^{-1}	1×10^{-9} Pa^{-1}
p_l	200 bar	2×10^7 Pa
p_r	100 bar	1×10^7 Pa

Table 5.1: Basic data for example.

Exercise 5.2 Complete the derivation between Eq. (5.52) and Eq. (5.53) for the von Neumann criterion for the implicit case.

Exercise 5.3 Derive the von Neumann criterion for the Crank-Nicholson method.

6
Reservoir simulation of single-phase flow

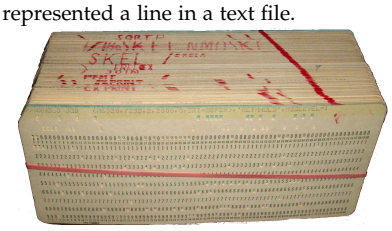
Rocket engines burning fuel so fast
Up into the night sky they blast
Through the universe the engines whine
Could it be the end of man and time?

Black Sabbath - Into The Void

In this chapter we will continue with the one-dimensional single phase case considered in Chap. 3 and Chap. 5. In this chapter we will solve the diffusivity equation, Eq. (3.52), using the reservoir simulator OPM-Flow. OPM-Flow is a reservoir simulator with up to three phases (oil, water and gas). It is not created for single phase flow, so even though we will use it for single phase flow, the simulator itself will solve multi-phase equations with only one non-zero phase. This also means that we need to define a two-phase model (e.g., oil and water), but we will only have water in the model.

OPM-Flow is based on providing an input file, by some called an input deck[1], which contain all reservoir data, fluid data and the initial and boundary conditions (e.g., well control). The input file will be described in the following.

[1] The term deck for a computer file goes back to the ancient (pre 1980) times of punched cards. Each card represented a line in a text file.



Picture by Arnold Reinhold (Wikipedia)

6.1 *OPM-Flow input files (`*.DATA`-files)*

The name of the main input file to OPM-Flow is commonly written with capital letters, and with the file extension `*.DATA`. These files are therefore sometimes referred to as DATA-files. The simulator input can be completely specified in the DATA-file, but it is most common to split the input over several files that are read in using `INCLUDE` keywords in the DATA-file. There is no naming convention for these included files.

The DATA-file is split into sections. Each section starts by a keyword, being the name of the section. Be aware that the sections need to be in correct order. We will describe all the DATA-file sections in the following, together with examples of keywords for the different sections.

6.1.1 *RUNSPEC*

The first section gives the overall settings of the simulation. This include

- Title
- Dimensions of the simulation grid
- Which phases are present (water, oil, gas)
- Start date for the simulation
- Unit system (metric, oil field units, etc.)
- Dimensions of parts of the system (e.g., number of wells, number of saturation points in the flow parameter tables, etc.)[2]

[2] We define the dimensions for legacy reasons: In old programming languages it was impossible to add memory dynamically. One therefore had to specify all memory usage at the beginning of the run.

An example of a RUNSPEC section is shown below. This example is for the single phase case considered in the previous and present chapters. It is not possible to run OPM-Flow with a single phase. To circumvent this, we tell the simulator to consider two phases, but we will later initialize the model with one phase only and inject only this phase into the model.

In the OPM-Flow *.DATA-files lines are commented out by a double dash: --. So the lines shown in green are commented out and not read by the simulator.

```
RUNSPEC
-- -----------------------------------------------

TITLE
 1D model for water


DIMENS
   102 1 1 /

-- The number of fluid property (PVT) tables is
-- inferred from the TABDIMS keyword; when no
-- data is included in the keyword, then the
-- default values are used.
TABDIMS
/

-- We do not have oil in our system; we still need
-- the OIL keyword since the Flow simulator is
-- a multiphase simulator.
OIL
WATER

-- The unit system to be used
METRIC

-- The start of simulation
START
   1 'JAN' 2015 /
```

```
WELLDIMS
-- Item 1: maximum number of wells in the model
--     - there are two wells in the problem;
--           injector and producer
-- Item 2: maximum number of grid blocks connected
--  to any one well
--     - must be one as the wells are located
--           at specific grid blocks
-- Item 3: maximum number of groups in the model
--     - we are dealing with only one 'group'
-- Item 4: maximum number of wells in any one group
--     - there must be two wells in a group
--           as there are two wells in total
   2 1 1 2 /

-- The UNIFOUT keyword is used to request the
-- output files to be on a unified format, i.e.
-- the output from different timesteps are
-- collected into a single file.
UNIFOUT
```

We use keywords to define properties in the `*.DATA`-file. These keywords are always written with capital letters. Note that the simulator is case-sensitive.

As an example, consider the keyword `DIMENS`. This keyword defines the dimensions of the model. Note that these values are not maximum values, they are the actual size of the model. And these dimensions will dictate the number of input data needed in other keywords. As an example, since our model has size $102 \times 1 \times 1 = 102$, we will need to pass 102 values for, e.g., porosity later in the DATA-file. All keywords are described in detail in the OPM Manual, including examples of how to use them.

The manual for OPM-Flow is found on the OPM website: `https://opm-project.org/`

In our one dimensional example considered earlier we used pressure (Dirichlet) boundary conditions. There are no option for such boundary conditions in OPM-Flow. We will therefore use dummy grid blocks to mimic a Dirichlet boundary condition for the whole cross-sectional area of the ends of the model. So our actual model is only the 100 grid cells in the middle, with two dummy cells at each side giving a total of 102 cells.

6.1.2 *GRID*

The second section, the `GRID` section, defines the geometry of the grid and petrophysical properties (porosity, permeability) of the grid cells. From the input in the `GRID` section the simulator calculates properties such as the cell's pore volume, depth, and inter-cell transmissibilities. Note that OPM-Flow is using a two-point flux approximation by default, as presented in Sec. 7.2.1. An example of the `GRID` section for our 1D example is shown below.

```
GRID
-- ----------------------------------------------------
```

```
-- The INIT keyword is used to request an .INIT file.
-- The .INIT file is written before the simulation
-- actually starts, and contains grid properties and
-- saturation tables as inferred from the input
-- deck. There are no other keywords which can be
-- used to configure what is written to the .INIT
-- file.
INIT

DX
-- There are in total 102 cells with length 2.0m
-- in x-direction. The main part of the model is
-- the 100 cells in the middle. The two outer
-- cells are dummy cells with high permeability
-- to distribute the flow from the wells to the
-- full cross-section of the model.
    102*2.0 /

DY
  102*2.0 /

DZ
  102*2.0 /

TOPS
-- The depth of the top of each grid block
  102*1000 /

PORO
-- Constant porosity of 0.2
    0.5 100*0.2 0.5 /

PERMX
-- Permeability 100mD for the internal cells, a
-- high permeability for the dummy cells
  100000.0 100*100.0 100000.0 /

PERMY
-- As the model is 1D, the PERMY is not relevant
  102*100.0 /

PERMZ
-- As the model is 1D, the PERMZ is not relevant
  102*100.0 /
```

In the above example the grid geometry is given by specifying each grid block as a horizontal rectangle with dimensions given by the keywords `DX`, `DY`, and `DZ`, and top depth given the keyword `TOPS`. This method is applicable to simple toy models only, in real reservoir models the geometry is specified using corner-point geometry (see page 113) through the keywords `COORD` ans `ZCORN`. Although in `ASCII` format, the data for these keywords are always generated by software and not meant to be read or edited by humans.

6.1.3 *EDIT*

The `EDIT` section enables changes to the grid structure defined in the `GRID` section. This includes pore volume multipliers, changes to the transmissiblities, etc. Most of the properties defined in the `EDIT` section can also be defined in the `GRID` section, thus not all models include an `EDIT` section. However, the `EDIT` section can be useful in, e.g., history-matching loops.

Our 1D example does not have an `EDIT` section.

6.1.4 *PROPS*

The `PROPS` section defines the flow parameters (relative permeability and capillary pressure), fluid properties (PVT) and rock properties (rock compressiblity). The `PROPS` section for our 1D example is shown below.

```
PROPS
-- ------------------------------------------------

SWOF
-- This keyword is not used in our case, since we
-- do not have oil in our system.
--    Sw      Krw    Krow     Pcow
--                            (bar)
 0.0 0.0 1.0 0.0
 1.0 1.0 0.0 0.0 /

DENSITY
--   Oil       Water     Gas
-- (kg/m3)   (kg/m3)   (kg/m3)
     849       1025      0.82   /

-- PVT tables for water. We only have one table,
-- consisting of a single line (only the reference
-- pressure).
PVTW
-- REF.PRES.  REF. FVF  COMPRESSIBILITY  REF.VISC.
--        -> VISCOSIBILITY
--  (bar) (m3/m3) (1/bar)    (cP)       (1/bar)
     150   1.01   1.0e-4     1.0        0.0e+0   /

-- PVT tables for oil. This keyword is not used in
-- our case, since we do not have oil in our system.
PVDO
--  PRES.   FVF.   VISC.
    20.68   1.05   2.85
    55.16   1.02   2.99
   551.58   1.01   3.00
/

ROCK
-- REF.PRES   COMPRESSIBILITY
--  (bars)        (1/bars)
     150          0.0e-6   /
```

6.1.5 *REGIONS*

The REGIONS section is used to define sub-regions of the reservoir model where different properties applies. As an example, one can define different fluid properties for different regions in the PROPS section. Then one will specify the spatial distribution of the different fluid property classes in this section. The REGIONS section can also define regions with different model initialization, defined in the SOLUTION section.

Our 1D example does not have an REGIONS section as we only have a single region for both fluid properties and initialization.

6.1.6 *SOLUTION*

The SOLUTION section defines how to initialize the model. This is usually done by defining the depth of the fluid–fluid contacts (oil–water contact and gas–oil contact), and the pressure at a specified depth. The corresponding keyword is EQUIL. Alternatively the state of each grid-block (pressure and saturation) can be specified directly. In our 1D example we use the latter method.

```
SOLUTION
-- ------------------------------------------------

-- Initial water saturation for each grid cell
-- We have only water, so the saturation is set
-- to unity.
SWAT
  102*1.0
/

-- Initial pressure for each grid cell.
PRESSURE
  102*100.0
/
```

6.1.7 *SUMMARY*

The SUMMARY section defines what should be saved from the simulation. It is common to include the keyword ALL; despite its name this keyword does not include all output, but all the commonly used keywords. The SUMMARY section for our 1D example is shown below.

```
SUMMARY
-- ------------------------------------------------

-- This keyword enable printing out all the most
-- commonly used data.
ALL

-- This keyword enable printing out the
```

```
-- cummulative CPU use at different time steps.
TCPU
```

6.1.8 *SCHEDULE*

The SCHEDULE section defines when and where to add wells, and how to operate these wells. It also defines how long the simulation shall run, and what should be printed to RESTART files. As the RESTART files store most of the grid data, such as pressure and saturation values, they are not only used for restarting simulations, but also for visualizing the simulation results. Thus the time steps between your simulation visualization in ResInsight needs to be specified in this section. The SCHEDULE section for our 1D example is shown below. Note that this section ends with the keyword END, which indicates the end of the simulation input.

```
SCHEDULE
-- ------------------------------------------------

-- This keyword enable printing out restart files.
RPTRST
 BASIC=1 /


-- The WELSPECS keyword is used to defined some
-- general data for the wells; where they enter
-- the grid, and what kind of fluids they handle.
WELSPECS
-- Item #:
--  1  2  3 4 5  6
 'PROD' 'G1'  102 1 1005  'WATER' /
 'INJW' 'G1'  1 1 1005  'WATER' /
/

-- The COMPDAT keyword defines how a well is
-- connected to the reservoir (where it is
-- completed/perforated)
COMPDAT
-- Item #:
-- 1  2   3 4 5  6   7  8 9
 'PROD' 102 1 1 1 'OPEN' 1* 1*  0.005 /
 'INJW' 1   1 1 1 'OPEN' 1* 1*  0.005 /
/

-- This keyword defines well constrains for
-- production wells
WCONPROD
-- Item #:1 2      3    4-8 9
  'PROD' 'OPEN' 'BHP' 5*  99.9 /
/

-- This keyword defines well constrains for
-- injection wells
WCONINJE
```

```
-- Item #:1  2   3   4  5-6 7
  'INJW'  'WATER' 'OPEN'  'BHP' 2*  199.9 /
/

-- This keyword defines times when the simulation
-- shall advance. One can use both dates and times.
TSTEP
 5*0.02 /

TSTEP
 4*0.1 /

END
```

6.2 *Reservoir simulation of single phase flow*

In this section we will run the 1-dimensional single phase reservoir simulation model described in the previous section. We assume the input data in the previous section has been collected into an `ASCII` file named `SINGLEPHASE.DATA`. You can collect the data using any text editor.

To run the input deck, write the following in the command line

```
flow SINGLEPHASE1D.DATA
```

This will run the simulation model, and should produce an output that starts similar to

```
****************************************************************************
*                                                                          *
*                        This is flow 2022.10                              *
*                                                                          *
* Flow is a simulator for fully implicit three-phase black-oil flow,       *
*             including solvent and polymer capabilities.                  *
*          For more information, see https://opm-project.org               *
*                                                                          *
****************************************************************************

Using 1 MPI processes with 2 OMP threads on each

Reading deck file 'SINGLEPHASE1D.DATA'
```

The restart file is stored using the file extension `*.UNRST` (unified restart). While the name of the restart file links it to restarting the simulation at different time-steps, the restart file is probably more frequently used for visualization purposes. Storing data needed for a restart involves storing e.g. pressure and saturation for all grid cells, which opens up for visualizing the same data in three dimensions.

The data stored in the restart file can be visualized using ResInsight. In Fig. 6.1 is an example showing the pressure at time step 5 for our grid.

We now want to plot the resulting data using the Python package `ecl`. For this, we start by loading the specific `ecl` package we need in Python, in our case the package `EclFile`:

```
from ecl.eclfile import EclFile
```

Figure 6.1: Reservoir simulation of the single phase 1D case visualized in ResInsight. This visualization shows the pressure distribution along the model at time step 5.

We can then load the restart file using the `ecl` package `EclFile`, and then extract the pressure data at the different time steps, as follows:

```
hRestartFile=EclFile('SINGLEPHASE1D.UNRST')
iTimeSteps=hRestartFile.num_report_steps()

afxGrid=np.arange(1.0,fModelLength,2.0)
plt.figure()
fMaxtime=hRestartFile.iget_restart_sim_days(iTimeSteps-1)
cmap = plt.get_cmap('gnuplot')

for iTimeStep in range(1,iTimeSteps):
        fTime=hRestartFile.iget_restart_sim_days(iTimeStep)
        afPressure=hRestartFile.iget_named_kw('PRESSURE',iTimeStep
    ↪ )
        plt.plot(afxGrid,afPressure[1:-1]*1E5,color=cmap(1-fTime/
    ↪ fMaxtime))
```

Here we are using several build in functions for the restart-file structure in `ecl`, including getting the number of restart steps (`.num_report_steps()`), the physical time of the different restart steps (`.iget_restart_sim_days()`), and extracting the data we are interested in, which in our case is the pressure (`.iget_named_kw('PRESSURE',iTimeStep)`). You can obtain a full description of all built-in functions by typing

```
>>> help(hRestartFile)
```

in the Python command line (note that the >>> should not be typed in, this is just the standard line-start in Python when running Python in the command line window). The `help()` function should give an output similar to the following.

```
Help on EclFile in module ecl.eclfile.ecl_file object:

class EclFile(cwrap.basecclass.BaseCClass)
 |  Method resolution order:
 |      EclFile
 |      cwrap.basecclass.BaseCClass
 |      __builtin__.object
 |
 |  Methods defined here:
 |
 |  __contains__(self, kw)
 |      Check if the current file contains keyword @kw.
 |
 |  __getitem__(self, index)
 |      Implements [] operator; index can be integer or key.
 |
 |      Will look up EclKW instances from the current EclFile
 |      instance. The @index argument can either be an integer, in
 |      which case the method will return EclKW number @index, or
 |      alternatively a keyword string, in which case the method will
 |      return a list of EclKW instances with that keyword:
 |
```

```
|       restart_file = ecl_file.EclFile("ECLIPSE.UNRST")
```

In the Python script above we have also included a line for plotting the pressure at the different time-steps, where the color coding represent the physical time for the plotted pressure distribution. Running the script gives the plot shown in Fig. 6.2.

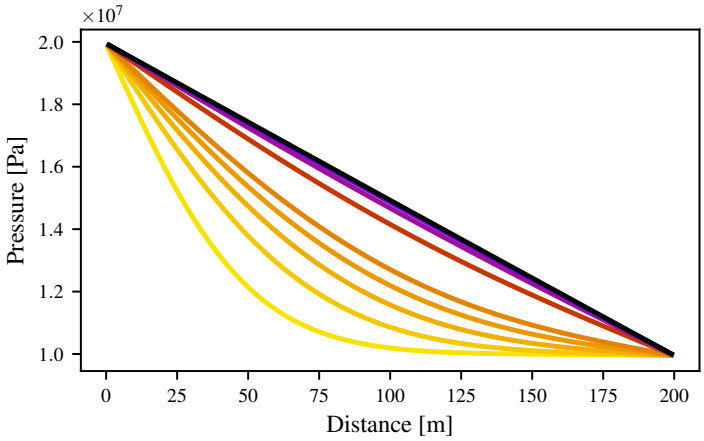


Figure 6.2: The pressure distribution of the reservoir simulation of the single phase 1D case at different time steps.

The pressure at the different time steps can then be compared to the analytical solution, as shown in Fig. 6.3. For most time steps, the solution from OPM-Flow and the analytical solution match up to a fair degree. However, for the early time steps we see large discrepancies. For the OPM-Flow solution there is a strong kink from a smooth curve to a pressure which equal the initial pressure. This numerical error linked to the mass balance tolerance; tightening this from the default 10^{-6} to 10^{-10} will remove these kinks. Improved match with the exact solution also requires shorter time steps.

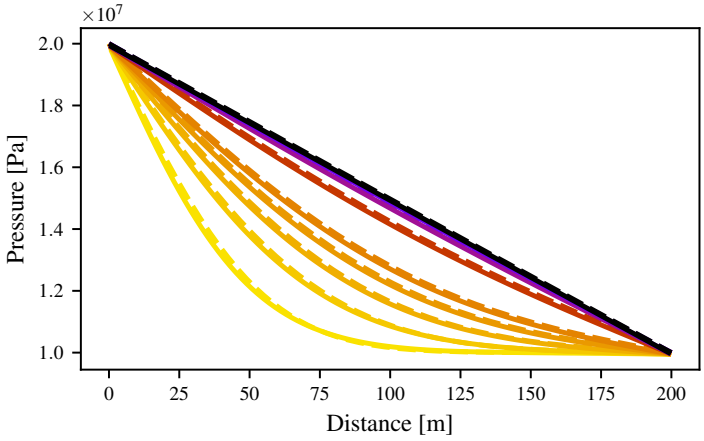


Figure 6.3: Comparison of the reservoir simulation results with the analytical solution. The full-drawn line are the simulation results, while the dashed line is the analytical solution. The results shows some clear discrepancies.

In the following command line we have specified the maximal time step explicitly, and we have also set a tighter mass balance tolerance:

```
flow SINGLEPHASE1D.DATA --tolerance-mb=1e-10 --solver-max-time-step-in-days=0.005
```

The resulting pressure plot is shown in Fig. 6.4. The plot now shows an almost perfect match. This example shows that one always need to carefully assess the numerical solution with respect to time step size and grid size.

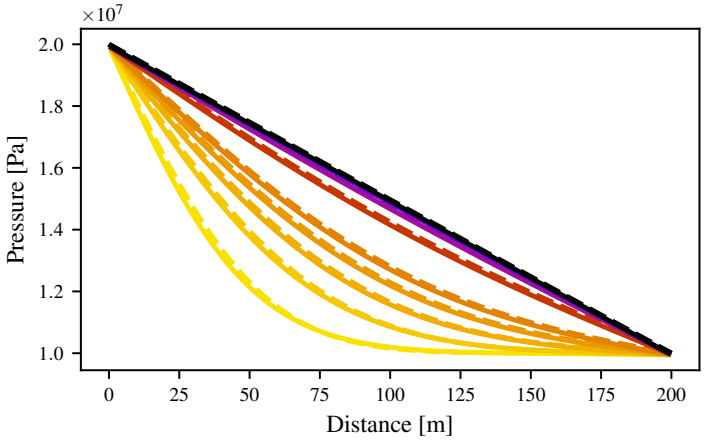


Figure 6.4: Comparison of the reservoir simulation results with the analytical solution, after running the simulation with reduced time step length and stricter mass balance tolerance.

In might be surprising that our earlier simple Python scripts is doing a better job in capturing the evolution of the pressure in our 1D model than the rather large and complicated software OPM-Flow. Keep in mind that OPM-Flow is not build for such simple cases, but rather for multi-phase simulations on complex grids.

6.3 *Well inflow modeling*

Modeling of well inflow and outflow is at the same time both very important and very challenging. We will only briefly introduce it in this section. We will sketch a well inflow model derived by Peaceman, and which is included in OPM-Flow. We will also compare the simulation results from OPM-Flow with external calculations of extensions of the Peaceman model.

In our one-dimensional OPM-Flow example above, we used dummy grid blocks to mimic a Dirichlet boundary condition for the whole cross-sectional area of the ends of the model. When modeling reservoir fluid flow, the injection and production of fluids happen through wells. We therefore need to couple the wells and the reservoir through an appropriate model for well inflow and outflow. Ideally, we would have a fine scale grid able to fully resolve the well so that we could have used either a Dirichlet or von Neumann boundary condition on the rock surface towards the well. Except for specialized near well-bore models, this is unrealistic, as the amount of grid cells would make the computational cost inhibiting. We therefore need a model for coupling the well to the reservoir.

Our inflow model is based on assuming radial flow in the near well-bore region. This is a fair approximation for a vertical well when the reservoir height is almost constant in the near well region.

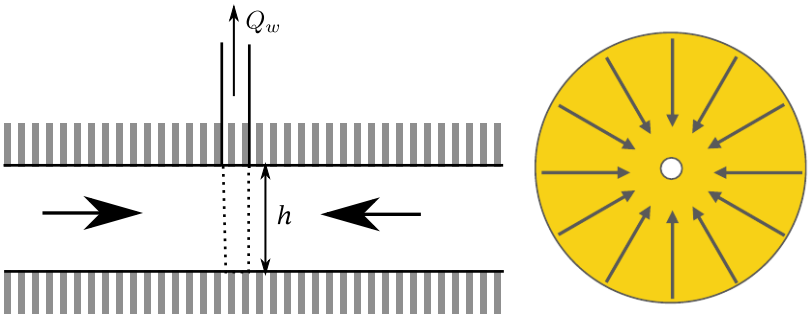


Figure 6.5: Vertical fully penetrating well in a reservoir of constant thickness h.

Such a situation is illustrated in Fig. 6.5. Here, the dashed line indicates the perforated part of the well, of height h equal the height of the reservoir. With a constant porosity and permeability field this idealized example yields horizontal radial flow.

Starting with the diffusivity equation, Eq. (3.52), we can reformulate it for radial flow:

$$\eta \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial p}{\partial r}\right) = \frac{\partial p}{\partial t} \quad . \tag{6.1}$$

As pressure diffuse relative fast, we can assume a steady state solution close to the well-bore. At steady state, the time derivative in Eq. (6.1) is zero, yielding the equation

$$\frac{\partial}{\partial r}\left(r\frac{\partial p}{\partial r}\right) = 0 \quad . \tag{6.2}$$

Integrating this equation once gives

$$\frac{\partial p}{\partial r} = \frac{p_c}{r} \quad , \tag{6.3}$$

where p_c is a constant. To determine this constant, we will investigate the fluid flow right at the interface between the well and the reservoir. The well surface towards the reservoir rock has an area $A = 2\pi rh$, where h is the height of the perforated part of the well, here assumed to be equal the height of the reservoir. Then the Darcy velocity at the rock surface is

$$q = -\frac{Q_w}{A} = -\frac{Q_w}{2\pi rh} \quad , \tag{6.4}$$

where Q_w is the is the volumetric flow rate in the well at reservoir conditions. By convention, we use a positive Q_w for production. Note the distinction between the volumetric flow rate Q_w and the Darcy velocity given by q.

From Darcy's law, Eq. (3.19), now using cylinder coordinates, we get

$$-\frac{k}{\mu}\frac{\partial p}{\partial r} = q = -\frac{Q_w}{2\pi rh} \quad . \tag{6.5}$$

Combining Eq. (6.5) with Eq. (6.3) we see that the constant p_c is given as

$$p_c = \frac{Q_w \mu}{2\pi kh} \quad . \tag{6.6}$$

This constant, which has dimension Pa, is called the *characteristic pressure*, and is an important constant in radial flow.

Characteristic pressure

Let r_w be the well-bore radius, and integrate the pressure from the well-bore and to a distance r into the reservoir. Using Eq. (6.3) we then get

$$\int_{p_w}^{p(r)} dp = \int_{r_w}^{r} \frac{p_c}{r} dr$$
$$p(r) - p_w = p_c \left(\ln(r) - \ln(r_w)\right) = p_c \ln\left(\frac{r}{r_w}\right) \quad . \tag{6.7}$$

Rearranging, the radial pressure distribution $p(r)$ is given by the equation

$$p(r) = p_w + p_c \ln\left(\frac{r}{r_w}\right) \quad , \tag{6.8}$$

where r is the radial distance from the well, p_w is the pressure in the well, r_w is the well radius, while p_c is the characteristic pressure as given by Eq. (6.6). This equation is well known from well-testing.

You will find a more in-depth derivation of these well-pressure equations in *Lecture notes in well-testing:* `https://folk.ntnu.no/carlfrbe/books/wellTesting.pdf`.

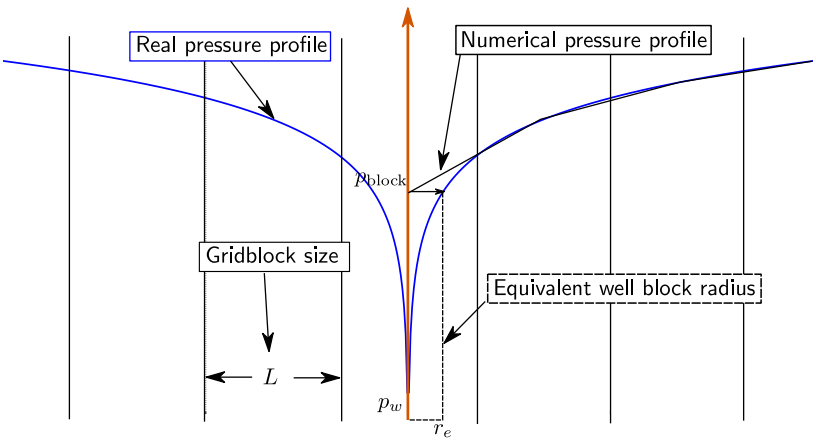


Figure 6.6: An illustration of the radial pressure profile around a well together with the numerical pressure profile for the grid. The block pressure p_b and equivalent radius r_e are indicated in the figure.

We will now use the radial pressure distribution for steady-state radial flow, as given by Eq. (6.8), to derive a coupling between the well pressure p_w and the well block pressure p_b. The well block pressure is the numerical pressure in the grid cell containing the well, and used for to calculate inter-cell flows from the two-point flux approximation (Sec. 7.2.1). Both the well pressure and block pressure are indicated in Fig. 6.6. This figure also indicate the numerical pressure profile; the pressures in the grid cells surrounding the grid block containing the well.

We are seeking an well inflow model of the form

$$Q_w = \lambda T_{wb}(p_b - p_w) \quad , \tag{6.9}$$

where λ is the phase mobility (in single phase the reciprocal of the viscosity μ), while T_{wb} is the well connection transmissibility factor.

As indicated in Fig. 6.6, there exists a radius r_e such that the flowing grid block pressure p_b is equivalent to the pressure $p(r_e)$ given by Eq. (6.8). This radius r_e is called the equivalent well-block

radius. From Eq. (6.8), the well block pressure $p_b = p(r_e)$ is related to the well pressure p_w as

$$p_b = p(r_e) = p_w + p_c \ln\left(\frac{r_e}{r_w}\right) \quad . \tag{6.10}$$

The equivalent well-block radius is sometimes called the pressure equivalent radius.

Writing out the characteristic pressure p_c, then restructuring Eq. (6.10) gives

$$Q_w = \frac{2\pi kh}{\mu}\left(\ln\left(\frac{r_e}{r_w}\right)\right)^{-1}(p_b - p_w) \quad . \tag{6.11}$$

Thus we see that the flow into a well from a grid block is determined by the pressure difference between the grid block pressure p_b and the well pressure p_w. With the form given by Eq. (6.9) we have $\lambda = 1/\mu$, while

$$T_{wb} = \frac{2\pi kh}{\ln\left(\frac{r_e}{r_w}\right)} \quad . \tag{6.12}$$

The only unknown still to be determined in the above equation is the equivalent well-block radius r_e.

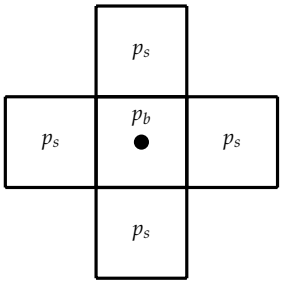


Figure 6.7: A two-dimensional grid around a central cell with a well in the middle.

We will only derive r_e for a regular grid where cell-dimensions in x and y direction are equal; $\Delta x = \Delta y$. To determined r_e, we start with the assumption that Eq. (6.8) holds, and consider the two-dimensional grid around a block with a well in the middle, as indicated in Fig. 6.7. Let p_b denote the pressure in the central cell. Assuming symmetry, we can assume that the four surrounding cells have the same pressure p_s.

Start with the general equation for single phase flow, as given by Eq. (3.36), but here extended with a sink/source term consisting of the volumetric flux times the density giving the mass flux $Q_w\rho$:

$$\nabla \cdot \left(\frac{k\rho}{\mu}\nabla p\right) = \frac{\partial}{\partial t}(\phi\rho) + \delta Q_w \rho \quad . \tag{6.13}$$

Here δ is the Dirac-delta function. Assuming constant density ρ, permeability k and viscosity μ, and further assuming steady-state conditions so that the time derivative becomes zero, we can simplify to

The Dirac-delta function is zero everywhere except at a single point, and the integral over the entire space is equal to one.

$$\frac{k}{\mu}\nabla^2 p = \delta Q_w \quad . \tag{6.14}$$

Using the centered second-difference quotient, Eq. (4.44), for the second derivative in two dimensions, we have

$$\begin{aligned}\frac{\partial^2 p}{\partial x^2}(x,y) + \frac{\partial^2 p}{\partial y^2}(x,y) \simeq \\ \frac{p(x+\Delta x, y) - 2p(x,y) + p(x-\Delta x, y)}{\Delta x^2} \\ + \frac{p(x, y+\Delta y) - 2p(x,y) + p(x, y-\Delta y)}{\Delta y^2} \quad .\end{aligned} \tag{6.15}$$

When $\Delta x = \Delta y$ and all pressure in all the surrounding cells are equal p_s, as indicated in Fig. 6.7, we then have

$$\frac{\partial^2 p}{\partial x^2}(x,y) + \frac{\partial^2 p}{\partial y^2}(x,y) \simeq \frac{4p_s - 4p_b}{\Delta x^2} \quad . \tag{6.16}$$

Using the approximation Eq. (6.16) in Eq. (6.14), we then have

$$\frac{k}{\mu}\frac{4p_s - 4p_b}{\Delta x^2} = \frac{Q_w}{\Delta x \Delta y h} \quad . \tag{6.17}$$

As we discretize, the point source given by the Dirac delta function is transformed to a division by the cell volume. As $\Delta x = \Delta y$, we then get

$$\frac{4kh}{\mu}(p_s - p_b) = Q_w \quad . \tag{6.18}$$

From Eq. (6.8) we have

$$p_s = p(\Delta x) = p_w + p_c \ln\left(\frac{\Delta x}{r_w}\right) \quad . \tag{6.19}$$

Combining Eq. (6.19) with Eq. (6.18), and writing out the characteristic pressure p_c, we get

$$p_b = p_w + \frac{Q_w \mu}{2\pi kh}\left(\ln\left(\frac{\Delta x}{r_w}\right) - \frac{\pi}{2}\right) = p_w + p_c \ln\left(\frac{e^{-\frac{\pi}{2}}\Delta x}{r_w}\right) \quad . \tag{6.20}$$

Assuming Eq. (6.8) holds, then for a well in the center of a squared grid cell of side-length Δx we have shown that

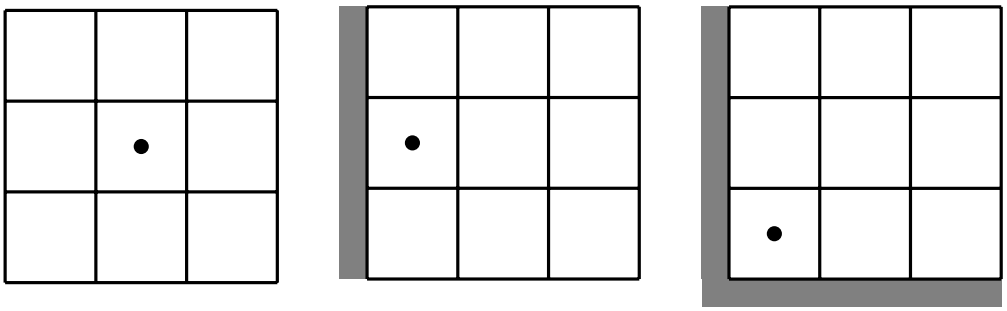
Equivalent well-block radius

$$r_e = e^{-\frac{\pi}{2}}\Delta x \quad . \tag{6.21}$$

This was shown in (Peaceman, 1978), where Peaceman also showed that he obtained similar results numerically for a repeated five-spot pattern of injectors and producers. These results were extended in (Peaceman, 1983), where it was shown that for a well perforated vertically in the center of a grid block with side-lengths Δx and Δy in an anisotropic permeability field he equivalent well-block radius is given by

$$r_e = 0.28\,\frac{\sqrt{\Delta x^2\sqrt{k_y/k_x} + \Delta y^2\sqrt{k_x/k_y}}}{(k_y/k_x)^{\frac{1}{4}} + (k_x/k_y)^{\frac{1}{4}}} \quad . \tag{6.22}$$

Note that $e^{-\frac{\pi}{2}}\sqrt{2} \approx 0.294$, so Eq. (6.22) is only approximately equal to to Eq. (6.21) when $\Delta x = \Delta y$ and $k_x = k_y$. Eq. (6.22) is the formula used in OPM-Flow when calculating T_{wb}.



$r_e \simeq e^{-\frac{\pi}{2}}\Delta x$ $\quad$ $r_e \simeq e^{-\frac{2\pi-\ln(4)}{3}}\Delta x$ $\quad$ $r_e \simeq e^{-\frac{2\pi-\ln(20)}{2}}\Delta x$

Figure 6.8: Figure indicating the well placement close to no-flow boundaries (indicated by thick gray lines), and the associated equivalent well-block radius r_e.

Some examples of pre-factors for wells close to no-flow boundaries are given in (Kuniansky and Hillestad, 1980). Three of them

are shown in Fig. 6.8. We see that for a well placed in the corner of the grid will have a pre-factor $e^{-\frac{2\pi-\ln(20)}{2}} \simeq 0.193$, a difference from an internal point with pre-factor $e^{-\frac{\pi}{2}} \simeq 0.208$. While such changes might be detectable, they are comparatively small compared to changes due to different placement of the well within the grid block.

To check the inflow performance in OPM-Flow, we will consider a two-dimensional example. We created a 9×9 grid, with a producer in each corner and an injector in the middle grid cell, as illustrated in Fig. 6.3. The model has a thickness of Δz =10.0 m, horizontal dimensions of $\Delta x = \Delta y$ =10.0 m, and a permeability of 100.0 mD. The wells have inner diameter of 0.2 m. To write out information on the wells, we use the following keyword before the first time step, and terminate the same keyword right after the first time step:

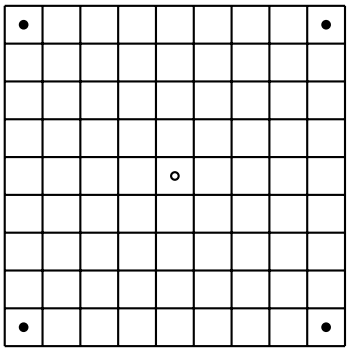


Figure 6.9: A 9×9 grid with a producer in each corner and an injector in the middle.

```
RPTSCHED
  'WELLS=1' 'WELSPECS' FIP=1 /

-- This keyword defines times to when the simulation
-- shall advance. One can use both dates and times.

TSTEP
 0.01 /


RPTSCHED
  'NOTHING' /
```

This keyword add information to the print file, which has the output extension *.PRT. In the print file we find the following output:

```
-------------------------------------------------------------------------------
:   WELL   :     GRID     : CMPL:  CENTRE : OPEN: SAT :  CONNECTION :   INT   :
:   NAME   :     BLOCK    : NO# :  DEPTH  : SHUT: TAB :   FACTOR*   :  DIAM   :
:          :              :     :  METRES :     :     :   CPM3/D/B   : METRES :
-------------------------------------------------------------------------------
: INJW     :    5,  5,  1 :   1 :  1005.0 : OPEN:  1  :   17.944893 :0.200000:
-------------------------------------------------------------------------------
: PROD4    :    9,  9,  1 :   1 :  1005.0 : OPEN:  1  :   17.944893 :0.200000:
-------------------------------------------------------------------------------
: PROD3    :    1,  9,  1 :   1 :  1005.0 : OPEN:  1  :   17.944893 :0.200000:
-------------------------------------------------------------------------------
: PROD2    :    9,  1,  1 :   1 :  1005.0 : OPEN:  1  :   17.944893 :0.200000:
-------------------------------------------------------------------------------
: PROD1    :    1,  1,  1 :   1 :  1005.0 : OPEN:  1  :   17.944893 :0.200000:
-------------------------------------------------------------------------------
```

Thus the simulator has calculated the connection factor 17.944 cP m^3 d/bar. Converting this to SI units, this corresponds to 2.078×10^{-12} m^3.

When the well is in the center of the grid cell, then the OPM-Flow simulator calculates the well connection factor T_{wb} using the equivalent radius r_e as given by Eq. (6.22). Since we have constant grid

cell sizes of $\Delta x = \Delta y$ =10.0 m and constant permeability, we get

$$r_e = 0.28 \frac{\sqrt{\Delta x^2 \sqrt{k_y/k_x} + \Delta y^2 \sqrt{k_x/k_y}}}{(k_y/k_x)^{\frac{1}{4}} + (k_x/k_y)^{\frac{1}{4}}} = 0.28 \frac{10\,\text{m}\sqrt{2}}{2} \simeq 1.98\,\text{m} \quad . \tag{6.23}$$

The well connection factor, as given by Eq. (6.12), can then be calculated as (100 mD = 9.87×10^{-14} m^2):

$$T_{wb} = \frac{2\pi kh}{\ln\left(\frac{r_e}{r_w}\right)} = \frac{2\pi \cdot 9.87 \times 10^{-14}\,\text{m}^2 \cdot 10\,\text{m}}{\ln\left(\frac{1.98\,\text{m}}{0.1\,\text{m}}\right)} = 2.077 \times 10^{-12}\,\text{m}^3 \quad . \tag{6.24}$$

We see that this equals the number used by OPM-Flow.

When the well is no longer in the center of the grid cell, no longer vertical, no longer straight, etc., the calculation of the equivalent radius becomes complicated. It is quite common to use special stand-alone software for calculating the well connection factor, and then use this externally calculated well connection factor instead of letting OPM-Flow calculate the connection factor from the well radius. The well connection factor can be included in the `COMPDAT` keyword. Including the well connection factor as shown below would yield the same solution as a well diameter of 0.2 m.

```
COMPDAT
-- Item #:
-- 1  2   3 4 5  6   7  8 9
 'PROD1'  1 1 1 1 'OPEN' 1* 17.944893 /
 'PROD2'  9 1 1 1 'OPEN' 1* 17.944893 /
 'PROD3'  1 9 1 1 'OPEN' 1* 17.944893 /
 'PROD4'  9 9 1 1 'OPEN' 1* 17.944893 /
 'INJW'       5 5 1 1 'OPEN' 1* 17.944893 /
/
```

6.4 Exercises

Exercise 6.1 Compile the Flow input deck from the chapter above, and run the simulation case. Create a Python script to compare the pressure from the numerical solution to the analytical solution. The basic data is given in Table 5.1.

a) Open the simulation results in ResInsight. Plot the water injection rate and the average reservoir pressure over time. Write a python script to plot the same results.

b) Coarsen the reservoir grid, first to 50, then to 10 grid cells. Observe and discuss how the grid size affects the resulting pressure distribution.

	SPE Metric	SI
l	200 m	200 m
k	100 mD	1.0×10^{-13} m^2
μ	1 cP	1×10^{-3} Pa s
ϕ	0.2	0.2
c_t	1×10^{-4} bar^{-1}	1×10^{-9} Pa^{-1}
p_l	200 bar	2×10^7 Pa
p_r	100 bar	1×10^7 Pa

Table 6.1: Basic data for example.

Exercise 6.2 Extend the 1D grid to a quadratic 2D grid as indicated in Fig. 6.3. The number of grid cells in the figure is just indicative; your model should have 100×100 cells with a side-length of 200 m. The initial pressure is 150 bar, while the injection pressure is 200 bar and the production pressure is 100 bar. Run the model until steady state.

a) Investigate the pressure profile in ResInsight.

b) Use the `ecl` python library to plot the pressure along the diagonal between the wells.

c) Change the well control to injection and production of $25\,\mathrm{m}^3/\mathrm{d}$. Plot the pressure along the diagonal. Optional: Can you describe the pressure profile using the infinite acting line source solution for drawdown and the superposition principle.

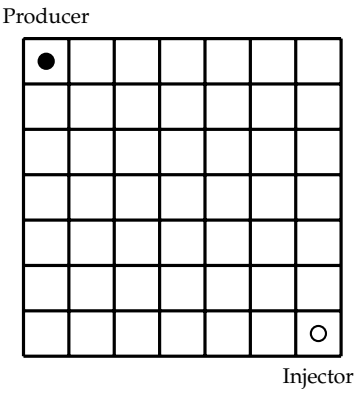


Figure 6.10: Figure indicating the grid cells for the 2D square. Remember that the number of grid cells in the figure are just for illustration purposes.

Exercise 6.3 In this exercise we will investigate grid size sensitivity, the effect of wells placed off center in cells, and wells placed in neighbor cells.

Consider a reservoir of size $1890 \times 1890 \times 50$ m. Permeability is 100 mD, and porosity is 0.2. The reservoir depth (top of reservoir) is at 1500 m, and initial pressure (at reference depth 1525 m) is 200 bar. The reservoir is water filled.

All wells are vertical and completed in the whole thickness of the reservoir, and have a diameter of 0.2 m

A water injector is placed at the (x,y) coordinate (135 m,1755 m), and is injecting at a constant bottom hole pressure (BHP) of 300 bar and a maximum rate of $1500\,\mathrm{m}^3/\mathrm{d}$.

We will consider a number of producer placements. All producers will produce at a constant BHP of 170 bar and maximum rate of $10\,000\,\mathrm{m}^3/\mathrm{d}$.

a) A single producer is placed at (945 m,945 m). Compare the production rate for four grid call sizes: 270×270 m, 90×90 m, 30×30 m, and 10×10 m, and inspect the pressure field in ResInsight.

b) A single producer is placed at (985 m,985 m). Compare the production rate for three grid call sizes: 90×90 m, 30×30 m, and 10×10 m. Also compare with the rates in a)

c) A single producer is placed at (985 m,905 m). Compare the production rate for the three grid call sizes in b).

d) Two producers are placed at (945 m,945 m) and (945 m,1035 m). Increase the maximum injection rate to $300\,\mathrm{m}^3/\mathrm{d}$ and compare the production rate for the three grid cell sizes in b).

e) Comment on the results from a)–d), and draw conclusions.

Notes:

- An input file, `CENTER90.DATA`, for the 90×90 m case in a) is provided in the repository mentioned in Section 2.5.
- A python script, `centerplot.py`, for plotting results in a) can also be found in the repository.
- Running OPM-Flow with default settings will produce some artifacts at early times, both for production and pressure profiles. Run flow as:
  ```
  flow CENTER90.DATA --tolerance-mb=1e-10
  --initial-time-step-in-days=0.01
  ```
  The last option is to get some short initial time steps for plotting on the coarsest grids.
- **Optional:** Run OPM-Flow as
  ```
  flow CENTER90.DATA
  ```
  investigate the pressure field in ResInsight and identify the artifacts

7
Grids in reservoir simulation

> Agricultural commodities are intrinsically renewable, whereas nonagricultural commodities intrinsically depletable. (...) The revenues from agricultural commodities are therefore predominantly a return on past investment and current work. In contrast, minerals are valuable over-and-above the investment and work needed to extract them. Agricultural commodities, in short, are less subject to plunder than are minerals.
>
> Paul Collier - The plundered planet

In this chapter we will define grids and their constituents. Further, we will extend the presented one-dimensional discretisation of flow equations to more complex grids than the regular grids we have looked at in the previous chapters. In particular, we will introduce *finite volume methods* which is the fundamental discretisation in most reservoir simulation codes.

7.1 Grid

Grid

A *grid* (or a mesh) is a division of a geometric domain into (simple) subobjects, and is a common technique for numerical methods in many sciences. In our case the geometric domain is the reservoir. The subobjects are defined as *cells*. It is also common to call the grid cells for *control volumes*, as conservation laws can be applied to these discrete volumes which are covering the geometric domain. Another term on common use is *grid blocks*.

Grid cells

Grid generation is the initial step for numerical modeling. We have already seen an example of a grid of a spatial geometry when we discretized a one-dimensional model as a pseudo two-dimensional grid as depicted in Fig. 4.9. A grid might refer to a discretization of the space consisting of both the time and spatial domain, as visualized for the aforementioned one-dimensional model in Fig. 4.8. However, in reservoir simulation a grid usually refer to the spatial domain only. In reservoir simulation we thus use grids to refer to the tessellation of the geometric domain consiting of the subsurface reservoir. The simulation progresses through time

in steps, and the time step lengths are determined based on the dynamic state of the simulation and not set in advance on a time-grid.

The main geological features of a reservoir are vertical changes in lithology (rock types), which gives rise to horizontal surfaces, and faults, which give rise to vertical (or inclined) surfaces (see Fig. 8.2). These two main geological features guide the division of the reservoir into grid cells. Additionally we have finer structures that may or may not be resolved, such a horizontal changes in rock properties, e.g., channels with different properties than the surrounding background material.

See Sec. 8.2 for a more throughout discussion on geological features and the framework model.

There is usually a small part of a larger connected porous medium that is of primary interest for our simulations. For hydrocarbon reservoirs we are mostly interested in the part containing hydrocarbons, but this part is interconnected with a much larger porous medium, including the aquifer below where the hydrocarbons have migrated upwards into the trap which is our current reservoir. While the hydrocarbon filled porous medium is of primary interest, part of the aquifer is commonly also included into the grid for better handling of the boundary conditions. This aquifer can be very large in comparison with the hydrocarbon reservoir, so representing the full aquifer with the same degree of accuracy as the hydrocarbon reservoir would therefore be computationally heavy. As the flow in the aquifer is single phase and the driving mechanism is fluid expansion due to pressure depletion (or, when water injection increases the pressure of the reservoir, fluid compression due to pressure increase), the flow process in the aquifer is simple compared to the reservoir. The major part of the aquifer can therefore be represented by boundary conditions without significantly deterring the accuracy.

There are many types of grids used in reservoir simulation. They are typically distinguished by the geometry of the individual grid cells and how these grid cells are interconnected. Regular grids consisting of cuboid grid cells are the simplest version. We have already used such simple grids when treating one-dimensional models earlier. In the next section we will present how to derive flow equations using finite volume methods for general grid cell geometries. We will in particular present the two point flux approximation, which is the most common discretization in commercial codes and what is also used in OPM-Flow. Grids with different geometries for the individual grid cells will be presented in Sec. 7.5.

7.2 Finite volume methods

Up until now we have worked with one dimensional examples. For these examples the finite difference methods work well. When starting to treat more complex grids it is more convenient to use finite volume methods.

In a finite volume method, time is discretized into time-steps Δt, and all flows are assumed piece-wise constant in time, i.e., constant

Flow is assumed constant during each time step

during each time step.

As in Eq. (3.30), the fluid mass is given by an integral over the cell volume as

$$m_i = \int_{V_i} \rho\phi \, dV \quad . \tag{7.1}$$

Here m_i is the fluid mass inside cell i, and V_i is the cell volume. We denote by Δm_i the change in fluid mass during the time step Δt. Note that Δm_i would be the change in fluid mass whether m_i is the fluid mass or total mass inside cell i, as the mass of the solid rock can be considered constant.

Conservation of mass applied directly to each grid cell, assuming piece-wise constant rates, gives

Finite volume methods are mass conservative.

$$\Delta m_i = -\Delta t \int_{A_i} \rho q \cdot \vec{n} \, dA - \Delta t \, \rho Q_{wi} \quad , \tag{7.2}$$

where the left hand side is the change of mass inside cell i, Q_{wi} is the rate of flow from the cell into wells, while the right hand side integral, which runs over the bounding area of the cell, is the net mass flow rate out of the cell into adjacent cells. As before, and $\vec{n}$ is the outward pointing unit normal on the surface.

Let the vector $j = \rho q$ denote the mass flux density (mass flow per area). The integral in Eq. (7.2) can be split into a sum over the cell faces (see Fig. 7.1):

$$\frac{\Delta m_i}{\Delta t} = -\sum_j J_{ij} - \rho Q_{wi} \quad , \tag{7.3}$$

where J_{ij} is the mass flow from cell i to cell j defined as

$$J_{ij} = \int_{A_{ij}} j \cdot \vec{n} \, dA = \int_{A_{ij}} \rho q \cdot \vec{n} \, dA \quad . \tag{7.4}$$

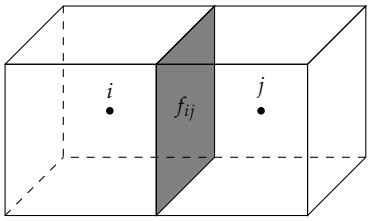


Figure 7.1: Two cells i and j, with the face f_{ij} between the two cells indicated by the gray surface.

Since the mass transport on each side of a surface is just the inverse of each other, $J_{ij} = -J_{ji}$, total mass of each component is always conserved irrespective of how the flux q is discretized. Thus for finite volume methods no fluids are artificially created or removed due to discretization errors.

For multiphase flow (and compositional simulations) mass conservation, as given by Eq. (7.2), is applied separately for each component. We will only treat single phase flow in this chapter.

Properties are assumed constant within a grid cell

In general all properties except pressure are assumed constant in each grid cell, and fluid properties are calculated assuming thermodynamic equilibrium at a pressure equal to the pressure in the center of the cell. The integral Eq. (7.1) is approximated by

Cells are assumed to be in thermodynamic equilibrium.

$$m_i = \rho_{\mathrm{EOS}}(p_i) \, \phi_i(p_i) \, V_i \quad , \tag{7.5}$$

where V_i is the volume of grid cell i, p_i is the pressure at the *center* of the grid cell, and $\rho_{\mathrm{EOS}}(p_i)$ is the fluid density determined by an *equation of state*.

Equation of state

Each grid cell is assigned a single value for (reference) porosity, ϕ_{0i}, and formation (rock) compressibility, $c_{\phi i}$, so that

$$\phi_i(p_i) = \phi_{0i} \exp\left(c_\phi(p_i - p_r)\right) \approx \phi_{0i}\left(1 + c_{\phi i}(p_i - p_r)\right) \quad , \tag{7.6}$$

where p_r is a reference pressure. A similar model is usually also applied for the density of water

$$\rho_{\text{EOS}}(p) = \rho_w(p) = \rho_{w0}\left(1 + c_w(p - p_r)\right) \quad , \tag{7.7}$$

where ρ_w is the density of water and c_w is the compressibility of water. When using the `CO2STORE` keyword in OPM-Flow for CO_2 sequestration, both the CO_2 phase density and the water phase density are calculated from the cell pressure, the amount of dissolved components, and the temperature. For hydrocarbon reservoirs we typically do not include temperature in our simulations. In the black oil model (see Section 10.2) the density for hydrocarbons is given by tabulated values for the formation volume factors for oil B_o and gas B_g depending on pressure only:

$$\rho_{\text{EOS}}(p) = \rho_p(p) = \frac{\rho_{ps}}{B_p(p)} \quad , \tag{7.8}$$

where the subscript p represents either oil o or gas g.

7.2.1 *The two point flux approximation*

We will discretize the flux over the cell boundaries by finite differences. In this case the finite volume method is often referred to as conservative finite difference method or simply finite difference method. When using finite difference methods to discretize the flux over the boundaries, the two methods are so related that they can easily be confused. In particular, for Cartesian grids the finite volume method reduces to the finite difference method when one use finite differences to discretize the flux over the boundaries. In the petroleum literature the distinction is sometimes not clarified, while sometimes the finite volume methods are called finite difference control volume methods, e.g., in (Aziz, 1993; Ertekin et al., 2001; Abou-Kassem et al., 2006). We will stick with the common name in mathematics and physics, namely finite volume, when we want to highlight the distinction.

In general, the permeability is not isotropic and can not be represented by a scalar. Typically permeability is lower in the direction orthogonal to the bedding (vertical permeability) than in the parallel directions (horizontal permeability), and is sometimes represented by scalar values for the vertical and horizontal directions only. The permeability is actually a tensor, $\mathbf{K}$, and can be represented by a 3×3 matrix. The matrix will include off-diagonal cross-terms if the coordinate system is not aligned with the characteristic directions for permeability (i.e., the eigenvectors of $\mathbf{K}$)

Visit Sec. 3.2.2 for a more in-depth discussion on permeability and the Darcy equation.

An anisotropic permeability field $\mathbf{K}$ is represented by a symmetric 3×3 matrix

$$\mathbf{K} = \begin{bmatrix} k_{xx} & k_{xy} & k_{xz} \\ k_{yx} & k_{yy} & k_{yz} \\ k_{zx} & k_{zy} & k_{zz} \end{bmatrix} \quad . \tag{7.9}$$

Recall from Sec. 3.2.2 that the permeability tensor is symmetric, i.e., $k_{ij} = k_{ji}$.

According to Darcy's equation, Eq. (3.24), we can express the mass flux as

$$\vec{j} = \rho\vec{q} = -\frac{\rho}{\mu}\mathbf{K}\cdot\nabla p \quad , \tag{7.10}$$

which, when substituted in (7.4), gives

$$J_{ij} = -\int_{A_{ij}} \frac{\rho}{\mu}\left(\mathbf{K}\cdot\nabla p\right)\cdot\vec{n}\, dA \quad . \tag{7.11}$$

In the two point flux approximation we have

$$J_{ij} = -\lambda T_{ij}\left(p_j - p_i\right) = \lambda T_{ij}\left(p_i - p_j\right) \quad , \tag{7.12}$$

where $\lambda = \rho/\mu$ is the mobility[1], and T_{ij} is the transmissibility. Mobility is a fluid property, while transmissibility is a function of permeability and grid geometry.

[1] For multiphase flow the mobility also include relative permeability. The mobility is usually evaluated in the upstream block. Mobility weighting is discussed in detail in Chap. 10

The transmissibility is found by applying a finite difference approximation to the pressure gradient. It would be tempting to use a centered difference coefficient

$$\frac{\partial p}{\partial x} \simeq \frac{p_j - p_i}{\vec{d}_{ij}\cdot\hat{x}} \quad , \tag{7.13}$$

where $\vec{d}_{ij}$ is the vector between cell centers, and $\hat{x}$ is the unit vector in the x direction. Unfortunately, this is not consistent with a different permeability between the grid cells. We will instead approximate the gradient on each side of the interface. We then have one equation for each side of the interface, given as

$$\begin{aligned} p_x &\simeq p_i + \left(\frac{\partial p}{\partial x}\right)_i \vec{d}_{ix}\cdot\hat{x} + \left(\frac{\partial p}{\partial y}\right)_i \vec{d}_{ix}\cdot\hat{y} + \left(\frac{\partial p}{\partial z}\right)_i \vec{d}_{ix}\cdot\hat{z} \\ p_x &\simeq p_j + \left(\frac{\partial p}{\partial x}\right)_j \vec{d}_{jx}\cdot\hat{x} + \left(\frac{\partial p}{\partial y}\right)_j \vec{d}_{jx}\cdot\hat{y} + \left(\frac{\partial p}{\partial z}\right)_j \vec{d}_{jx}\cdot\hat{z} \end{aligned} \quad , \tag{7.14}$$

where $\vec{d}_{ix}$ is the vector from grid center i to the center of the interface, equivalently, $\vec{d}_{jx}$ is the vector from grid center y to the center of the interface, while p_x is the pressure at the center of the interface as shown in Fig. 7.2. If we use a local coordinate system in

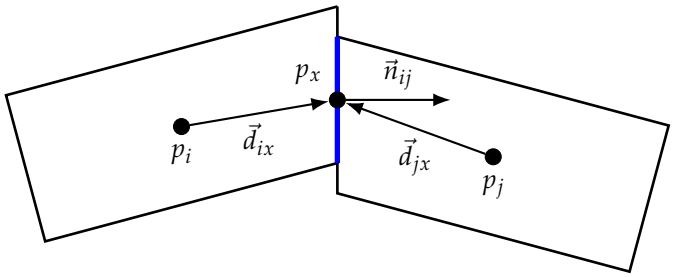


Figure 7.2: Definition of points and vectors used in the two point flux approximation. The flux from cell i (left) to cell j (right) through the common face (in blue) is proportional to the pressure difference $(p_i - p_j)$. Note that $\vec{n}_{ji} = -\vec{n}_{ij}$.

each block where the local cell direction $\hat{x}_i$ is parallel to $\vec{d}_{ix}$, i.e., $\hat{x}_i = \vec{d}_{ix}/\left|\vec{d}_{ix}\right|$, then $\vec{d}_{ix}\cdot\hat{y}_i = 0 = \vec{d}_{ix}\cdot\hat{z}_i$. Thus the two last terms in the first equation in Eq. (7.14) become zero (equivalently for the second equation in Eq. (7.14)), so that

$$\begin{aligned} \left(\frac{\partial p}{\partial x}\right)_i &\simeq \frac{p_x - p_i}{\vec{d}_{ix}\cdot\hat{x}_i} \\ \left(\frac{\partial p}{\partial x}\right)_j &\simeq \frac{p_x - p_j}{\vec{d}_{jx}\cdot\hat{x}_j} \end{aligned} \quad . \tag{7.15}$$

As seen from Eq. (7.11), we need an approximation for the vector $\mathbf{K}^i \cdot \nabla p$, where $\mathbf{K}^i$ is the permeability tensor in cell i. If the permeability tensor has nonzero off diagonal elements, all elements of this vector will depend on $\partial p/\partial y$ and $\partial p/\partial z$, while Eq. (7.15) gives us $\partial p/\partial x$ only. The off diagonal matrix elements k_{xy} and k_{xz} are zero if $\hat{x}_i$ and $\hat{x}_j$ are eigenvectors of the corresponding permeability tensors $\mathbf{K}^i$ and $\mathbf{K}^j$ in their respective cells. Equivalently, the off-diagonal element k_{xy} and k_{xz} are zero if the vector $\vec{d}_{ix}$ from the grid cell center to the interface is parallel to a characteristic direction for permeability.

When deriving the two-point flux approximation, it is assumed that grid cells are aligned with the characteristic directions for permeability.

In the following, we will assume that the vector $\vec{d}_{ix}$ from the grid cell center to the interface is parallel to a characteristic direction for permeability, so that we have

$$\left(\mathbf{K}^i \cdot \nabla p\right) \cdot \vec{n}_{ij} \simeq k^i_{xx} \left(\frac{\partial p}{\partial x}\right)_i \hat{x}_i \cdot \vec{n}_{ij} + k^i_{yy} \left(\frac{\partial p}{\partial y}\right)_i \hat{y}_i \cdot \vec{n}_{ij} + k^i_{zz} \left(\frac{\partial p}{\partial z}\right)_i \hat{z}_i \cdot \vec{n}_{ij} \quad , \tag{7.16}$$

where the subscript i of the derivatives indicate that these are in the local coordinate system, and $\hat{y}_i$ and $\hat{z}_i$ are the two orthogonal directions in the local coordinate system. In general, we need to include the two last terms in this equation. However, for most grids the values of $\hat{y}_i \cdot \vec{n}_{ij}$ and $\hat{z}_i \cdot \vec{n}_{ij}$ are small. To simplify our notation, we will therefore omit them in the following, i.e., we assume that $\hat{y}_i$ and $\hat{z}_i$ are orthogonal to the normal vector $\vec{n}_{ij}$. If we assume that the pressure gradient is constant on the interface, we can apply Eq. (7.15) and Eq. (7.16) to the integral Eq. (7.11) to get

$$\begin{aligned} J_{ij} &\simeq -\left(\frac{\rho}{\mu}\right)_w k^i_{xx} \frac{(p_x - p_i)\ \vec{d}_{ix} \cdot \vec{a}_{ij}}{|d_{ix}|^2} \\ J_{ji} &\simeq -\left(\frac{\rho}{\mu}\right)_w k^j_{xx} \frac{(p_x - p_j)\ \vec{d}_{jx} \cdot \vec{a}_{ji}}{|d_{jx}|^2} \end{aligned} \quad , \tag{7.17}$$

where the subscript w indicates a weighting of the mobility ρ/μ for the two grid cells (see Chap. 10 for more details), while

$$\vec{a}_{ij} = \int_{A_{ij}} \vec{n}_{ij}\, dA \quad , \tag{7.18}$$

that is the interface area as a vector. Note that $\vec{a}_{ij} = \vec{n}_{ij} \left|A_{ij}\right|$ for planar surfaces A_{ij}.

We can express Eq. (7.17) in terms of cell–to–face transmissimilities

Cell–to–face transmissimilities

$$\begin{aligned} T_{ix} &= k^i_{xx} \frac{\vec{d}_{ix} \cdot \vec{a}_{ij}}{\vec{d}_{ix} \cdot \vec{d}_{ix}} \\ T_{jx} &= k^j_{xx} \frac{\vec{d}_{jx} \cdot \vec{a}_{ij}}{\vec{d}_{jx} \cdot \vec{d}_{jx}} \end{aligned} \quad , \tag{7.19}$$

to get the following simplified equations:

$$\begin{aligned} J_{ij} &\simeq -\lambda T_{ix} \left(p_x - p_i\right) \\ J_{ji} &\simeq -\lambda T_{jx} \left(p_x - p_j\right) \end{aligned} \quad , \tag{7.20}$$

where $\lambda = (\rho/\mu)_w$ is the mobility term. Using the second equation above we can find an expression for the unknown p_x as

$$p_x = p_j - \frac{J_{ji}}{\lambda T_{jx}} = p_j + \frac{J_{ij}}{\lambda T_{jx}} \quad , \tag{7.21}$$

as $J_{ij} = -J_{ji}$. Filling this expression for p_x into the first equation in Eq. (7.20) we have

$$J_{ij} = -\lambda T_{ix} (p_x - p_i) \tag{7.22}$$

$$J_{ij} = -\lambda T_{ix} \left(p_j - p_i + \frac{J_{ij}}{\lambda T_{jx}} \right) \tag{7.23}$$

$$J_{ij} \left(1 + \frac{T_{ix}}{T_{jx}} \right) = -\lambda T_{ix} (p_j - p_i) \tag{7.24}$$

$$J_{ij} = -\lambda \frac{1}{\frac{1}{T_{ix}} + \frac{1}{T_{jx}}} (p_j - p_i) \quad . \tag{7.25}$$

Thus, we get two point flux equation Eq. (7.12) with

$$\frac{1}{T_{ij}} = \frac{1}{T_{ix}} + \frac{1}{T_{jx}} \quad . \tag{7.26}$$

Inverse transmissibility is analogous to a resistance in an electrical network

If we interpret *inverse* transmissibility as a resistance, we see that the resistance between cell centers i and j is the sum of the resistance from center i to the common cell face and the restistance from the face to cell center j. This is in complete analog with an electric resistor network. For simple cartesian grids with cuboid grid cells we can alternatively define *cell transmissibilites*, $T_i^x = \frac{1}{2} T_{ix}$ analogous to Eq. (7.19), by replacing $\vec{d}_{ix}$ with the cell length $2d_{ix}$. We see from Eq. (7.26) that the cell–to-cell transmissibility is the harmonic average of the cell transmissibilities

$$\frac{1}{T_{ij}} = \frac{1}{2} \left(\frac{1}{T_i^x} + \frac{1}{T_j^x} \right) \quad . \tag{7.27}$$

Cell transmisibilities can however not be defined for general cell shapes.

7.3 *Numerical errors due to the grid*

When deriving the two point flux approximation, we made two assumptions with relation to the grid; that the grid cells are aligned with the characteristic directions for permeability, and that the lines joining the grid cell centers are orthogonal to the common faces. Grids where the lines joining the cell centers are orthogonal to their respective common cell faces are called orthogonal. An orthogonal grid where the cells are aligned with the characteristic directions for the permeability are called **K**-orthogonal. If the permeability is isotropic, i.e., we have the same permeability k in all directions, then any orthogonal grid is **K**-orthogonal. Fig. 7.3 shows examples of non-**K**-orthogonal and **K**-orthogonal grids for an anisotropic permeability field.

K-orthogonal grid

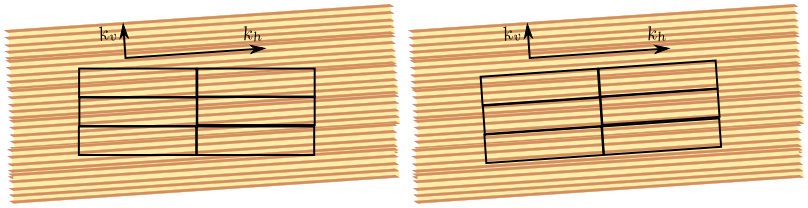


Figure 7.3: Illustration of **K**-orthogonality. In a reservoir zone comprising a layered facies (rock type), the permeability is higher parallel to the layering than orthogonal. A grid that does not follow the layering (left) is not **K**-orthogonal.

Errors that are not reduced by refining the grid can be called 0-th order errors, thus the error is of order $\mathcal{O}((\Delta x)^0) = \mathcal{O}(1)$ and not dependent on grid size Δx. Grids that are not **K**-orthogonal can introduce 0-th order errors in the two point flux approximation.

Another grid related error source is the grid orientation effect, which is a numerical artifact that also exist on **K**-orthogonal grids. Assume we have two injectors and one producer, as illustrated in Fig. 7.4. We observe that the length of the discretized path for the fluids moving parallel to the grid (the blue arrows) equals the distance between this injector and the producer. In contrast, the fluids transported from the lower injector to the producer will have a discretized path that is $2/\sqrt{2} = \sqrt{2}$ times longer than the distance between the injector and producer. For this reason, changes in fluid saturations are propagated faster along grid axes than diagonally. Thus this error is only an issue for for multiphase flow. The errors are not reduced by grid refinement, thus they are 0-th order errors.

Grid orientation effects

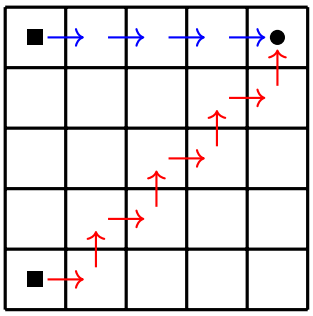


Figure 7.4: Illustration of flow paths parallel and diagonal to the grid. The rectangles indicates injectors, while the circle indicate a producer. With the two-point flux approximation, the fluid fronts will move faster parallel to the grid (indicated by blue arrows) than diagonally (indicated by red arrows).

Most simulators include an option for more advanced multi-point flux approximations. These reduce the errors induced by non-**K**-orthogonal grids and the grid orientation effect. In spite of this, multi-point flux approximations are not commonly used.

7.4 *Grids and geology*

The subsurface permeability fields can usually be treated as slowly varying within geological objects[2], such as sand bodies, but may change abruptly across zone boundaries, faults, and between geological objects. This needs special attention since when we formulate a problem in terms of differential equations it is implicitly assumed that the underlying fields are slowly varying. Abrupt changes in material properties typically have to be handled through interface- or boundary- conditions. The finite volume methods are however, in part, based on an *integral* formulation, and the permeability field is assumed to be piecewise constant. The two point flux approximation is derived assuming a piecewise linear pressure drop inside the grid cells, and works well as long as abrupt changes in the permeability are confined to cell boundaries. It is thus preferable that zone boundaries, faults, and boundaries between geological objects coincide with cell faces. It follows that these geological features should define a framework for grid generation.

[2] See Sec. 8.2 for the defintion of a geological object. In a properly built reservoir model, the property fields within geological objects are slowly varying after geological upscaling, see Sec. 8.3

Zone boundaries, faults, and boundaries between geological objects, should define a framework for gridding.

An additional reason for preferring the grids to follow zone boundaries and faults is that these often are associated with thin

zones with reduced permeability; known as zone barriers and the fault damage zone. These features are usually so thin that including them explicitly would severely decrement numeric stability and increase the running time of the simulations. The effect of these small volumes of reduced permeability at grid cell boundaries can however easily be implemented in the two point flux approximation by simply adding an extra resistance in series with the cell-to-face resistances in Eq. (7.26). A further discussion of this, and how the effect is represented in many reservoir simulators through transmissibility multipliers, can be found on page 125.

Thin zones of reduced permeability can be represented as an additional resistance in series.

7.5 Common grid types

We will now present the most common grid types used in reservoir simulation. Note that OPM-Flow only supports cornerpoint grids.

7.5.1 Cornerpoint grid

The cornerpoint (or pillar) grid is a legacy grid format that is supported by all reservoir simulators, and it is the most widely used grid type. In many companies corner point grids is the only grid type used in official models. The grid type is illustrated in Fig. 7.5.

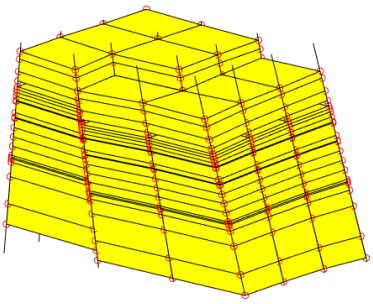


Figure 7.5: A cornerpoint grid (From `https://www.sintef.no/projectweb/mrst`). At the top of the model we can observe the horizontal zig-zag pattern representing a fault.

Grid cells are ordered (labeled) using a regular i, j, k scheme. All grid cells have 8 corners, but the corners may be collapsed vertically so that actual grid cells can have between 4 and 8 effective corners. Cells with 4 corners are totally collapsed with zero volume. All corners are defined on pillars. The corners of grid cell (i, j, k) lies on pillars $(i-1, j-1)$, $(i-1, j)$, $(i, j-1)$, and (i, j). Pillars are normally not vertical, but normal to the layering, as this gives better **K**-orthogonality.

We distinguish between a logical grid, which in essence is only a way to number cells and corners, but can be thought of as a regular grid as illustrated in the left figure in Fig. 7.6. In the corner point grid, logical neighbors do not need to share a common face, and the height of the grid corners along the pillars could be different for different grid cells, as illustrated in the right figure in Fig. 7.6. Each logical corner in the regular i, j, k grid is shared by 8 grid cells. Different depths for the same logical corner can be used for representing faults and non-reservoir gaps.

It would seem straight forward to represent 2D fault patterns by defining pillars along the fault planes. However, the resulting grids will not be **K**-orthogonal, so zig-zag faults (see Fig. 7.5) is actually preferred. Compartment volumes may be severely affected by this approximation. Well paths close to faults may also need to be moved in the model so that they penetrate the correct compartments. Complex fault patterns in 3D are impossible to represent without an additional zig-zag representation in the vertical. Unless the faults are completely sealing, the calculation of across fault communication can in this case at best be characterized as compli-

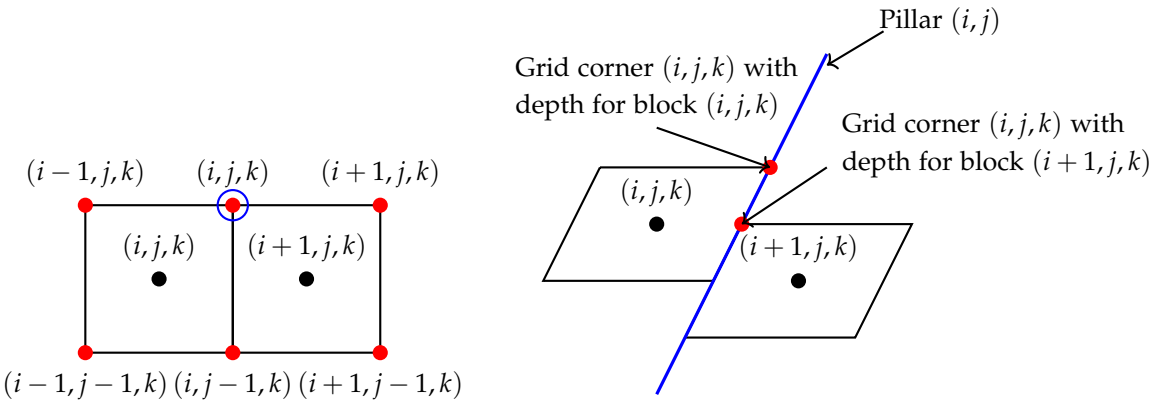


Figure 7.6: A constant j-slice through a corner point grid (right), and its corresponding logical grid seen from above (left). Black circles are cell centers, and red circles are grid corners. The (i,j)-pillar is shown in blue. The corner point grid have cells which are logical neighbors, but have different depths for the same logical corners.

cated.

Cornerpoint grids can not be adjusted to well paths. As a result, it is in most cases impossible to get accurate reservoir–well couplings with this grid type. This problem is discussed in more detail in Sec. 8.7.

7.5.2 2D Voronoi grid

Voronoi grids have grid-cells with varying number of corners. Similar to the cornerpoint grids, the corners are defined on pillars. The term 2.5D PEBI grid is often used for this grid type, which is a generalization of the cornerpoint grid where the pillar pattern have general shapes while maintaining horizontal **K**-orthogonality.

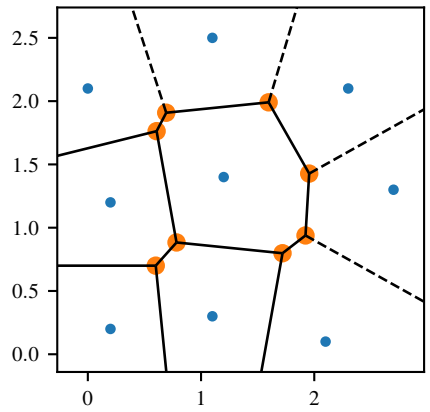


Figure 7.7: Example of a 2D Voronoi grid. The blue points are the cell centers (seeds), given as input. The lines divide the 2D space into sets of points closest to each of the cell centers. The orange points would be the positions of the pillars in the 3D grid.

Voronoi construction

A Voronoi construction builds grid cells starting from grid cell centers. This is similar to the point-centered grids presented in Sec. 4.4. Let $\{c_i\}$ be a set of grid cell centers. The (two-dimensional) space is then divided into regions V_i defined by the closest cell center c_i:

$$V_i = \left\{ x \in \mathbb{R}^2 \mid \|x - c_i\| < \|x - c_j\| \quad \forall j \neq i \right\} \quad . \tag{7.28}$$

Separation of the plane $\mathbb{R}^2$ by this method is indicated in Fig. 7.7. As a direct consequence of the Voronoi construction, for all points inside a grid cell the cell center of this grid cell is the closest cell center, as seen in Fig. 7.7. Furthermore, the cell edges are midway between the respective cell centers.

A Voronoi tessellation is called *regular* if all grid vertices connect exactly three grid edges. The Voronoi tessellation in Fig. 7.7 is regular. A regular Voronoi tessellation has a one-to-one map to a Delaunay triangulation, where the triangles are constructed by connecting the three cell centers around the different grid edges in the Voronoi tessellation.

A Delaunay triangulation is a triangulation where the circumcircle of any triangle only contains the vertices of that given triangle.

For Voronoi grids, the Voronoi construction is used for the pillar pattern. The pillar position for a 2D cross-section is indicated by the orange points in Fig. 7.7. This grid construction guarantees **K**-orthogonality if the permeability is horizontally isotropic.

The major strength of this grid type is the ability to accurately represent fault patterns, albeit only in 2D. Further, this grid type can adjust the grid to straight well paths. Furthermore, the grid orientation effect (as illustrated in Fig. 7.4) is also minimized if hexagonal grid cells are used in combination with radial grids around wells.

Major weak points for this type of grids are the lack of vertical grid refinements, representation of 3D faults, and that the grid cannot adapt to complex well paths. Furthermore, the complex grid structure is computationally hard for linear solvers.

7.5.3 Unstructured grids

As indicated by the name, unstructered grids may have any shape grid-cells, but they are often built using a 3D Voronoi construction. The obvious strength of this grid type is the ability to accurately represent reservoir compartments, as for instance defined by complex fault patterns, in 3D. Even more important, as well design becomes increasingly complex, unstructured grids can easily be adjusted to even the most complex well path.

Major weak points for unstructured grids are a complex description of cell properties from the underlying geo-models, the unstructured grids are missing **K**-orthogonality (if not accounted for in the Voronoi definition), and as with the 2D Voronoi grids they can be computationally challenging for the linear solvers.

7.5.4 Local grid refinements

As indicated by the von Neumann stability analysis, see Sec. 5.3, the accuracy of our simulations are linked to grid size. As changes in pressure and saturation are most rapid close to wells, increasing the grid resolution around wells would increase the stability and accuracy of our simulations. Examples of local grid refinements around wells are illustrated in Fig. 7.8. In addition to grid refinement using the same grid type, also radial grid refinements

are common, as seen in the right grid in Fig. 7.8. As the fluid flow around a well is close to radial, the radial grid refinements better captures the flow directions.

Most reservoir simulators have capabilities for handling local grid refinements[3]. There are two main options for how to numerically handle grid refinements: The remainder of the original grid and the refined part of the grid can be solved independently, with boundary conditions between the two grids being updated at each time step. Alternatively, both grids can be solved simultaneously. The benefit of solving the flow equations independently is computational efficiency, e.g., different time step size for the two grids. The drawback is that the boundary conditions between the two grids could lead to unstability and convergence problems.

[3] Note that the current version of OPM-Flow (2021-04) does not support local grid refinements.

While it is possible to refine grids by simply dividing the grid cells into smaller cells and inherit the cell properties from the original coarser grid cells, grid refinements should preferably be resampled on the geo-model to ensure correct up-scaling. In cases with large geological heterogeneity on several scales one should consider building separate fine scale geo-models in the near-well region.

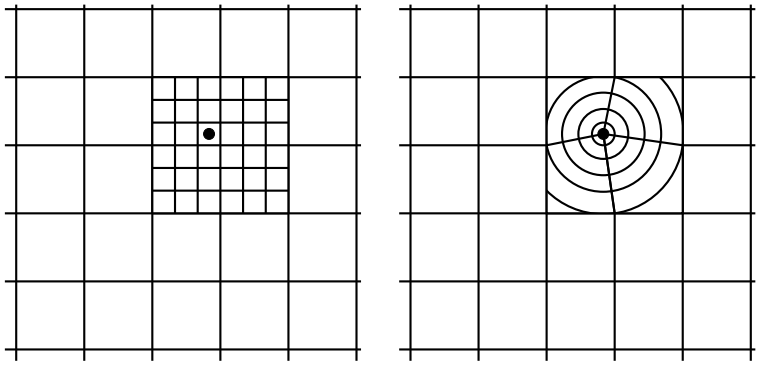


Figure 7.8: Examples of local grid refinements around wells. The wells are indicated by a black filled circle.

7.6 *Index ordering and preconditioning*

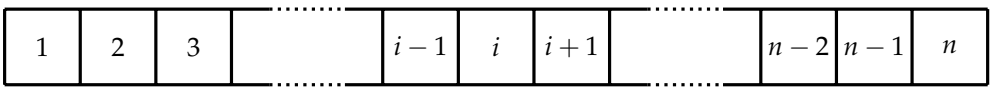


Figure 7.9: Indexing of grid cells in the pseudo-1D slab.

One-dimensional grid

In this section we will briefly present how the ordering of grid cells gives different matrices in the implicit method. Let us start by returning to the one-dimensional grid, as given by Fig. 7.9. From Eq. (5.27), we saw that the implicit method for this grid could be written on matrix form as

$$\mathbf{A}\vec{P}^{t+\Delta t} = \vec{P}^{t} + \vec{e} \quad , \tag{7.29}$$

where the unknowns are the pressures at time-step $t + \Delta t$, i.e. the vector $\vec{P}^{t+\Delta t}$. This matrix form then represents n-equations of the

form

$$a_i P_{i-1} + b_i P_i + c_i P_{i+1} = d_i \quad , \tag{7.30}$$

where by convention we have dropped the $t + \Delta t$ superscript. Comparing to Eq. (5.26), we see that $a_i = -\alpha$, $b_i = 2\alpha + 1$ and $c_i = -\alpha$, where $\alpha = \eta \Delta t / \Delta x^2$, except for the boundaries. Thus the matrix is of form

$$\mathbf{A} = \begin{bmatrix} b_1 & c_1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & \cdots & 0 & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & b_{n-2} & c_{n-2} & 0 \\ 0 & 0 & 0 & 0 & \cdots & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & 0 & 0 & 0 & \cdots & 0 & a_n & b_n \end{bmatrix} \quad . \tag{7.31}$$

As seen, the matrix is given by a compact band of non-zero elements around the diagonal. The compact band consist of three diagonal strips, represented by the a_i, b_i and c_i elements respectively. Such a compact diagonal sparse matrix can be inverted effectively.

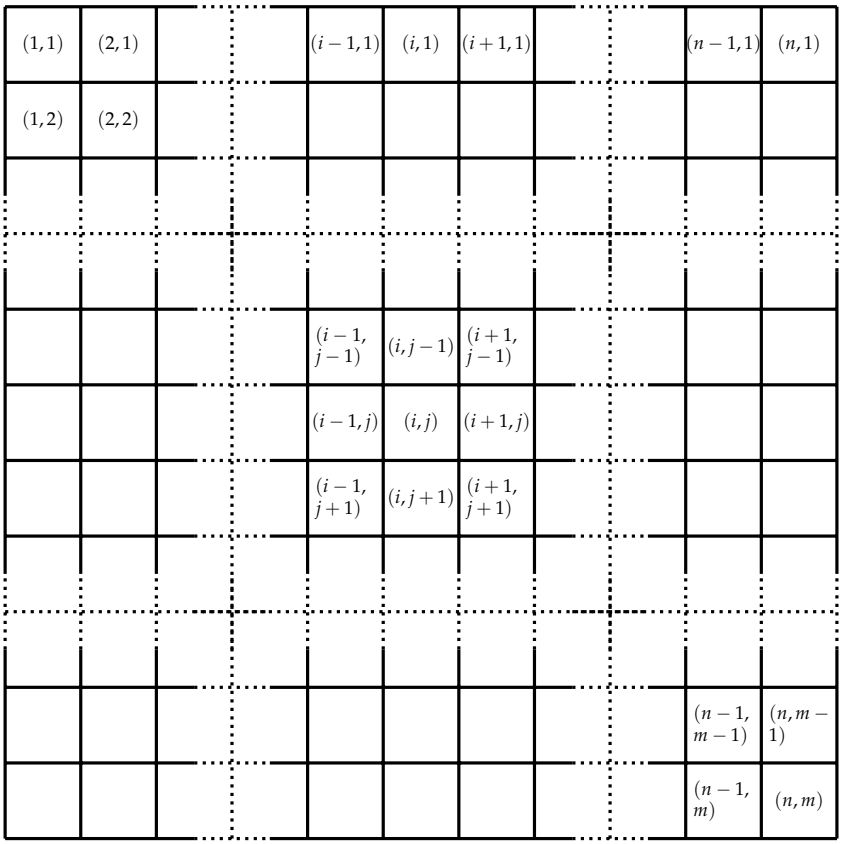


Figure 7.10: Indexing of grid cells in a two-dimensional grid.

Two-dimensional grid

Let us now consider a two-dimensional grid, as illustrated in Fig. 7.10. Here we represent the grid cells by a two-tuple (i, j). When we want to solve the implicit case, and therefore want our system of equations written in matrix form, we need to write the pressures as a vector. A vector is by definition one dimensional. We

thus need to represent all the cells in the two-dimensional grid by a one-dimensional vector. There are many possibilities for the map from two dimensions to one dimension, and which map is used will affect how easy it is to solve the system. We will return to this soon.

For now we stay with the two-dimensional representation given by the two-tuples (i, j). We then get pressure equations on the form

$$e_{i,j}P_{i,j-1} + a_{i,j}P_{i-1,j} + b_{i,j}P_{i,j} + c_{i,j}P_{i+1.j} + f_{i,j}P_{i,j+1} = d_{i,j} \quad . \tag{7.32}$$

This will again result in a matrix where the elements of $b_{i,j}$ are on the diagonal, with $a_{i,j}$ and $c_{i,j}$ on each side of the diagonal. Also $e_{i,j}$ and $f_{i,j}$ will be aligned diagonally, however the distance from the central diagonal will depend on the size of the grid, as we will illustrate in the following.

For a specific example, consider a 3×4-sized grid, where we order the indices as indicated in Fig. 7.11. This will give a matrix as

$$\mathbf{A}_{3,4} = \begin{bmatrix} b_1 & c_1 & 0 & f_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & f_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & a_3 & b_3 & 0 & 0 & f_3 & 0 & 0 & 0 & 0 & 0 & 0 \\ e_4 & 0 & 0 & b_4 & c_4 & 0 & f_4 & 0 & 0 & 0 & 0 & 0 \\ 0 & e_5 & 0 & a_5 & b_5 & c_5 & 0 & f_5 & 0 & 0 & 0 & 0 \\ 0 & 0 & e_6 & 0 & a_6 & b_6 & 0 & 0 & f_6 & 0 & 0 & 0 \\ 0 & 0 & 0 & e_7 & 0 & 0 & b_7 & c_7 & 0 & f_7 & 0 & 0 \\ 0 & 0 & 0 & 0 & e_8 & 0 & a_8 & b_8 & c_8 & 0 & f_8 & 0 \\ 0 & 0 & 0 & 0 & 0 & e_9 & 0 & a_9 & b_9 & 0 & 0 & f_9 \\ 0 & 0 & 0 & 0 & 0 & 0 & e_{10} & 0 & 0 & b_{10} & c_{10} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & e_{11} & 0 & a_{11} & b_{11} & c_{11} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & e_{12} & 0 & a_{12} & b_{12} \end{bmatrix} . \tag{7.33}$$

Here we see that the central diagonal and its two neighbors are similar to the one-dimensional case: For each row in the grid you get a corresponding part on the diagonal that is organized just as the matrix we obtained for the one-dimensional case. In addition to these central one-dimensional replicas, you have two diagonal strips further out representing the grid cells neighbors in the vertical direction in the grid.

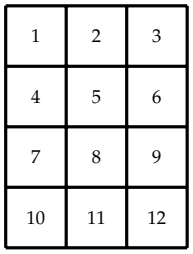


Figure 7.11: Indices of grid cells in a 3×4 sized grid.

Let us order the same 3×4-sized grid in the other direction, starting with the longest axis first. Then the indices will be as indi-

cated in Fig. 7.12. This will give a matrix as

$$\mathbf{A}_{4,3} = \begin{bmatrix} b_1 & c_1 & 0 & 0 & f_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & 0 & f_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & 0 & 0 & f_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & a_4 & b_4 & 0 & 0 & 0 & f_4 & 0 & 0 & 0 & 0 \\ e_5 & 0 & 0 & 0 & b_5 & c_5 & 0 & 0 & f_5 & 0 & 0 & 0 \\ 0 & e_6 & 0 & 0 & a_6 & b_6 & c_6 & 0 & 0 & f_6 & 0 & 0 \\ 0 & 0 & e_7 & 0 & 0 & a_7 & b_7 & c_7 & 0 & 0 & f_7 & 0 \\ 0 & 0 & 0 & e_8 & 0 & 0 & a_8 & b_8 & 0 & 0 & 0 & f_8 \\ 0 & 0 & 0 & 0 & e_9 & 0 & 0 & 0 & b_9 & c_9 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & e_{10} & 0 & 0 & a_{10} & b_{10} & c_{10} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & e_{11} & 0 & 0 & a_{11} & b_{11} & c_{11} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & e_{12} & 0 & 0 & a_{12} & b_{12} \end{bmatrix} . \tag{7.34}$$

We observe that the matrix $\mathbf{A}_{3,4}$ has its non-zero elements in a narrower band than the matrix $\mathbf{A}_{4,3}$. If you solve these two matrices by Gaussian elimination, you will need fewer operations for the matrix with a narrower band. Thus, it is then computationally easier to solve $\mathbf{A}_{3,4}$ than $\mathbf{A}_{4,3}$.

1	5	9
2	6	10
3	7	11
4	8	12

Figure 7.12: Indices when ordered with the longest axis first in a 3×4 sized grid.

Observe that we have the same elements in the two matrices $\mathbf{A}_{3,4}$ than $\mathbf{A}_{4,3}$ (the names are not the same though). Assume we can obtain one matrix from the other by linear transformations, which we can obtain by multiplying by another matrix $\mathbf{P}$:

$$\mathbf{P}\mathbf{A}_{4,3} = \mathbf{A}_{3,4} \quad . \tag{7.35}$$

Thus, when we have the matrix form of our implicit equation as

$$\mathbf{A}\vec{P}^{t+\Delta t} = \vec{P}^t + \vec{e} \quad , \tag{7.36}$$

there might exist a matrix $\mathbf{P}$ so that $\mathbf{PA}$ is easier to solve. We then solve the equivalent system

$$\mathbf{P}\mathbf{A}\vec{P}^{t+\Delta t} = \mathbf{P}\left(\vec{P}^t + \vec{e}\right) \quad . \tag{7.37}$$

Such a matrix $\mathbf{P}$ that makes the system easier to solve is called preconditioner.

Preconditioner

Similar effects as discussed above also holds for sparse matrix solvers; matrices with a narrower band of non-zero elements are easier to solve. Thus you want to organize your indices so that they give non-zero elements as close to the main diagonal as possible. This means that in general you should start indexing in the shortest direction first, and the subsequently longer directions. However, there are other indexing strategies that might yield even narrower bands. You might further ease your numerical solution by multiplying your system by a preconditioner.

Development of preconditioner matrices is a scientific subject still under rapid development, and new procedures for preconditioning systems are still added to software for reservoir simulation. The Constraint Pressure Residual (CPR) preconditioner, as formulated in (Scheichl et al., 2003), was recently added to OPM-Flow. The

CPR preconditioner can be invoked using a keyword, while default is a LU-factorization (Rasmussen et al., 2021).

7.7 Exercises

Exercise 7.1 Derive the matrix for a three dimensional grid, e.g. a grid of dimensions $3 \times 3 \times 3$. How many non-zero diagonal strips do you get in the matrix?

Exercise 7.2 The checkerboard ordering, or A3 ordering, order the cells in a grid by always skipping one cell. An example for the 3×4 grid is shown in Fig. 7.13. Indicate all the non-zero elements in the associated matrix.

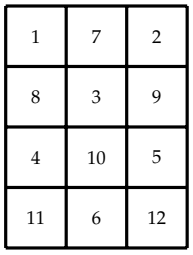


Figure 7.13: A3 ordering of grid cells in a 3×4 sized grid.

8
Building a reservoir model

> Every oil-pool in the world was different – each one a riddle, with colossal prizes for the men who could guess it! (...) He had felt the temptation of grandeur, and knew what it must be to a boy. It was pleasant to have a lot of money; but you must set up a skeleton at the feast, and while you quaffed the wine of success, you must hear a voice behind you whispering, "Memento mori!"
>
> Upton Sinclair, *Oil!*

In this chapter we will *present the main concepts* of reservoir models and reservoir model building. The goal is to summarize what is needed for a reservoir engineer to communicate effectively within a multidisciplinary reservoir modeling group. For a more complete exposition we will recommend the book on reservoir modeling by Philip Ringrose and Mark Bentley.[1]

[1] *"Reservoir Model Design A Practitioner's Guide"* (Ringrose and Bentley, 2015)

Note that the perspectives in this chapter are undoubtedly colored by the fact that the authors have mainly worked with offshore fields in the Norwegian Sea and the North Sea.

The main purpose of building a reservoir model is to capture information of the subsurface in a form that is suitable for making *quantitative* statements about the reservoir and reservoir behavior. The model is based on available subsurface information, such as seismic, well logs, core material, observed fluid contacts, and so on, all with associated uncertainties. The model predictions serve as a basis for managerial decisions, e.g. development plans, well placement targets, optimization of recovery etc. This overall purpose is illustrated in the simple flowchart in Fig. 8.1.

In the context of reservoir simulation, that is the simulation of reservoir fluid flow, the model must contain geological, and other sub-surface, knowledge in a form suitable for creating reservoir simulation model realizations. These realizations are specific reservoir simulator input data that, when run through the simulator, reproduce the behavior of a possible reservoir. Note that since the present text is a book on reservoir simulation, the term "reservoir simulation model", and often even "reservoir model", is used many places instead of the more correct, but more cumbersome, "reservoir

Reservoir simulation model realization

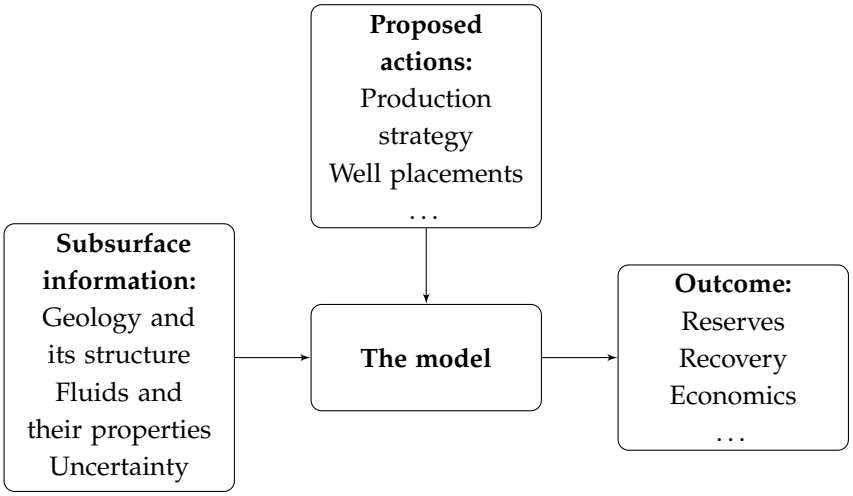


Figure 8.1: Flow chart illustrating the overall purpose of reservoir modeling.

simulation model realization". This usage is consistent with many other text on the subject, but in the general reservoir modeling context it is important to maintain the distinction between "the model", which represent all reservoirs that are consistent with our subsurface knowledge, and a "model realization", which represents a single of these possible reservoirs.

From the introduction we remember that the main model classification is static and dynamic models. The reservoir model has a *static model* that comprise the geological features that do not change during production, and a *dynamic model* which contain the additional features needed for simulating fluid flow during production. Note that when we say that a static model *do not change*, that need not be entirely true, e.g., the pore volume could be allowed to change due to compaction.

Static model

Dynamic model

The static model has a *framework model*, which describes faults and horizons, an *object model*, that describes the spatial distribution of geological objects, and a *property model*, describing the spatial distribution of properties. In addition, the static model will contain a model for initial fluid-contacts. This *contact model* needs to be consistent with the model for fault seal and compartment communication. In order to be able to calculate initial in place hydrocarbon volumes, the static model must also have a model for capillary pressure and a, possibly simplified, fluid model. The fluid model, and models for capillary pressure and relative permeability are, however, normally viewed as being part of the dynamic model.

Framework model

Object model

Property model

Contact model

The dynamic model contains a *fluid model*, which describes the properties of the reservoir fluids, and their spatial distribution (pVT regions). A model for relative permeability and capillary pressure functions, including the spatial distribution of rock-types (saturation-function regions, keyword `SATNUM` in OPM-Flow), is also needed. The coupling between the reservoir and the wells, and the flow in the wells, is described in the *well model*, and in order to correctly predict production from a reservoir with multiple wells,

Fluid model

Well model

the *production strategy* must be described in a robust fashion that can be understood by the reservoir simulator.

Production strategy

8.1 Framework model and compartment communication

The framework model is built with two main goals: to define a set of (non)communicating *compartments*, and to define a reasonable zonation for geological modeling (the object model). It consists of the main faults and zone boundaries, as shown in Fig. 8.2. Often

Figure 8.2: The framework model contain the faults (left) and one zone boundary (right) of the Norne model.

the term "structural model" is used as an alternate name for the framework model. The main data source is seismic, while well observations of faults and zone boundaries are used for anchoring seismic data (in the "time" domain) in space.

The seismic interpretation (Fig. 8.3), which is a representation of faults and horizons, is not to be confused with the framework model. It is, however, the *starting point* for building the framework model. For reservoir modeling purposes it is important to note that

Seismic interpretation $\neq$ Framework model

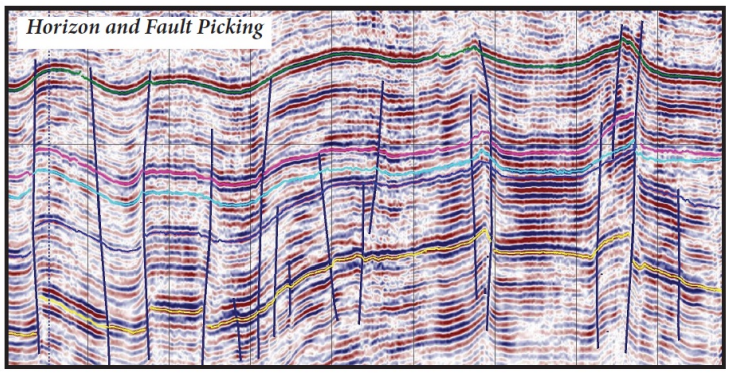


Figure 8.3: Seismic with interpreted horizons and faults. `http://www.enageo.com/seismic_interpretation.html`

seismic data has poor vertical resolution and the data quality degrades near faults. Faults with throw $\leq 10m$ (at best) are invisible on seismic, but can be very important for reservoir flow, even displacing complete zones. If the interpretation is performed in the

"time" domain, then depth conversion will also introduce artifacts. Most modern model work-flows demand that the framework model is consistent and "water tight", i.e. that it divides space into a number of separate unfaulted blocks. A seismic interpretation, and indeed nature itself, usually does not adhere to this restriction. Note that faults are also often only interpreted where the data quality is sufficient, which leaves artificial "holes" close to fault intersections where data quality is degraded.

Note also that the seismic reflectors (horizons) represent sharp changes in seismic properties, such as the speed of sound, and this does not necessarily coincide with the zone boundaries that are needed in reservoir models. As a general rule there are also more zone boundaries than reflectors, and the location of additional zone boundaries and vertical barriers are found on well-logs.

When it comes to zonation, a framework model can be built based on one of two alternative philosophies: chrono-stratigraphy or litho-stratigraphy. The two approaches are illustrated in Fig. 8.4.

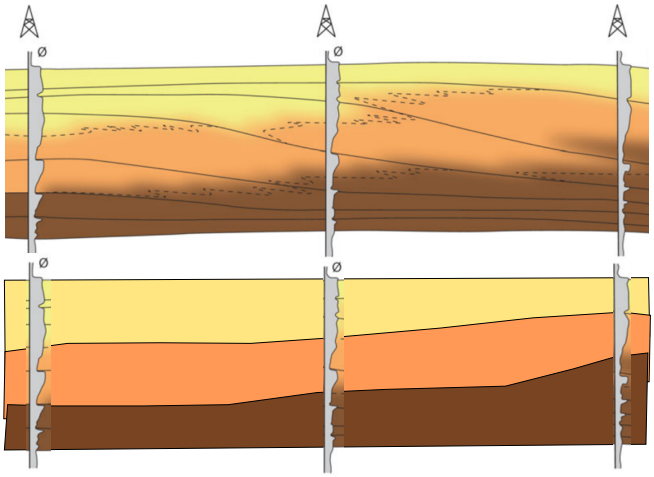


Figure 8.4: Chrono-stratigraphy (top), with zone boundaries defined by geological time (solid lines), vs. litho-stratigraphy (bottom), where zone boundaries are defined at rock-type boundaries.

In a litho-stratigraphic model, the zones are defined based on the rock types (lithography), while in a chrono-stratigraphic model, the zones are defined by events in geological time, i.e. a layer corresponds to a specific time (chronos) of deposition. Very often the lithographic layering will correspond to the seismic horizons, while a chrono-stratigraphic model is generally most suitable both for geological modeling (the object model) and for describing communication in reservoir simulation models. In practice, the difference between the two stratigraphic approaches are not always that clear; Some framework model zone boundaries may be defined based on lithology, while others are based on chronology. Also, the geological events that are associated with chrono-stratigraphic markers are often associated with lithographic changes. Finally, the lithographic makeup within a chrono-stratigraphic zone is often constant over reservoir size distances.

Inter-compartment communication

Inter-compartment communication is the 0-th order dynamic model. In gas reservoirs and high permeable reservoirs a mass-

balance model with compartments as smallest unit is often sufficient for describing the dynamic behavior of the reservoir. Inter-compartment communication is also often the largest uncertainty in the reservoir description. Production data, and well test data, are the main sources of information for reducing this uncertainty, but the inter-compartment communication often remain a large contributor to uncertainty even in fields with a considerable production history.

The communication between compartments is influenced by the reduced permeability near zone boundaries and fault planes. The altered zone is too thin to be represented explicitly in the grid and is modelled as a plane with characteristic property (See Fig. 8.5)

$$\beta = \frac{k_f}{L_f} \quad , \tag{8.1}$$

where k_f is the permeability, and L_f is the thickness, of the altered zone.

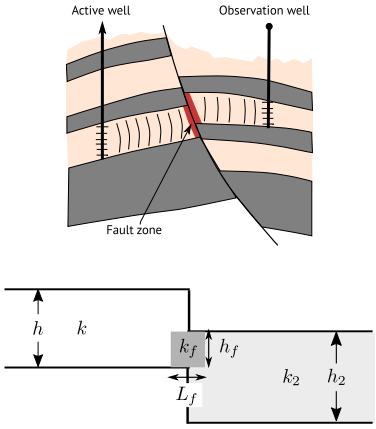


Figure 8.5: Representing compartment communication on a grid.

In the two-point flux approximation the communication between grid-cells is represented as transmissibilities, T_{ij}. In most simulators, including OPM-Flow, the fault properties are not given explicitly, but are input as multipliers M_{ij} on the transmissibilities T^0_{ij} that are calculated based on grid cell geometry and grid cell permeabilities

Transmissibility multiplier

$$T_{ij} = M_{ij} T^0_{ij} \quad . \tag{8.2}$$

The fault geometry, in particular the reduced contact area between grid cells, is taken into accont in T^0_{ij}, while the effect of the reduced permeability in the fault zone is included in the multiplier M_{ij}.[2]

In order to investigate the connection between the fault zone property β_{ij} and the corresponding multiplier, we will consider a simplified geometry. Let us align the grid cells horizontally and consider a fault between two grid cells as shown in the upper figure in Fig. 8.6. We also consider the corresponding grid without the fault as in the lower figure in Fig. 8.6.

[2] Further information on fault multipliers can be found in (Manzocchi et al., 1999).

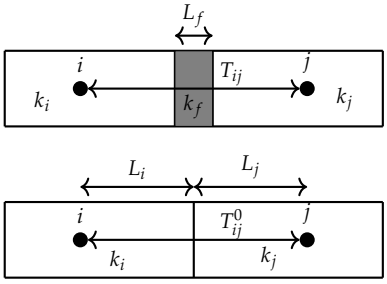


Figure 8.6: Simple representations of two grid cells with and without a fault zone in between. Note that in this figure the fault zone thickness L_f is artificially large for visualization purposes.

The transmissibility is in both cases given by

$$T_{ij} = \frac{A_{ij}}{L_i + L_j} k_e \quad , \tag{8.3}$$

where A_{ij} is the area of the common cell face, and k_e is the effective permeability between the cell centers. The effective permeability without the reduced permeability zone (lower figure 8.6) is

$$k^0_e = \frac{L_i + L_i}{\frac{L_i}{k_i} + \frac{L_j}{k_j}} \quad , \tag{8.4}$$

while the effective permeability including the zone (upper figure 8.6) is

$$k_e = \frac{L_i + L_i}{\frac{L_i - \frac{L_f}{2}}{k_i} + \frac{L_f}{k_f} + \frac{L_j - \frac{L_f}{2}}{k_j}} \quad . \tag{8.5}$$

From (8.2)–(8.5) we get

$$M_{ij} = \frac{k_e}{k_e^0} = \frac{1}{1 + L_f \frac{\frac{1}{k_f} - \frac{1}{2k_i} - \frac{1}{2k_j}}{\frac{L_i}{k_i} + \frac{L_j}{k_j}}} \tag{8.6}$$

In the case of constant grid cell size L and permeability k this is reduced to

$$M_{ij} = \frac{1}{1 + \frac{L_f}{L}\left(\frac{k - k_f}{k_f}\right)} \simeq \frac{L\beta_{ij}}{L\beta_{ij} + k} \quad , \tag{8.7}$$

where the last equality is valid if we treat the fault zone as a plane, $L_f \ll L$. We see from (8.7) that transmissibility multipliers are grid dependent; they vary with grid size through the length L. Under grid refinements where L is reduced, the multipliers M_{ij} must be reduced accordingly. For small multipliers, that is strongly sealing faults with $\beta_{ij} \ll \frac{k}{L}$, the multiplier is proportional to grid size.

Transmissibility multipliers are grid dependent

In addition to a reduced permeability, a fault zone and layers between reservoir zones often have a sealing property known as *capillary seal*. Water wet rock with low permeability and porosity tend to have high water saturations due to capillary pressure. In many cases the hydrocarbon pressure must also exceed a capillary entry pressure in order to flow into, and through, the zone. In any case the permeability of the sealing zone will be phase dependent. There will typically be a significantly reduced, or even zero, permeability for hydrocarbon, while there will be a possibility for water to flow.

Capillary seal

The transmissibility multipliers described above do not account for capillary seal. In most cases it is possible to include the effect by a combination of threshold pressure and directional relative permeabilities, [3] but in some cases an explicit grid representation of the zone may be neded. Since flow through small grid cells often is associated with numerical problems, it is advisable to make the zones artificially thick and adjust the properties of the zone and its neighboring grid cells accordingly.

[3] See the OPM-Flow manual for details on threshold pressure and directional relative permeability. OPM-Flow does currently (2020-10) not support directional relative permeabilities.

8.2 Object model

The basis for all reservoir modeling is the conceptual model. The conceptual model is a mental image of the reservoir shared by everyone involved, and sketches of reservoir sections are highly informative and brings clarity to this mental image. Without a common conceptual model, the professionals from different disciplines are forced to work in separate "silos" and no real communication is possible. The main question that is addressed in the conceptual model is the identification of the fundamental geological reservoir building blocks. These building blocks are called model elements. The conceptual model must also determine the general shape of these model elements, and how they stack up in space. [4] The selection of appropriate model elements is not a job for geologists alone;

Conceptual model

Model elements

[4] In the literature, the term "flow unit" is sometimes used for model element. Here we follow the nomenclature of (Ringrose and Bentley, 2015), this also avoids confusion with other uses of the term flow unit.

Single- and multi-phase flow properties are very important when selecting model elements, and there should be a close connection between model elements and the "rock-types" used for assigning multi-phase properties in the flow-simulation model (`SATNUM` regions).

The object model describes the spatial distribution of geological objects. These objects are associated with the model elements in the conceptual model. Model elements usually form a hierarchy, where one element comprise other elements. As an example (See figure 8.7); the reservoir zones and sub-zones are model elements comprised of various types sands and non-sands, which are also model elements. In the process of creating model realizations, the spatial distribution of objects within a zone is often generated by algorithms in the class of "object modeling", but it should be noted that there is no other connection between these algorithms and the object model. The existence of a geological object model that comprise geological objects which belong to model elements is independent of the algorithms used.

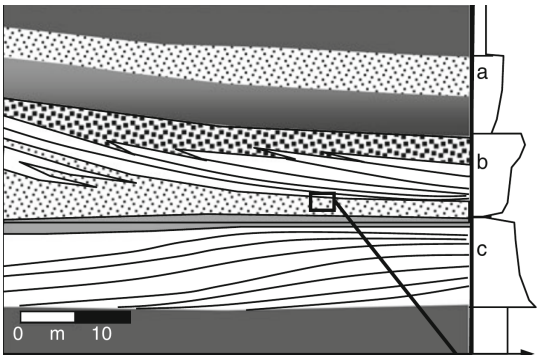


Figure 8.7: A hierarchy of model elements

8.3 Property model

Each geological object in the object model is filled with property fields, i.e. they have properties defined everywhere in space inside the object. The spatial distribution of the property values is determined by the model element the object belongs to, and conditioned to available measurements in wells.

When generating model realizations, the properties are represented on a grid, this implies that the object geometries are also sampled on a grid. For simulation, properties should be slowly varying between grid-cells inside each object, and objects should cover more than a few grid-blocks in each direction. Sadly, the property variation between neighboring grid-cells that belong to the same geological object is often too high in many simulation models used in the industry today. The key for getting this right is proper understanding of the concept of a representative elementary volume (REV) for a model element.

Objects and properties are sampled on a grid.

Properties such as permeability are scale dependent, and only

exist in specific scale ranges known as REV scales. This is illustrated in Fig. 8.8. The REV concept is probably best explained in

Representative elementary volume (REV)

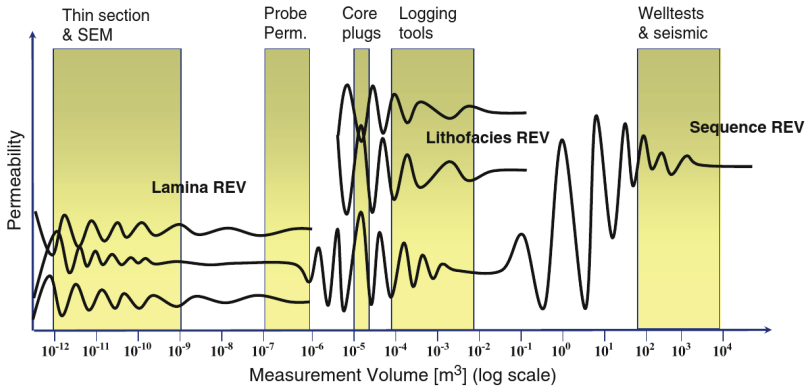


Figure 8.8: The concept of a representative elementary volume (REV)

terms of a moving "average": If we calculate a property for a certain volume, such as the average porosity inside a cube, and that property varies slowly on the characteristic length scale for the volume when the volume is moved around, then the volume is a representative elementary volume for that property. This procedure is performed inside a given object corresponding to a model element, and determines the REV for that model element. The center position of the averaging volume can also be kept fixed, while varying the size, as in Fig. 8.8. In that case, size ranges that show little dependency on size for the calculated property correspond to REV scales. Model elements can be defined on any of these scales.

Ideally we should have full *separation of scales*, defined as

Separation of scales

$$\text{Grid-cell size} \ll \text{REV} \ll \text{Object size} \quad ,$$

in our models. This is, however, often impossible to obtain in practice, and we will have to contend with approximate separation

$$\text{Grid-cell size} \approx \text{REV} < \text{Object size} \quad .$$

Property fields are usually generated using voxel based geostatistics, such as a Gaussian field with a given spatial correlation structure (or more correctly variogram). With these algorithms it is

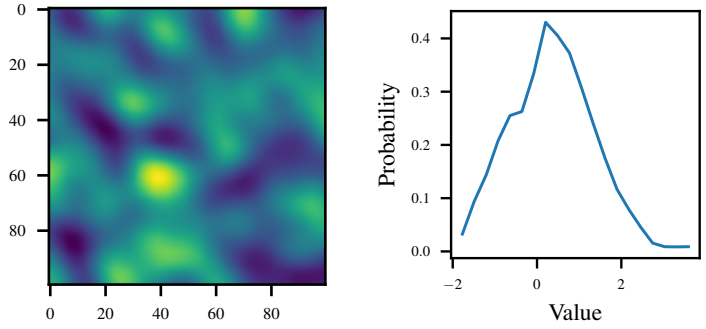


Figure 8.9: A stochastic realization from a gaussian random field with a correlation structure.

Data conditioning

easy to condition to data from well-logs by keeping values in corresponding voxels fixed at measured values. Due to the scale difference between the well-log observation and the property REV, this "hard conditioning" is very often too restrictive. Log-measurements may sometimes also not correspond to any REV at all, in which case they cannot be used as direct observations for conditioning. The alternative "soft conditioning", where voxel values are not fixed but have known probability distributions derived from measurements, is usually most correct, but it is surprisingly seldom used. Note also that permeability is not measured by well logging tools, and hard conditioning to well-log permeability is thus never correct.

The properties the simulator sees are porosity, ϕ, permeability, k_v and k_h,[5] net to gross ratio[6], and rock type numbers[7]. Apart from ϕ it is actualy often not a good idea to model these directly in the geological property model. The question of which properties should be directly simulated by geo-statistical algorithms, and which should be derived based on functional dependencies, are tightly tied to the fact that reservoir modeling always involve up-scaling of properties. In general there are two types of up-scaling involved; geological up-scaling and grid based up-scaling. For property modeling it is geological up-scaling that is relevant. Grid based up-scaling will be discussed separately (page 130).

[5] Alternatively often input as k_h and $\frac{k_v}{k_h}$

[6] The use of NTG $\neq 1$ is discouraged !!

[7] These numbers may be used in conjunction with endpoint scaling of relative permeability, in which case the model also have endpoint values in each grid-cell. For details see the OPM-Flow manual.

Geological up-scaling

Geological up-scaling is the process of incorporating measurements on smaller scales, and geological knowledge, in order to obtain model element properties and property correlations. This includes transforming well-log data, possibly measured on non-REV scales, so that they can be used for conditioning, and the up-scaling of multiphase properties, such as relative permeability (Chapter 12.4.1).

The main concepts of geological up-scaling is best explained by going through a simple example:

Example of geological up-scaling

- At small scales a model element comprise a layered structure of sand and mud with variable sand fraction, as shown in Fig. 8.10.

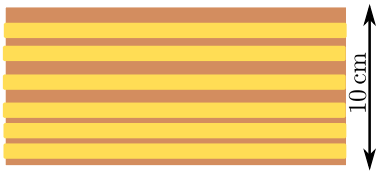


Figure 8.10: At small scale model element comprising a layered structure of sand and mud

- The average porosity, smoothed over 30cm, is available in well-logs.

- Measurements on core material give

	Porosity	Permeability (mD)
mud	0.15	1.5
sand	0.31	900.0

We see that the permeability and porosity of the model element is determined by the sand fraction, s. Thus, it is natural to model the spatial distribution of sand fraction s using geo-statistics, and calculate the porosity and permeability based on it. For the simple

layered system, we have the following expressions (See page 201)

$$\begin{aligned} \text{Porosity:} \quad & \phi = 0.31 \cdot s + 0.15 \cdot (1 - s) \\ \text{Horizontal permeability:} \quad & k_h = (900.0 \cdot s + 1.5 \cdot (1 - s))\,\text{mD} \\ \text{Vertical permeability:} \quad & k_v = \frac{1.5 \cdot 900.0}{1.5 \cdot s + 900.0 \cdot (1 - s)}\,\text{mD} \end{aligned} \tag{8.8}$$

The measured porosity data at the wells are transformed to sand fraction,

$$s_{\text{well}} = \frac{\phi_{\text{well}} - 0.15}{0.16} \quad , \tag{8.9}$$

and used for conditioning.

As mentioned earlier, the well-log data should usually not be used directly as "hard" conditioning data. Based on an analysis of the well data, core material, and the conceptual geological model, a probability model for the sand fraction on the appropriate REV scale

$$p(s^{\text{REV}}_{\text{conditioning}} | s_{\text{well}}) \quad , \tag{8.10}$$

should be developed. For geo-statistical modeling, a correct REV scale variogram must be derived, and conditioning data should be drawn from (8.10). However, how to infer the probability model used for geo-statistical modeling is beyond the scope of the present text.

Grid based up-scaling is the process of transforming properties

Grid based up-scaling

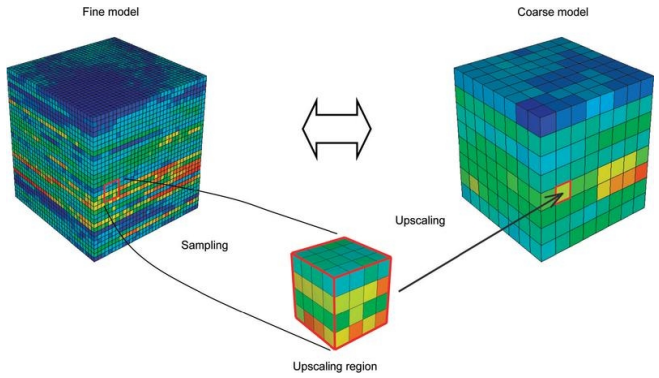


Figure 8.11: The concept og grid based up-scaling. Source: `http://www.epgeology.com/static-modeling-f39/how-upscale-permeability-t6045.html`

that are represented on a fine grid into properties on a coarser grid as shown in Fig. 8.11. Typically there is a need for this up-scaling when the static model realizations[8] are built on a finer scale than what, due to constraints on simulation run-time, can be used for dynamic simulation. This situation is far from ideal, but is very common in practice.

[8] A static model is often called geo-model, and the dynamic model simulation-model.

Additive properties, such as porosity and saturation, can be trivially up-scaled using respectively the volume weighted and pore volume weighted average. A representative up-scaled absolute permeability can usually be found by solving local flow problems with suitable boundary conditions. This is called flow based up-scaling, and is always better than averaging. If the coarse-scale grid does not follow object boundaries, the coarse grid cells may

Flow based up-scaling

contain regions from several geo-objects. Since rock types, with corresponding multiphase properties (relative permeability), are associated with model elements, the assignment of rock types to coarse-scale grid cells is problematic in these cases.

Assigning rock types to coarse-scale grid cells is non-trivial.

8.4 Grids for geomodeling

In most geo-modeling software, the geological object model and property model is modeled in an *unfaulted and uncompacted rectangular* coordinate system conventionally known as the simulation box. The simulation box represent the geometry at the time when the sediments were deposited. In many popular tools, such as Petrel and RMS, the property values in the simulation box are simply moved to cell values in the actual gridded framework model using direct *ijk*-correspondence. This leads to restrictions on the grids that may be used for geological modeling in these tools. Grid cells should be as uniform in size and direction as possible, and artifacts are unavoidable across faults. Alternative and potentially more correct transformations, such as GeoChron (Moyen, 2005), is used in some tools.

The simulation box

8.4.1 Geo-grid vs Simulation-grid compatibility

The sampling of properties on the geo-grid onto the simulation grid is surprisingly often problematic in many software tools. A direct mother–child correspondence is often preferred to minimize these problems. The coarser grid meant for dynamic simulation is considered the mother grid, while a finer child grid used for geological modeling is constructed by refining the mother grid. This approach clearly impose strong limitations on the available grid types. If the simulation box to real geometry transformations use direct *ijk*-correspondence, this, and the need for **K**-orthogonality, will enforce almost rectangular cornerpoint grids. With improved algorithms and software, the mother–child approach with its limitations could become obsolete.

Mother–child grids

8.5 Fluid model

The fluid model comprise the fluid model proper, often referred to as the pVT model, and the fluid–rock model. The pVT model describe the properties of the reservoir fluids, while the multiphase transport properties, which depend both on the fluids and the rock, are described in the fluid–rock model, typically in terms of saturation functions such as relative permeability.

8.5.1 pVT model

In the context of reservoir engineering, the term pVT-model is used for the fluid model proper. Note, however, that this should

be understood as the "thermodynamic model", since it describes much more that the relation between pressure (p), volume (V), and temperature (T). Moreover, the fluid models employed in reservoir simulation are mostly isothermal. The pVT model determines the density, saturation, composition, and viscosity of the different phases as a function of pressure, total composition, and temperature.

Two classes of pVT models are most common in reservoir simulation, "black oil" and "compositional". The *black oil model* has three components (water, oil, and gas), that distribute among three phases (water, oil, and gas). That the components and phases in the black oil model have identical names is admittedly confusing, and the components may be more correctly described as surface-water/oil/gas or produced-water/sales-oil/sales-gas. Invariably the water-phase contains only the water-component, while the oil- and gas-phases contain no water component. The oil- and gas-components split between the oil- and gas-phases dependent on pressure and total composition, and a black oil pVT- model is input to a reservoir simulator in the form of tables. Reservoir simulation with a black oil pVT-model is discussed in detail in chapters 10, and 11. Most compositional models also have three possible phases (water, oil, and gas), but the number of components may vary and they are typically connected to actual chemical constituents. In a compositional model, densities, saturations, phase compositions, viscosities, and other fluid properties are based on equations of state or correlations. As a result, these models are computationally more demanding than the table based black oil model. Compositional models with a large number of components are very heavy in terms of cpu- and wall-clock- time, and the number of components in models used in reservoir simulation are usually around 10 or less.

Black oil model

Compositional model

The data basis for the pVT model is measurements on downhole fluid samples or re-combined surface samples. Both black oil and compositional models are typically based on a compositional model with a large number of components which is tuned in order to reproduce these measurements. This is done in specialist pVT software which is subsequently also used for creating black oil tables. Compositional models for reservoir simulation have a reduced number of components, and the selection of the reduced component set and the parameters in the reduced model is also generated in the pVT software. This process is called component lumping.

Component lumping

Different recovery methods require separate fluid models. In the case of primary depletion or water injection, a black oil model is usually sufficient, while processes which result in larger changes in reservoir fluid composition, such as miscible gas-injection, will require a compositional model. Gas condensate reservoirs also typically require compositional simulation.

The *viscosity correlations* included in pVT software are good, and the viscosity values generated for Black-Oil tables tend to be of suf-

Viscosity correlations

ficient accuracy. However, these correlations are computationally heavy, and compositional simulation usually employ simpler, faster to evaluate, but less accurate, correlations. Compositional simulation also involves a reduced number of components compared to the tuned internal correlations in pVT software. As a consequence, it is important to pay special attention to the tuning of the viscosity correlations for compositional simulation. Note that some of the more popular viscosity correlations are unstable outside the parameter range used for tuning. The calculated viscosity values should therefore always be checked.

In reservoir simulators the grid-cell volumes are generally assumed to be in *thermodynamic equilibrium*, and it is clear that this assumption is violated if there are fast state changes in the typical large grid cellss used in full field simulation. Compositional simulators usually have no available tricks for compensating for this, while some ad-hoc fixes, such as the Todd–Logstaff approach for miscible gas displacement and the `DRSDT` keyword[9] allowing gas and oil to be not fully equilibrated, exist for black oil.

[9] See OPM-Flow manual

8.5.2 *Saturation functions*

The saturation functions, i.e. relative permeability and capillary pressure, are normally provided in the form of tables, while some simulators allow for a limited number of parameterized correlations. These tables are then associated with spatial regions using region numbers which are fields of integers that are assigned to grid cells. In OPM-Flow the corresponding keyword is `SATNUM`. The regions are associated with geological model elements. If finer control is needed, for instance having relative permeability in a model element a function of permeability or initial saturation, a technique called end-point scaling is typically available.

Region numbers

The laboratory measurements pertaining to relative permeability and capillary pressure are called special core analysis (SCAL). It is well known that saturation functions are highly dependent on the wetting state of the fluid–rock system, and reproducing the correct reservoir wetting conditions in core samples is highly non-trivial. More so since the true reservoir wetting state is uncertain. Also, only a few core samples are subjected to SCAL. Up-scaling from core-plug to model-element scale is usually also required. Thus, relative permeability tend to have a large uncertainty, and is consequently a prime candidate for a tuning parameter when conditioning the model to production data.

Saturation functions are associated with large uncertainty

Data for (drainage) capillary pressure is available from saturation-logs

8.6 *Contact model and model initialization*

The contact model describes the initial fluid contacts, i.e. the water–oil and gas–oil contact, in each of the reservoir compartments. The

initial, pre production, state of the reservoir, that is initial pressures, compositions, and saturations, are determined by the contact model and the capillary pressure functions. The different contact regions are typically given to the simulator in terms of corresponding region numbers, and in OPM-Flow the relevant keyword is `EQLNUM`.

In order to establish initial hydrocarbon in place volumes, the saturations are often calculated on a fine gridded static geo-model. If detailed consistency in fluid distribution between the static model realizations and the coarser model realizations used for dynamic simulation is an issue, grid-based up-scaling of the saturations is performed. It should however also be noted that geo-modeling software often have simplified fluid models. Endpoint scaling can be used to ensure that the dynamic model is in initial gravitational equilibrium, either by scaling P_c (vertical scaling) or by setting the critical water saturation S_{wcr} equal to the initial saturation S_{wi} (horizontal scaling).

Correct initialization of compositional models with near-critical fluids (condensate reservoirs), where the composition can depend strongly on depth, requires special attention.

8.7 Well model

The boundary conditions to the reservoir simulation model are defined in wells. Wells are either controlled by rates or by pressure. The flow rate from a grid-cell into a well is proportional to the pressure difference between the well pressure and the grid-cell pressure. The proportionality constant C_i is called the connection factor:

Connection factor

$$Q_i = C_i \frac{k_{ri}(S_i)}{\mu} \left(p_i^{\text{cell}} - p_i^{\text{well}} \right) \quad . \tag{8.11}$$

The connection factor is often calculated using the Peaceman formula (6.11). This formula is derived based on the assumption that the well is vertical and running through the center of the grid-cell. Also, interference between neighboring wells is ignored. For non-vertical wells in cornerpoint grids, real well paths do not go through cell centers, as illustrated in figure 8.12. The perforation

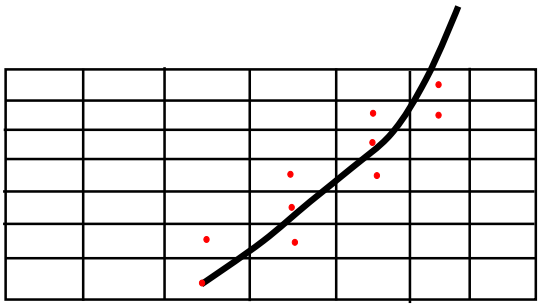


Figure 8.12: A well path through a cornerpoint grid. The grid-cellss indicated by a red dot are connected to the well.

length in each grid-cell has to be taken into account when calculating the connection factors, and normally also the depth difference

between grid-cell center and the corresponding well perforations is accounted for in the pressure difference calculation. However, the inflow from the different perforated grid-cells are interdependent, and the simple rate-independent connection factor representation (8.11) cannot fully account for the complex well inflow.

Wells with advanced inflow control equipment, and branched wells, are even more challenging. In a segmented-well model (Figure 8.13) the well is divided into segments. The rate and saturation dependent pressure drops in each segment is accounted for and the state and flow in each segment is included in the the overall simulation model equation system. Special in-well equipment, such as inflow control devices, is included as separate segments. The workings of most advanced in-well equipment, and any complex branching pattern can be accurately modeled with a segmented-well model. The main weakness in the approach is the use of rate independent connection factors, which cannot account for the interference between flow in neighboring perforated grid cells. In the case of branched wells, a grid cell may even be connected to more than one branch. Custom gridding around the wellbore is preferred when simulating "smart" or branched wells since the errors introduced by the constant connection factors may invalidate any conclusions related to the operation of the well.

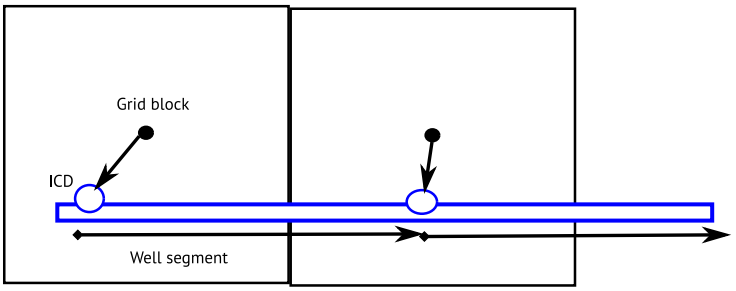


Figure 8.13: A well equipped with inflow control devices (ICDs). In a segmented-well model both pressure drops in the ICDs and in the wellbore segments are accounted for.

8.8 Production strategy

When a model is used for predicting future production the production strategy must be *operationalized* in a manner that can be understood by the simulator or by external software that interact with the simulator. In OPM-Flow this is typically expressed in terms of well and group controls, and `ACTIONX` keywords. The operationalized production strategy must be *robust* both with respect to numerics and event combinations.

8.9 Exercises

Exercise 8.1 In this exercise we will create a set of model realizations with varying porosity and permeability fields. We will consider a 2D model with 100×100 grid cells, and we will create 5 realizations that are assume equally probable.

Create a set of five different porosity realizations from a gaussian random field. From the porosity values we create a set of permeability values from the simple equation

$$k(\phi) = 10^{10\phi} \quad . \tag{8.12}$$

You can use a python code similar to the following to create your porosity field:

```
from gstools import SRF, Gaussian

fMu = 0.5 #mean value
fVariance = 1 #variance
inn=100 #number of grid cells in one direction

x=y=range(inn)
sModel = Gaussian(dim=2, var=fVariance, len_scale=10)
sSrf = SRF(sModel, seed=1)
aafField=sSrf.structured([x, y])
aafField+=fMu
```

For the gaussian random field for the porosity values you should use a correlation length of 10 (grid cells), a variance of 10^{-3}, and a mean of 0.2. Note that you need to change the seed to get different realizations. From the porosity fields, create corresponding permeability fields using the functional relationship in Eq. (8.12).

For your OPM-Flow runs you can use the example *.DATA file provided in the repository. Note that the path to the `PORO` and `PERMX` include-files needs to be updated in the provided *.DATA file.

`https://bitbucket.org/ntnu_petroleum/ressimbook-material/src/master/flow/Exercise81`

Run the five realizations using OPM-Flow. Compare the water production. Please comment on how the porosity distribution affects the production rates. Do you find any correlation between average field porosity or average field permeability and steady-state rates?

9
Basic theory for two phase flow

Playing with fire tastes much better when I'm on

Blood Command - Quitters don't smoke

We will now move from working with a single fluid phase to start working with multi-phase settings. A large part of managing sub-surface resources includes multi-phase flow; air and water in shallow aquifers, brine and CO_2 in CO_2 sequestration, and a three-phase system of oil, water and gas in hydrocarbon recovery. To start off our journey into multi-phase flow we will review the one-dimensional flow equations for horizontal flow, this time for two fluids instead of only one. In this two-phase setting we will look at analytical solutions of pressure and saturation as function of position and time. These equations are derived using an extension to the single phase Darcy equation; relative permeability. We will also devote a section to numerical diffusion, a non-physical diffusion that occurs when solving for several phases.

Throughout this chapter the two fluids under consideration are will have subscripts w and n. These letters can represent different things; a water phase and non-water phase, a wetting phase and a non-wetting phase, an aqueous phase and a non-aqueous phase (sometimes called a non-aqueous phase liquid (NAPL)). The subscript n is sometimes also interpreted as naphtha, in the meaning oil. What is important for us is that we have to immiscible phases, i.e., two phases that are separated by a fluid-fluid interface.

9.1 Extending mass conservation to two phases

By considering mass conservation, we obtained the following equation, Eq. (3.35), in Chapter 3:

$$-\frac{\partial}{\partial x}(\rho q) = \frac{\partial}{\partial t}(\phi\rho) \quad , \tag{9.1}$$

here written as a 1D equation. Let s_p denote the saturation of phase i, hence s_w is the aqueous phase saturation and s_n is the non-aqueous phase saturation. If we assume that the phases are immiscible, in the sense that their composition remain constant and

Immiscible phases

there is no exchange of components, we can apply mass conservation to each phase to obtain

$$-\frac{\partial}{\partial x}\left(\rho_p q_p\right) = \frac{\partial}{\partial t}\left(\phi \rho_p s_p\right) \quad , \tag{9.2}$$

where ρ_p and q_p is the density and Darcy velocity of phase p, respectively. Assuming solely two phases, we have $s_n + s_w = 1$.

We will now extend the Darcy equation to one equation for each phase. The extended Darcy equation is written as

Extended Darcy equation

$$q_p = -\frac{k k_{rp}}{\mu_p}\frac{\partial p_p}{\partial x} \quad . \tag{9.3}$$

Here k_{rp} is the *relative permeability* of phase p, and p_p is the pressure in phase p. The phase pressures are linked through the capillary pressure p_c as

$$p_c = p_n - p_w \quad , \tag{9.4}$$

where it is a convention to subtract the wetting phase pressure from the non-wetting phase pressure.

Substituting the extended Darcy equation, Eq. (9.3), into the continuity equation, we obtain:

Two-phase flow equation

$$\frac{\partial}{\partial x}\left(\rho_p \frac{k k_{rp}}{\mu_p}\frac{\partial p_p}{\partial x}\right) = \frac{\partial}{\partial t}\left(\rho_p \phi s_p\right) \quad . \tag{9.5}$$

For incompressible fluids we can remove the densities ρ_p, giving the following set of equations:

We will consider multi-phase flow of compressible fluids in Sec. 10.2.

Incompressible two-phase flow equations

$$\begin{aligned}\frac{\partial}{\partial x}\left(\frac{k k_{rn}}{\mu_n}\frac{\partial p_n}{\partial x}\right) &= \frac{\partial}{\partial t}\left(\phi s_n\right) \\ \frac{\partial}{\partial x}\left(\frac{k k_{rw}}{\mu_w}\frac{\partial p_w}{\partial x}\right) &= \frac{\partial}{\partial t}\left(\phi s_w\right) \quad .\end{aligned} \tag{9.6}$$

To solve such a system we need to describe our new parameters, the relative permeability k_{rp} and capillary pressure p_c.

9.2 Relative permeability and capillary pressure

For CO_2 and water it is typically water that has most adherence to the rock surface, and is therefore considered the wetting fluid. Also for water and air systems the water is considered the wetting fluid. For water, oil and gas systems, water is considered most wetting, oil intermediate and gas the least wetting fluid. Be aware that the actual wetting varies throughout the pore space and with fluid and rock properties. You could have hydrocarbon reservoirs where oil is more wetting than water. In the following we will consider water the wetting fluid, if another fluid is actually the wetting fluid, this will be captured by the capillary pressure and relative permeability.

A process where the wetting fluid is displaced by a non-wetting fluid is called drainage, while a process where the non-wetting fluid is displaced by a wetting fluid is called imbibition. Thus, CO_2 injection into an aquifer is a drainage process. Pumping water

Drainage and imbibition

out of a shallow aquifer, so that the water is displaced by air, is also a drainage process, while a recharging of the aquifer through precipitation is an imbibition process.

Oil accumulation into a hydrocarbon reservoir starts at a water saturation of one, and draining the water down to the initial (i.e., initial when we start to produce the reservoir) water saturation is called primary drainage. This primary drainage process takes place over geological time. As the initial saturation is the result of a drainage process, the primary drainage capillary pressure curve is required for initialization the saturation using pressure.

Primary drainage

When we start to inject water into and produce oil out of the reservoir, we are reversing the geological-time drainage process to a much faster imbibition process. While imbibition is the general process when producing a reservoir, both drainage and imbibition can happen simultaneously at different parts of the reservoir, e.g., when creating an oil bank due to an enhanced recovery method. An increase in water saturation during oil production is considered a secondary drainage process. Modeling both drainage and imbibition requires several flow parameter curves for the different processes, and is therefore significantly more complex than pure imbibition.

Secondary drainage

In general we need different relative permeability and capillary pressure curves for drainage (decreasing wetting saturation) and imbibition (increasing wetting saturation). Unfortunately, a possible secondary drainage process might require a third set of curves. In general you only need the capillary pressure curve for the primary drainage process, as we seldom model the fluid flow over geological time. The two properties relative permeability and capillary pressure are thus not fully defined by saturation. It can be shown that the rate of change is also affecting the flow parameter curves.

In this chapter we will however assume that both curves can be described as a function of saturation only, thus disregarding any changes between drainage and imbibition curves. This assumption is fair as long as the saturation changes monotonically and the rates are in a limited range.

The relative permeability curves are sometimes parameterized to simplify its description. The most common parameterization is the (Brooks-)Corey curves (Brooks and Corey, 1964), which is a power law in the normalized saturation:

Corey functions for relative permeability

$$\begin{aligned} k_{rn}(s_w) &= (1 - s_{wn})^{n_n} \\ k_{rw}(s_w) &= k_{rw}^n s_{wn}^{n_w} \\ s_{wn} &= \frac{s_w - s_{wir}}{1 - s_{wir} - s_{nrw}} \end{aligned} , \tag{9.7}$$

where s_{wir} is the irreducible wetting saturation, while s_{nrw} is the irreducible non-wetting saturation, s_{wn} is the normalized water saturation, $k_{rw}^n = k_{rw}(s_{nrw})$ is the wetting relative permeability at irreducible non-wetting saturation, and n_n and n_w are the power law parameters for the non-wetting and wetting (water) case, re-

spectively.

These functions can be implemented in Python as:

```
def normSw(fSw,fSwirr,fSnrw):
    return (fSw-fSwirr)/(1.0-fSnrw-fSwirr)

def coreyWater(fSw,fNw,fKrwn):
    return fKrwn*fSw**fNw

def coreyNAPL(fSw,fNn):
    return (1-fSw)**fNn
```

Common values for the power law parameters are $n_n = 2 = n_w$. If we assume that $k^n_{rw} = 0.4$ and $s_{wir} = 0.2 = s_{nrw}$, then this will give the relative permeability curves shown in Fig. 9.1.

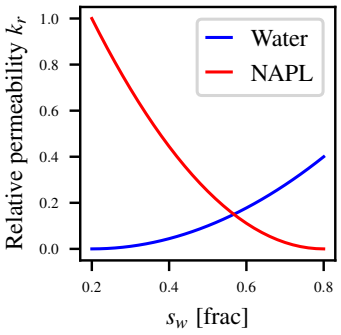


Figure 9.1: Corey relative permeability curves using $n_n = 2 = n_w$.

There exist other parameterization methods, e.g., the LET-parameters (Lomeland et al., 2005). These have more parameters, and can thus give more sophisticated shapes. One use of relative permeability parameterization is in history matching. To obtain a match one often need the added flexibility given by curves parameterized by several parameters. One drawback of using curve shapes described by more parameters is that the curves are then non-unique with respect to the parameters, i.e., two different set of parameters can give similar curves. This non-uniqueness can be a challenge for automation of optimization processes.

Also the capillary pressure curves can be parameterized. One method is the Skjæveland method (Skjaeveland et al., 1998), were the imbibition capillary pressure curve is described by the function

$$p_c(s_w) = \frac{c_w}{\left(\frac{s_w - s_{wir}}{1 - s_{wir}}\right)^{a_w}} + \frac{c_n}{\left(\frac{s_n - s_{nrw}}{1 - s_{nrw}}\right)^{a_n}} \quad . \tag{9.8}$$

This can be implemented in Python as

```
def skjaevland(fSw,fSwirr,fSnrw,fCw,fAw,fCn,fAn):
    return fCw/((fSw-fSwirr)/(1-fSwirr))**fAw+fCn/((1-fSw-fSnrw)
    ↪ /(1-fSnrw))**fAn
```

An example using the parameters $a_w = 2 = a_n$, $c_w = 5 \times 10^{-3}$ bar, $c_n = -5 \times 10^{-3}$ bar, and $s_{wir} = 0.2 = s_{nrw}$ is shown Fig. 9.2.

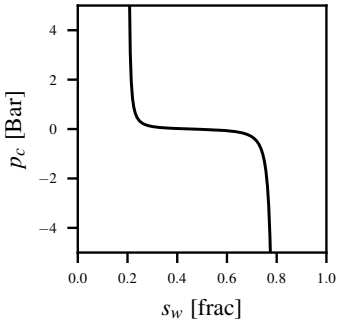


Figure 9.2: Skjævland capillary pressure curve using $a_w = 2 = a_n$, $c_w = 5 \times 10^{-3}$ bar and $c_n = -5 \times 10^{-3}$ bar.

Using $c_n = 0$ will give a primary drainage capillary pressure curve, with $p_c(s_w) > 0$ for all saturation values.

We define the fractional flow of the wetting phase (water) f_w as the fraction of wetting phase Darcy velocity q_w to the total Darcy velocity $q_t = q_w + q_n$ (equivalently, the fraction of wetting phase flow rate to the total flow rate):

$$f_w = \frac{q_w}{q_t} = \frac{q_w}{q_w + q_n} = \frac{1}{1 + \frac{q_n}{q_w}} \quad . \tag{9.9}$$

If we assume that the capillary pressure is zero, so that $p_n = p_w$, then from Eq. (9.3) we have

$$\frac{q_n}{q_w} = \frac{-\frac{kk_{rn}}{\mu_n}\frac{\partial p_n}{\partial x}}{-\frac{kk_{rw}}{\mu_w}\frac{\partial p_w}{\partial x}} = \frac{\frac{k_{rn}}{\mu_n}}{\frac{k_{rw}}{\mu_w}} = \frac{k_{rn}\mu_w}{k_{rw}\mu_n} \quad . \tag{9.10}$$

This gives the factional flow as

$$f_w(s_w) = \frac{1}{1 + \frac{q_n(s_w)}{q_w(s_w)}} = \frac{1}{1 + \frac{k_{rn}(s_w)\mu_w}{k_{rw}(s_w)\mu_n}} \quad . \tag{9.11}$$

Fractional flow

Using the same parameters as for the Corey relative permeability curves in Fig. 9.1, and using a wetting phase viscosity of 1×10^{-3} Pa s and oil viscosity of 2×10^{-3} Pa s, we get the fractional flow curve depicted in Fig. 9.3.

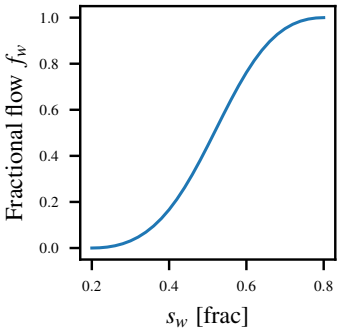


Figure 9.3: Fractional flow curve of the relative permeability curves in Fig. 9.1 with $\mu_w = 1 \times 10^{-3}$ Pa s and $\mu_n = 2 \times 10^{-3}$ Pa s.

9.3 *Buckley–Leverett equation*

In this section we will present a solution to the set of two partial differential equations, Eqs. (9.6), describing incompressible two-phase flow in a porous medium. This set of equations can not be solved analytical in general. However, it exists a semi-analytical solution for a one-dimensional system, the *Buckley-Leverett* equation. For this solution we need the additional assumptions of incompressible rock and zero capillary pressure.

The Buckley-Leverett equation is named after the paper (Buckley and Leverett, 1942).

We will now assume a constant injection rate q_l on the left side, and a constant production rate q_r on the right side.

If you assume a pseudo-1D slab of cross-section A, then you need to assume a constant injection rate $Q_p = Aq_p$.

Given incompressible fluids, our mass balance equation, Eq. (9.2), can be simplified to

$$-\frac{\partial q_p}{\partial x} = \phi \frac{\partial}{\partial t}(s_p) \quad , \tag{9.12}$$

where the porosity has been moved outside the derivative as the rock is assumed incompressible. The wetting phase equation can then be written as

$$-\frac{\partial q_w}{\partial x} = \phi \frac{\partial s_w}{\partial t} \quad . \tag{9.13}$$

Since the fluids are incompressible, we have that the total flow q_t is constant. In particular, the flow rate boundary condition on the left and right side of the model equals the constant flow rate: $q_l = q_r = q_t$. Further, the wetting phase flow is given by $q_w = q_t f_w$, where f_w is the fractional flow (of the wetting phase). This gives the Buckley-Leverett equation

$$-\frac{\partial f_w}{\partial x} = \frac{\phi}{q_t} \frac{\partial s_w}{\partial t} \quad . \tag{9.14}$$

Buckley-Leverett equation

In the following we will find an analytical solution to this equation. For this end, we start by employing the chain rule to get

$$-\frac{df_w}{ds_w}\frac{\partial s_w}{\partial x} = \frac{\phi}{q_t}\frac{\partial s_w}{\partial t} \quad . \tag{9.15}$$

Expand the partial ds_w to

$$ds_w = \left(\frac{\partial s_w}{\partial x}\right) dx + \left(\frac{\partial s_w}{\partial t}\right) dt \quad . \tag{9.16}$$

At a constant saturation, $ds_w = 0$, we then have

The vertical line symbol $|_{s_w}$ indicates that the derivative is at a constant value of s_w.

$$\frac{\partial s_w}{\partial t} = -\left.\frac{dx}{dt}\right|_{s_w} \frac{\partial s_w}{\partial x} \quad . \tag{9.17}$$

We can then eliminate $\partial s_w/\partial t$ from Eq. (9.15), and get

$$\frac{df_w}{ds_w} = \frac{\phi}{q_t}\left.\frac{dx}{dt}\right|_{s_w} \quad . \tag{9.18}$$

Denote the velocity of the constant saturation as u_{s_w}, then we have

$$u_{s_w} = \left.\frac{dx}{dt}\right|_{s_w} = \frac{df_w}{ds_w}\frac{q_t}{\phi} \quad . \tag{9.19}$$

Thus, the velocity of a constant saturation is proportional to the derivative of the fractional flow curve.

We will now consider a wetting phase flooding scenario; consider a one dimensional model initially filled with non-wetting and wetting phases at $s_w = s_{wir}$. The wetting phase is injected at $x = 0$ with a rate $q_w(t)$ According to Eq. (9.19), the coordinate x_{s_w} where the saturation is s_w then moves with the velocity u_{s_w}. The coordinate x_{s_w} for where the saturation is s_w is then given by

$$x_{s_w}(t) = \int_0^t u_{s_w}(t')dt' = \frac{W(t)}{\phi}\frac{df_w}{ds_w} \quad , \tag{9.20}$$

where

$$W(t) = \int_0^t q_w(t')\,dt' \tag{9.21}$$

is the total volume of wetting phase injected.

The derivative of the fractional flow curve shown in Fig. 9.3 is shown in Fig. 9.4. Using the description of the Corey curves, this was implemented in Python as

```
def fractionalFlowCorey(fSw,fNw,fNn,fKrwn,fMuw,fMun):
    return 1/(1+(1-fSw)**fNn*fMuw/(fKrwn*fSw**fNw*fMun))
```

We see from Eq. (9.20) that flipping axis for this curve, as shown in Fig. 9.5, should give the saturation distribution at a particular time. However, as we see from the plot, there are two saturation values for each distance and given time. This is unphysical, at most one of these saturations represent the true solution.

At high wetting phase saturations, velocity u_{s_w} decrease with saturation, thus the saturation profile is spreading with time. This situation is called a *rarefaction wave*. On the other hand, at low wetting phase saturation, velocity u_{s_w} *increase* with saturation, thus the higher saturations will overtake the lower forming a self sharpening *shock front*. At the front, the saturation has an instant change from the shock front saturation s_{wf} to the initial saturation, s_{wir}. Behind the front, the saturation is described by Eq. (9.20). The saturation distribution is illustrated in Fig. 9.6. We will derive the front position, x_f, and saturation, s_{wf}, below.

The average saturation behind the front is given by the total volume of injected wetting phase fluid divided by pore volume, in addition to the initial wetting fluid:

$$\langle s_w \rangle = s_{wir} + \frac{W(t)}{\phi x_f} = s_{wir} + \frac{1}{\left.\frac{df_w}{ds_w}\right|_{s_{wf}}} \quad , \tag{9.22}$$

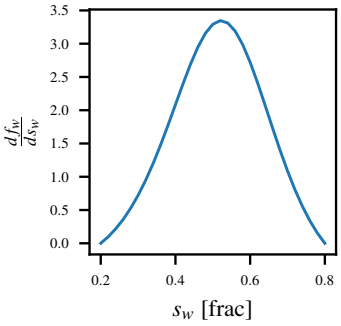


Figure 9.4: The derivative with respect to s_w of the fractional flow curve in Fig. 9.3.

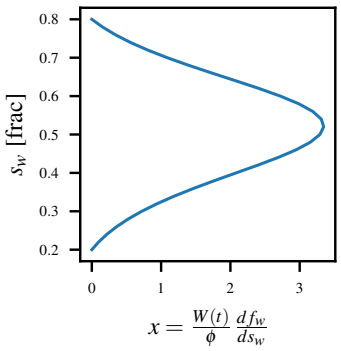


Figure 9.5: Getting the saturation profile by changing axis for the plot in Fig. 9.4.

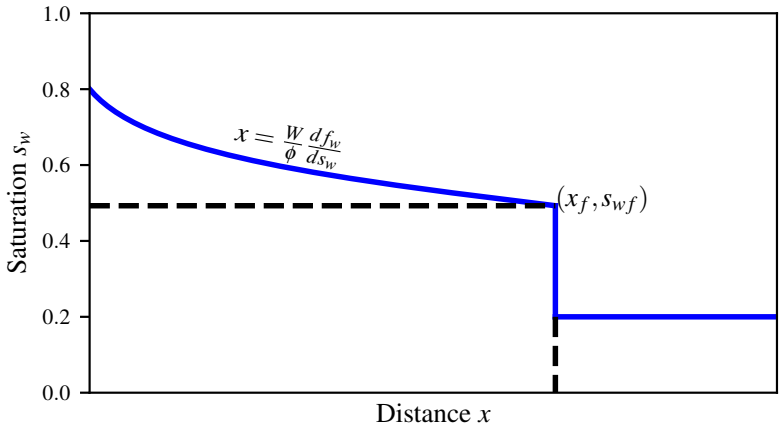


Figure 9.6: Saturation profile for the Buckley–Leverett problem.

where we have used Eq. (9.20) for the saturation s_{wf} at the shock front x_f. We can also find the average saturation by integrating over the saturation profile:

$$\langle s_w \rangle = \frac{\int_0^{x_f} s_w \, dx}{x_f} \quad . \tag{9.23}$$

We see from Fig. 9.6 that

$$\begin{aligned} \int_0^{x_f} s_w \, dx &= x_f s_{wf} + \int_{s_{wf}}^{1-s_{nrw}} \frac{W}{\phi}\frac{df_w}{ds_w} \, ds_w \\ &= x_f s_{wf} + \frac{W}{\phi}\Big(1 - f_w(s_{wf})\Big) \quad , \end{aligned} \tag{9.24}$$

where the term $x_f s_{wf}$ gives the area of the square outlined by the black dashed lines in Fig. 9.6, while the integral term gives the area between the blue curve and the top of the square. Note that this integral is with respect to the water saturation, so we integrate the area between the y-axis and the blue curve between the two given saturation values; the residual non-wetting saturation $1 - s_{nrw}$ and the shock front saturation s_{wf}. The integral is solved by changing from integrating on s_W to integrating on $(df_w/ds_w)ds_w = df_w$.

Filling Eq. (9.24) into Eq. (9.23) gives

$$\langle s_w \rangle = s_{wf} + \frac{1 - f_w(s_{wf})}{\left.\frac{df_w}{ds_w}\right|_{s_{wf}}} \quad . \tag{9.25}$$

Eliminating $\langle s_w \rangle$ from Eqs. (9.22) and (9.25) gives

$$\left.\frac{df_w}{ds_w}\right|_{s_{wf}} = \frac{f_w(s_{wf})}{s_{wf} - s_{wir}} \quad . \tag{9.26}$$

Due to the shape of the fractional flow curve, this equation holds for only one wetting phase saturation s_{wf}, shown by the secant line in Fig. 9.7. The saturation jump at the shock front is thus constant in time, and, with constant injection rate, the front moves at

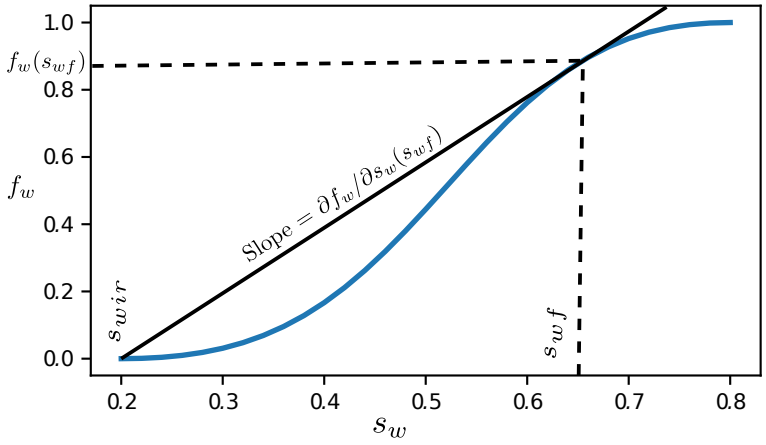


Figure 9.7: The secant indicating the shock front saturation for the one-dimensional two-phase flow.

a constant speed. The saturation at the shock front is given by the tangent to the fractional flow curve which cross the point $(s_{wir}, 0)$.

To find the shock front saturation, we observe that the line between $(s_{wir}, 0)$ and $(s_w, f_w(s_w))$ is at is steepest for $s_w = s_{wf}$. The `scipy` package has several optimization modules, including a module for finding the minimal value. As we want to find the maximal value of the slope, we use this package on the negative of the slope. This can be implemented in Python as:

```
def findFrontSaturation(fSwirr,fSnrw,fNw,fNn,fKrwn,fMuw,fMun):
    from scipy import optimize
    def tangentFFCorey(fSw):
        return -fractionalFlowCorey(normSw(fSw,fSwirr,fSnrw),fNw,
    ↪ fNn,fKrwn,fMuw,fMun)/(fSw-fSwirr)
    fSwf=optimize.fmin(tangentFFCorey,((1.0-fSnrw)-fSwirr)/2.0)
    return fSwf
```

Here `fSwf` will be the shock front saturation.

We can then calculate the Buckley-Leverett function as:

```
def buckleyLeverettSolution(fTime,fSwirr,fSnrw,fNw,fNn,fKrwn,fMuw,
    ↪ fMun,fModelLenght,fPoro,fDarcyVelocity):
    fSwf=findFrontSaturation(fSwirr,fSnrw,fNw,fNn,fKrwn,fMuw,fMun)
    from scipy.misc import derivative
    def funcFFCorey(fSw):
        return fractionalFlowCorey(normSw(fSw,fSwirr,fSnrw),fNw,
    ↪ fNn,fKrwn,fMuw,fMun)
    afSw=np.arange(fSwf,1.0-fSnrw,0.001)
    afDerivativeFFCorey=derivative(funcFFCorey,afSw,dx=1E-6)
    afDerFFDarcyVeloPhi=afDerivativeFFCorey*fDarcyVelocity/fPoro
    return afDerFFDarcyVeloPhi*fTime,afSw
```

This can be plot as follows:

```
cmap = plt.get_cmap('gnuplot')

fTime=0.0
fDeltaT=0.1
fMaxTime=0.5
while(fTime<fMaxTime):
```

```
  fTime=fTime+fDeltaT
  afDistance,afSw=buckleyLeverettSolution(fTime,fSwirr,fSorw,fNw,
    ↪ fNo,fKrwo,fMuw,fMuo,fModelLength,fPoro,fDarcyVelocity)
  plt.plot(afDistance,afSw,color=cmap(1-fTime/fMaxTime))
  plt.plot([afDistance[0],afDistance[0]],[afSw[0],fSwirr],color=
    ↪ cmap(1-fTime/fMaxTime))
  plt.plot([afDistance[0],fModelLength],[fSwirr,fSwirr],color=cmap
    ↪ (1-fTime/fMaxTime))

plt.xlabel(r'Distance [m]')
plt.ylabel(r'Saturation $s_w$')
```

This code plots as shown in Fig. 9.8.

As seen from the derivation and in the plot, the Buckley-Leverett equation gives a discontinuous saturation at the wetting phase front. If we include capillary pressure, then a large saturation derivative will give large capillary pressure differences. This will induce flow that will smear out the shock front. As we will see in the next chapter, also numerical errors will smear out the shock front.

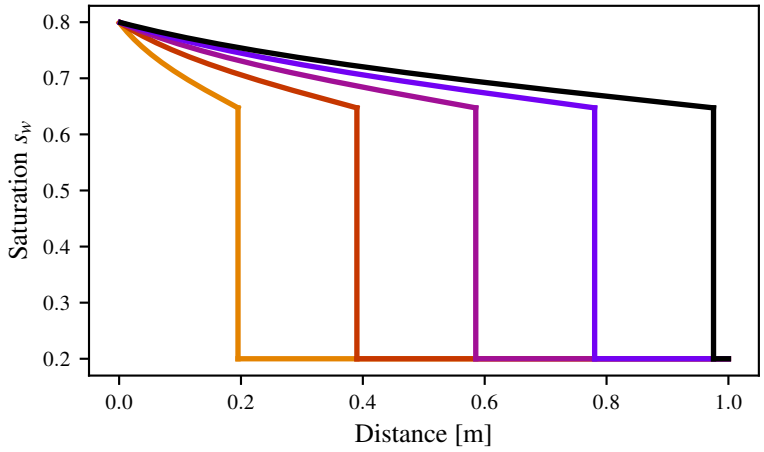


Figure 9.8: The Buckley-Leverett equation for 10 different time steps.

9.4 *Numerical diffusion*

The analytical Buckley-Leverett solution gives a perfectly sharp front, as seen in Fig. 9.8. Real life displacements will never give such a perfect sharp front, both due to mixing and due to capillary forces the smear out any front. Without any relation this physical diffusion, also numerical solutions give a smeared out front. It is this numerical smearing of the front that will be treated in this section. This phenomena is called *numerical diffusion*. While the phenomena has nothing to do with actual diffusion in the porous medium, the numerical methods yield solutions that resembles the solution of a advection-diffusion equation (also known as the convection-diffusion equation). In most cases relevant for reservoir simulation, numerical diffusion is much stronger than the physical diffusion.

Numerical diffusion

9.4.1 *Advection–diffusion equation*

The advection–diffusion equation describes a system where a quantity, e.g., temperature, solute concentration, etc., is transferred due to two processes, namely advection and diffusion. Transfer by advection means that the quantity is transported by fluid flow. Let c be the quantity, e.g., salt concentration in a brine, and let $\vec{u}$ be the velocity field of the fluid flow (not necessarily inside a porous medium), then the local change in the quantity $\partial c/\partial t$ due to advection of the quantity is given by gradient in the quantity, i.e., ∇c, in the direction of the flow field $\vec{u}$: thus the change is given as $\vec{u} \cdot \nabla c$. The change in quantity due to advection is

$$\frac{\partial c}{\partial t} = -\vec{u} \cdot \nabla c \quad . \tag{9.27}$$

Diffusion is the process of movement due to a quantity gradient, usually a movement from high towards low quantity. Typical examples are temperature and solute concentration, with heat transported from high temperature towards lower temperature, and solutes transported from high concentrations towards lower concentrations. Diffusion is described by Fick's first law, where the diffusion flux vector $\vec{j}$ of the quantity c is proportional to the quantity gradient ∇c by a diffusion coefficient D:

$$\vec{j} = -D\nabla c \quad . \tag{9.28}$$

Fick's first law

This flux then give a change in quantity c with time as given by Fick's second law:

$$\frac{\partial c}{\partial t} = -\nabla \cdot \vec{j} = \nabla \left(D\nabla c \right) = D\nabla^2 c \quad , \tag{9.29}$$

Fick's second law

where the last equality only holds for a constant diffusion constant D.

If we have a quantity that is transferred by both advection and diffusion, e.g., a drop of ink in a stream, we will get a combination of the advection equation, Eq. (9.27), and the diffusion equation, Eq. (9.29), yielding the *advection-diffusion equation*:

Advection-diffusion equation

$$\frac{\partial c}{\partial t} = -\vec{u} \cdot \nabla c + D\nabla^2 c \quad . \tag{9.30}$$

Note that this equation describes physical systems where you have diffusion. For the one-dimensional example described by the Buckley-Leverett equation, we have no diffusion, or to be more precise; we have ignored the diffusive term related to capillary pressure. In the following we will describe an artificial diffusion that arise in the numerical solutions. This numerical diffusion has no physical counterpart.

9.4.2 *Numerical diffusion*

We now want to investigate the solution of the Buckley-Leverett equation, Eq. (9.14), by different numerical methods. These numerical methods will be applied in the next chapter, in this section we

will only investigate the consequence of applying different numerical solution methods.

After employing the chain rule, we have the Buckley-Leverett equation on the form given by Eq. (9.15):

$$-\left(\frac{df_w}{ds_w}\right)\frac{\partial s_w}{\partial x} = \frac{\phi}{q_t}\frac{\partial s_w}{\partial t} \quad . \tag{9.31}$$

This equation can be solved explicitly as

$$-\left(\left(\frac{df_w}{ds_w}\right)\frac{\partial S_w}{\partial x}\right)_i^n = \frac{\phi}{q_t}\left(\frac{\partial S_w}{\partial t}\right)_i^n \quad , \tag{9.32}$$

or implicitly as

$$-\left(\left(\frac{df_w}{ds_w}\right)\frac{\partial S_w}{\partial x}\right)_i^{n+1} = \frac{\phi}{q_t}\left(\frac{\partial S_w}{\partial t}\right)_i^{n+1} \quad . \tag{9.33}$$

We will consider both the explicit and implicit case, and compare them in the end.

We will later need the following reformulation of the second-order time derivative to the second-order spatial derivative:

$$\begin{aligned}
\frac{\partial^2 s_w}{\partial t^2} &= \frac{\partial}{\partial t}\left(\frac{\partial s_w}{\partial t}\right) \\
&= -\frac{\partial}{\partial t}\left(\frac{q_t}{\phi}\left(\frac{df_w}{ds_w}\right)\frac{\partial s_w}{\partial x}\right) \\
&= -\frac{q_t}{\phi}\left(\frac{df_w}{ds_w}\right)\frac{\partial}{\partial t}\left(\frac{\partial s_w}{\partial x}\right) \\
&= -\frac{q_t}{\phi}\left(\frac{df_w}{ds_w}\right)\frac{\partial}{\partial x}\left(\frac{\partial s_w}{\partial t}\right) \\
&= -\frac{q_t}{\phi}\left(\frac{df_w}{ds_w}\right)\frac{\partial}{\partial x}\left(-\frac{q_t}{\phi}\left(\frac{df_w}{ds_w}\right)\frac{\partial s_w}{\partial x}\right) \\
&= \left(\frac{q_t}{\phi}\frac{df_w}{ds_w}\right)^2\frac{\partial^2 s_w}{\partial x^2} \quad .
\end{aligned} \tag{9.34}$$

Let us now continue by trying to solve the Buckley-Leverett solution numerically. We start by looking at the time derivative of the saturation. Using Taylor series of the saturation s_w around the time t^n we obtain

$$\begin{aligned}
s_w(t^{n+1}) = s_w(t^n) &+ \frac{\partial s_w}{\partial t}(t^n)(t^{n+1}-t^n) \\
&+ \frac{1}{2}\frac{\partial^2 s_w}{\partial t^2}(t^n)(t^{n+1}-t^n)^2 + \mathcal{O}((t^{n+1}-t^n)^3) \quad .
\end{aligned} \tag{9.35}$$

With $\Delta t = t^{n+1} - t^n$, we can then rewrite this equation as

$$\frac{s_w^{n+1}-s_w^n}{\Delta t} = \left(\frac{\partial s_w}{\partial t}\right)^n + \frac{\Delta t}{2}\left(\frac{\partial^2 s_w}{\partial t^2}\right)^n + \mathcal{O}((\Delta t)^2) \quad , \tag{9.36}$$

where the superscript indicates at which time step the function is evaluated. The first part of this equation is then the forward difference quotient approximation of the first-order derivative.

From a Taylor series expansion of saturation s_w around the time t^{n+1} we obtain

$$s_w(t^n) = s_w(t^{n+1}) + \frac{\partial s_w}{\partial t}(t^{n+1})(t^n - t^{n+1}) + \frac{1}{2}\frac{\partial^2 s_w}{\partial t^2}(t^{n+1})(t^n - t^{n+1})^2 + \mathcal{O}((t^n - t^{n+1})^3) \quad , \quad (9.37)$$

which can be rewritten as

$$\frac{s_w^{n+1} - s_w^n}{\Delta t} = \left(\frac{\partial s_w}{\partial t}\right)^{n+1} - \frac{\Delta t}{2}\left(\frac{\partial^2 s_w}{\partial t^2}\right)^{n+1} + \mathcal{O}((\Delta t)^2) \quad . \quad (9.38)$$

The first part then gives a backward-difference quotient shifted in time.

To obtain the spatial derivative of the saturation, we again use the Taylor series, but this time centered around the spatial coordinate x_i of grid cell i, and evaluated at the coordinate x_{i-1} of the upstream grid-cell x_{i-1}:

$$s_w(x_{i-1}) = s_w(x_i) + \frac{\partial s_w}{\partial x}(x_i)(x_{i-1} - x_i) + \frac{1}{2}\frac{\partial^2 s_w}{\partial x^2}(x_i)(x_{i-1} - x_i)^2 + \mathcal{O}((x_{i-1} - x_i)^3) \quad . \quad (9.39)$$

Denoting $s_w(x_i)$ as s_{wi}, and letting $\Delta x = x_i - x_{i-1}$, we get

$$\frac{s_{wi} - s_{wi-1}}{\Delta x} = \left(\frac{\partial s_w}{\partial x}\right)_i - \frac{\Delta x}{2}\left(\frac{\partial^2 s_w}{\partial x^2}\right)_i + \mathcal{O}((\Delta x)^2) \quad . \quad (9.40)$$

We now have all we need to investigate the numerical versions of the Buckley-Leverett equation. Let us start by the explicit version, as given by Eq. (9.32). If we use the forward difference quotient for the time differential and the upstream difference quotient for the spatial derivative, we get the following numerical equation for the explicit equation Eq. (9.32):

$$\frac{s_{wi}^{n+1} - s_{wi}^n}{\Delta t} = -\frac{q_t}{\phi}\left(\frac{df_w}{ds_w}\right)\frac{s_{wi}^n - s_{wi-1}^n}{\Delta x} \quad (9.41)$$

We know that this equation has an error in order $\mathcal{O}(\Delta t)$ and $\mathcal{O}(\Delta x)$ in time and space, respectively. We will now investigate this error by reformulating the equation to an equation with errors of higher order.

For this, we use Eq. (9.36) and Eq. (9.40) to replace the difference quotients with partial differential equations:

$$\left(\frac{\partial s_w}{\partial t}\right)_i^n + \frac{\Delta t}{2}\left(\frac{\partial^2 s_w}{\partial t^2}\right)_i^n = -\frac{q_t}{\phi}\left(\frac{df_w}{ds_w}\right)\left[\left(\frac{\partial s_w}{\partial x}\right)_i^n - \frac{\Delta x}{2}\left(\frac{\partial^2 s_w}{\partial x^2}\right)_i^n\right] \quad . \quad (9.42)$$

Removing the numerical coordinates, and using Eq. (9.34) to transform the second-order time derivative with a spatial derivative of

the same order, we get

$$\begin{aligned}\frac{\partial s_w}{\partial t} &= -\frac{q_t}{\phi}\left(\frac{df_w}{ds_w}\right)\frac{\partial s_w}{\partial x} + \left(\frac{q_t}{\phi}\left(\frac{df_w}{ds_w}\right)\frac{\Delta x}{2} - \left(\frac{q_t}{\phi}\frac{df_w}{ds_w}\right)^2\frac{\Delta t}{2}\right)\frac{\partial^2 s_w}{\partial x^2} \\ &= -\frac{q_t}{\phi}\left(\frac{df_w}{ds_w}\right)\frac{\partial s_w}{\partial x} + \frac{q_t}{\phi}\left(\frac{df_w}{ds_w}\right)\left(\frac{\Delta x}{2} - \frac{q_t}{\phi}\left(\frac{df_w}{ds_w}\right)\frac{\Delta t}{2}\right)\frac{\partial^2 s_w}{\partial x^2} \quad .\end{aligned} \tag{9.43}$$

We observe that this equation is on the same form as the advection-diffusion equation, where the first term

$$u = \frac{q_t}{\phi}\left(\frac{df_w}{ds_w}\right) \quad , \tag{9.44}$$

reflect the velocity in the advection term, while

$$D_e = u\left(\frac{\Delta x}{2} - u\frac{\Delta t}{2}\right) \quad , \tag{9.45}$$

reflect the diffusion constant in the diffusion term. We can thus write Eq. (9.43) as

$$\frac{\partial s_w}{\partial t} = -\frac{q_t}{\phi}\left(\frac{df_w}{ds_w}\right)\frac{\partial s_w}{\partial x} + D_e\frac{\partial^2 s_w}{\partial x^2} \quad . \tag{9.46}$$

We see that the first two terms are the original Buckley-Leverett equation. Thus, in this form of the explicit equation, Eq. (9.41), where we have reduced the error to an order $\mathcal{O}((\Delta t)^2)$ and $\mathcal{O}((\Delta x)^2)$, we see that the difference between the Buckley-Leverett equation and the explicit numerical approximation is given by an extra diffusion term. That the equation resembles an advection-diffusion equation, and that the error is given by the diffusion term, is the reason such numerical errors are called numerical diffusion.

Doing the same algebraic manipulations for the implicit method, we get the equation

$$\frac{\partial s_w}{\partial t} = -\frac{q_t}{\phi}\left(\frac{df_w}{ds_w}\right)\frac{\partial s_w}{\partial x} + D_i\frac{\partial^2 s_w}{\partial x^2} \quad , \tag{9.47}$$

where the diffusion constant for the implicit method is

$$D_i = u\left(\frac{\Delta x}{2} + u\frac{\Delta t}{2}\right) \quad . \tag{9.48}$$

We see that the diffusion term will smear out the solution, and it will smear it out relative to the magnitude of the diffusion term. We also observe that $D_i > D_e$, thus there is more numerical diffusion in the implicit method than the explicit. By choosing

The implicit method nas more numerical diffusion than the explixit

$$\Delta x = u\Delta t \quad , \tag{9.49}$$

we can remove the numerical diffusion when using the explicit method. Note that this choice of Δx and Δt will give other numerical issues, however, it is possible to reduce the numerical diffusion to a minimum by choosing values close to the equality in Eq. (9.49).

9.5 Exercises

Exercise 9.1 Create a Python script to calculate the Buckley-Leverett solution for the saturation. The basic data is given in Table 9.1.

a) Plot the saturation versus distance from inlet for a set of times that illustrate the transient period.

b) Do a sensitivity study on the oil viscosity to investigate how the oil viscosity influence:

- The fractional flow curve
- The shock front saturation
- The Buckley-Leverett curve

	SPE Metric	SI
l	200 m	200 m
μ_w	1 cP	1×10^{-3} Pa s
μ_n	2 cP	2×10^{-3} Pa s
ϕ	0.2	0.2
n_w	2.0	2.0
n_n	2.0	2.0
k_{rwo}	0.4	0.4
q_t	0.2 m/d	2.3148 m/s

Table 9.1: Basic data for example.

Exercise 9.2 Derive the numerical diffusion constant D_e given by Eq. (9.48) for the implicit numerical solution method.

10
Numerical methods for two phase flow

> You wanna know how to rhyme you
> better learn how to add
>
> Mos Def - Mathematics

In this chapter we will solve numerically a one-dimensional two-phase flow problem. Our derivations can easily be extended to higher dimensional problems. The reasons for restricting our derivations to one dimension is to keep the notation simple and that we have an analytical solution through the Buckley-Leverett analysis in the previous section. Thus, we will start by investigating the same system as was considered for the Buckley-Leverett equation in the previous section, i.e., the set of equations for two-phase incompressible flow, as given by Eqs. (9.6). These equations will now be solved numerically. After solving these equations for incompressible fluids and rock, we will consider the more general case with compressible fluids. For the incompressible cases the numerical solution will be conducted by two methods, one method called *IMplicit Pressure, Explicit Saturation*, or IMPES, and another called *fully implicit*. The compressible case will be solved by the IMPES method only.

There exist a range of other solution methods. One notable method is the simultaneous solution method; this method requires a non-zero capillary pressure function, and uses the capillary pressure to express the saturations in terms of pressures, thereby obtaining a set of equations in pressure only. This method is explained in detail by Aziz and Settari (1979). Another notable mention is finite element methods, which is covered in details by Chen et al. (2006). The finite element methods have strengths when it comes to handling more complex grids, boundary conditions and reduced grid orientation effects. Finite element methods are newer than finite difference methods, and have not yet caught on in reservoir simulation.

10.1 Incompressible flow

We will start by emulating the conditions for the Buckley-Leverett equation, thus we want to consider a 1D system where both fluids

and the rock are incompressible.

Consider the set of equations given by Eqs. (9.6):

$$\begin{aligned}\frac{\partial}{\partial x}\left(\frac{kk_{ro}}{\mu_o}\frac{\partial p_o}{\partial x}\right) &= \frac{\partial}{\partial t}\left(\phi s_o\right)\\ \frac{\partial}{\partial x}\left(\frac{kk_{rw}}{\mu_w}\frac{\partial p_w}{\partial x}\right) &= \frac{\partial}{\partial t}\left(\phi s_w\right) \quad .\end{aligned} \tag{10.1}$$

The left side of these equations are of the form given by Eq. (4.53) in the chapter on finite differences:

$$\frac{\partial}{\partial x}\left(a_p(x)\frac{\partial p_p}{\partial x}\right) \quad , \tag{10.2}$$

where $a_p(x) = k(x)k_{rp}(x)/\mu_p(x)$, and p is either o for oil or w for water.

Following the discretization given by Eq. (4.55), we have the following discretization for the left side of the equations above.

$$\frac{\partial}{\partial x}\left(a_p(x)\frac{\partial p_p}{\partial x}\right) \simeq \frac{a_p\left(x+\frac{\Delta x_r}{2}\right)\frac{P_p(x+\Delta x_r)-P_p(x)}{\Delta x_r} - a_p\left(x-\frac{\Delta x_l}{2}\right)\frac{P_p(x)-P_p(x-\Delta x_l)}{\Delta x_l}}{\frac{\Delta x_r+\Delta x_l}{2}} \quad . \tag{10.3}$$

For a 1D grid with varying grid size, we have $\Delta x_r = (\Delta x_i + \Delta x_{i+1})/2$ and $\Delta x_l = (\Delta x_i + \Delta x_{i-1})/2$. Assuming all cells have equal x-length, then $\Delta x_r = \Delta x = \Delta x_l$, and we get

$$\begin{aligned}\frac{\partial}{\partial x}\left(a_p(x)\frac{\partial p_p}{\partial x}\right) \simeq & \\ & \frac{a_p\left(x+\frac{\Delta x}{2}\right)}{(\Delta x)^2}\left(P_p(x+\Delta x)-P_p(x)\right)\\ & -\frac{a_p\left(x-\frac{\Delta x}{2}\right)}{(\Delta x)^2}\left(P_p(x)-P_p(x-\Delta x_l)\right) \quad .\end{aligned} \tag{10.4}$$

From Eq. (10.3) we observe that we need to evaluate a_p at a point between the cell centers, e.g.:

$$a_p\left(x+\frac{\Delta x_r}{2}\right) = \frac{k\left(x+\frac{\Delta x_r}{2}\right)k_{rp}\left(x+\frac{\Delta x_r}{2}\right)}{\mu_p\left(x+\frac{\Delta x_r}{2}\right)} \quad . \tag{10.5}$$

For the permeability, it is common to associate the permeability between two cell centers as the harmonic mean of the permeability in each cell center:

$$k\left(x_i+\frac{\Delta x_r}{2}\right) = \left(\frac{\Delta x_i}{2}+\frac{\Delta x_{i+1}}{2}\right)\frac{1}{\frac{\frac{\Delta x_i}{2}}{k(x_i)}+\frac{\frac{\Delta x_{i+1}}{2}}{k(x_{i+1})}} \quad , \tag{10.6}$$

When considering cells in series, we take the harmonic mean, while cells in parallel gives the arithmetic mean.

When using the index notation, this can be written as

$$k_{i+1/2} = \frac{\Delta x_i + \Delta x_{i+1}}{\frac{\Delta x_i}{k_i}+\frac{\Delta x_{i+1}}{k_{i+1}}} \quad . \tag{10.7}$$

For a constant cell size Δx, this simplifies to

$$k_{i+1/2} = \frac{2}{\frac{1}{k_i} + \frac{1}{k_{i+1}}} \quad . \tag{10.8}$$

Filling Eq. (10.7) into Eq. (10.3), we get

$$\begin{aligned}\frac{\partial}{\partial x}\left(a_p(x)\frac{\partial p_p}{\partial x}\right)(x_i) \simeq \\ \frac{2k_{rp_{i+1/2}}}{\frac{\Delta x_r+\Delta x_l}{2}\mu_{p_{i+1/2}}\left(\frac{\Delta x_i}{k_i}+\frac{\Delta x_{i+1}}{k_{i+1}}\right)}\left(P_{p_{i+1}}-P_{p_i}\right) \\ -\frac{2k_{rp_{i-1/2}}}{\frac{\Delta x_r+\Delta x_l}{2}\mu_{p_{i-1/2}}\left(\frac{\Delta x_i}{k_i}+\frac{\Delta x_{i-1}}{k_{i-1}}\right)}\left(P_{p_i}-P_{p_{i-1}}\right) \quad .\end{aligned} \tag{10.9}$$

This equation is commonly written as

$$\frac{\partial}{\partial x}\left(a_p(x)\frac{\partial p_p}{\partial x}\right)(x_i) \simeq T_{i+\frac{1}{2}}\lambda_{p_{i+\frac{1}{2}}}\left(P_{p_{i+1}}-P_{p_i}\right) - T_{i-\frac{1}{2}}\lambda_{p_{i-\frac{1}{2}}}\left(P_{p_i}-P_{p_{i-1}}\right) \quad , \tag{10.10}$$

where T is called the transmissibility, and

$$\lambda_p = k_{rp}/\mu_p \tag{10.11}$$

is the mobility of phase p. For constant permeability, and cell size Δx, we get the transmissibilities as

$$T_{i-\frac{1}{2}} = T_{i+\frac{1}{2}} = T = \frac{k}{(\Delta x)^2} \tag{10.12}$$

Note the similarity between the expression for the flux over faces in the finite volume formulation, Eq. (7.17), and Eq. (10.10) with the transmissibility Eq. (10.12).

As the relative permeability, and thereby mobility, is dependent on saturation, our discretized equation is dependent on both pressure and saturation, and we need to choose how to approximate the mobility. This is usually done by weighting the mobilities in the upstream and downstream block:

Mobility weighting

$$\lambda_{p_{i+\frac{1}{2}}} = \frac{\alpha\Delta x_i\lambda_{p_i} + (1-\alpha)\Delta x_{i+1}\lambda_{p_{i+1}}}{\alpha\Delta x_i + (1-\alpha)\Delta x_{i+1}} \quad . \tag{10.13}$$

If the grid cells have equal size Δx, then this simplifies to

$$\lambda_{p_{i+\frac{1}{2}}} = \alpha\lambda_{p_i} + (1-\alpha)\lambda_{p_{i+1}} \quad . \tag{10.14}$$

We will only consider two cases. The first is the upstream weighting where we select the mobility in the upstream cell, that is the cell where the flow is coming from. If we assume that $p_i > p_{i+1}$ that means $\alpha = 1$:

Upstream weighting: select the mobility in the upstream cell

$$\lambda_{p_{i+\frac{1}{2}}} = \lambda_{p_i} \quad . \tag{10.15}$$

The second case is the average selection where $\alpha = 1/2$:

Average selection

$$\lambda_{p_{i+\frac{1}{2}}} = \frac{\Delta x_i\lambda_{p_i} + \Delta x_{i+1}\lambda_{p_{i+1}}}{\Delta x_i + \Delta x_{i+1}} \quad . \tag{10.16}$$

For constant grid cell size Δx

$$\lambda_{p_{i+\frac{1}{2}}} = \frac{1}{2}\left(\lambda_{p_i} + \lambda_{p_{i+1}}\right) \quad . \tag{10.17}$$

For the right hand side of Eqs. (10.1) we have

$$\frac{\partial}{\partial t}\left(\phi s_p\right) = \phi \frac{\partial s_p}{\partial t} \quad , \tag{10.18}$$

and we want to use a backward difference quotient of the time derivative, as given by:

$$\left(\phi \frac{\partial s_p}{\partial t}\right)_i^{t+\Delta t} \simeq \phi \frac{S_{p_i}^{t+\Delta t} - S_{p_i}^t}{\Delta t} \quad . \tag{10.19}$$

To simplify notation, we will drop the $t + \Delta t$ superscript:

Superscript t denotes current time, while no superscript denotes the state after next time step, that is at time $t + \Delta t$

$$\left(\phi \frac{\partial s_p}{\partial t}\right)_i \simeq \phi \frac{S_{p_i} - S_{p_i}^t}{\Delta t} \quad . \tag{10.20}$$

As the oil saturation is $s_o = 1 - s_w$, we have that

$$\frac{\partial s_o}{\partial t} = -\frac{\partial s_w}{\partial t} \quad , \tag{10.21}$$

thus we eliminate oil saturation s_o and write the equations as equations of water saturation s_w only.

For simplicity we will in the following assume a constant transmissibility, and the discrete forms of the oil and water equations can now be written as:

We will assume constant permeability and cell size; Eq. (10.12)

$$\begin{aligned} T\lambda_{o_{i+\frac{1}{2}}}(P_{o_{i+1}} - P_{o_i}) - T\lambda_{o_{i-\frac{1}{2}}}(P_{o_i} - P_{o_{i-1}}) &= -\phi \frac{S_{w_i} - S_{w_i}^t}{\Delta t} \\ T\lambda_{w_{i+\frac{1}{2}}}(P_{w_{i+1}} - P_{w_i}) - T\lambda_{w_{i-\frac{1}{2}}}(P_{w_i} - P_{w_{i-1}}) &= \phi \frac{S_{w_i} - S_{w_i}^t}{\Delta t} \quad . \end{aligned} \tag{10.22}$$

Here we use capital letters to indicate that this is part of our numerical scheme, thus the pressures P_i^t and saturation S_w^t are the pressure and saturation in cell i at time t.

The boundary conditions for multi-phase flow are as for one phase flow, but rates and pressures can be specified for each of the phases. Normally, we inject water at constant rate or at constant pressure, and produce oil and water at constant rate or at a constant pressure. For our 1D case, the constant rate corresponds to a constant Darcy velocity.

Constant water injection Darcy velocity is a fairly simple condition to handle. The derivation is similar to the single phase left Neumann boundary condition, Eq. (4.65). First, we have that the left Darcy velocity for water is given by

Boundary conditions for constant water injection rate on left hand side

$$q_l = -\frac{kk_{rw}}{\mu_w}\frac{\partial p_w}{\partial x}\left(x_{1/2}\right) = -a_{w_{1/2}}\frac{\partial p_w}{\partial x}(x_{1/2}) \quad , \tag{10.23}$$

From Eq. (4.56) we then have the following equation for water:

$$\begin{aligned} \frac{\partial}{\partial x}\left(a_w \frac{\partial}{\partial x} p_w\right)_1 &\simeq \frac{a_{w_{1+1/2}}\left(P_{w_2} - P_{w_1}\right)}{(\Delta x)^2} - \frac{a_{w_{1/2}}\frac{\partial p_w}{\partial x}(x_{1/2})}{\Delta x} \\ &= T\lambda_{w_{1+1/2}}\left(P_{w_2} - P_{w_1}\right) + \frac{q_l}{\Delta x} \quad . \end{aligned} \tag{10.24}$$

For oil the Darcy velocity is zero, so $a_{o_{1/2}} = 0$, and

$$\frac{\partial}{\partial x}\left(a_o \frac{\partial}{\partial x} p_o\right)_1 \simeq \frac{a_{o_{1+1/2}}\left(P_{o_2} - P_{o_1}\right)}{(\Delta x)^2} - \frac{a_{o_{1/2}} \frac{\partial p_o}{\partial x}(x_{1/2})}{\Delta x}$$
$$= T\lambda_{o_{1+1/2}}\left(P_{o_2} - P_{o_1}\right) \quad . \tag{10.25}$$

For the right hand side we keep a constant pressure boundary condition. If we in Eq. (10.9) assume a dummy grid block $n+1$ with infinite permeability $k_{n+1} = \infty$ and pressure p_r, we will get a constant pressure p_r on the hand right side:

Constant pressure boundary condition at right hand side outlet

$$\begin{aligned}
&\frac{\partial}{\partial x}\left(a_p(x)\frac{\partial p_p}{\partial x}\right)(x_n) \\
&\simeq \frac{2k_{rp_{n+1/2}}}{\Delta x \mu_p\left(\frac{\Delta x}{k_n} + \frac{\Delta x}{k_{n+1}}\right)}\left(P_{p_{n+1}} - P_{p_n}\right) - \frac{2k_{rp_{n-1/2}}}{\Delta x \mu_p\left(\frac{\Delta x}{k_n} + \frac{\Delta x}{k_{n-1}}\right)}\left(P_{p_n} - P_{p_{n-1}}\right) \\
&= \frac{2kk_{rp_{n+1/2}}}{(\Delta x)^2\mu_p}\left(p_r - P_{p_n}\right) - \frac{kk_{rp_{n-1/2}}}{(\Delta x)^2\mu_p}\left(P_{p_n} - P_{p_{n-1}}\right) \\
&\simeq 2T\lambda_{p_{n+1/2}}\left(p_r - P_{p_n}\right) - T\lambda_{p_{n-1/2}}\left(P_{p_n} - P_{p_{n-1}}\right) \quad .
\end{aligned} \tag{10.26}$$

If we use the capillary pressure to replace all the wetting pressures P_w by non-wetting pressures P_n, then in the equations above the non-wetting pressure P_n and wetting saturation S_w are the unknowns to be solved for. All the mobilities, and the capillary pressure, are functions of the saturation. Thus, to solve the equations we need to calculate the mobilities and capillary pressures, and we can not calculate these before we know the saturation. In the more general case considered later, where we allow for compressible fluids and compressible rock, the mobilities and porosity are also a function of pressure, and we have extra coefficients also dependent on pressure and saturation.

As we considered the Buckley-Leverett equation without capillary pressure, we will use $p_c = 0$ in the following. Thus the non-wetting and wetting pressures are equal; $p_n = p = p_w$. The discrete forms of the non-wetting and wetting equations, Eqs. (10.22), can then be simplified to:

$$\begin{aligned}
T\lambda_{n_{i+\frac{1}{2}}}(P_{i+1} - P_i) - T\lambda_{n_{i-\frac{1}{2}}}(P_i - P_{i-1}) &= -\phi\frac{S_{w_i} - S^t_{w_i}}{\Delta t} \\
T\lambda_{w_{i+\frac{1}{2}}}(P_{i+1} - P_i) - T\lambda_{w_{i-\frac{1}{2}}}(P_i - P_{i-1}) &= \phi\frac{S_{w_i} - S^t_{w_i}}{\Delta t} \quad .
\end{aligned} \tag{10.27}$$

Adding these two equations then gives the equation for pressure as

$$T\left(\lambda_{n_{i+\frac{1}{2}}} + \lambda_{w_{i+\frac{1}{2}}}\right)(P_{i+1} - P_i) - T\left(\lambda_{n_{i-\frac{1}{2}}} + \lambda_{w_{i-\frac{1}{2}}}\right)(P_i - P_{i-1}) = 0 \quad . \tag{10.28}$$

Note that, since the mobilities in the incompressible case are functions of saturation only, this equation does not involve the pressure at the previous time step. The reason for this is that pressure travels (diffuse) infinitely fast. Thus the pressure distribution can be obtained by solving the pressure equation above.

The pressure equation only involve pressure after next time step

We can define the total mobility λ_t as $\lambda_o + \lambda_w$ and rewrite

Total mobility

Eq. (10.28) as

$$T\lambda_{t_{i-\frac{1}{2}}}P_{i-1} - T\left(\lambda_{t_{i-\frac{1}{2}}} + \lambda_{t_{i+\frac{1}{2}}}\right)P_i + T\lambda_{t_{i+\frac{1}{2}}}P_{i+1} = 0 \quad . \tag{10.29}$$

One possible solution method is to iterate on the solution and update coefficients until convergence is reached. This will be investigated in subsection 10.1.3. We will start by looking at the simpler IMPES method; this method evaluates the mobilities at time t instead of time $t + \Delta t$.

10.1.1 IMPES solution to incompressible flow

IMPES = IMplicit Pressure Explicit Saturation

IMPES stands for "IMplicit Pressure Explicit Saturation", and since the mobilities, in the incompressible case, are functions of saturation only, they are evaluated explicitly, i.e. at time t. We will, as before, denote this by the superscript t. We can regain generality with respect to a variable permeability field by defining the product of transmissibility and mobility as Λ:

$$\Lambda^t_{i\pm\frac{1}{2}} = T_{i\pm\frac{1}{2}}\lambda^t_{t_{i\pm\frac{1}{2}}} \quad . \tag{10.30}$$

Pressures are found by an implicit method, thus evaluated the at the next time step $t + \Delta t$. Eq. (10.29) is then

$$\Lambda^t_{i-\frac{1}{2}}P_{i-1} - \left(\Lambda^t_{i-\frac{1}{2}} + \Lambda^t_{i+\frac{1}{2}}\right)P_i + \Lambda^t_{i+\frac{1}{2}}P_{i+1} = 0 \quad . \tag{10.31}$$

We thus have a system of linear equations where the unknowns are pressures only.

The system of equations given by Eq. (10.31), and the boundary conditions (10.24), (10.25), and (10.26) can be expressed in matrix form as

$$\mathbf{AP} + \mathbf{e} = 0 \quad , \tag{10.32}$$

where

$$\mathbf{A} = \begin{bmatrix}
-\Lambda^t_{1+\frac{1}{2}} & \Lambda^t_{1+\frac{1}{2}} & 0 & 0 & \cdots & 0 & 0 & 0 \\
\Lambda^t_{2-\frac{1}{2}} & -\left(\Lambda^t_{2-\frac{1}{2}} + \Lambda^t_{2+\frac{1}{2}}\right) & \Lambda^t_{2+\frac{1}{2}} & 0 & \cdots & 0 & 0 & 0 \\
0 & \Lambda^t_{3-\frac{1}{2}} & -\left(\Lambda^t_{3-\frac{1}{2}} + \Lambda^t_{3+\frac{1}{2}}\right) & \Lambda^t_{3+\frac{1}{2}} & 0 & \cdots & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & 0 & \cdots & -\left(\Lambda^t_{n-2-\frac{1}{2}} + \Lambda^t_{n-2+\frac{1}{2}}\right) & \Lambda^t_{n-2+\frac{1}{2}} & 0 \\
0 & 0 & 0 & 0 & \cdots & \Lambda^t_{n-1-\frac{1}{2}} & -\left(\Lambda^t_{n-1-\frac{1}{2}} + \Lambda^t_{n-1+\frac{1}{2}}\right) & \Lambda^t_{n-1+\frac{1}{2}} \\
0 & 0 & 0 & 0 & \cdots & 0 & \Lambda^t_{n-\frac{1}{2}} & -\left(\Lambda^t_{n-\frac{1}{2}} + \Lambda^t_{n+\frac{1}{2}}\right)
\end{bmatrix} , \tag{10.33}$$

and

$$\mathbf{e} = \begin{bmatrix} \frac{q_l}{\Delta x} \\ 0 \\ \vdots \\ 0 \\ 2\Lambda^t_{n+\frac{1}{2}}p_r \end{bmatrix} \quad . \tag{10.34}$$

After solving for pressure, we can then update the saturation from either equation in Eqs. (10.22):

$$S_{w_i} = S^t_{w_i} - \frac{\Delta t}{\phi}\left(T_{i+\frac{1}{2}}\lambda^t_{n\,i+\frac{1}{2}}(P_{i+1} - P_i) - T_{i-\frac{1}{2}}\lambda^t_{n\,i-\frac{1}{2}}(P_i - P_{i-1})\right) \quad . \tag{10.35}$$

10.1.2 IMPES – Python implementation

In this section we will implement the method outlined in section 10.1.1, and compare the numerical solution with the Buckley–Leverett equation (Section 9.3).

The simulator is implemented as a class `Simulator1DIMPES` in a file `IMPES1D.py`. Class instances are initialized with default values. This class has two methods; `doTimestep` and `simulateTo`. The following code creates a 200 m long model with 100 cells, and simulate for 30 days with default parameters:

Each *instance* of a class has its own version of the data

```
from IMPES1D import  Simulator1DIMPES
from conversion import daysToSeconds,secondsToDays

simulator = Simulator1DIMPES(100,200.0)
simulator.simulateTo(daysToSeconds(30))

print(simulator.pressure)
print(simulator.saturation)
```

The conversion is a straight forward conversion from days to seconds, as we use SI units in the simulator itself.

Initialization of Simulator1DIMPES instances is, as for all classes, performed by the `__init__` method:

```
    def __init__(self,Ncells,length):
        '''
        Args:
           Ncells:   Number of cells
           length:   Total length [m]
        '''

        self.Ncells = Ncells

        self.length = length
        self.deltaX = length/Ncells

        self.poro  = 0.2*np.ones(Ncells)
        #This next line will also define the transmissibilities
        self._perm = self.setPermeabilities(1.0E-13*np.ones(Ncells
    ↪ ))

        self.pressure   = 1.0E7*np.ones(Ncells)
        self.saturation = 0.2*np.ones(Ncells)

        self.rightPressure     = 1.0E7
        self.leftDarcyVelocity = 2.315E-6 * self.poro[0]

        self.mobilityWeighting = 1.0
        self.deltat = daysToSeconds(1)
```

```
        self.time = 0.0

        self.oilViscosity   = 2.0E-3
        self.waterViscosity = 1.0E-3

        self.relpermWater = coreyWater(2.0,0.4,0.2,0.2)
        self.relpermOil   = coreyOil(2.0,0.2,0.2)
```

We see that the members `relpermWater` and `relpermOil` are initiated with instances of the classes `coreyWater` and `coreyOil` respectively. These classes are defined in the file `corey.py` as

```
class coreyWater:
    '''
    Water relative permeability
    '''
    def __init__(self,Nw,Krwo,Swirr,Sorw):
        '''
        Args:
           Nw:    Exponent
           Krwo:  Relperm at 1-S_{orw}
           Swirr: S_{wi}
           Sorw:  S_{orw}
        '''
        self.Nw   = Nw
        self.Krwo = Krwo
        self.Swirr = Swirr
        self.Sorw = Sorw

    def __call__(self,Sw):
        nSw = normSw(Sw,self.Swirr,self.Sorw)
        return  self.Krwo*nSw**self.Nw
```

and

```
class coreyOil:

    def __init__(self,No,Swirr,Sorw):
        '''
        Args:
           No:    Exponent
           Swirr: S_{wi}
           Sorw:  S_{orw}
        '''
        self.No = No
        self.Swirr = Swirr
        self.Sorw = Sorw


    def __call__(self,Sw):
        nSw = normSw(Sw,self.Swirr,self.Sorw)
        return (1.0-nSw)**self.No
```

Any change in the permeability should trigger a recalculation of the transmissibilities. We therefore calculate the transmissibilities every time we set the permeability values. The permeability field is decleared as a *private member* `_perm`. The permeability field is changed through the `setPermeabilities` method, which then also set the transmissibilities.

A *private member* of a class is classically a member that is denied access from outside the class. Private instances does not actually exist in Python, however, it is a convention that a name prefixed with an underscore (such as `_perm`) should be treated as private.

```
    def setPermeabilities(self,permVector):
        '''
        Set permeabilities

        Args:
           permVector: A numpy array of length
           self.Ncells with perm values
        '''

        self._perm = permVector

        self._Tran = (2.0/(1.0/self._perm[:-1]+1.0/self._perm[1:])
    ↪ )/self.deltaX**2
        self._TranRight = self._perm[-1]/self.deltaX**2
```

The actual simulation of one time step is performed by the method `doTimestep`

```
    def doTimestep(self):
        '''
        Do one time step of length self.deltat
        '''

        mobOil   = self.relpermOil(self.saturation)/self.
    ↪ oilViscosity
        mobWater = self.relpermWater(self.saturation)/self.
    ↪ waterViscosity

        upW   = self.mobilityWeighting
        downW = 1.0-upW
        mobOilW   = mobOil[:-1]*upW   +  mobOil[1:]*downW
        mobWaterW = mobWater[:-1]*upW +  mobWater[1:]*downW

        oilTrans   = self._Tran*mobOilW
        waterTrans = self._Tran*mobWaterW
        oilTransRight   = self._TranRight*mobOil[-1]
        waterTransRight = self._TranRight*mobWater[-1]
        totalTrans = oilTrans + waterTrans
        totalTransRight = oilTransRight + waterTransRight

        # ----------------------------
        # Solve implicit for pressure:
        #
        # We solve a linear system matrixA pressure = vectorE
        #
        # Since the system is small and 1D we can buid a
        # dense matrix and use explicit inversion

        #    --- Build matrixA:

        matrixA = np.zeros((self.Ncells,self.Ncells))

        # First row
        matrixA[0,0] = -totalTrans[0]
        matrixA[0,1] =  totalTrans[0]

        # Middle rows
        for ii in np.arange(1,self.Ncells-1):
```

```
            matrixA[ii,ii-1] =  totalTrans[ii-1]
            matrixA[ii,ii]   = -totalTrans[ii-1]-totalTrans[ii]
            matrixA[ii,ii+1]  = totalTrans[ii]

        # Last row
        matrixA[-1,-2] =    totalTrans[-1]
        matrixA[-1,-1] = -2*totalTransRight - totalTrans[-1]

        #    ------

        #    --- Build vectorE:

        vectorE     = np.zeros(self.Ncells)
        vectorE[0]  = -self.leftDarcyVelocity/self.deltaX
        vectorE[-1] = -2.0*totalTransRight*self.rightPressure

        #    ------

        #    --- Solve linear system:

        matrixAInv = np.linalg.inv(matrixA)
        pressure = np.dot(matrixAInv,vectorE)

        # --------------------------------
        # Solve explicitly for saturation:

        dtOverPoro = self.deltat/self.poro

        self.saturation[1:-1] = self.saturation[1:-1] - dtOverPoro
    ↪ [1:-1]*(oilTrans[1:]*(pressure[2:]-pressure[1:-1]) +
    ↪ oilTrans[:-1]*(pressure[:-2]-pressure[1:-1]))
        self.saturation[0]    = self.saturation[0]  - dtOverPoro
    ↪ [0]*oilTrans[0]*(pressure[1]-pressure[0])
        self.saturation[-1]   = self.saturation[-1] + dtOverPoro
    ↪ [-1]*(2*waterTransRight*(self.rightPressure-pressure[-1])-
    ↪ waterTrans[-1]*(pressure[-1]-pressure[-2]))

        maxsat = 1.0-self.relpermOil.Sorw
        minsat = self.relpermOil.Swirr
        self.saturation[ self.saturation>maxsat ] = maxsat
        self.saturation[ self.saturation<minsat ] = minsat

        # --------------------------------

        self.pressure = pressure

        self.time = self.time + self.deltat
```

The method `simulateTo` simply calls `doTimestep` sufficiently many times to advance the simulation to a prescribed time

```
    def simulateTo(self,time):
        '''
        Progress simulation to specific time with
        a constant timestep self.deltat

        Args:
           time: Time to advance to [s]
```

```
        '''

        baseDeltat = self.deltat
        while self.time < time:
            if self.time + baseDeltat >= time:
                self.deltat = time - self.time
                self.doTimestep()
                self.deltat = baseDeltat
                self.time = time
            else:
                self.doTimestep()
```

Fig. 10.1 shows an example of an IMPES solution using an upstream mobility weighting. We see that the front moves forward with time. As explained in the section on numerical diffusion, Sec. 9.4, the IMPES method is expected to have a significant numerical diffusion. This will smear out the front, so it will not be as sharp as in the analytical solution. This smearing of the front is clearly visible in the figure.

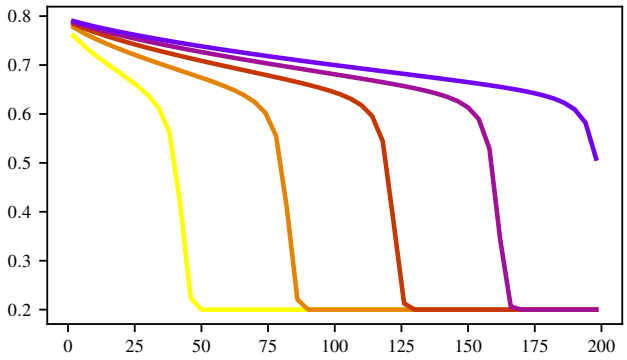


Figure 10.1: The IMPES solution using an upstream mobility weighting. The numerical model has 50 grid cells, the time step size is 1 day, and a line is plotted for each 100 days.

To see the effect of mobility weighting, we have compared the upstream weighting with a weighting of $\alpha = 0.85$ in Fig. 10.2. This figure also contain the Buckley-Leverett solution.

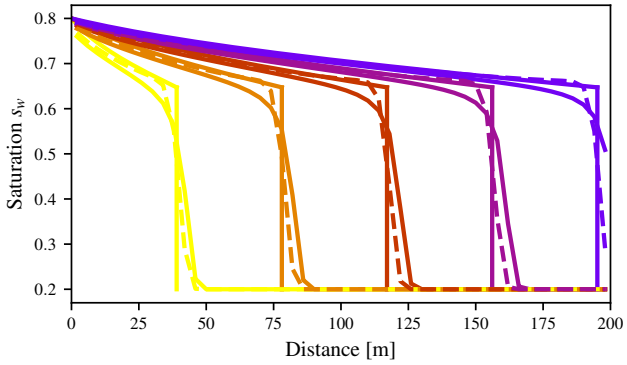


Figure 10.2: The Buckley–Leverett solution compared to the IMPES solution for an upstream mobility weighting and a weighting of $\alpha = 0.85$. The numerical model has 50 grid cells, the time step size is 1 day, and a line is plotted for each 100 days.

In this figure the $\alpha = 0.85$ seems closer to the analytical solution, and could therefore be preferable. If we try even smaller mobility weighting α the solutions get truly un-physical.

10.1.3 *Fully–implicit solution to incompressible flow*

The fully-implicit method needs to solve Eqs. (10.22) simultaneously. As the mobilities are functions of saturation, through relative permeability, this equation system is non-linear. In order to solve the system of non-linear equations we will use the iterative Newton method.

For a single-variable function f where the derivative f' is known, a root of f can be found by the iterative Newton method

The Newton method is an iterative method for solving non-linear equations.

$$x^{(m+1)} = x^{(m)} - \frac{f(x^{(m)})}{f'(x^{(m)})} \quad , \tag{10.36}$$

where $x^{(m)}$ is the m'th iteration for x, which is consequently moving closer to a root for f. Thus, for a sufficiently large m we have $f(x^{(m)}) \simeq 0$.

Newton's method can be extended to a set of k variables and k real-valued functions. The k variables can be written on vector form as $\mathbf{x} = (x_1, x_2, \ldots, x_k)$, and the k functions can be written as a single function $\mathbf{F} = (f_1, f_2, \ldots, f_k)$, where $f_i \colon \mathbb{R}^k \to \mathbb{R}$, thus $\mathbf{F} \colon \mathbb{R}^k \to \mathbb{R}^k$. We can find the root of $\mathbf{F}$ by the following iterative procedure:

The Newton method can be extended to *systems* of non-linear equations by substituting the derivative with the Jacobian.

$$\mathbf{x}^{(m+1)} = \mathbf{x}^{(m)} - \mathbf{J}_F^{-1}(\mathbf{x}^{(m)}) F(\mathbf{x}^{(m)})) \quad , \tag{10.37}$$

where $\mathbf{J}_F^{-1}$ is the inverse of the Jacobian matrix

$$\mathbf{J}_F = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \quad . \tag{10.38}$$

We now want to reformulate our finite difference formulations, Eqs. (10.22), to obtain functions in pressure and saturation which become zero-valued for the pressure and saturation solution we seek. For this end we move all terms over to one side:

$$\begin{aligned} T\lambda_{n_{i+\frac{1}{2}}}(P_{n_{i+1}} - P_{n_i}) - T\lambda_{n_{i-\frac{1}{2}}}(P_{n_i} - P_{n_{i-1}}) + \phi\frac{S_{w_i} - S_{w_i}^t}{\Delta t} = 0 \\ T\lambda_{w_{i+\frac{1}{2}}}(P_{w_{i+1}} - P_{w_i}) - T\lambda_{w_{i-\frac{1}{2}}}(P_{w_i} - P_{w_{i-1}}) - \phi\frac{S_{w_i} - S_{w_i}^t}{\Delta t} = 0 \end{aligned} \quad . \tag{10.39}$$

The left side of these equations are zero-valued exactly when the free variables, i.e., the pressures and saturations, are a solution. Whenever we have a set of pressures and saturation values that are not a solution, then the equations above do not hold, thus the left sides are not zero. The difference from zero is called the residual and denoted by R:

$$\begin{aligned} R_{n_i} &= T\lambda_{n_{i+\frac{1}{2}}}(P_{n_{i+1}} - P_{n_i}) - T\lambda_{n_{i-\frac{1}{2}}}(P_{n_i} - P_{n_{i-1}}) + \phi\frac{S_{w_i} - S_{w_i}^t}{\Delta t} \\ R_{w_i} &= T\lambda_{w_{i+\frac{1}{2}}}(P_{w_{i+1}} - P_{w_i}) - T\lambda_{w_{i-\frac{1}{2}}}(P_{w_i} - P_{w_{i-1}}) - \phi\frac{S_{w_i} - S_{w_i}^t}{\Delta t} \end{aligned} \quad . \tag{10.40}$$

As an initial guess for the solution of Eqs. (10.22) it is common to use the saturation and pressure at the previous time step. Except for trivial cases, the initial guess will give non-zero residuals. We can then use the Newton's method, an iterative process, to converge to the root of the residuals.

While the fully-implicit method holds in general, we will now restrict ourselves to try to solve the Buckley-Leverett case; thus assume $p_c = p_n - p_w = 0$, so that we only have one pressure $P_n = P_w = P$. Let $S_w^{(m)}$ be the water saturation value, $P^{(m)}$ be the pressure, and $\lambda_p^{(m)}$ the mobility of phase p at iteration m. The residuals are then of the form

$$\begin{aligned} R_{n_i}^{(m)} &= T\lambda_{n_{i+\frac{1}{2}}}^{(m)}\left(P_{i+1}^{(m)} - P_i^{(m)}\right) - T\lambda_{n_{i-\frac{1}{2}}}^{(m)}\left(P_i^{(m)} - P_{i-1}^{(m)}\right) + \phi\frac{S_{w_i}^{(m)} - S_{w_i}^t}{\Delta t} \\ R_{w_i}^{(m)} &= T\lambda_{w_{i+\frac{1}{2}}}^{(m)}\left(P_{i+1}^{(m)} - P_i^{(m)}\right) - T\lambda_{w_{i-\frac{1}{2}}}^{(m)}\left(P_i^{(m)} - P_{i-1}^{(m)}\right) - \phi\frac{S_{w_i}^{(m)} - S_{w_i}^t}{\Delta t} \end{aligned} \quad . \tag{10.41}$$

Note that the residuals are not considered a function of the mobilities, as these are functions of pressure and saturation. The vector of residuals evaluated at iteration m, $\mathbf{R}^{(m)} = \mathbf{R}(\mathbf{P}^{(m)}, \mathbf{S_w^{(m)}})$, is a measure for how far we are from the solution; a measure we want to reduce to zero (or, in practice, below a pre-described threshold) by iterating using Newton's method.

$\mathbf{R}^{(m)}$: The vector of residuals at iteration m
$\mathbf{P}^{(m)}$: The vector of pressure values at iteration m
$\mathbf{S}^{(m)}$: The vector of saturation values at iteration m

We need to add boundary conditions at the inlet and outlet; The residual at the first grid cell (inlet) is (see Eq. (10.23), and Eq. (10.24))

Boundary conditions

$$\begin{aligned} R_{n_1}^{(m)} &= T\lambda_{n_{1+\frac{1}{2}}}^{(m)}\left(P_2^{(m)} - P_1^{(m)}\right) + \phi\frac{S_{w_1}^{(m)} - S_{w_1}^t}{\Delta t} \\ R_{w_1}^{(m)} &= T\lambda_{w_{1+\frac{1}{2}}}^{(m)}\left(P_2^{(m)} - P_1^{(m)}\right) + \frac{q_l}{\Delta x} - \phi\frac{S_{w_1}^{(m)} - S_{w_1}^t}{\Delta t} \end{aligned} \quad , \tag{10.42}$$

while the residual at the last cell (outlet) is (see Eq. (10.26))

$$\begin{aligned} R_{n_n}^{(m)} &= 2T\lambda_{n_{n+\frac{1}{2}}}^{(m)}\left(p_r - P_n^{(m)}\right) - T\lambda_{n_{n-\frac{1}{2}}}^{(m)}\left(P_n^{(m)} - P_{n-1}^{(m)}\right) + \phi\frac{S_{w_n}^{(m)} - S_{w_n}^t}{\Delta t} \\ R_{w_n}^{(m)} &= 2T\lambda_{w_{n+\frac{1}{2}}}^{(m)}\left(p_r - P_n^{(m)}\right) - T\lambda_{w_{n-\frac{1}{2}}}^{(m)}\left(P_n^{(m)} - P_{n-1}^{(m)}\right) - \phi\frac{S_{w_n}^{(m)} - S_{w_n}^t}{\Delta t} \end{aligned} \quad . \tag{10.43}$$

The terms in the residual that stem from the spatial discretization (proportional to T) are called flow terms, while the terms that stem from time discretization (proportional to $\frac{1}{\Delta t}$) are called accumulation terms. Terms from the boundary conditions ($\frac{q_l}{\Delta x}$ in Eq. (10.42), and $2T\lambda_{w_{n+\frac{1}{2}}}^{(m)}\left(p_r - P_n^{(m)}\right)$ in Eq. (10.43)) are called source terms. Note that source terms that extract mass from the reservoir are often alternatively called sink terms, reserving the word source term for injection. Terms that represent wells are the main source terms in reservoir simulation. Wells are either governed by rate, giving terms proportional to the rate similar to the source term in Eq. (10.42), or by pressure, giving terms proportional

Flow terms

Accumulation terms

Source terms

to the difference between well pressure and cell pressure similar to the source term in Eq. (10.43).

For the Newton method, we write the unknowns, i.e., the pressures P_i and saturation values S_{w_i}, as a vector $\mathbf{X}$. We also write the residuals as a vector $\mathbf{R}$:

$$\mathbf{X} = \begin{bmatrix} P_1 \\ S_{w_1} \\ P_2 \\ S_{w_2} \\ \vdots \\ P_n \\ S_{w_n} \end{bmatrix}, \quad \mathbf{R} = \begin{bmatrix} R_{n_1} \\ R_{w_1} \\ R_{n_2} \\ R_{w_2} \\ \vdots \\ R_{n_n} \\ R_{w_n} \end{bmatrix} \quad . \tag{10.44}$$

Observe that $\mathbf{X}$ and $\mathbf{R}$ are of equal length $2n$, thus we have the same number of equations as unknowns.

$\mathbf{X}^{(m)}$: The vector of unknowns at iteration m

If we denote the m'th iteration as $\mathbf{X}^{(m)}$, then a single iteration in Newton's method is given by (see Eq. (10.37))

$$\mathbf{X}^{(m+1)} = \mathbf{X}^{(m)} - \mathbf{J}_R^{-1}(\mathbf{X}^{(m)})\mathbf{R}^m \quad , \tag{10.45}$$

where $\mathbf{J}_R$ is the Jacobian matrix:

$$\mathbf{J}_R = \begin{bmatrix} \frac{\partial R_{n_1}}{\partial P_1} & \frac{\partial R_{n_1}}{\partial S_{w_1}} & \frac{\partial R_{n_1}}{\partial P_2} & \frac{\partial R_{n_1}}{\partial S_{w_2}} & \cdots & \frac{\partial R_{n_1}}{\partial S_{w_{n-1}}} & \frac{\partial R_{n_1}}{\partial P_n} & \frac{\partial R_{n_1}}{\partial S_{w_n}} \\ \frac{\partial R_{w_1}}{\partial P_1} & \frac{\partial R_{w_1}}{\partial S_{w_1}} & \frac{\partial R_{w_1}}{\partial P_2} & \frac{\partial R_{w_1}}{\partial S_{w_2}} & \cdots & \frac{\partial R_{w_1}}{\partial S_{w_{n-1}}} & \frac{\partial R_{w_1}}{\partial P_n} & \frac{\partial R_{w_1}}{\partial S_{w_n}} \\ \frac{\partial R_{n_2}}{\partial P_1} & \frac{\partial R_{n_2}}{\partial S_{w_1}} & \frac{\partial R_{n_2}}{\partial P_2} & \frac{\partial R_{n_2}}{\partial S_{w_2}} & \cdots & \frac{\partial R_{n_2}}{\partial S_{w_{n-1}}} & \frac{\partial R_{n_2}}{\partial P_n} & \frac{\partial R_{n_2}}{\partial S_{w_n}} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \frac{\partial R_{w_{n-1}}}{\partial P_1} & \frac{\partial R_{w_{n-1}}}{\partial S_{w_1}} & \frac{\partial R_{w_{n-1}}}{\partial P_2} & \frac{\partial R_{w_{n-1}}}{\partial S_{w_2}} & \cdots & \frac{\partial R_{w_{n-1}}}{\partial S_{w_{n-1}}} & \frac{\partial R_{w_{n-1}}}{\partial P_n} & \frac{\partial R_{w_{n-1}}}{\partial S_{w_n}} \\ \frac{\partial R_{n_n}}{\partial P_1} & \frac{\partial R_{n_n}}{\partial S_{w_1}} & \frac{\partial R_{n_n}}{\partial P_2} & \frac{\partial R_{n_n}}{\partial S_{w_2}} & \cdots & \frac{\partial R_{n_n}}{\partial S_{w_{n-1}}} & \frac{\partial R_{n_n}}{\partial P_n} & \frac{\partial R_{n_n}}{\partial S_{w_n}} \\ \frac{\partial R_{w_n}}{\partial P_1} & \frac{\partial R_{w_n}}{\partial S_{w_1}} & \frac{\partial R_{w_n}}{\partial P_2} & \frac{\partial R_{w_n}}{\partial S_{w_2}} & \cdots & \frac{\partial R_{w_n}}{\partial S_{w_{n-1}}} & \frac{\partial R_{w_n}}{\partial P_n} & \frac{\partial R_{w_n}}{\partial S_{w_n}} \end{bmatrix}, \tag{10.46}$$

A small digression on numerical solutions to linear systems is in place here: Although the explicit expression (Eq. (10.45)) for the updated values $\mathbf{X}^{(m+1)}$ is mathematically correct and simple it is not used in practice; Instead the update is formulated as

$$\mathbf{X}^{(m+1)} = \mathbf{X}^{(m)} + \mathbf{\Delta}^{(m+1)} \quad , \tag{10.47}$$

where $\mathbf{\Delta}^{(m+1)}$ is the solution of a system of linear equations

A system of linear equations is solved for each Newton iteration

$$\mathbf{J}_R^{(m)}\mathbf{\Delta}^{(m+1)} + \mathbf{R}^{(m)} = 0 \quad . \tag{10.48}$$

We will see below that the Jacobian is a *sparse* matrix with $(a+1)n$ nonzero elements, where a is the average number of cell neighbors, while the inverse Jacobian is *dense* with n^2 elements. Sparse linear systems like Eq. (10.48) are solved using an iterative method[1], and there is no need to store the inverse Jacobian, which for large systems would be prohibitively huge. The inverse Jacobian is also actually found by solving a system of linear equations with n right hand sides, while in Eq. (10.48) there is only one ($-\mathbf{R}^{(m)}$). The

[1] The iterations used by the linear solver are known as inner- or linear-iterations, and should not be confused with the Newton iterations.

formulation as a system of linear equations is thus always more efficient than Eq. (10.45).

The residuals for a given cell only depend on the state of that cell and its neighbors. As a consequence, most of the elements in the Jacobian matrix, which are partial derivatives of the residual in a cell with respect to the state of another cell, are zero. The Jacobian matrix $\mathbf{J}_R$ has a block structure similar to the structure of the **A**-matrix in Eq. (10.32) for the IMPES solution. The Jacobian matrix consist of the blocks

$$\mathbf{J}_{i,j} = \begin{bmatrix} \frac{\partial R_{n_i}}{\partial P_{n_j}} & \frac{\partial R_{n_i}}{\partial S_{w_j}} \\ \frac{\partial R_{w_i}}{\partial P_{n_j}} & \frac{\partial R_{w_i}}{\partial S_{w_j}} \end{bmatrix} \quad , \tag{10.49}$$

where $\mathbf{J}_{i,j} = 0$ for all $i \notin \{i-1, i, i+1\}$. This gives

$$\mathbf{J}_R = \begin{bmatrix} \mathbf{J}_{1,1} & \mathbf{J}_{1,2} & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ \mathbf{J}_{2,1} & \mathbf{J}_{2,2} & \mathbf{J}_{2,3} & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & \mathbf{J}_{3,2} & \mathbf{J}_{3,3} & \mathbf{J}_{3,4} & 0 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & \mathbf{J}_{n-1,n-2} & \mathbf{J}_{n-1,n-1} & \mathbf{J}_{n-1,n} \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \mathbf{J}_{n,n-1} & \mathbf{J}_{n,n} \end{bmatrix} . \tag{10.50}$$

We can calculate the nonzero elements in the Jacobian matrix from Eqs. (10.41), (10.42), and (10.43). We will use that the mobilities are functions of saturation through the relative permeability, but are independent of pressure for an incompressible system. The porosity is also constant.

The derivatives of the residual of phase p with respect to pressure are

$$\begin{aligned} \frac{\partial R_{p_i}}{\partial P_{i-1}} &= T\lambda^{(m)}_{p_{i-\frac{1}{2}}} \\ \frac{\partial R_{p_i}}{\partial P_i} &= -T\lambda^{(m)}_{p_{i+\frac{1}{2}}} - T\lambda^{(m)}_{p_{i-\frac{1}{2}}} \\ \frac{\partial R_{p_i}}{\partial P_{i+1}} &= T\lambda^{(m)}_{p_{i+\frac{1}{2}}} \end{aligned} \quad , \tag{10.51}$$

for all interior cells,

$$\begin{aligned} \frac{\partial R_{p_1}}{\partial P_1} &= -T\lambda^{(m)}_{p_{1+\frac{1}{2}}} \\ \frac{\partial R_{p_1}}{\partial P_2} &= T\lambda^{(m)}_{p_{1+\frac{1}{2}}} \end{aligned} \quad , \tag{10.52}$$

for the first cell, and

$$\begin{aligned} \frac{\partial R_{p_n}}{\partial P_{n-1}} &= T\lambda^{(m)}_{p_{n-\frac{1}{2}}} \\ \frac{\partial R_{p_n}}{\partial P_n} &= -2T\lambda^{(m)}_{p_{n+\frac{1}{2}}} - T\lambda^{(m)}_{p_{n-\frac{1}{2}}} \end{aligned} \quad , \tag{10.53}$$

for the last cell.

We will use upstream weighting for the mobilities. The mobility at an interface is then a function of the saturation of the upstream cell only. In general, what is the upstream cell is determined by the,

dynamically changing, pressure field. In our simple one dimensional case it is however always the left cell. The derivatives of the residual of phase p with respect to saturation are then[2]

$$\begin{aligned}
\frac{\partial R_{p_i}}{\partial S_{w_{i-1}}} &= -\frac{T}{\mu} k'_{rp}(S^{(m)}_{w_{i-1}}) \left(P^{(m)}_i - P^{(m)}_{i-1}\right) \\
\frac{\partial R_{p_i}}{\partial S_{w_i}} &= \frac{T}{\mu} k'_{rp}(S^{(m)}_{w_i}) \left(P^{(m)}_{i+1} - P^{(m)}_i\right) \pm \frac{\phi}{\Delta t} \\
&\quad (\pm \text{ is } + \text{ for non-wetting and } - \text{ for wetting}) \\
\frac{\partial R_{p_i}}{\partial S_{w_{i+1}}} &= 0
\end{aligned} \quad , \tag{10.54}$$

for all interior cells,

$$\begin{aligned}
\frac{\partial R_{p_1}}{\partial S_{w_1}} &= \frac{T}{\mu} k'_{rp}(S^{(m)}_{w_1}) \left(P^{(m)}_2 - P^{(m)}_1\right) \pm \frac{\phi}{\Delta t} \\
\frac{\partial R_{p_1}}{\partial S_{w_2}} &= 0
\end{aligned} \quad , \tag{10.55}$$

for the first cell, and

$$\begin{aligned}
\frac{\partial R_{p_n}}{\partial S_{w_{n-1}}} &= -\frac{T}{\mu} k'_{rp}(S^{(m)}_{w_{n-1}}) \left(P^{(m)}_n - P^{(m)}_{n-1}\right) \\
\frac{\partial R_{p_n}}{\partial S_{w_n}} &= \frac{2T}{\mu} k'_{rp}(S^{(m)}_{w_n}) \left(p_r - P^{(m)}_n\right) \pm \frac{\phi}{\Delta t}
\end{aligned} \quad , \tag{10.56}$$

for the last cell.

A solution, with Corey functions (Eq. (9.7)) for the relative permeability, is shown in Fig. 10.3. This figure contains both the Buckley-Leverett solution, the solution from OPM-Flow, and the solution using a fully implicit Python code. As can be observed, the numerical methods are smeared out compared to the Buckley-Leverett solution. However, there is small differences between the Python code and OPM-Flow.

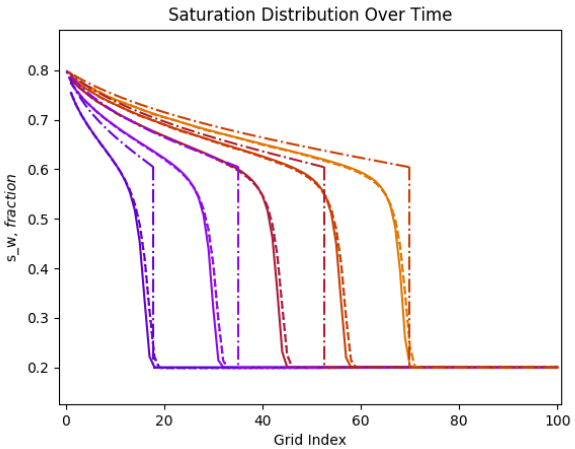


Figure 10.3: A comparison between a solution using a Python script versus both the analytical Buckley-Leverett solution and a solution using OPM-Flow. The full-drawn line is the Python solution, the dashed line is OPM-Flow, while the dash-dot line is the Buckley-Leverett solution. This plot is created by Fadhil Berylian.

This plot indicates a common problem with the fully implicit method: Numerical diffusion. Comparing with the result from

[2] $k'_{rp}(\cdot)$ is the derivative of the relative permeability function.

the IMPES method, the fully implicit method has more numerical diffusion. This is consistent with the results obtained in section 9.4.

The fully implicit method is still preferable over the IMPES method for most realistic reservoir simulation problems due to less restrictive stability criteria. One can therefore use larger time steps. However, the issue of numerical diffusion could make IMPES preferable in special cases.

10.2 Compressible flow

For a compressible fluid, we cannot eliminate the density from the two–phase flow equations. We will also include rock compressibility, so that the porosity is non-constant. Reservoir temperature is assumed constant so density and viscosity are known functions of pressure for each of the immiscible phases. We will in this section also include capillary pressure.

In this chapter, and in chapter 9, we assume that the phases are immiscible in the sense that their composition remain constant and there is no exchange of components. In that case mass is conserved for each phase separately. The flow equation for phase p is

$$\frac{\partial}{\partial x}\left(\rho_p \frac{kk_{rp}}{\mu_p}\frac{\partial p_p}{\partial x}\right) = \frac{\partial}{\partial t}\left(\rho_p \phi s_p\right) \quad . \tag{10.57}$$

Before we go into the details of the numerics, we can compare Eq. (10.57) with the incompressible equation

$$\frac{\partial}{\partial x}\left(\frac{kk_{rp}}{\mu_p}\frac{\partial p_p}{\partial x}\right) = \phi\frac{\partial}{\partial t}\left(s_p\right) \quad . \tag{10.58}$$

in order to anticipate how the introduction of compressibility will affect the IMPES and fully implicit equations.

By comparing the left hand sides of Eqs. (10.57) and (10.58), we see that the mobility, $\lambda_p = \frac{k_{rp}}{\mu_p}$, which is a function of saturation only, is replaced by $\lambda_p^c = \frac{\rho_p k_{rp}}{\mu_p}$, which is a function of both pressure and saturation. This is of little consequence for the IMPES equations, since the mobility terms are evaluated explicitly. In the fully implicit formulation extra terms, proportional to $\frac{d\lambda_p^c}{dP}$, will be added to the nonzero elements of the Jacobi matrix in Eqs. (10.51), (10.52), and (10.53).

Mobility is a function of both pressure and saturation

By comparing the right hand sides of Eqs. (10.57) and (10.58), we see that the time discretisation will give extra terms proportional to $(P_i - P_i^t)$.

In IMPES, the terms proportional to P_i will add to the diagonal of the **A** matrix (Eq. (10.33)), while the terms proportional to P_i^t will contribute to the **e** vector (Eq. (10.34)). For incompressible flow, the **e** vector only depends on the boundary conditions and the pressure solution is independent of the current pressure. For compressible flow the pressure solution will depend on current pressure, since the **e** vector contains the current pressure.

In the fully implicit formulation, the extra terms will add contributions to the diagonal element $\frac{\partial R_{n_i}}{\partial P_{n_i}}$ of the Jacobian matrix (Eqs. (10.49), and (10.50)). The structure, and complexity, of the equation system remain unchanged.

The left hand side similar to the incompressible case (Eq. (10.10));

$$\frac{\partial}{\partial x}\left(k\lambda_p^c\frac{\partial p_p}{\partial x}\right) \simeq T\lambda^c_{p_{i+1/2}}\left(P_{p_{i+1}}-P_{p_i}\right) - T\lambda^c_{p_{i.1/2}}\left(P_{p_i}-P_{p_{i-1}}\right) \quad , \tag{10.59}$$

with the generalized mobility

$$\lambda_p^c = \frac{\rho_p k_{rp}}{\mu_p} \quad . \tag{10.60}$$

Now we will investigate the right hand side of Eq. (10.57). Both the density ρ_p and the porosity ϕ are pressure dependent, and therefore variables with respect to the time derivative.

We start by writing out the partial as:

$$\frac{\partial}{\partial t}\left(\rho_p\phi s_p\right) = \rho_p\phi\frac{\partial s_p}{\partial t} + s_p\frac{\partial}{\partial t}\left(\rho_p\phi\right) \quad . \tag{10.61}$$

Following the same procedure as in Chap. 9, we can write

$$\begin{aligned}\frac{\partial}{\partial t}\left(\rho_p\phi\right) &= \rho_p\phi\left(\frac{1}{\phi}\frac{\partial\phi}{\partial p_p}+\frac{1}{\rho_p}\frac{\partial\rho_p}{\partial p_p}\right)\frac{\partial p_p}{\partial t} \\ &= \rho_p\phi\left(c_\phi + c_p\right)\frac{\partial p_p}{\partial t}\end{aligned} \quad , \tag{10.62}$$

where we have introduced the rock compressibility

$$c_\phi = \frac{1}{\phi}\frac{\partial\phi}{\partial p} \quad , \tag{10.63}$$

and the fluid phase compressibility

Compressibility of fluid and rock

$$c_p = \frac{1}{\rho_p}\frac{\partial\rho_p}{\partial p} \quad . \tag{10.64}$$

Applying the backward difference quotient for the time derivatives we get

$$\left(\frac{\partial}{\partial t}\left(\rho_p\phi s_p\right)\right)_i \simeq \frac{\rho_p\phi}{\Delta t}\left(S_{p_i}-S^t_{p_i}\right) + \frac{s_p\rho_p\phi(c_\phi+c_p)}{\Delta t}\left(P_{p_i}-P^t_{p_i}\right) \quad , \tag{10.65}$$

which can be written as

$$\left(\frac{\partial}{\partial t}\left(\rho_p\phi s_p\right)\right)_i \simeq \frac{1}{\Delta t}\left(\Sigma_{p_i}\left(S_{p_i}-S^t_{p_i}\right) + \Pi_{p_i}\left(P_{p_i}-P^t_{p_i}\right)\right) \quad , \tag{10.66}$$

where

$$\Sigma_p = \rho_p\phi \quad , \tag{10.67}$$

and

$$\Pi_p = s_p\rho_p\phi(c_\phi+c_p) \quad . \tag{10.68}$$

To simplify the notation we will also define

$$\Lambda_n = \Delta t T\lambda^c_n \quad . \tag{10.69}$$

Using Eqs. (10.59) and (10.66) for the left- and right hand side of Eq. (10.57), we then get the discrete expression

$$\begin{array}{l}\Lambda_{p_{i+1/2}}\left(P_{p_{i+1}}-P_{p_i}\right)-\Lambda_{p_{i-1/2}}\left(P_{p_i}-P_{p_{i-1}}\right)= \\ \Sigma_{p_i}\left(S_{p_i}-S_{p_i}^t\right)+\Pi_{p_i}\left(P_{p_i}-P_{p_i}^t\right)\end{array} \quad . \tag{10.70}$$

We have one equation (Eq. (10.70)) for each phase per grid cell, and we can use that $s_o + s_w = 1$, and that $p_o - p_w = P_c(s_w)$, in order to get one unknown saturation and one unknown pressure per grid cell. Thus we have the same number of equations as unknowns.

In petroleum applications it is common to use water saturation S_w, and oil pressure P_o, as unknowns. Following that convention, we will use wetting saturation S_w and non-wetting pressure P_n, and we will drop the phase subscripts

$$\begin{array}{l}S_{w_i}\rightarrow S_i \\ S_{n_i}\rightarrow 1-S_i \\ P_{w_i}\rightarrow P_i-P_c(S_i) \\ P_{n_i}\rightarrow P_i\end{array} \quad . \tag{10.71}$$

The two resulting discrete equations are

$$\begin{array}{l}\Lambda_{n_{i+1/2}}\left(P_{i+1}-P_i\right)-\Lambda_{n_{i-1/2}}\left(P_i-P_{i-1}\right)= \\ -\Sigma_{n_i}\left(S_i-S_i^t\right)+\Pi_{n_i}\left(P_i-P_i^t\right)\end{array} \quad , \tag{10.72}$$

and

$$\begin{array}{r}\Lambda_{w_{i+1/2}}\left(P_{i+1}-P_i-P_{c_{i+1}}+P_{c_i}\right) \\ -\Lambda_{w_{i-1/2}}\left(P_i-P_{i-1}-P_{c_i}+P_{c_{i-1}}\right)= \\ \Sigma_{w_i}\left(S_i-S_i^t\right)+\Pi_{w_i}\left(P_i-P_i^t\right)\end{array} \quad . \tag{10.73}$$

If we multiply Eq. (10.72) with Σ_{w_i} and Eq. (10.73) with Σ_{n_i}, and add the two equations, we get

Pressure equation

$$\begin{array}{l}\left(\Sigma_{w_i}\Lambda_{n_{i+1/2}}+\Sigma_{n_i}\Lambda_{w_{i+1/2}}\right)\left(P_{i+1}-P_i\right) \\ -\left(\Sigma_{w_i}\Lambda_{n_{i-1/2}}+\Sigma_{n_i}\Lambda_{w_{i-1/2}}\right)\left(P_i-P_{i-1}\right) \\ -\Sigma_{n_i}\Lambda_{w_{i+1/2}}\left(P_{c_{i+1}}-P_{c_i}\right)+\Sigma_{n_i}\Lambda_{w_{i-1/2}}\left(P_{c_i}-P_{c_{i-1}}\right) \\ = \\ \left(\Sigma_{w_i}\Pi_{n_i}+\Sigma_{n_i}\Pi_{w_i}\right)\left(P_i-P_i^t\right)\end{array} \quad . \tag{10.74}$$

In IMPES, saturations, capillary pressures, and mobilities are evaluated explicitly.

In IMPES, we evaluate all the saturations at the current time step. The capillary pressures in Eq. (10.74) can thus also be treated as explicitly known. We will also evaluate the mobilities, λ_p^c explicitly. In this case, Eq. (10.74) is an equation where only pressure is unknown. After solving for pressure, we can then apply Eq. (10.72) to find the updated saturation in each cell. If we in addition also treat the coefficients Σ_p and Π_p explicitly, the system of equations (One equation (Eq. (10.74) for each cell plus boundary conditions) is also *linear*. The linear system can be expressed as

Linearized pressure equation in IMPES

$$(\mathbf{A}-\mathbf{C})\cdot\mathbf{P}+\mathbf{C}\cdot\mathbf{P}^t+\mathbf{e}=0 \quad , \tag{10.75}$$

where the vector **e** contain source/sink-terms and capillary pressure contributions, and he matrix **A** has the same sparse structure as in the incompressible case. **C** is a *diagonal* matrix which is proportional to the compressibility, while $\mathbf{P}^t$ is the pressures at the previous time step.

Note that the relative change in Σ_p and Π_p (Eqs. (10.67) and (10.68))) during time step is proportional to $c_p \Delta p$. This is also the size of the extra terms $\mathbf{C} \cdot \left(\mathbf{P} - \mathbf{P}^t\right)$ in Eq. (10.75) compared with Eq. (10.32) in the incompressible case. Thus, it is not completely consistent to treat Σ_p and Π_p explicitly. The solution of the pressure equation will normally require an iterative approach also in IMPES. The nonlinearity is however usually not very strong (See Section 10.2.1), and the linear system is just half the size of the fully implicit system. An IMPES time step thus requires much less CPU time than a fully implicit time step.

10.2.1 Models for density and porosity

A model for the density of a substance as a function of pressure, and in general of temperature and composition, is called an equation of state (EOS). Simple EOS models for oil and gas commonly used in the petroleum industry was briefly discussed in Section 7.2. We will expand on these here, and derive the corresponding coefficients in the numerical formulation.

Porosity is usually modelled with a reference porosity, ϕ_0, and a constant compressibility, c_ϕ, (Eq. (7.6))

$$\begin{aligned} \phi(p) &= \phi_0 \exp\left(c_\phi(p - p_r)\right) \\ &\approx \phi_0 \left(1 + c_\phi(p - p_r)\right) \end{aligned} \quad . \tag{10.76}$$

The approximate linear model here, and in subsequent equations, may be used in order to avoid the CPU-heavy evaluations of the exponential.

In the black oil model[3], the oil and gas densities are expressed in terms of formation volume factors, B. These are the ratios of fluid volumes at reservoir pressure and temperature and volumes at standard conditions. In the symmetric black oil model, where the gas component can dissolve in reservoir oil, and the oil component can dissolve in reservoir gas, the oil density is

$$\rho_o(p, R_s) = \frac{\rho_{os} + \rho_{gs} R_s}{B_o(p, R_s)} \quad , \tag{10.77}$$

and the gas density is

$$\rho_g(p, R_v) = \frac{\rho_{gs} + \rho_{os} R_v}{B_g(p, R_v)} \quad . \tag{10.78}$$

Here, ρ_{os} is the density of oil at standard conditions[4], ρ_{gs} is the density of gas at standard conditions[5]. The solution ratios (R_s is the solution-gas-oil ratio, and R_v is the solution-oil-gas ratio) are proxies for the oil and gas composition. In this chapter we consider phases with constant composition only, R_s and R_v are thus

[3] The black oil model is explained in detail in Chapter 11

[4] Oil at standard conditions is by definition a fluid with 100% oil component

[5] Gas at standard conditions is by definition a fluid with 100% gas component

constants. The density model for the hydrocarbon phases are then simply

$$\rho_p(p) = \frac{C_p}{B_p(p)} \quad . \tag{10.79}$$

The formation volume factors are typically supplied to the reservoir simulator in the form of lookup tables.

Water density is usually modelled with a reference density at a reference pressure and a constant compressibility

$$\begin{aligned} \rho_w(p) &= \frac{\rho_{ws}}{B_{w0}} \exp(+c_w(p - p_r)) \\ &\approx \frac{\rho_{ws}}{B_{w0}} \left(1 + c_w(p - p_r)\right) \end{aligned} \quad . \tag{10.80}$$

If we use the above density models, the coefficients in Eq. (10.72) and (10.73) are[6]

$$\begin{aligned} \Sigma_w &= \frac{\rho_{ws}\phi_0}{B_{w0}} \exp\left((c_w + c_\phi)(p - p_r)\right) \\ &\approx \frac{\rho_{ws}\phi_0}{B_{w0}} \left(1 + c_w(p - p_r)\right)\left(1 + c_\phi(p - p_r)\right) \\ \Sigma_o &= \frac{C_o\phi_0}{B_o(p)} \exp\left(c_\phi(p - p_r)\right) \\ &\approx \frac{C_o\phi_0}{B_o(p)} \left(1 + c_\phi(p - p_r)\right) \\ \Pi_w &= s_w(c_\phi + c_w)\Sigma_w \\ &\approx s_w \left(\frac{c_\phi}{1 + c_\phi(p - p_r)} + \frac{c_w}{1 + c_w(p - p_r)}\right)\Sigma_w \\ \Pi_o &= (1 - s_w)\left(c_\phi - \frac{1}{B_o}\frac{\partial B_o}{\partial p}\right)\Sigma_o \\ &\approx (1 - s_w)\left(\frac{c_\phi}{1 + c_\phi(p - p_r)} - \frac{1}{B_o}\frac{\partial B_o}{\partial p}\right)\Sigma_o \end{aligned} \quad . \tag{10.81}$$

[6] Note that the "approximate" expressions in Eq. (10.81) are the exact expressions for the linear models of Eqs. (10.76) and (10.80) used as replacement for the exponentials in the constant compressibility case.

As also Λ_p on the left hand side of Eqs. (10.72) and (10.73) is proportional to density, we see that all the constants ρ_{ws}, B_{w0}, and C_o, will actually cancel out from the final versions of Eqs. (10.72), (10.73), and (10.74).

10.2.2 Mass conservative finite volume formulation

The formulation leading up to Eqs. (10.72) and (10.73) is not fully mass conservative. In order to derive mass conservative equations we will now reformulate the problem using a finite volume formulation as explained in Section 7.2.

Applying the backward difference quotient for the time derivative on the right hand side of Eq. (10.57) we get

$$\left(\frac{\partial}{\partial t}\left(\rho_p\phi s_p\right)\right)_i \simeq \frac{1}{\Delta t}\left(\rho_p(P_i)\phi(P_i)S_{p_i} - \rho_p(P_i^t)\phi(P_i^t)S_i^t\right) \quad , \tag{10.82}$$

and when using Eqs. (10.59) and (10.82) for the left- and right hand side of Eq. (10.57), we get the discrete expression

$$\begin{aligned} &\Delta t T\lambda^{\mathrm{c}}_{p_{i-1/2}}\left(P_{p_{i-1}} - P_{p_i}\right) - \Delta t T\lambda^{\mathrm{c}}_{p_{i+1/2}}\left(P_{p_i} - P_{p_{i+1}}\right) = \\ &\rho_p(P_i)\phi(P_i)S_{p_i} - \rho_p(P_i^t)\phi(P_i^t)S^t_{p_i} \end{aligned} \quad . \tag{10.83}$$

The left hand side is the mass that flows into a grid cell during a time step through the left boundary, minus the mass that flows out of the cell through the right boundary. The right hand side is the mass in the cell at the end of the time step minus the mass in the cell at the beginning of the time step, that is the mass accumulated in the cell during a time step. This is exactly the finite volume formulation.

We have one equation (Eq. (10.83)) for each phase per grid cell, and we can use that $s_n + s_w = 1$, and that $p_n - p_w = P_c(s_w)$, in order to get one unknown saturation and one unknown pressure per grid cell. We will use the same substitutions as before (Eq. (10.71)), with wetting saturation S_w, and non-wetting pressure P_n, as unknowns.

To simplify the notation we will define

$$\Lambda_n = \Delta t T \lambda_n^c \quad . \tag{10.84}$$

We then have the two discrete equations

$$\begin{array}{l} \Lambda_{o_{i-1/2}} \left[P_{i-1} - P_i\right] - \Lambda_{o_{i+1/2}} \left[P_i - P_{i+1}\right] = \\ \rho_n(P_i)\phi(P_i)(1 - S_i) - \rho_n(P_i^t)\phi(P_i^t)(1 - S_i^t) \end{array} \quad , \tag{10.85}$$

and

$$\begin{array}{l} \Lambda_{w_{i-1/2}} \left[P_{i-1} - P_i + P_c(S_{i-1}) - P_c(S_i)\right] \\ -\Lambda_{w_{i+1/2}} \left[P_i - P_{i+1} + P_c(S_i) - P_c(S_{i+1})\right] = \\ \rho_w(P_i)\phi(P_i)S_i - \rho_w(P_i^t)\phi(P_i^t)S_i^t \end{array} \quad . \tag{10.86}$$

In IMPES, we evaluate all the saturations at the current time step. The capillary pressures in Eq. (10.86) can thus also be treated as explicitly known. We will also evaluate the mobility terms, Λ_p, explicitly with upstream weighting.

IMPES: Saturations, capillary pressures, and mobilities are evaluated explicitly.

By multiplying the oil equation with $\rho_w(P_i)$, and the water equation with $\rho_n(P_i)$, and adding the two, we get the pressure equation

IMPES pressure equation

$$\begin{array}{l} \left[\Lambda_{o_{i-1/2}}\rho_w(P_i) + \Lambda_{w_{i-1/2}}\rho_n(P_i)\right]\left[P_{i-1} - P_i\right] \\ -\left[\Lambda_{o_{i+1/2}}\rho_w(P_i) + \Lambda_{w_{i+1/2}}\rho_n(P_i)\right]\left[P_i - P_{i+1}\right] \\ +\rho_n(P_i)\left(\Lambda_{w_{i-1/2}}P_{c_{i-1}}^t - \left(\Lambda_{w_{i-1/2}} - \Lambda_{w_{i+1/2}}\right)P_{c_i}^t + \Lambda_{w_{i+1/2}}P_{c_{i+1}}^t\right) \\ = \\ \phi(P_i)\rho_w(P_i)\rho_n(P_i) - \phi(P_i^t)\left[\rho_w(P_i)\rho_n(P_i^t)(1 - S_i^t) + \rho_n(P_i)\rho_w(P_i^t)S_i^t\right] \end{array} \quad . \tag{10.87}$$

The compressible pressure equation is non linear

In the incompressible case, all densities are constant, and we get a *linear* equation system. Also, the right hand side of Eq. (10.87) is zero as we had in Eq. (10.32). The compressible pressure equation is nonlinear and must be solved using an iterative scheme. However, compared to a fully implicit formulation, where both pressure and saturation are found with an iterative method, the size of the equation system is half. The nonlinearity is also not very strong, at least not for liquid–liquid systems, so the number of iterations needed to reach convergence will be small.

10.3 Two-phase flow in OPM-Flow

The IMPES formulation is not implemented in the reservoir simulator OPM-Flow. If you use the keyword `IMPES` in the `RUNSPEC` section, it will read the key-word, but it will not change the solution method, thus the solution method will still be fully implicit. In the following, we will simulate a similar case as the 1D model described in Chap. 6 using OPM-Flow. This model needs to be extended to two phases. We will use the Corey curves described before, where both exponents are 2, and $k_{rwo} = 0.4$. This can be tabulated as:

```
SWOF
 0.2     0.0     1.0     0.0
 0.23158 0.00111 0.89751 0.0
 0.26316 0.00443 0.80055 0.0
 0.29474 0.00997 0.70914 0.0
 0.32632 0.01773 0.62327 0.0
 0.35789 0.0277  0.54294 0.0
 0.38947 0.03989 0.46814 0.0
 0.42105 0.05429 0.39889 0.0
 0.45263 0.07091 0.33518 0.0
 0.48421 0.08975 0.27701 0.0
 0.51579 0.1108  0.22438 0.0
 0.54737 0.13407 0.17729 0.0
 0.57895 0.15956 0.13573 0.0
 0.61053 0.18726 0.09972 0.0
 0.64211 0.21717 0.06925 0.0
 0.67368 0.24931 0.04432 0.0
 0.70526 0.28366 0.02493 0.0
 0.73684 0.32022 0.01108 0.0
 0.76842 0.359   0.00277 0.0
 0.8     0.4     0.0     0.0 /
```

While the simulator OPM-Flow handles compressible fluids, we will compare to our fluids used in the IMPES code implemented in Python above, thus we want to keep the fluids quite incompressible:

```
PVTW
-- REF.PRES.  REF. FVF  COMPRESSIBILITY  REF.VISC.
--         -> VISCOSIBILITY
--  (bar) (m3/m3) (1/bar)    (cP)       (1/bar)
     150   1.01   1.0e-4     1.0        0.0e+0   /

PVDO
--
--
--  PRES.   FVF.   VISC.
    20.68   1.01   0.99
    55.16   1.00   1.00
   551.58   0.99   1.01
/

ROCK
-- REF.PRES   COMPRESSIBILITY
--  (bars)        (1/bars)
     150          0.0e-6   /
```

As this is a two phase case, we define the initial saturation:

```
SWAT
  1.0 101*0.2
/
```

To keep the simulation model numerically stable, we want to extend the grid cells perpendicular to the flow direction, so that the grid cells get a larger volume:

```
DX
-- There are in total 102 cells with length 2.0m
-- in x-direction. The main part of the model is
-- the 100 cells in the middle. The two outer
-- cells are dummy cells with high permeability
-- to distribute the flow from the wells to the
-- full cross-section.
    102*2.0 /

DY
  102*10.0 /

DZ
  102*10.0 /
```

To get a flow rate in the order of a meter per day, we then use an injection rate of

$$Q = \phi A q = 0.2 \cdot 10\,\mathrm{m} \cdot 10\,\mathrm{m} \cdot 1\,\mathrm{m/d} = 20\,\mathrm{m}^3/\mathrm{d} \quad . \tag{10.88}$$

```
WCONINJE
-- Item #:1  2   3   4  5-6 7
  'INJW'  'WATER' 'OPEN'  'RESV'  1* 20.0 /
/
```

We run this input deck using OPM-Flow. The resulting saturation versus distance is plotted in Fig. 10.4. This plot also include the Buckley-Leverett result for the same system. We see that the simulation results is much more smoothed out around the shock front than what is predicted by the Buckley-Leverett solution.

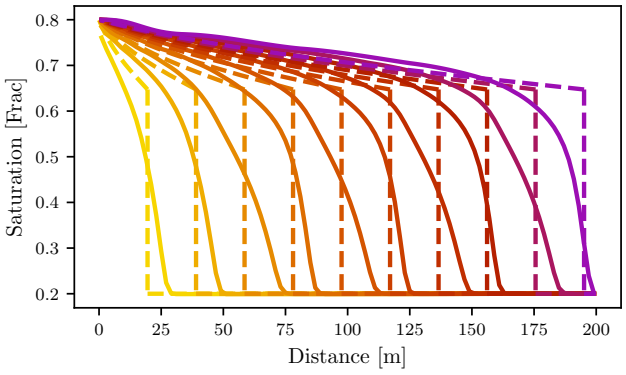


Figure 10.4: The simulation results from OPM-Flow compared to the Buckley-Leverett solution. We have plotted one line for each restart step of 10 d.

To have a simulation comparable to the Python implementation of the IMPES method, we want to have a comparable amount of

time steps. The Python implementation used time steps of size 1×10^{-3} s with a flow rate of 1 m/s. This would give approximately 1000 times steps to reach the time it takes to move the front through the model. In our simulation model we have a flow rate around 1 m/d. As the model is 200 m long, we would need a time step of 0.2 d to have approximately 1000 time steps for the time it takes to move the shock front through the model. To reduce the time step to 0.2 d, we use the following command line argument:

```
flow TWOPHASE1D.DATA --solver-max-time-step-in-days=0.2
```

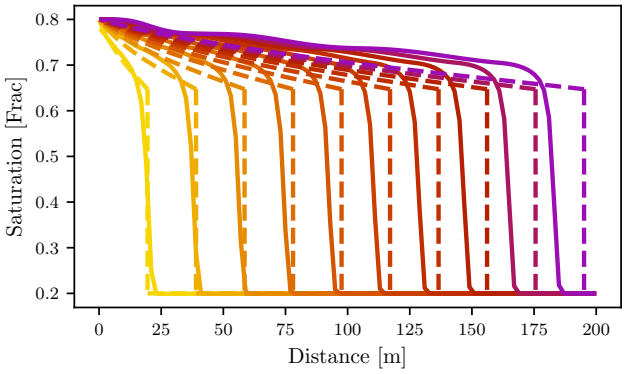


Figure 10.5: The simulation results from OPM-Flow compared to the Buckley-Leverett solution, when we have limited the time steps to 0.2 d. We have plotted one line for each restart step of 10 d.

The resulting saturation versus distance is plotted in Fig. 10.5. We see that the reduced time steps gives a sharper shock front. However, the plot also indicate that we have larger uncertainty behind the front, where the saturation is too high compared to the Buckley-Leverett solution. As the saturation is too large behind the front, the front ends up moving to slow compared to Buckley-Leverett solution. The saturation profile is not smooth either. Thus, while the smaller time steps improves on the shock front sharpness, it depreciate the solution with respect to other features.

10.4 Exercises

Exercise 10.1 Use a Python script for simulating 1D incompressible IMPES. The basic data is given in Table 10.1.

a) Plot the saturation versus distance from inlet for a set of times that illustrate how the fluid front moves through the medium.

b) Do a sensitivity study on the Corey parameters to investigate how the relative permeability curves influence the displacement process. Discuss your results with respect to the

- The fractional flow curve
- The shock front saturation

c) Try to change the mobility weighting. At which weighting does your simulation break down, and why?

	SPE Metric	SI
l	200 m	200 m
μ_w	1 cP	1×10^{-3} Pa s
μ_n	2 cP	2×10^{-3} Pa s
ϕ	0.2	0.2
n_w	2.0	2.0
n_n	2.0	2.0
k_{rwo}	0.4	0.4
q_t	0.2 m/d	2.315×10^{-6} m/s

Table 10.1: Basic data for example.

Exercise 10.2 Write a Python script for simulating incompressible fully implicit with upstream weighting.

a) Compare the solution for IMPES and fully implicit for different number of grid cells and time step length using the same data as in Exersise 10.1

11
Black oil simulations

> Were we a rational society, a virtue of which we have rarely been accused, we would husband our oil and gas resources.
>
> Marion King Hubbert

In the previous chapter we discussed the simulation of flows with two phases of constant composition. This is relevant for water–gas systems, such as gas reservoirs with aquifer support or produced with support from water injection, and simple water–air systems in hydrology. It is also relevant for oil–water systems, such as oil reservoirs produced by water injection. If we ignore the effect of the dissolution of CO_2 in formation water, the methods can also be used to describe injection of CO_2 into aquifers.

In this chapter we will extend the flow description to situations where the composition of the phases can change. This includes the production of oil reservoirs by pressure depletion, where the oil composition will change when pressure falls under the bubble point, the production of oil reservoirs supported by gas injection, and the modelling of CO_2 sequestration in aquifers, where the dissolution of CO_2 into the water may be important for the long term behavior. We will however only discuss the absolute simplest model for the description of compositional change; the *black oil model*.

11.1 The black oil model

Real subsurface systems are multi component systems with a large number of molecular species. Such systems typically have a rich phase behavior as a function of pressure, temperature and composition. Extended knowledge of this behavior is of crucial importance for the design and operation of top-side equipment, e.g., design and operation of separators to optimize liquid production and avoid precipitation of waxes and asphaltenes. The full space of pressure and composition is however far from being visited in the subsurface reservoir, even for the most exotic of recovery processes. Reservoir temperature can also usually be treated as being constant, despite possible strong temperature gradients around injectors.

A simplified description is thus possible, and is also required for reservoir simulation.

The black oil model is a two component model for the description of the phase behavior and density of a fluid system. In its original form, it is based on direct tabulation of measured volume ratios in a fairly simple pressure depletion experiment, and gives a good description for oil reservoirs produced by pressure depletion. The model has been extended also to other scenarios, and the black oil model is by far the most common phase behavior description employed in petroleum reservoir simulation.

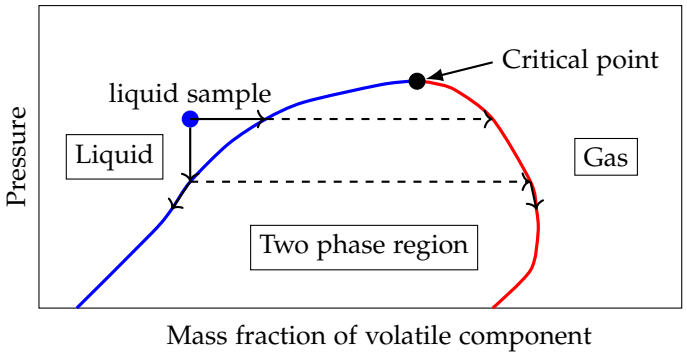


Figure 11.1: Sketch of phase behavior for a two component system at constant temperature. In the pressure range shown, one ("non-volatile") component is a liquid, while the other ("volatile") component is a gas. The bubble point line is shown in blue, and the dew point line in red. A liquid with a given composition at a pressure above bubble point is shown as a blue dot. Other lines and arrows are explained in the main text.

The general phase behavior for a two component system at constant temperature is illustrated in Fig. 11.1. Dependent on the total fraction of each component, the system can be either a gas or a liquid. We assume that one component is non-volatile, that is the single component fluid is in a liquid phase at all pressures of interest, and also at standard conditions. The other component is volatile, and the corresponding single component system is in the gas phase in the same pressure range, and also at standard conditions. At intermediate mass fractions, the system is separated into two phases. At a given pressure, the gas and liquid are at equilibrium with different compositions, given respectively by the composition on the dew point line (red) and on the bubble point line (blue).

If we take a liquid sample (blue dot) and add the volatile component, the system will move to the right in the phase diagram (horizontal arrow), and at some point reach the bubble point line where gas, with composition given by the dew point line, will appear. Further addition of the volatile component will not change the composition of the liquid or gas, only the relative amount of each phase. Also, if we reduce the pressure of the liquid sample (vertical arrow), gas will appear when we reach the bubble point line. The composition of the gas is again given by the composition at the dew point line. However, if we reduce the pressure further, we will not only get more of the gas phase, but the composition of the gas and liquid will also change, as determined by the bubble point and dew point lines.

In a two component system, the properties of the single phase liquid is a function of pressure and the fraction of volatile com-

In the two phase region, the properties of liquid and gas is a function of pressure only.

ponent, and the properties of single phase gas is a function of the pressure and the fraction of non-volatile component. However, in the two phase region the two phases will be in equilibrium, therefore the properties of liquid and gas is a function of pressure only.

The black oil model is a model for a two component system where the non-volatile component is called the oil component, and the volatile component is called the gas component. The liquid phase is usually called the oil phase, and the gas phase is usually called the gas phase. That the components and phases have identical names is unfortunate and definitely is confusing for students and professionals alike.[1] While the black oil model can be applied to other systems with other phases and components, e.g., a CO_2-brine system, in the following we will follow the traditional use of the black oil model and use gas and oil.

[1] To distinguish the phases from the components, sometimes the term "oleic phase" is used for the liquid phase (e.g., in (Lie, 2016)), and "gaseous phase" for the gas phase. In petroleum reservoirs we also have brine, which is an "aqueous phase".

Note that the oil and gas components are not directly associated with any actual hydrocarbon species. As used in most reservoir simulation models, the oil component is associated with the sales, or stock tank, oil produced when the production stream is processed through a given separator system, and the gas component is associated with the corresponding produced sales gas. The phase behavior of a black oil model is in principle identical to the two component system described above (Fig. 11.1), but as implemented it has some additional restrictions. In particular, the black oil model does not easily accommodate a critical point.

Figure 11.2: Illustration of reservoir oil phase and gas phase (lower rectangles) versus oil phase and gas phase at standard conditions (upper rectangles).

The black oil model is based on volumes and volume ratios. As illustrated in Fig. 11.2, let V_{or} denote the volume of liquid phase at reservoir conditions. When brought to surface, this volume of liquid phase separates into a volume V_{os} of liquid with 100% oil component at standard conditions and a volume V_{gs} of gas with 100% gas component at standard conditions.

The black oil model is expressed in terms of volume ratios

Let m_o and m_g be the mass of the oil V_{os} and gas V_{gs} at standard conditions from the oil V_{or} at reservoir conditions. Then the density of the liquid phase at reservoir conditions is $\rho_{or} = \rho_o = (m_o + m_g)/V_{or}$, while the density of liquid and gas at standard conditions are $\rho_{os} = m_o/V_{os}$ and $\rho_{gs} = m_g/V_{gs}$, respectively. We can then write

In this chapter, an upper index (superscript) represents the component, while a lower index (subscript) represent the phase.

$$\rho_o(p, R_s) = \frac{m_o + m_g}{V_{or}} = \frac{\frac{m_o}{V_{os}} + \frac{m_g}{V_{gs}}\frac{V_{gs}}{V_{os}}}{\frac{V_{or}}{V_{os}}} = \frac{\rho_{os} + \rho_{gs}R_s}{B_o(p, R_s)} \quad , \qquad (11.1)$$

where $B_o = V_{or}/V_{os}$, and $R_s = V_{gs}/V_{os}$. This gives the following

equation for the liquid phase density at reservoir conditions:

$$\rho_o(p, R_s) = \frac{\rho_{os} + \rho_{gs} R_s}{B_o(p, R_s)} \quad . \tag{11.2}$$

Using equivalent derivations, we obtain the following equation for the gas phase density

$$\rho_g(p, R_v) = \frac{\rho_{gs} + \rho_{os} R_v}{B_g(p, R_v)} \quad . \tag{11.3}$$

Thus the composition of the oil phase is expressed through the solution-gas-oil ratio, R_s, and the composition of the gas phase is expressed through the solution-oil-gas ratio, R_v. For the liquid and gas density at reservoir conditions we also need the liquid and gas formation volume factors B_o and B_g.

R_s is a proxy for oil phase composition.
R_v is a proxy for gas phase composition.
Formation volume factor

Thus, we see that in the black oil model, the volume ratios B_o, B_g, R_s, and R_v are defined as follows:

Model definitions of the volume ratios

- Take a volume of reservoir oil V_{or} and separate out the gas component. At standard conditions, the remaining oil has volume V_{os}, and the separated gas has volume V_{gs}. Then $B_o = V_{or}/V_{os}$, and $R_s = V_{gs}/V_{os}$.
- Take a volume of reservoir gas V_{gr} and separate out the oil component. At standard conditions, the remaining gas has volume V_{gs}, and the separated oil has volume V_{os}. Then $B_g = V_{gr}/V_{gs}$, and $R_v = V_{os}/V_{gs}$.

These definitions are approximately the same as the operational definitions:

Operational definitions of the volume ratios

- Take a volume of reservoir oil V_{or} and process it through the separator system where it separates into a sales-oil and a sales-gas. The sales-oil has volume V_{os}, and the sales-gas has volume V_{gs}. Then $B_o = V_{or}/V_{os}$, and $R_s = V_{gs}/V_{os}$.
- Take a volume of reservoir gas V_{gr} and process it through the separator system where it separates into a sales-oil (condensate) and a sales-gas. The sales-gas has volume V_{gs}, and the sales-oil (condensate) has volume V_{os}. Then $B_g = V_{gr}/V_{gs}$, and $R_v = V_{os}/V_{gs}$.

We see from the model definitions of volume ratios and densities that the mass fraction of gas (light component) in the oil phase is

Volume ratios translated into mass fractions

$$x_o^g = \frac{m_g}{m_o + m_g} = \frac{\rho_{gs} R_s}{\rho_{os} + \rho_{gs} R_s} \quad , \tag{11.4}$$

and the mass fraction of gas (light component) in the gas phase is

$$x_g^g = \frac{\rho_{gs}}{\rho_{gs} + \rho_{os} R_v} \quad . \tag{11.5}$$

Eqs. (11.2), (11.3), (11.4), and (11.5) serve as a direct link between the black oil model expressed in terms of pressure and volume

ratios, and the general two component model expressed in terms of pressure and mass fraction.

The black oil model is input to OPM-Flow in the form of tables.

In reservoir simulators, the black oil model is invariably provided in the form of tables. This is also the case for OPM-Flow. The format of these tables and how they relate to the phase diagram and the density calculations is described below.

The properties of the oil (liquid) phase, and the location of the bubble point line, is defined in a table provided under the keyword PVTO. An example is shown if Fig. 11.3. Pairs of values in the first

```
PVTO
--  Rs       P         Bo          mu_o
   44.09    110.00    1.16437     0.880
            185.00    1.14973     1.012
            210.00    1.14547     1.056  /
   70.14    170.00    1.22704     0.698
            220.00    1.21558     0.759
            270.00    1.20555     0.821  /
   99.39    230.00    1.29586     0.622
            280.00    1.28300     0.661
            330.00    1.27171     0.699  /
  130.23    285.33    1.36737     0.5335
            335.33    1.35313     0.5638
            385.33    1.34059     0.5934 /
  150.01    317.23    1.41282     0.4614
            367.23    1.39773     0.4863
            417.23    1.38443     0.5107  /
  179.63    360.80    1.48028     0.39503
            410.80    1.46398     0.41502 /
  209.18    399.99    1.54700     0.35239
            424.99    1.53800     0.36089 /
/
```

Figure 11.3: Example of input to OPM-Flow for the oil phase properties. Simplified data from the Norne model (Ch. 14).

two columns (R_s and p) define the bubble point line. Corresponding values in the third (B_o) and fourth (viscosity, μ_o) column, are oil phase properties at the bubble point, and thus also in the two phase region. Lines where R_s is missing contain oil properties in the single phase liquid region (undersaturated oil) at the pressure given in the second column and R_s given by the nearest value above in the first column.

The properties of the gas phase, and the location of the dew point line, is defined in a table provided under the keyword PVTG. An example is shown if Fig. 11.4. Pairs of values in the first two columns (p and R_v) define the dew point line. Corresponding values in the third (B_g) and fourth (viscosity, μ_g) column, are gas phase properties at the dew point, and thus also in the two phase region. Lines where pressure is missing contain gas properties in the single phase gas region at R_v given in the second column, and pressure given by the nearest value above in the first column.

In many cases, a simplified version of the black oil model with dry gas, is sufficient. In the dry gas version, it is assumed that the

Dry gas model: $R_v = 0$, and no single phase gas region

```
PVTG
--  p           Rv          Bg          mu_g
    110.00    0.00000798   0.011072    0.01609
              0.00000000   0.011081    0.01605 /
    170.00    0.00001786   0.007156    0.01834
              0.00000000   0.007172    0.01819 /
    230.00    0.00003767   0.005402    0.02121
              0.00001883   0.005412    0.02095
              0.00000000   0.005422    0.02071 /
    250.80    0.00004756   0.005013    0.02234
              0.00002378   0.005022    0.02197
              0.00000000   0.005032    0.02162 /
    285.33    0.00006853   0.004511    0.02438
              0.00003427   0.004518    0.02375
              0.00000000   0.004525    0.02315 /
    317.23    0.00009313   0.004165    0.02648
              0.00004657   0.004166    0.02549
              0.00000000   0.004169    0.02456 /
    346.80    0.00012100   0.003917    0.02863
              0.00006050   0.003911    0.02719
              0.00000000   0.003906    0.02585 /
    374.31    0.00015188   0.003735    0.03087
              0.00000000   0.003705    0.02703 /
    399.99    0.00018571   0.003598    0.03320
              0.00000000   0.003545    0.02810 /
/
```

Figure 11.4: Example of input to OPM-Flow for the gas phase properties. Simplified data from the Norne model (Ch. 14).

reservoir gas contain no dissolved oil so that R_v is always zero. In this case, there is no single phase gas region in the phase diagram. Dry gas properties as a function of pressure is provided to OPM-Flow with the keyword `PVDG`.

The component mass densities, needed for the calculation of reservoir mass densities, are defined using the keyword `DENSITY`. An example is shown in Fig. 11.5.

```
DENSITY
--    Oil     Water    Gas
      859.5  1033.0    0.854  /
```

Figure 11.5: Example of input to OPM-Flow for component mass densities. Data from the Norne model (Ch. 14).

Figures 11.6 and 11.7 show the pressure–mass-fraction phase diagrams that correspond to the tables in Figs. 11.3, 11.4, and 11.5. These figures should be compared with the general two–component phase diagram of Fig. 11.1. In the two–phase region, the phase properties (here gas phase and oil phase density) are only a function of pressure, given by the properties of the equilibrium fluids on the bubble and dew point lines. Note also that the oil density is *decreasing* with increased pressure, while the gas density is increasing. This is consistent with a critical point, where the gas and oil phase are identical at high pressure.

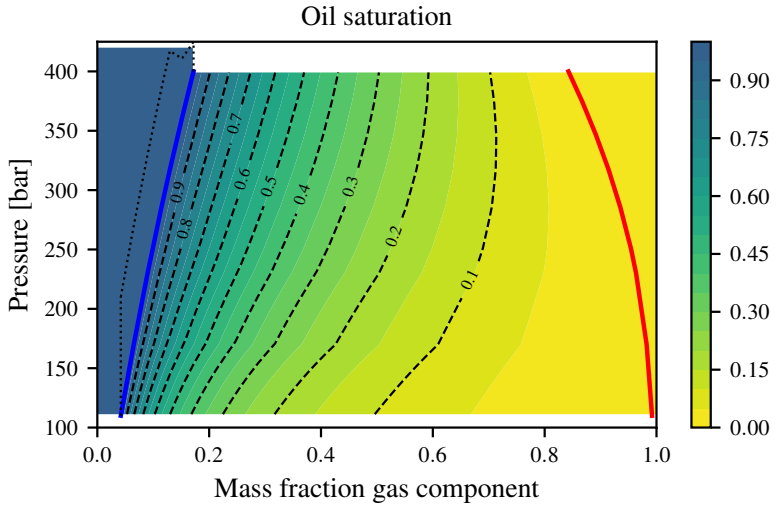


Figure 11.6: Phase diagram corresponding to the simplified Norne black oil tables in Figs. 11.3 and 11.4. Bubble point line in blue, and dew point line in red. Contours and colors show the volume fraction of oil phase in the two–phase region. The dotted line in the single phase oil region show the region that are covered by the property data (B_o, and μ_o) for undersaturated oil in the `PVTO` table. Properties are extrapolated outside of this region.

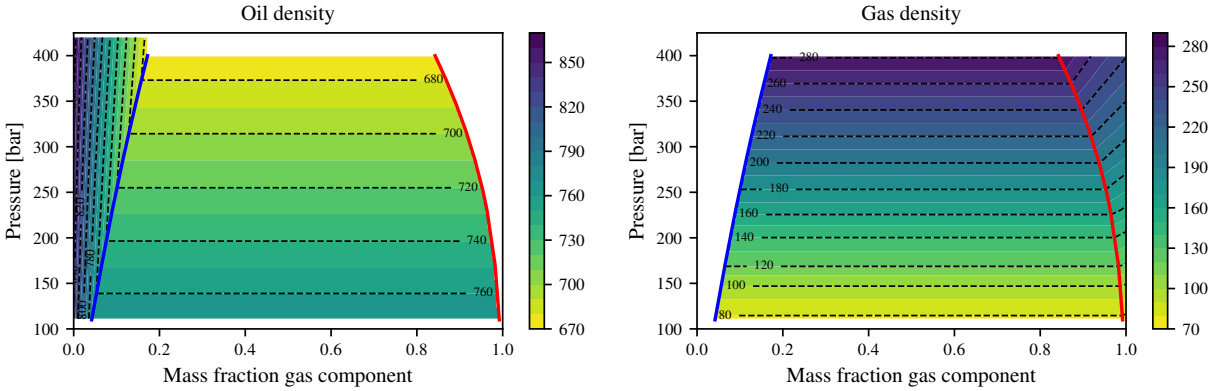


Figure 11.7: Density of the oil and gas phase from the simplified Norne black oil tables in Figs. 11.3 and 11.4. In the two–phase region, the properties are only a function of pressure.

11.2 *Finite volume discretization*

In this section we will develop a finite volume discretization using the two-point flux approximation for a general grid, as described in section 7.2.1. We will also include gravity terms, which we have ignored in the earlier numerics chapters.

In previous chapters we have assumed that the composition of the flowing phases remain constant, and we could write one mass conservation equation per phase (Eq. 10.83). For gas–liquid systems described by the black oil model, the gas and liquid phase compositions are no longer constant, and we will need one mass conservation equation per *component*. In petroleum reservoirs we will always have water present in addition to oil and gas, both as a phase (formation water with dissolved salts), and as a component. In the black oil model, the water component and the dissolved salts are not present in the gas and oil phase, and the water composition is assumed constant. The mass conservation of water can thus be treated as before. In order to simplify the presentation, we will, in this section, treat the water as immobile. Thus, we will concentrate

For water, component conservation is equivalent to phase conservation.

our derivations on one liquid (oil) and one gas phase only. We will also, for the same reason, assume that the capillary pressures (gas–oil and oil–water) are zero.

Similar to the mass balance per phase (Eq. 10.83), we have one mass balance equation per component. The oil (non-volatile) component is distributed in both the liquid and the gas phase. The mass of oil component M^o in a grid cell with bulk volume V is thus

Remember that we use upper-case letters for numerical grid cell values.

$$M^o = \rho_{os} V \phi \left(\frac{1}{B_o} S_o + \frac{R_v}{B_g} S_g \right) \quad , \tag{11.6}$$

where the first term is the mass of oil component in the oil phase, and the second term is the mass of oil component in the gas phase. The accumulation term is the change of component mass over a time step. The black oil model is expressed in terms of volumes, and both this accumulation term and the flow terms for oil are proportional to ρ_{os}. Thus we can eliminate this constant from the equations and work with component volumes at standard conditions instead:

Conserved quantity: Component volume at standard conditions

$$V^o = V \phi \left(\frac{1}{B_o} S_o + \frac{R_v}{B_g} S_g \right) \quad . \tag{11.7}$$

The volume of gas component is distributed over the oil and gas phase, giving

$$V^g = V \phi \left(\frac{1}{B_g} S_g + \frac{R_s}{B_o} S_o \right) \quad . \tag{11.8}$$

If we ignore capillary pressure, the generalized Darcy law for *phase n* is (see Eq. 3.17)

$$q_n = -\frac{k k_{rn}}{\mu_n} \left(\nabla p + \rho_n g \nabla z \right) \quad . \tag{11.9}$$

The oil flux carries dissolved gas, and the gas flux carries dissolved oil, so that the *component* fluxes are

$$\begin{aligned} q^o &= \frac{1}{B_o} q_o + \frac{R_v}{B_g} q_g \\ q^g &= \frac{1}{B_g} q_g + \frac{R_s}{B_o} q_o \end{aligned} \quad , \tag{11.10}$$

and we may combine Eqs. 11.9 and 11.10 to get

$$\begin{aligned} q^o &= -k \left[\frac{1}{B_o} \frac{k_{ro}}{\mu_o} + \frac{R_v}{B_g} \frac{k_{rg}}{\mu_g} \right] \nabla p - k \left[\frac{1}{B_o} \frac{\rho_o k_{ro}}{\mu_o} + \frac{R_v}{B_g} \frac{\rho_g k_{rg}}{\mu_g} \right] g \nabla z \\ q^g &= -k \left[\frac{1}{B_g} \frac{k_{rg}}{\mu_g} + \frac{R_s}{B_o} \frac{k_{ro}}{\mu_o} \right] \nabla p - k \left[\frac{1}{B_g} \frac{\rho_g k_{rg}}{\mu_g} + \frac{R_s}{B_o} \frac{\rho_o k_{ro}}{\mu_o} \right] g \nabla z \end{aligned} \quad , \tag{11.11}$$

If we define phase mobilities[2] as

$$\lambda_n = \frac{1}{B_n} \frac{k_{rn}}{\mu_n} \quad , \tag{11.12}$$

then Eqs. (11.11) can be expressed as

$$\begin{aligned} q^o &= -k \left[\lambda_o + R_v \lambda_g \right] \nabla p - k \left[\rho_o \lambda_o + \rho_g R_v \lambda_g \right] g \nabla z \\ q^g &= -k \left[\lambda_g + R_s \lambda_o \right] \nabla p - k \left[\rho_g \lambda_g + \rho_o R_s \lambda_o \right] g \nabla z \end{aligned} \quad . \tag{11.13}$$

[2] If we *ignore* the gravity term, we see that we can define *component* mobilities

$$\lambda^o = \frac{1}{B_o} \frac{k_{ro}}{\mu_o} + \frac{1}{B_g} \frac{R_v k_{rg}}{\mu_g}$$

and

$$\lambda^g = \frac{1}{B_g} \frac{k_{rg}}{\mu_g} + \frac{1}{B_o} \frac{R_s k_{ro}}{\mu_o} \quad .$$

Eq. (11.13) is in that case equivalent to the equation for compressible immiscible flow, Eq. (10.59), substituting mass flow with component flow.

In the two point flux approximation, using Eq. (11.13), the net component flow into grid cell i during a time step Δt is

$$\begin{aligned}\Delta V_i^o &= \Delta t \sum_j T_{ji} \left[\left(\lambda_o + R_v \lambda_g\right) \left(P_j - P_i\right) + \left(\rho_o \lambda_o + \rho_g R_v \lambda_g\right) g \left(z_j - z_i\right) \right] \\ \Delta V_i^g &= \Delta t \sum_j T_{ji} \left[\left(\lambda_g + R_s \lambda_o\right) \left(P_j - P_i\right) + \left(\rho_g \lambda_g + \rho_o R_s \lambda_o\right) g \left(z_j - z_i\right) \right]\end{aligned} \quad , \tag{11.14}$$

where the sum runs over the neighboring cells, T_{ji} is the cell to cell transmissibility (See Eqs. 7.19 and 7.26), and ΔV are the volumes transported during the time step Δt.

Summing up, we have two mass balance equations for each grid cell:

Mass balance

$$\begin{aligned}V^{ot} - V^o - \Delta V_i^{ot} + \Delta t Q^{ot} = 0 \\ V^{gt} - V^g - \Delta V_i^{gt} + \Delta t Q^{gt} = 0\end{aligned} \quad , \tag{11.15}$$

where Q^o and Q^g are source/sink terms that will be discussed below. A superscript t denotes, as before, that the term has to be evaluated implicitly. Mobilities may be evaluated explicitly, corresponding to an IMPES method, or implicitly, corresponding to a fully implicit method. The pressures in Eq. (11.14) are evaluated implicitly. Upstream weighting is applied to the mobilities. Note that, due to the gravity term, the oil and gas may flow in opposite directions even with zero capillary pressure; therefore the upstream cell must be determined independently for the gas and oil mobilities.

The upstream cell is determined independently for each phase

We have three unknowns in each grid cell, V^o, V^g, and P, and mass balance provide two equations (Eq. (11.15)) per grid cell. In order to proceed, we need an additional equation. This additional equation is the volume balance equation which express the condition that the sum of oil and gas phase volumes must fill the cell:

Volume balance

$$V_{or} + V_{gr} = \phi V \quad . \tag{11.16}$$

We can use the volume balance equation to eliminate one unknown in each cell, and express the two mass balance equations in terms of the remaining two.

A grid cell can be in one of three states:

- single phase oil
- single phase gas
- two phase oil and gas in equilibrium

and the natural independent solution variables are state dependent.

In the single phase oil region, $V_{gr} = 0$, and Eq. (11.16) is simply

$$V_{or} = B_o V^o = \phi V \quad , \tag{11.17}$$

and using the definition of R_s, we see that we can make the substitutions

$$\begin{aligned}V^o &= V \frac{\phi(p)}{B_o(p, R_s)} \\ V^g &= V \frac{\phi(p)}{B_o(p, R_s)} R_s\end{aligned} \quad , \tag{11.18}$$

The natural unknowns to solve for are then pressure, P, and solution-oil-gas ratio, R_s. In the single phase gas region, we see in a similar fashion that V^o and V^g can be expressed as functions of pressure and R_v. The natural unknowns to solve for are then pressure, P, and solution-gas-oil ratio, R_v. In the two phase region, the fluid properties, R_s, R_v, B_o, and B_g, are functions of pressure only, but V^o and V^g is the sum of contributions proportional to the saturation of each phase:

$$\begin{aligned} V^o &= V\phi(p)\left(\frac{1}{B_o(p)}S_o + \frac{R_v(p)}{B_g(p)}(1-S_o)\right) \\ V^g &= V\phi(p)\left(\frac{R_s}{B_o(p)}S_o + \frac{1}{B_g(p)}(1-S_o)\right) \end{aligned} \quad . \tag{11.19}$$

The natural unknowns are then pressure and saturation. These relations between grid cell state and unknowns are summarized in Table 11.1.

State	Unknowns	V^o/V	V^g/V
Single phase oil	P and R_s	$\frac{\phi}{B_o}$	$\frac{\phi R_s}{B_o}$
Single phase gas	P and R_v	$\frac{\phi R_v}{B_g}$	$\frac{\phi}{B_g}$
Two phase	P and S_o	$\left(\frac{1}{B_o}S_o + \frac{R_v}{B_g}(1-S_o)\right)\phi$	$\left(\frac{R_s}{B_o}S_o + \frac{1}{B_g}(1-S_o)\right)\phi$

Table 11.1: Cell states, corresponding unknowns, and substitutions made in Eq. (11.15).

State changes during a time step need special attention in the non-linear solver.

Note that grid cells may change state as a result of changes in composition and pressure. Since derivatives are not continuous across the phase boundaries (See Figs. 11.6 and 11.7), this can create convergence problems when solving the non-linear equation system with a Newton method.

11.2.1 Fully implicit solution

As explained in in section 10.1.3, the set of non linear equations (11.15) is solved using an iterative Newton method. We have two unknowns per grid cell, P_i and X_i, where X_i is either R_s, R_v, or S_o depending on the grid cell state. We also have two equations per grid cell, and evaluating the left hand side of each of these with a given set of values for the unknowns yields residuals R_{o_i} and R_{g_i}:

$$\mathbf{X} = \begin{bmatrix} P_1 \\ X_1 \\ P_2 \\ X_2 \\ \vdots \\ P_N \\ X_N \end{bmatrix}, \quad \mathbf{R} = \begin{bmatrix} R_{o_1} \\ R_{g_1} \\ R_{o_2} \\ R_{g_2} \\ \vdots \\ R_{o_N} \\ R_{g_N} \end{bmatrix} \quad . \tag{11.20}$$

If $\mathbf{X}^{(m)}$ is the current value for the unknowns (m-th iterate), and $\mathbf{R}^{(m)}$ is the corresponding residuals, an updated value $\mathbf{X}^{(m)}$ is found

as

$$\mathbf{X}^{(m+1)} = \mathbf{X}^{(m)} + \mathbf{\Delta}^{(m+1)} \quad , \tag{11.21}$$

where $\mathbf{\Delta}^{(m+1)}$ is the solution of a system of linear equations

$$\mathbf{J}_R^{(m)} \mathbf{\Delta}^{(m+1)} + \mathbf{R}^{(m)} = 0 \quad . \tag{11.22}$$

Here $\mathbf{J}_R$ is the *Jacobian* matrix; the derivatives of the residual with respect to the unknowns. The Jacobian matrix consist of blocks for each pair (i, j) of grid cells:

$$\mathbf{J}_R = \begin{bmatrix} \mathbf{J}_{1,1} & \mathbf{J}_{1,2} & \mathbf{J}_{1,3} & \cdots & \mathbf{J}_{1,N-1} & \mathbf{J}_{1,N} \\ \mathbf{J}_{2,1} & \mathbf{J}_{2,2} & \mathbf{J}_{2,3} & \cdots & \mathbf{J}_{2,N-1} & \mathbf{J}_{2,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{J}_{N-1,1} & \mathbf{J}_{N-1,2} & \mathbf{J}_{N-1,3} & \cdots & \mathbf{J}_{N-1,N-1} & \mathbf{J}_{N-1,N} \\ \mathbf{J}_{N,1} & \mathbf{J}_{N,2} & \mathbf{J}_{N,3} & \cdots & \mathbf{J}_{N,N-1} & \mathbf{J}_{N,N} \end{bmatrix} \quad , \tag{11.23}$$

where

$$\mathbf{J}_{i,j} = \begin{bmatrix} \dfrac{\partial R_{o_i}}{\partial P_j} & \dfrac{\partial R_{o_i}}{\partial X_j} \\ \dfrac{\partial R_{g_i}}{\partial P_j} & \dfrac{\partial R_{g_i}}{\partial X_j} \end{bmatrix} \quad . \tag{11.24}$$

We have $\mathbf{J}_{i,j} = 0$ except when the blocks i and j are neighbors, making the Jacobian very sparse.

The state of a cell can be different from the state of its neighbors, so it is not possible to give a single expression for the nonzero elements of the Jacobian. The cell state, and thus the unknowns, may also change from one iteration to the next. For single–phase oil cells, a possible state change is checked by comparing $R_s^{(m+1)}$ with R_s on the bubble-point line, or alternatively comparing $P^{(m+1)}$ with p at the bubble point line. For single–phase gas cells, state change is checked by comparing $R_v^{(m+1)}$ with R_v on the dew-point line. In two–phase cells, $S_o^{(m+1)} < 0$ signals a change to single–phase gas, and $S_g^{(m+1)} < 0$ signals a change to single–phase oil.

In terms of B_o, B_g, R_s, and R_v, the expressions for the different derivatives that are needed for building the Jacobian are quite complex, especially in the two phase case. Since the reservoir input is table based, it is however most efficient to use the input to build internal tables for the functions

$$\begin{aligned} \beta_o(p, R_s) &= 1/B_o(p, R_s) \\ \beta_g(p, R_v) &= 1/B_g(p, R_v) \\ \beta_o^*(p) &= 1/B_o(p) \\ \beta_g^*(p) &= 1/B_g(p) \\ \zeta_o(p) &= R_s(p)/B_o(p) \\ \zeta_g(p) &= R_v(p)/B_g(p) \end{aligned} \quad . \tag{11.25}$$

The star on β_p^* and ζ_p^* indicates that the pressure derivatives in two–phase cells are taken along the bubble-point and dew-point lines.

An overview of different derivatives that are needed for building the Jacobian in terms of the tabulated functions defined in Eq. (11.25) can be found in Table 11.2.

	Single–phase oil	Single–phase gas	Two–phase
	$X = R_s$	$X = R_v$	$X = S_o$
$\frac{\partial V^o}{\partial p}$	$\beta_o \frac{d\phi}{dp} + \phi \frac{\partial \beta_o}{\partial p}$	$\left(\beta_g \frac{d\phi}{dp} + \phi \frac{\partial \beta_g}{\partial p}\right) R_v$	$S_o \frac{d\beta_o^*}{dp} + (1 - S_o) \frac{d\zeta_g}{dp} + \frac{V^o}{\phi} \frac{d\phi}{dp}$
$\frac{\partial V^g}{\partial p}$	$\left(\beta_o \frac{d\phi}{dp} + \phi \frac{\partial \beta_o}{\partial p}\right) R_s$	$\beta_g \frac{d\phi}{dp} + \phi \frac{\partial \beta_g}{\partial p}$	$(1 - S_o) \frac{d\beta_g^*}{dp} + S_o \frac{d\zeta_o}{dp} + \frac{V^o}{\phi} \frac{d\phi}{dp}$
$\frac{\partial \lambda_o}{\partial p}$	$\left(\frac{1}{\beta_o} \frac{\partial \beta_o}{\partial p} - \frac{1}{\mu_o} \frac{\partial \mu_o}{\partial p}\right) \lambda_o$	0	$\left(\frac{1}{\beta_o^*} \frac{d\beta_o^*}{dp} - \frac{1}{\mu_o^*} \frac{d\mu_o^*}{dp}\right) \lambda_o$
$\frac{\partial \lambda_g}{\partial p}$	0	$\left(\frac{1}{\beta_g} \frac{\partial \beta_g}{\partial p} - \frac{1}{\mu_g} \frac{\partial \mu_g}{\partial p}\right) \lambda_g$	$\left(\frac{1}{\beta_g^*} \frac{d\beta_g^*}{dp} - \frac{1}{\mu_g} \frac{d\mu_g^*}{dp}\right) \lambda_g$
$\frac{\partial \rho_o}{\partial p}$	$\left(\frac{1}{\beta_o} \frac{\partial \beta_o}{\partial p}\right) \rho_o$	0	$\left(\frac{1}{\beta_o^*} \frac{d\beta_o^*}{dp}\right) \rho_o$
$\frac{\partial \rho_g}{\partial p}$	0	$\left(\frac{1}{\beta_g} \frac{\partial \beta_g}{\partial p}\right) \rho_g$	$\left(\frac{1}{\beta_g^*} \frac{d\beta_g^*}{dp}\right) \rho_g$
$\frac{\partial V^o}{\partial X}$	$\left(\frac{1}{\beta_o} \frac{\partial \beta_o}{\partial R_s}\right) V^o$	$\left(\beta_g + R_v \frac{\partial \beta_g}{\partial R_v}\right) \phi V$	$\left(\beta_o^* - \zeta_g\right) \phi V$
$\frac{\partial V^g}{\partial X}$	$\left(\beta_o + R_s \frac{\partial \beta_o}{\partial R_s}\right) \phi V$	$\left(\frac{1}{\beta_g} \frac{\partial \beta_g}{\partial R_v}\right) V^g$	$-\left(\beta_g^* - \zeta_o\right) \phi V$
$\frac{\partial \lambda_o}{\partial X}$	$\left(\frac{1}{\beta_o} \frac{\partial \beta_o}{\partial R_s} - \frac{1}{\mu_o} \frac{\partial \mu_o}{\partial R_s}\right) \lambda_o$	0	$\frac{\beta_o^*}{\mu_o^*} \frac{dk_{ro}}{dS_o}$
$\frac{\partial \lambda_o}{\partial X}$	0	$\left(\frac{1}{\beta_g} \frac{\partial \beta_g}{\partial R_v} - \frac{1}{\mu_g} \frac{\partial \mu_g}{\partial R_v}\right) \lambda_g$	$\frac{\beta_g^*}{\mu_g^*} \frac{dk_{rg}}{dS_o}$
$\frac{\partial \rho_o}{\partial X}$	$\rho_{gs} \beta_o + \frac{\rho_o}{\beta_o} \frac{\partial \beta_o}{\partial R_s}$	0	0
$\frac{\partial \rho_g}{\partial X}$	0	$\rho_{os} \beta_g + \frac{\rho_g}{\beta_g} \frac{\partial \beta_g}{\partial R_v}$	0

Table 11.2: Derivatives for building the Jacobian as given by Eqs. (11.23) and (11.24).

11.3 Wells and source/sink terms

Source/sink terms are added to the mass balance equations in cells that are penetrated by wells. With the exception of very simple wells and well models, wells will also add extra unknowns and equations, associated with the physical state of the well.

Wells are typically either pressure controlled (Dirichlet) or rate controlled (Neumann). In all cases the sink term (production wells) for component n has the form

Sink term

$$Q_n = C_{wi} \lambda_i^n \left(P_i^t - P_{wi}\right) \quad , \tag{11.26}$$

where C_{wi} is the connection factor, typically calculated using the Peaceman formula (see Section 6.3), and P_{wi} is the wellbore or completion pressure.

Pressure controlled wells

If the well is pressure controlled, and connected to a single cell without any fancy equipment in the completion, the well pressure P_{wi} is a known constant, and no extra equations or unknowns are introduced. Note also that the sink term in this case has the same form as the term for a constant pressure boundary (See Eq. 10.26). A constant pressure boundary condition can therefore always be simulated by introducing dummy wells with judicially selected connection factors. If the well is connected to several cells, or the well pressure is defined at a location different from the completion in the cell, P_{wi} will be different from the controlling well pressure. The difference is typically at least dependent on the hydraulic head, that is the density of the fluid column in the well multiplied with the height difference between where the well pressure is specified and the cell. Since the well fluid density is a function of the reservoir fluid densities and the fractional flow f_o, and hence of the mobility ratio λ_i^o/λ_i^g, P_{wi} will depend on the properties in all cells that are connected to the well. The well state P_{wi} can in this case, at least in principle, be specified without introducing any new unknowns, and the sink term (11.26) will contribute to extra non-zero terms to the Jacobian. However, in practice, instead of hand coding special cases, at least one extra equation is typically added to the equation system for each well.

Rate controlled wells

Due to the way the black oil model is defined, where surface components and surface phases are identical, single cell connected wells that are controlled by the surface rate of one phase, are especially simple to model: Pressures and connection factors can be eliminated from the sink term, and the rates are entered directly. No extra unknowns are introduced. In the case of a specified surface gas rate Q_g^s, we get the sink terms

$$\begin{aligned} Q_o &= \frac{\lambda_i^o}{\lambda_i^g} Q_g^s \\ Q_g &= Q_g^s \end{aligned} \quad . \tag{11.27}$$

Rate control can also be specified in terms of phase specific or total reservoir rates, and a well is typically also connected to more than one cell. In these cases, extra equations must be added to the equation system for each well.

11.4 CO_2 sequestration

This section will describe the use of the black oil model to simulate CO_2 injection into sub-surface reservoirs. CO_2 sequestration is the process of storing CO_2, either by natural processes such as growing forest, or by artificial processes such as capturing CO_2, e.g., from cement production or electricity production from hydrocarbons, and then pumping the CO_2 into subsurface reservoirs. Carbon capture and sequestration into subsurface reservoirs is considered a necessary part to reach the worlds climate goals.

When injected into the subsurface, the CO_2 is typically in a supercritical form. Here we will denote supercritical CO_2 as $scCO_2$, but we will be sloppy with our notation and often use CO_2 also to denote the supercritical phase. The injection of CO_2 into depleted hydrocarbon reservoirs is a complicated process, with multiple phases (hydrocarbon gas, oil, brine and CO_2) and interactions between all of these phases. In this section we will focus on the simpler process of CO_2 being injected into an aquifer. Thus we will have only two phases: CO_2 and brine. The CO_2 component can be dissolved into the brine phase, and the brine component can evaporate into the CO_2 phase. When it comes to fluid properties, a good representation of the evaporation of water into the CO_2 phase is less critical than the dissolution of CO_2 into the brine phase, as the change in properties of the CO_2 phase is minor compared to the change of properties of the brine phase. However, the evaporation of brine into the CO_2 phase increase the salinity of the remaining brine, and can lead to salt precipitation. This can lead to clogging of the near well zone, and thereby to injection problems. Such near well problems will not be treated here.

$scCO_2$ = supercritical CO_2

Note that most reservoirs considered for CO_2 injection are so deep that the CO_2 phase is supercritical, i.e., the pressure and temperature in the reservoir are so high that the CO_2 is above its critical point. Above the critical point the liquid and gas phases do not exist, but rather a supercritical phase.

The supercritical CO_2 is stored by four different trapping mechanisms:

- *Structural trapping:* Structural trapping is the trapping under a cap rock or other stratigraphic units of the supercritical CO_2 during its upward migration due to buoyancy. This is the main trapping mechanism in the short term, i.e., on the time-scale of the injection, on the order of years. This trapping mechanism is similar to the trapping mechanism that create hydrocarbon reservoirs.

- *Capillary trapping:* This is the residual saturation of supercritical CO_2 in the porous medium. During the upward movement of CO_2, the CO_2 will first displace the brine phase (drainage), but after passing through the CO_2 will be displaced by the brine phase again (imbibition). The remaining CO_2 is mostly disconnected and capillary trapped, and thus immobile. This trapping mechanism occurs on the same time-scale as structural trapping.

- *Solubility trapping:* This is CO_2 dissolved in the brine phase. As the density of the brine phase increase with increasing amount of dissolved CO_2, buoyancy will drive this dissolved CO_2 downwards, and securely trapping it. As the density contrast between brine with and without dissolved CO_2 is small (on the order of $\Delta\rho \simeq 10\,\mathrm{kg/m^3}$), the convective transport of dissolved CO_2 is a slow process, and occurs at time-scales longer than the structural and capillary trapping. It is however considered a more secure form of storage.

- *Mineral trapping:* Chemical processes can transform the CO_2 into a solid phase, considered a stable and secure trap for the CO_2. Some sequestration projects target sub-surface structures where

the reservoir minerals react with the injected CO_2, and then rapidly trap the CO_2 as solids. For most sub-surface aquifers the precipitation process will be slow, slower than the other trapping mechanisms, however, it is considered as one of the safest trapping mechanisms.

The $scCO_2$ phase has a lower density than the brine phase, thus due to buoyancy it will rise in the reservoir and form a gas-cap at the top. When CO_2 dissolves into the brine, the brine density increase. This leads to convective mixing of the brine, where brine with high CO_2 content fingers downwards in the reservoir. Such convective mixing is important for CO_2 storage, as large amounts of CO_2 can be transported deep down in the reservoir, reducing the free $scCO_2$ phase and thereby the size of the gas cap. It might eventually remove the gas cap entirely.

In the black-oil model, there is traditionally most interaction between the oil and gas phases, with gas being dissolved in the oil phase. In some (extended) black-oil models the water phase can evaporate, and other components than water can dissolve into the water phase, however, this is not the main feature of the model. For water and CO_2 the dissolution of CO_2 into the water phase is an important process, which therefore needs to be included in the modeling. As the traditional black-oil model disregard interaction between the water phase and the other phases, it is better to treat the water as oil in the traditional black-oil model. So when using a traditional black-oil reservoir simulator to simulate CO_2 sequestration, it is usual to treat the water phase as the oil phase and the $scCO_2$ phase as the gas phase.

In contrast to typical oil-gas systems, where the density of the oil phase is lowered with an increased amount of dissolved gas, in a water-CO_2 system the density of water increase with increased amount of dissolved CO_2. This can be problematic for some black-oil simulators when we use them to simulate CO_2 sequestration, as the simulators might have built-in warnings and errors for the situation with lower oil phase density for increased amount of dissolved gas.

There is an internal PVT model for water-CO_2 systems implemented in OPM-Flow. This CO_2-water PVT model can be invoked by the keyword `CO2STORE`. Using this model removes the need for PVT tables, e.g., `PVTO`, which are disregarded if included when using the `CO2STORE`. The PVT model invoked by the `CO2STORE` keyword use analytical expressions for calculating the amount of dissolved CO_2, the brine (water) density, etc. Note that these functional relationships are also strongly dependent on brine salinity and temperature, so accurate values for these parameters should be provided by the `SALINITY` and `TEMP` keywords.

The model used in `CO2STORE` was introduced in (Sandve et al., 2021).

Another keyword critical for using OPM-Flow to model CO_2 sequestration is the `DRSDT` keyword, which sets an upper limit for the allowed change in R_S values relative to the time-step, i.e., an upper limit for $\partial R_S / \partial t$. The maximum dissolution rate is unfortunately

dependent on the grid block size and geometry. When using the `CO2STORE` keyword, the `DRSDT` keyword is replaced by the `DRSDTCON` keyword, giving the upper limit in a dimensionless form dependent on cell size and maximum dissolution at the current state (pressure and temperature). This form alleviates some of the size dependency. It is still necessary to estimate the dimensionless parameter. e.g., by fine scale simulations where the grid cells are so small that one can assume instant equilibrium between the phases within each grid-cell. Such a fine scale simulation is shown in Fig. 11.8. In this simulation we start with CO_2 as a free phase on the top of the model, and brine in the remainder. When the CO_2 dissolves into the brine phase, the brine becomes heavier, and starts to flow downwards due to buoyancy. Due to mass balance, brine with less amount of dissolved CO_2 then needs to flow upwards to replace the downward moving CO_2 rich brine. This results in the convective fingers observed in Fig. 11.8.

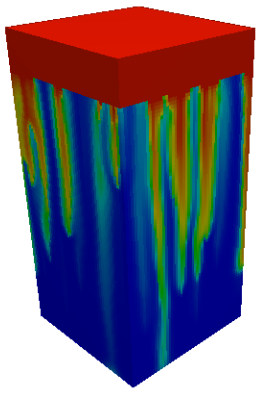


Figure 11.8: Three-dimensional simulation of CO_2 dissolution into the water phase, and convective fingering due to gravity contrast between water with different dissolved CO_2 content. This simulation was conducted by OPM-Flow using the `CO2STORE` keyword.

11.5 Examples

We will in this section present the data for simulating two simple 1D displacement processes with the black oil model. We will use the model data given in Figs. 11.3, 11.4, and 11.5[3].

The displacements involve displacing oil with either oil or gas, but simulations are run with three-phase data. The water saturation is set to $S_{wi} = 0.0001$, as determined by the input oil–water relative permeability table (codeword `SWOF`) shown in Fig. 11.9. Gas–oil relative permeabilities are shown in Fig. 11.10. We see that residual oil after gas injection is $S_{org} = 0.2$.

11.5.1 Gas–oil two phase displacement

In this example we will investigate the process of gas displacing oil. A two phase region will develop where the gas and oil phase are in thermodynamic equilibrium. In the black oil model, as in any two component model, the properties of the two phases in this region will be determined by the pressure.

The simulation model has 100 grid cells, and a total length of 1000 m. Grid cells are $10 \times 10 \times 10$ m. The initial pressure is 250 bar, and the whole reservoir is filled with undersaturated oil with $R_s = 90$. There are two wells: a producer in the last cell, with bottom hole pressure $p_{\mathsf{BHP}} = 249$ bar, and an injector, injecting gas with $R_v = 0$ at a constant reservoir volume rate of $10\,\mathrm{m}^3/\mathrm{d}$ in the first cell.

Results are shown in Fig. 11.11. We see that the gas breaks through in the producer after around 900 days, with a sharp rise in gas–oil ratio, resulting in a reduced oil production. The saturation plot shows that the gas is moving through the reservoir with a quite sharp front. However not as sharp as expected based on the Buckley-Leverett solution for two phase displacement. This indi-

[3] The simulator input files needed for running these examples can be found in the directory `flow/BOSimulations1D` in the repository at `https://bitbucket.org/ntnu_petroleum/ressimbook-material`

```
SWOF
--   Sw              Krw            Kro          Pc
    0.000000        0.000000       1.00000      0
    0.0001          0.000000       0.999        0 -- Swi
    0.0500000       0.000860000    0.847820     0
    0.100000        0.00263000     0.697460     0
    0.150000        0.00524000     0.557170     0
    0.200000        0.00877000     0.432860     0
    0.250000        0.0133800      0.327570     0
    0.300000        0.0192700      0.241770     0
    0.350000        0.0267200      0.174150     0
    0.400000        0.0360800      0.122370     0
    0.450000        0.0478100      0.0837400    0
    0.500000        0.0625000      0.0556500    0
    0.550000        0.0809000      0.0357200    0
    0.600000        0.103940       0.0219900    0
    0.650000        0.132770       0.0128400    0
    0.700000        0.168690       0.00699000   0
    0.750000        0.213020       0.00346000   0
    0.800000        0.266670       0.00149000   0
    0.850000        0.329180       0.000510000  0
    0.900000        0.397060       0.000120000  0
    0.950000        0.461030       0.00001      0
    1.00000         0.500000       0.000000     0
/
```

Figure 11.9: Input to OPM-Flow for the oil–water relative permeabilities. This is simplified data from the Norne model (Ch. 14). The capillary pressure, P_c, is set to zero.

```
SGOF
--  Sg              Krg            Kro            Pc
   0.000000        0.000000       1.00000        0.0
   0.0500000       0.00165500     0.806888       0.0
   0.100000        0.00691300     0.633562       0.0
   0.150000        0.0162130      0.485506       0.0
   0.200000        0.0299900      0.364043       0.0
   0.250000        0.0486550      0.267589       0.0
   0.300000        0.0725730      0.192992       0.0
   0.350000        0.102046       0.136554       0.0
   0.400000        0.137287       0.0946710      0.0
   0.450000        0.178402       0.0641510      0.0
   0.500000        0.225368       0.0423240      0.0
   0.550000        0.278030       0.0270350      0.0
   0.600000        0.336093       0.0165860      0.0
   0.650000        0.399135       0.00966200     0.0
   0.700000        0.466631       0.00525400     0.0
   0.750000        0.538000       0.002          0.0
   0.800000        0.612665       0.00           0.0 -- 1-Sorg
   0.850000        0.690169       0.00           0.0
   0.900000        0.770395       0.0000         0.0
   0.950000        0.854218       0.00000        0.0
   0.9999          0.9499         0.000000       0.0
   1.00000         0.950000       0.000000       0.0
/
```

Figure 11.10: Input to OPM-Flow for the gas–oil relative permeabilities. Amended data from the Norne model (Ch. 14); with residual oil to gas, S_{org}, set to 0.2.

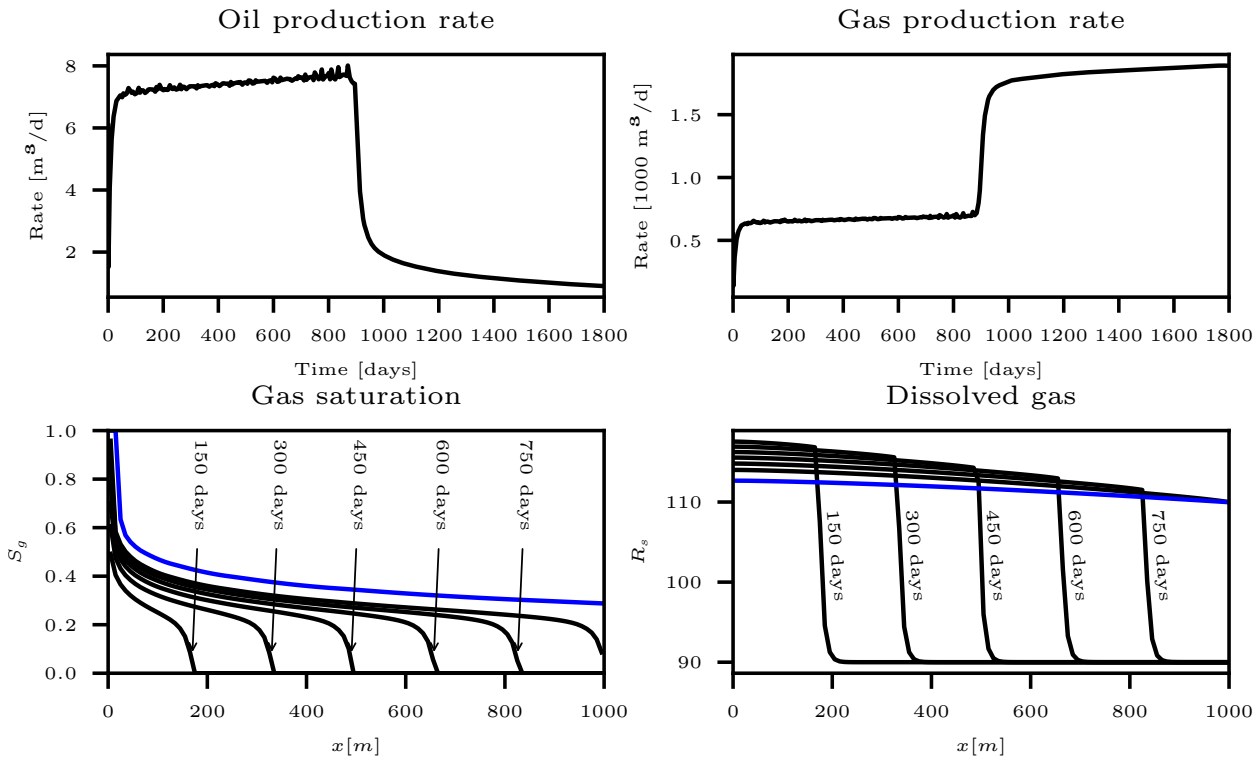


Figure 11.11: Gas displacing oil. **Top:** Production rates. **Bottom:** Gas saturation, and dissolved gas component in oil phase, R_s, as a function of distance from injector, at different times. The blue line is the state after 1800 days.

cates that there is some numerical diffusion. Notice also the sharp rise in gas saturation in the region close to the injector; at the end of simulation, the oil saturation in the cells close to the producer is zero. Since the oil component can be dissolved in the gas phase the amount of oil will be reduced by evaporation even when the oil is immobile. In real hydrocarbon systems, only the lightest components will evaporate, and the oil saturation will never be reduced to zero. The effect of evaporation is exaggerated in a two-component model like the black oil model.

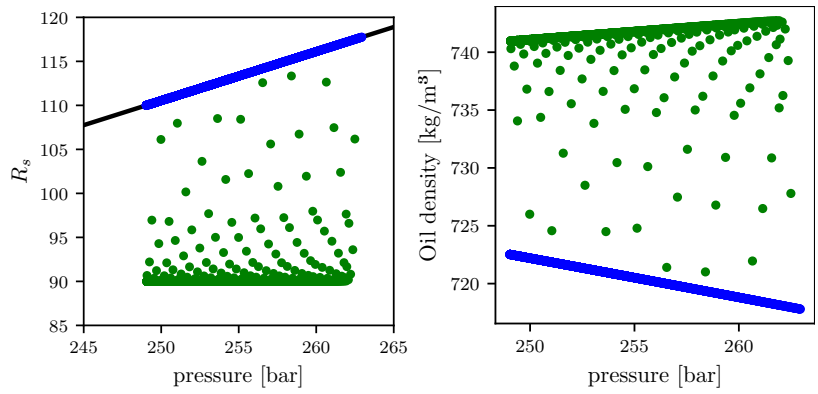


Figure 11.12: Dissolved gas, R_s, and oil mass density, ρ_o, in all cells at selected times (each 30 days). Green dots is single phase oil, blue is oil in the two phase region.

Figure 11.12 shows dissolved gas, R_s, and oil mass density, ρ_o, in all cells at selected times. We see clearly the oil properties in the two–phase region is a function of pressure alone.

11.5.2 Single phase displacement

In this example we will investigate a single phase process where a heavy oil with $R_s = 0$, is displacing a lighter oil with $R_s = 90.0$. The oil and gas components move together with in the oil phase at the same speed. Since we discard diffusion, and physical dispersion, there should be a sharp displacement front with $R_s = 0$ behind the front and $R_s = 90$ ahead of the front.

The simulation model has 100 grid cells, and a total length of 1000 m. Grid cells are $10 \times 10 \times 10$ m. The initial pressure is 250 bar, and most of the reservoir is filled with undersaturated oil with $R_s = 90$. The first 10 grid cells (100 m) is filled with oil with $R_s = 0$ There are two wells: a producer in the last cell, with bottom hole pressure $p_{\mathsf{BHP}} = 249$ bar, and an injector, injecting oil[4] with $R_s = 0$ at a constant reservoir volume rate of $10\,\mathrm{m}^3/\mathrm{d}$ in the first cell.

Results are shown in Fig. 11.13. We see that the gas-oil ratio, R_s, in the producer start changing at around 1500 days, but complete breakthrough of the injected oil is only experienced after 2500 days. The displacement front is initially quite sharp, but the width increase with time. The width of the front is more than hundred meters at the time of breakthrough. We can conclude that this fully implicit simulation suffers from severe numerical diffusion.

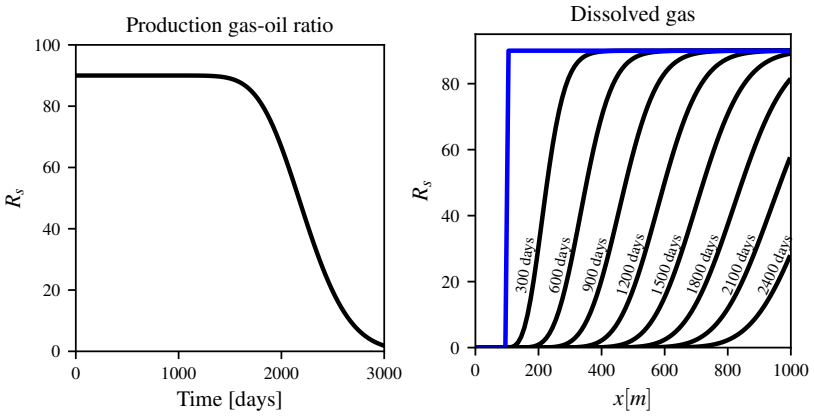


Figure 11.13: Oil displacing oil. **Left:** Gas-oil ratio, R_s, in the produced oil. **Right:** Composition profile at different times. Initial composition profile in blue.

[4] With the current version of OPM-Flow (2001.04), injected oil has always $R_s = 0$ and injected gas has $R_v = 0$. To inject gas or oil with other compositions we can use two (pseudo) wells.

11.6 Exercises

Exercise 11.1 A grid cell is in the two phase state, and contain V^o of oil component and V^g of gas component. Derive the formula for partial volume of oil phase (oil saturation).

Exercise 11.2 Using the data in Figs. 11.3, 11.4, and 11.5, find

- Bubble point pressure for an oil with $R_s = 90$
- R_s of oil in the two phase region at pressure 250 bar
- R_v of gas in the two phase region at pressure 250 bar

- Mass fraction of gas component in the oil phase and the gas phase at pressure 250 bar

Exercise 11.3 For the systems in section 11.5:

- Create simulation models with different grid resolution, and run these models with different maximal time steps. Maximum steps are controlled by the code word `TUNING`, and OPM-Flow should be run as
  ```
  flow --enable-tuning=1 --enable-opm-rst-file=1 XXX.DATA
  ```
- Estimate the amount of numerical diffusion.
- How is numerical diffusion effected by Δx and Δt?

12
Upscaling

> He looked in to my eyes and thought he saw a love light shine
> Nearly did he know that that light was dollar signs
>
> Dolly Parton - I'll Oilwells Love You

Upscaling

Upscaling is the process of replacing fine-scale properties by effective properties on a larger scale. For a grid, this means replacing a fine grid by a coarser grid, where the fine and coarse grid have similar response to changes in boundary conditions. This grid coarsening process is illustrated in Fig. 12.1.

Upscaling is unavoidable when we want to deal with fluid flow in a reservoir due to the vast scale and heterogeneities at all scales. It is theoretically possible to keep track of all fluid interfaces and the fluid composition throughout a reservoir, however, it is not practically possible. As the Germans say: *Es ist möglich, aber etwas unwahrscheinlich.* We therefore need upscaling.

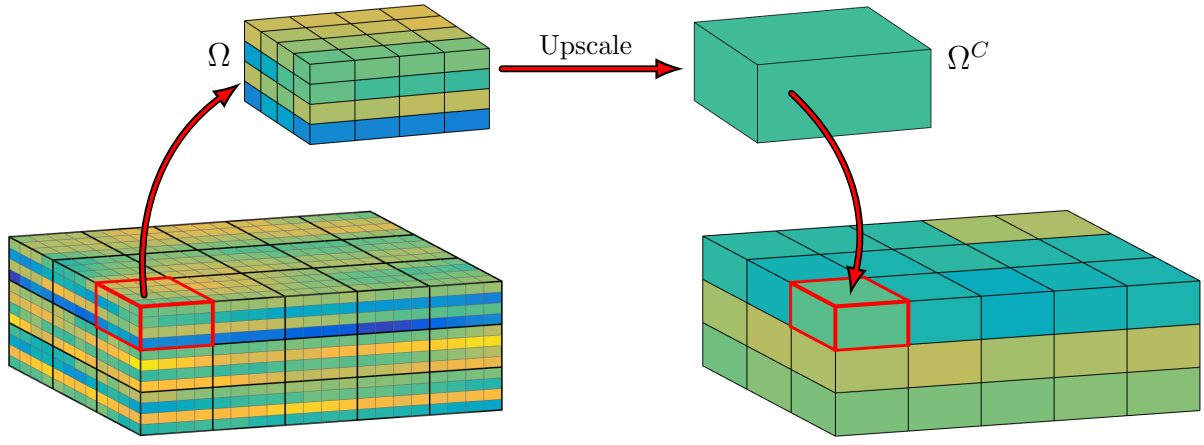


Figure 12.1: Upscaling from a fine grid Ω to a coarse grid Ω^C. The properties of sets of fine cells are replaced by effective properties for the coarse cells through upscaling.

We have already encountered upscaling. In Subsec. 3.2.1 we discussed the continuum model, which can be considered as an upscaled version of the pore scale. At the pore scale properties such as porosity and permeability are not well defined. Given a volume element at a larger scale, we can upscale the pore structure, i.e. the

full description of pore-matrix interfaces, to effective properties such a porosity and permeability for the volume element.

We have also encountered upscaling in our finite difference scheme. For properties at the boundary between two grid cells, e.g. at the point $i + 1/2$ between the cells i and $i + 1$ using our index system, we used the harmonic mean as an upscaled value for the permeability.

Geological structures in a reservoir exist at a range of length scales, and structures at one scale are build up of structures at smaller scales. Likewise, measured data are obtained at different scales, from core-flooding on plug samples in the range of cm to well tests which yield effective properties for the full height of the reservoir. To integrate data from different scales one need upscaling procedures. Further, it is not possible to represent all the finer structures in a finite grid, so the grid model only represents the larger geological structures. As small scale heterogeneities might be important for flow on larger scales, they need to be taken into account through upscaling.

In this chapter we will start by considering single phase properties such as porosity and permeability. Then, we will briefly investigate upscaling of two phase flow. We will start by investigating the force balance between gravity, viscous and capillary forces. Afterwards we will look into some upscaling examples, highlighting how different flow regimes will influence the coarse scale relative permeability and capillary pressure.

12.1 Additive properties

Upscaling of volumetric properties, such as porosity and saturation, is based on upscaling of volumes, which are additive properties. This type of upscaling is fairly straight forward. It is however illustrative for other upscaling processes, so we will describe it in detail in this section.

Consider a fine grid model with a corresponding coarse grid, and let $\mathcal{S}$ be the set of fine grid cells inside a coarse grid block, as illustrated in Fig. 12.1. For an additive property, such as total volume V_t and connected pore volume V_c, the coarse grid block values are just the sum of the values for the corresponding fine grid cells:

$$V_t^C = \sum_{i \in \mathcal{S}} V_t(i) \tag{12.1}$$

$$V_c^C = \sum_{i \in \mathcal{S}} V_c(i) \quad . \tag{12.2}$$

Here the superscript C indicates that this is a upscaled coarse grid property.

Porosity is not an additive property, but it is a fraction of the two additive properties connected pore volume V_c and total volume V_t as $\phi = V_c/V_t$. Thus we can use the upscaled versions of the

additive values to obtain the upscaled porosity:

$$\phi^C = \frac{V_c^C}{V_t^C} = \frac{\sum_{i \in \mathcal{S}} V_c(i)}{\sum_{i \in \mathcal{S}} V_t(i)} = \frac{\sum_{i \in \mathcal{S}} V_t(i)\phi(i)}{V_t^C} \quad . \tag{12.3}$$

We observer that the upscaled porosity is the volume-weighted average of the porosity for the corresponding fine scale grid cells. Here we use the porosity definition $\phi(i) = V_c(i)/V_t(i)$ for the individual fine scale grid cells.

Any volumetric property can be upscaled similarly to the upscaling of porosity. For example, saturation can be defined as a fraction of fluid volumes, and the fluid volumes are additive properties. We can use the upscaled fluid volumes to obtain an upscaled saturation. This is left as Exercise 12.1.

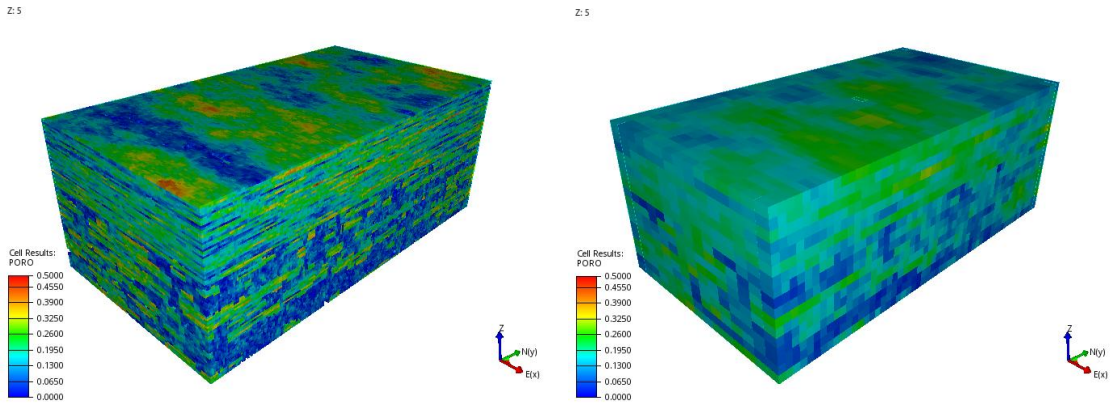


Figure 12.2: Porosity field for both the original fine scaled the SPE10 model and the upscaled coarse scale counterpart.

To illustrate upscaling of porosity, we will use model 2 from 10th SPE Comparative Solution Project (SPE10). The SPE models are a set of reference models for reservoir simulation. SPE10 was designed to be a benchmark case for upscaling of procedures. The original benchmark study can be found in (Christie and Blunt, 2001). SPE10 model 2 consists of two parts representing north sea reservoirs. Both parts are from the Brent reservoir: an upper part consisting of a shallow-marine Tarbert formation where the porosity values have a normal distribution, and a lower section consisting of a fluvial Upper Ness formation. The fluvial lower part is channelized, with high porosity in the channels and low porosity background sand. This leads to a bimodal porosity distribution.

The Brent reservoir is named after the coastal bird of the same name, and the initials of the formations in this reservoir forms the reservoir name: Broom, Rannoch, Etive, Ness and Tarbert.

The original SPE10 model has a grid size of $60 \times 220 \times 85$ cells, while the coarser model has $15 \times 55 \times 17$ cells. In Fig. 12.2 we have plotted the porosity values for SPE10 model 2, both the original fine model and a coarsened version where we have used Eq. (12.3) to calculate the upscaled porosity values. As can be seen from the two porosity fields, the finer details are lost when coarsening the model.

We also observe from Fig. 12.2 that the extreme values are lost when we coarsen. To highlight this fact, we have plotted the porosity values from the fine and coarse models in Fig. 12.3. We see that the variance in porosity is significantly reduced when we coarsen the model. The channelized lower part of the reservoir model con-

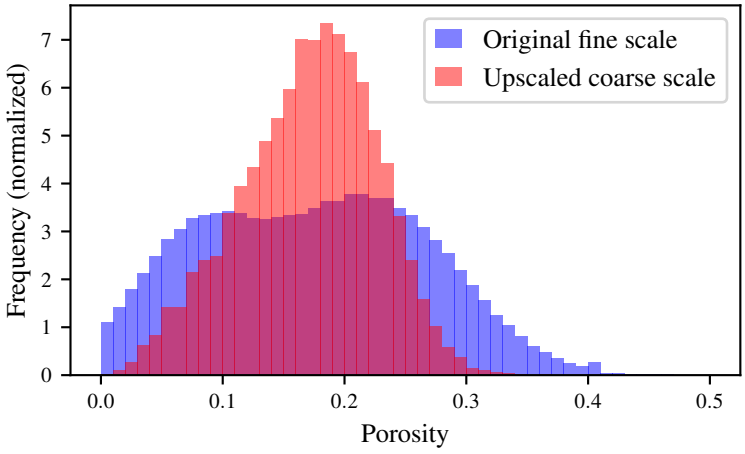


Figure 12.3: The distribution of porosity values for the fine and coarse models shown in Fig. 12.2.

sist of high porosity and permeability channels inside a low porosity and permeability background sand. This part of the reservoir has a strongly bi-modal porosity distribution; this bi-modal porosity distribution can also be observed in the overal porosity distribution for the reservoir in Fig. 12.3. However, we see that bi-modal signature of the channels are lost in the upscaled porosity distribution.

This porosity upscaling example is instructive for how to deal with property values in reservoir models, as all reservoir models are at some level of coarseness compared to the underlying property field. Using the variance in porosity from core plugs or from well logs are in no way representative for the variance in the larger scale grid blocks in a reservoir model. One should therefore never use the property variance from one scale to populate a property model at another scale.

12.2 Single phase transport

Upscaling of single phase transport, such as single phase flow, does not have an analytical solution. Thus *a* single solution does not exists for upscaling of permeability. Accordingly, and in contrast to the analytical upscaling solutions for porosity and saturation in the previous section, there is a range of different methods for upscaling single phase transport properties such as permeability.

We have already encountered upscaling of single phase transport when we upscaled from the pore scale to the continuum Darcy scale in Chap. 3. In that chapter we presented the concept of a representative elementary volume as an essential part of the upscaling procedure. In this section we will look at upscaling of permeability for a grid model, similar to the grid upscaling of porosity and saturation in the previous section, and as visualized in Fig. 12.1. We will consider averaging and flow based methods. We start with averaging methods.

As mentioned, in general there is no analytical solution, conse-

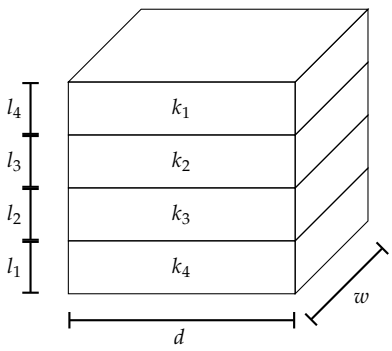


Figure 12.4: Figure indicating a layered model, with layers of different permeability.

quently there is a range of different methods with their strengths and weaknesses. The simplest methods are *averaging methods*, and we have encountered them before.

Averaging methods

Consider the layered model in Fig. 12.4. For this model we have analytical solutions for the permeability along the layers or perpendicular to the layers. Let us start by considering horizontal flow, and assume a constant pressure drop over the model, thus a left pressure p_l and a right pressure p_r that are constant over the full side-plane of the model. We then have Darcy's law for each layer as

$$Q_i = k_i w l_i \frac{p_r - p_l}{d} \quad , \tag{12.4}$$

where w is the width of the model, and d is the depth of the model, as indicated in Fig. 12.4. The total volumetric flow rate $Q = \sum_i Q_i$ is then related to an upscaled horizontal permeability k_h^C as

$$Q = k_h^C w l \frac{p_r - p_l}{d} \quad , \tag{12.5}$$

where $l = \sum_i l_i$ is the height of the model. Combining Eqs. (12.4) and (12.5), we get

$$\begin{aligned} Q &= \sum_i Q_i \\ k_h^C w \sum_i l_i \frac{p_r - p_l}{d} &= w \sum_i k_i l_i \frac{p_r - p_l}{d} \\ k_h^C &= \frac{1}{\sum_i l_i} \sum_i k_i l_i \quad . \end{aligned} \tag{12.6}$$

So, for horizontal flow, i.e., flow along the layers, the effective permeability is the height-weighted arithmetic mean

$$k_h^C = \frac{1}{\sum l_i} \sum l_i k_i \quad , \tag{12.7}$$

where the sum runs over all the layers i. We see that by multiplying all lengths with the horizontal area of the model, we get a volume-weighted arithmetic mean instead of a mean weighted by the layer-heights.

Let us now consider flow perpendicular to the layers, i.e., vertical flow in the model depicted in Fig. 12.4. In this case, due to mass balance, the volumetric flow rate through each layer much be the same, thus $Q_i = Q$ is constant. For each layer we have Darcy's law given as

$$Q_i = k_i w d \frac{\Delta p_i}{l_i} \quad , \tag{12.8}$$

where Δp_i is the pressure drop over the i-th layer (to include gravity, either consider the hydraulic head or include a $\rho g \Delta l_i$ term). The total volumetric flow rate $Q = Q_i$ is then related to an upscaled vertical permeability k_v^C as

$$Q = k_v^C w d \frac{\Delta p}{l} \quad , \tag{12.9}$$

where $\Delta p = \sum_i \Delta p_i$ is the total pressure drop in the vertical direction over the model, and $l = \sum_i l_i$ is the height of the model.

Equating Eqs. (12.8) and (12.9), we have

$$\begin{aligned} Q &= Q_i \\ k_v^C wd\frac{\Delta p}{l} &= Q_i \\ k_v^C &= \frac{1}{wd}\frac{l}{\sum \frac{\Delta p_i}{Q_i}} \\ k_v^C &= \frac{\sum_i l_i}{\sum \frac{l_i}{k_i}} \quad . \end{aligned} \tag{12.10}$$

Thus, for flow perpendicular to the layers, the effective permeability is the harmonic mean of the permeability values, weighted by the heights l_i:

$$k_v^C = \frac{\sum l_i}{\sum \frac{l_i}{k_i}} \quad . \tag{12.11}$$

From the equations for horizontal and vertical flow we can observe that the high-permeable layer dominates the effective permeability for the horizontal flow k_h^C, while the low-permeable layer dominates for the effective permeability for vertical flow k_v^C. Thus, the horizontal and vertical permeability values can be very different. They are equal if and only if all the layers have the same permeability, i.e., for a homogeneous medium. As the arithmetic average is always higher than the harmonic average, for all other cases than the homogeneous medium we have that the permeability along the layering is higher than the permeability perpendicular to the layering. Thus, it is always easier for the fluid to flow along the layers than perpendicular to them.

The two analytical averaging methods does not hold for any other grid than the layered one, and for any other direction than the layer direction of perpendicular to the layer direction. Still, the arithmetic and harmonic mean are outer bounds for the permeability. The geometric mean, which always gives a value between the arithmetic and harmonic mean, can be considered as a compromise. While the geometric mean can give fair values, see, e.g., Warren and Price (1961), it is off course nothing more than an approximation. Other popular approximations include the harmonic-arithmetic averaging, while renormalization might be a more effective upscaling technique for strong permeability contrasts (King, 1989).

The arithmetic and harmonic mean as outer bounds dates back to (Wiener, 1912), and are called Wiener bounds.

Another set of single phase upscaling methods are defined by the need to solve an incompressible steady-state flow, known as *flow based methods*. With these methods we need to solve the single phase flow equations, e.g., as described in Chap. 5. Flow based methods are considered better than averaging techniques, as they take the underlying permeability distribution into account.

Flow based methods

The flow based upscaling methods can roughly be divided into two categories; local and global methods. For the *local flow based upscaling* methods, each coarse grid block is upscaled individually, as indicated in Fig. 12.5. For each coarse block, the permeability can be upscaled in different directions: A constant pressure drop is applied over the two opposite sides in the upscaling direction. Either

Local flow based upscaling

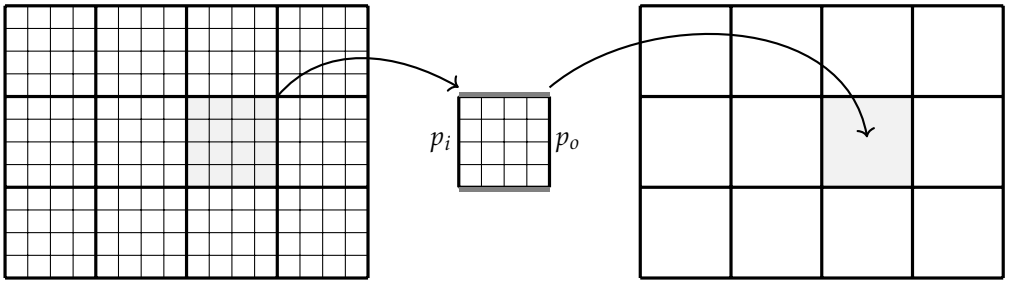


Figure 12.5: A sketch indicating the process of local flow based upscaling for a two dimensional permeability grid. The fine grid is indicated with thin lines, while the coarse grid is indicated by thicker lines. One coarse block is extracted for flow based upscaling.

no-flow or periodic boundary conditions are applied to the other sides. When using no-flow for the other sides, we only obtain the diagonal terms of the upscaled permeability tensor, while periodic boundary conditions can give off-diagonal terms too. The magnitude of the pressure drop $p_o - p_i$ does not matter, as this pressure drop will be scaled out. Equivalently, the final permeability solution is independent of the viscosity used in the solution of the single phase flow. Solving the flow field yields a total volume rate Q, and we then obtain the upscaled permeability in the given direction for the coarse grid block as

$$k^C = \frac{Q\mu l}{p_o - p_i} \quad , \tag{12.12}$$

where l is the length in of the coarse block in direction of the applied pressure drop.

Global flow based upscaling

The *global flow based upscaling* methods solve the fluid flow equations for the entire grid. After obtaining a global flow solution, the pressure drop and volumetric flow rate over the individual coarse grid cells are summed up to obtain the upscaled permeability values as

$$k^C = \frac{\overline{Q}\mu l}{\overline{p}_o - \overline{p}_i} \quad , \tag{12.13}$$

where the over-lines indicate average values.

In the example indicated in Fig. 12.6, we want to calculate the horizontal permeability in the extracted highlighted coarse block: The inlet pressure would be the pore-volume average of all the pressures on the left side, and the outlet pressure would be a similar calculation on the right side. The outlet volumetric rate would be the integral of the fluxes, in this case the sum of the cell-boundary volumetric rates q_i. The horizontal volumetric rate would be the average of the inlet and outlet flow rates.

The global upscaling approach will give different results for different boundary conditions used for the flow field solution. In the simplified example in Fig. 12.6 one could define constant pressure on two opposite side planes. Another possibility is to use the actual wells in the reservoir model, however, then the method is only valid for coarse blocks without wells. The global method gives good results on the coarse scale when using the same boundary conditions as was used to find the upscaled permeability values. However, it might be less robust for changes in boundary conditions compared to the local approach.

In addition to the local and global flow based methods, there are a range of hybrid methods. It is also common to upscale the transmissibility values instead of the permeability values.

12.3 Force balance

This section is based on the paper (Hilden and Berg, 2016).

We start by considering two phase incompressible flow. From Eq. (9.2) we have the mass balance for each phase given by

$$-\frac{\partial}{\partial x}\left(\rho_i q_i\right) = \frac{\partial}{\partial t}\left(\phi \rho_i s_i\right) \quad , \tag{12.14}$$

where ρ_i and q_i is the density and Darcy velocity of phase i, respectively. Assuming solely two phases, we have $s_o + s_w = 1$. As we consider incompressible flow, the densities ρ_i are constant, thus Eq. (12.14) reduces to

$$-\frac{\partial q_i}{\partial x} = \frac{\partial}{\partial t}\left(\phi s_i\right) \quad . \tag{12.15}$$

The extended Darcy equation was introduced in Eq. (9.3) as

$$q_i = -\frac{k k_{ri}}{\mu_i}\frac{\partial p_i}{\partial x} \quad . \tag{12.16}$$

Here k_{ri} is the relative permeability of phase i, and p_i is the pressure in phase i. The phase pressures are linked through the :

$$p_c = p_o - p_w \quad . \tag{12.17}$$

If we include gravity, then Eq. (12.16) can be extended as follows:

Darcy's law with gravity term

$$q_i = -\frac{k k_{ri}}{\mu_i}\left(\frac{\partial p_i}{\partial x} - \rho_i g \frac{\partial z}{\partial x}\right) \quad . \tag{12.18}$$

Here the last term gives the gravity contribution, where it is assume that the x-direction is tilted versus the depth given by z. Remember that the pressure and gravity terms can be joined into the head, as shown in Eq. (3.17).

We now introduce the phase mobility $\lambda_i = k_{ri}/\mu_i$, and obtain the two extended Darcy equations as follows:

$$\begin{aligned} q_o &= -k\lambda_o\left(\frac{\partial p_o}{\partial x} - \rho_o g \frac{\partial z}{\partial x}\right) \\ q_w &= -k\lambda_w\left(\frac{\partial p_w}{\partial x} - \rho_w g \frac{\partial z}{\partial x}\right) \end{aligned} \quad . \tag{12.19}$$

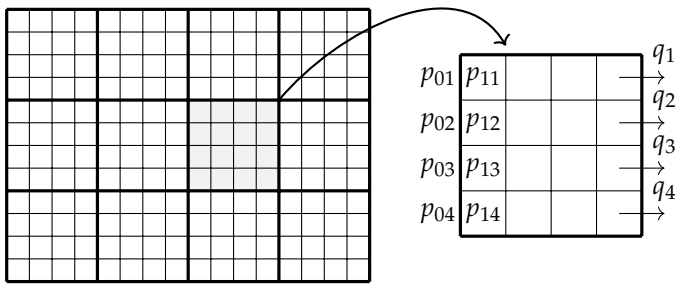


Figure 12.6: A sketch indicating the process of global flow based upscaling for a two dimensional permeability grid. One coarse block is extracted from the grid, and the pressure and flow rate are calculated for the different side planes.

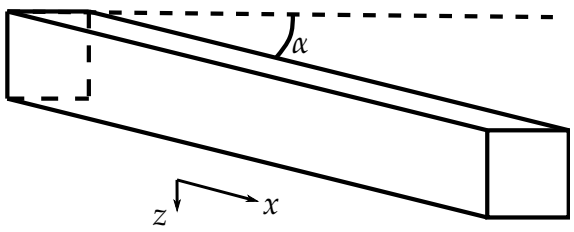


Figure 12.7: A pseudo-1D slab tilted an angle α from the horizontal. The cross-sectional area is constant.

If we subtract λ_o/λ_w times the water equation in Eq. (12.19) from the oil equation in Eq. (12.19), then we get

$$\begin{aligned} q_o - \frac{\lambda_o}{\lambda_w} q_w &= -k\lambda_o \left(\frac{\partial (p_o - p_w)}{\partial x} - (\rho_o - \rho_w) g \frac{\partial z}{\partial x} \right) \\ &= -k\lambda_o \left(\frac{\partial p_c}{\partial x} - \Delta\rho g \frac{\partial z}{\partial x} \right) \quad , \end{aligned} \tag{12.20}$$

where $p_c = p_o - p_w$ is the capillary pressure, and $\Delta\rho = \rho_o - \rho_i$ is the density contrast.

Let $q = q_o + q_w$ be defined as the total Darcy velocity, then

$$\begin{aligned} q_o - \frac{\lambda_o}{\lambda_w} q_w &= q - q_w - \frac{\lambda_o}{\lambda_w} q_w \\ &= q - \left(1 + \frac{\lambda_o}{\lambda_w} \right) q_w \\ &= q - \frac{\lambda_w + \lambda_o}{\lambda_w} q_w \\ &= q - \frac{1}{f_w} q_w \quad , \end{aligned} \tag{12.21}$$

where $f_w = \lambda_w/(\lambda_w + \lambda_o) = 1/(1 + \lambda_o/\lambda_w)$ is the related to the fractional flow of water .

Here we define f_w using the equation for fractional flow as given by Eq. (9.11), even though Eq. (9.11) was derived for a case without capillary pressure.

Replacing the left hand side of Eq. (12.20) with Eq. (12.21) and rearranging, we obtain

$$q_w = f_w q + k f_w \lambda_o \left(\frac{\partial p_c}{\partial x} - \Delta\rho g \frac{\partial z}{\partial x} \right) \quad . \tag{12.22}$$

If we insert Eq. (12.22) into Eq. (12.15), we get:

$$-\frac{\partial \phi s_w}{\partial t} = \frac{\partial f_w q}{\partial x} + \frac{\partial}{\partial x} \left(k f_w \lambda_o \left(\frac{\partial p_c}{\partial x} - \Delta\rho g \frac{\partial z}{\partial x} \right) \right) \quad . \tag{12.23}$$

Since $s_w + s_o = 1$, we can add the mass conservation equations, Eq. (12.15), for the two different phases to obtain

$$\begin{aligned} -\frac{\partial q_w}{\partial x} - \frac{\partial q_o}{\partial x} &= \frac{\partial}{\partial t} (\phi s_w) + \frac{\partial}{\partial t} (\phi s_o) \\ -\frac{\partial q_w + q_o}{\partial x} &= \frac{\partial}{\partial t} (\phi (s_w + s_o)) \\ -\frac{\partial q}{\partial x} &= \frac{\partial}{\partial t} (\phi) = 0 \quad , \end{aligned} \tag{12.24}$$

thus

$$\frac{\partial f_w q}{\partial x} = q \frac{\partial f_w}{\partial x} + f_w \frac{\partial q}{\partial x} = q \frac{\partial f_w}{\partial x} \quad . \tag{12.25}$$

Inserting Eq. (12.25) into Eq. (12.23) we get

$$-\frac{\partial \phi s_w}{\partial t} = q\frac{\partial f_w}{\partial x} + \frac{\partial}{\partial x}\left(k f_w \lambda_o \left(\frac{\partial p_c}{\partial x} - \Delta\rho g \frac{\partial z}{\partial x}\right)\right) \quad . \tag{12.26}$$

This equation is useful to investigate how different forces affect the flow behavior. We can split Eq. (12.26) into three parts, as follows:

$$-\frac{\partial \phi s_w}{\partial t} = \underbrace{q\frac{\partial f_w}{\partial x}}_{\text{viscous}} + \underbrace{\frac{\partial}{\partial x}\left(k f_w \lambda_o \frac{\partial p_c}{\partial x}\right)}_{\text{capillary}} - \underbrace{\frac{\partial}{\partial x}\left(k f_w \lambda_o \Delta\rho g \frac{\partial z}{\partial x}\right)}_{\text{gravitational}} \quad . \tag{12.27}$$

Here the first part scale with the total flow rate q, and is thus a measure of the viscous forces. The second part scales with the gradient in capillary pressure, and is thus a measure of capillary forces. The last part scales with the product of the density contrast and the z-gradient, and is a measure of the body force from gravity, and thus a measure of the contribution to fluid redistribution from gravitational segregation.

12.4 Limiting solutions

For high flow rates, i.e. when q is large, we observe that the viscous part will dominate the right hand side of Eq. (12.27). Thus, we can simplify the high flow rate solution to

High rate limit solution

$$-\phi\frac{\partial s_w}{\partial t} = \frac{\partial f_w}{\partial x} q \quad . \tag{12.28}$$

If we set the capillary pressure and density contrast to zero, our solution would be given by Eq. (12.28). At steady state we have $\delta s_w / \delta t = 0$, thus we get $\frac{\partial f_w}{\partial x} q = 0$. Hence the fractional flow is constant (at least for the non-trivial cases $q \neq 0$, while for the trivial case $q = 0$ the fractional flow f_w is undefined).

Generalizing the equations to three dimensions, it can be shown that we end up with the equation $\nabla f_w \cdot q = 0$. This equation tells us that the fraction flow of water is constant in the direction of flow, where the direction of flow is given by the Darcy velocity vector q. As the fractional flow of water is constant in the direction of flow, the fractional flow of water will be constant along each streamline.

See (Hilden and Berg, 2016) for a general derivation in three dimensions.

For low flow rates, i.e. when q is small, then $(\partial f_w / \partial x) q$ will diminish, and Eq. (12.28) will be dominated by the capillary and gravitational forces:

Low rate limit solution

$$-\phi\frac{\partial s_w}{\partial t} = \frac{\partial}{\partial x}\left(k f_w \lambda_o \left(\frac{\partial p_c}{\partial x} - \Delta\rho g \frac{\partial z}{\partial x}\right)\right) \quad . \tag{12.29}$$

At steady state, $\partial s_w / \partial t = 0$, this gives

$$\frac{\partial p_c}{\partial x} = \Delta\rho g \frac{\partial z}{\partial x} \quad . \tag{12.30}$$

Thus at steady state, the capillary forces is in equilibrium with gravity. Generalizing to three dimensions, we have

$$\nabla p_c = \Delta\rho g \nabla z \quad . \tag{12.31}$$

This can be further simplified if we assume that gravity is negligible, e.g. for small samples where the height difference is comparatively small. Then Eq. (12.31) is simplified to:

$$\nabla p_c = 0 \quad . \tag{12.32}$$

In this limit, the capillary pressure is equal throughout the model, as the gradient in capillary pressure has vanished.

12.4.1 Capillary limit upscaling

We can use Eq. (12.32) to upscale flow parameters in the capillary limit, i.e. where we have low flow rates and where the capillary forces are strong. As an example, consider a model consisting of high permeable and low permeable layers, as depicted in Fig. 12.8.

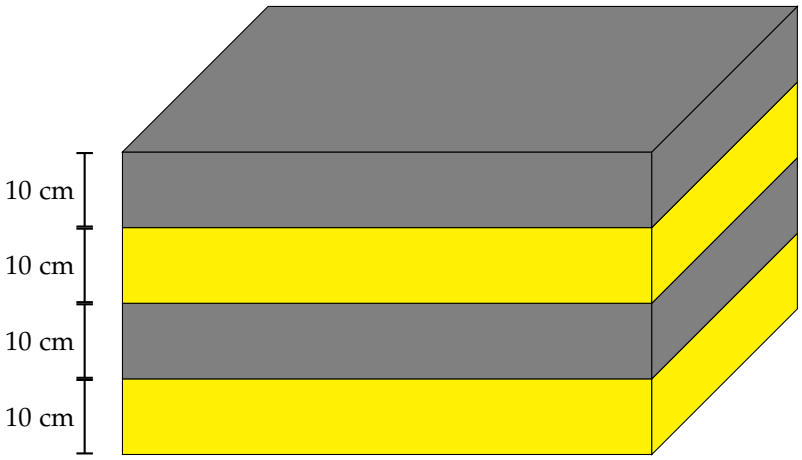


Figure 12.8: Figure indicating a layered model, with repeating layers of high and low permeability.

If we assume Eq. (12.32), then the capillary forces will redistribute the fluids until there are no gradients in the capillary pressure. Assume the capillary pressure curve for the high permeability and low permeability layers are as plotted in Fig. 12.9.

In the capillary limit, as the capillary pressure is equal throughout the model, both the high and low permeability layer should have the same capillary pressure. In Fig. 12.9 we have chosen a capillary pressure p_c. We can then find the saturation values for the two layers at capillary pressure p_c; the saturation in the high permeable layer is indicated as s_{wh}, while the saturation in the low permeability layer is indicated as s_{wl}. We see that the low permeable layer has a higher water saturation than the high permeable layer. This is because the rock we are considering is water wet, and then water will be sucked into the smaller pores.

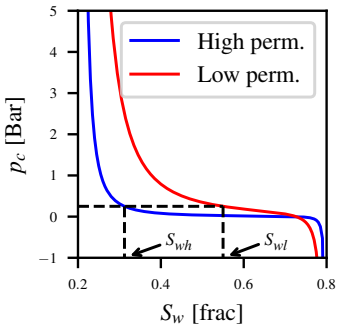


Figure 12.9: The capillary pressure curves for the high and low permeability layers in the previous figure. The dashed line indicates a single capillary pressure, and the corresponding saturation in the high and low permeable layers.

Given the saturation in each layer, we can then calculate the saturation in the full model as

$$\bar{s}_w(p_c) = \frac{s_{wl}(p_c)\phi_l V_l + s_{wh}(p_c)\phi_h V_h}{\phi_l V_l + \phi_h V_h} \quad , \tag{12.33}$$

where $V_i\phi_i$ is the pore volume of the high $i = h$ and low $i = l$ permeable layers. The resulting capillary pressure curve (the inverted

curve of $\bar{s}_w(p_c)$, i.e. the curve of $\bar{p}_c(s_w)$, where the bars indicate upscaled values) is shown in Fig. 12.10.

Given a saturation distribution (as indicated in Fig. 12.10), we can calculate the corresponding relative permeability for the two phases. For simplicity, we will assume that we only have one set of relative permeability curves k_{rw} given by the Corey exponents $n_w = 2.0 = n_0$ and $k_{rwo} = 0.4$. These curves are shown by the dashed lines in Fig. 12.11. Extending to two sets is simple. For a given set of saturation for the high s_{wh} and low s_{wl} permeability layers, we then have the corresponding relative permeability for the layers as $k_{rw}(s_{wh})$ and $k_{rw}(s_{wl})$.

We can then calculate the effective relative permeability for the whole model. Here we are assuming perfect layering, so that the effective horizontal permeability is just the height weighted average of the permeability.

$$\bar{k}_{rw}(\bar{s}_w) = \frac{k_{rw}(s_{wh})k_h h_h + k_{rw}(s_{wl})k_l h_l}{k_h h_h + k_l h_l} \quad , \tag{12.34}$$

where h_i is the height and k_i is the permeability of the high permeable $i = h$ or low permeable $i = l$ layer. The effective vertical permeability is similarly given by the harmonic mean.

Assume that Using a permeability 1000 mD for the high permeable layer and a permeability of 10 mD for the low permeable layer gives upscaled relative permeability curves as shown in Fig. 12.11.

As the low permeability layer is first filled up with water, there will be small changes to the horizontal relative permeability curves until the low permeability layer is almost completely filled with water. Then, the high permeable layer will start to be filled up, with a correspondingly large change in relative permeability. The effect is opposite for the vertical permeability. This example illustrates the fact that an anisotropic absolute permeability is usually accompanied with an anisotropic relative permeability.

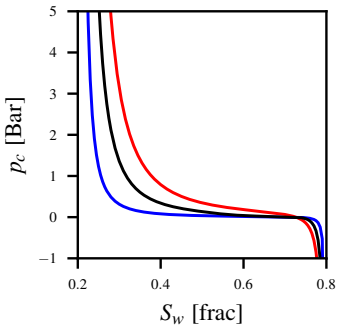


Figure 12.10: The capillary pressure curves for the high (blue) and low (red) permeability layer, and the upscaled capillary pressure curve for the full model in black.

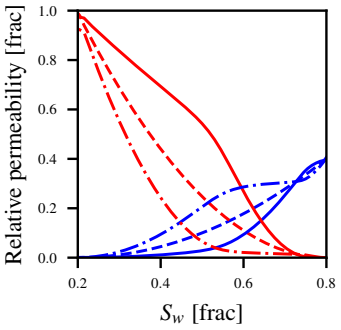


Figure 12.11: The upscaled relative permeability curves for the layered model. Horizontal relative permeability as full drawn lines, and vertical relative permeability as dash-dot lines, Dashed lines are the underlying fine scale relative permeability.

12.4.2 *Viscous dominated upscaling*

Viscous limit upscaling

We will now consider the opposite case of the capillary limit upscaling, namely a viscous dominated upscaling. We know from the previous section that the fractional flow is constant *along streamlines* for high rates (Eq. (12.28)). This is implemented in what is called viscous limit upscaling where the saturation distribution is determined by setting the fractional flow equal *everywhere* similar to how capillary pressure defines saturations in capillary limit upscaling. However, this approach is typically only applicable in cases where the rock-types are essentially randomly distributed, since the fractional flow may vary between streamlines.

Below we will consider a layered model in the viscous limit. The capillary pressure is set to zero, and the gravitational part of Eq. (12.27) is removed by setting the densities equal (OPM-Flow input shown in Fig. 12.12). The relative permeability curves are also set equal in each layer. In this case normal viscous limit upscaling

```
DENSITY
--   Oil       Water     Gas
-- (kg/m3)   (kg/m3)   (kg/m3)
     1025       1025      0.82   /

PVTW
-- REF.PRES.  REF. FVF  COMPRESSIBILITY  REF.VISC.
--         -> VISCOSIBILITY
--  (bar) (m3/m3) (1/bar)    (cP)       (1/bar)
     100   1.00   2.0e-4     1.0        0.0e+0   /

PVDO
--  PRES.   FVF.   VISC.
    50.0    1.01   1.0
   100.0    1.00   1.0
   200.0    0.98   1.0
/
```

Figure 12.12: Equal density of oil and water is obtained in OPM-Flow by setting surface densities and formation factors equal (Reservoir pressure is 100 bar). The shown input also sets compressibility and viscosity equal for the two phases.

would give upscaled relative permeability curves that are identical to the input curves. Contrary to this, we will see below that substantially different upscaled relative permeability curves are needed in order to reproduce fine simulation results in an upscaled model.

To simplify, we will only consider two layers, instead of the 4 in Fig. 12.8. We will have the same permeability distribution as for the capillary limit case, namely 1000 mD for the high permeable layer and a permeability of 10 mD for the low permeable layer. The model consists of 102 cells in x-direction, and 10 cells in z-direction. The cells are 10 m in x-direction, and 1 m in z-direction.

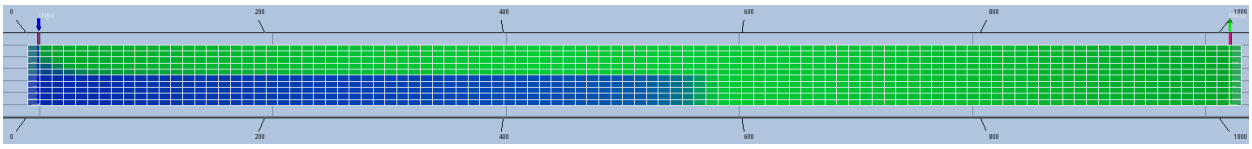


Figure 12.13: The fluid distribution in the two layer model during injection. We see that the water (blue) is intruding into the high-permeable layer, while the low permeable layer is still close to initial oil saturation.

Running the model gives a fluid distribution as shown in Fig. 12.13. We observe that the water is choosing the path of least resistance, displacing the oil in the high permeable layer, while the oil in the low permeable layer is close to initial oil saturation (i.e. residual water saturation).

To upscale the static model, we replace the fine grid with a coarse grid consisting of cells of size 10 m in the z-direction, thus reducing the number of layers from 2 to 1. In the x-direction, the first and last cell will have the same dimensions (to avoid upscaling of the wells), while the other cells will have size of 50 m.

For the relative permeability upscaling, we see that the water saturation in the high permeable layer is almost shifting from residual oil to irreducible water while the water saturation in the low permeable layer stays constant at residual oil. If we use this saturation distribution, and calculate the effective relative permeability from

Eq. (12.34), we obtain an upscaled curve as shown in Fig. 12.14.

Running the coarse scale model with the original relative permeability curves and the upscaled relative permeability curves gives an oil production as shown in Fig. 12.15. We see that the model with relative permeability upscaling has a much better match with the original model than the cooarsened model with only absolute permeability upscaling. Also shown in the figure is the procuction from a fine model with capillary pressure curves from section 12.4.1 and a realistic density difference of 245 kg/m^3. This model is neither in the viscous- nor in the capillary limit, and both capillary forces and gravity contribute to the transport.

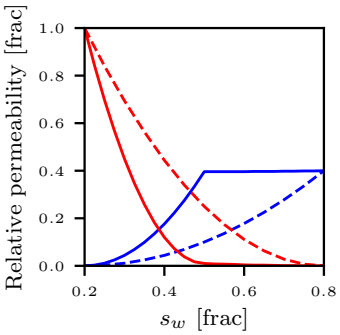


Figure 12.14: The original and upscaled relative permeability curves for the two layer model.

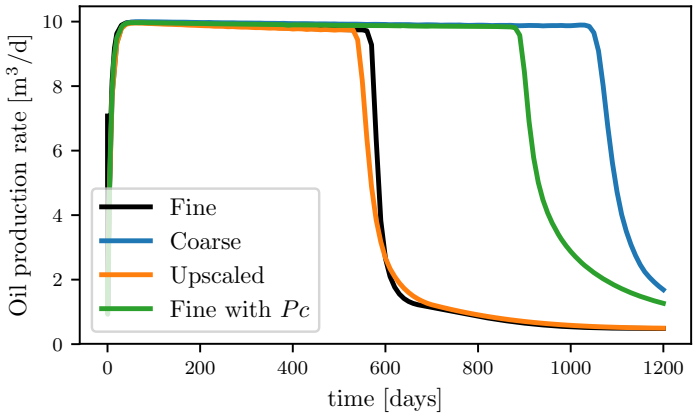


Figure 12.15: The oil production rate of different two-layer simulation models.

Note that the upscaling of the viscous dominated model give upscaled relative permeability curves that moves in the opposite direction of the upscaled relative permeability curves for the capillary limit case. Capillarity distributes water in the vertical direction, thus delaying water break through, while viscous forces segregate water through high permeable layers, thus promoting early water break trough.

Almost identical results to the present viscous limit upscaling of this layered model could have been obtained by using a net-to-gross model, that is defining the low permeable layer as non-reservoir. This would have the additional side effect of reducing the original oil in place to half, and doubling apparent recovery rates[1].

12.5 *Exercises*

Exercise 12.1 Use the fluid volumes to define an upscaled value for saturation. Show that the upscaled saturation is a pore-volume weighted average of the fine scale saturation values.

[1] Actual production (if implemented) is of course not influenced by the modeling choiches, so we leave it to the reader to ponder the consequences (for persons involved and buisness decisions) of selecting viscous dominated upscaling vs. NTG = 0.5, if the actual production would come out as the green-line in Fig. 12.15

13
Model uncertainty and updating

> We demand rigidly defined areas of doubt and uncertainty!
>
> Vroomfondel in ‘ ‘The Hitchhiker’s Guide to the Galaxy” by Douglas Adams

Our simulations will never perfectly predict or reproduce reservoir behavior. The main sources of uncertain predictions are that the true sub-surface remains unknown, that our models are coarse grained representations, and that the modeling algorithms have systematic errors. The total uncertainty should be taken into account when taking reservoir development and reservoir management decisions.

True uncertainty

Even if we were able to integrate all our knowledge about the state of the sub-surface into a perfect model with infinite prediction accuracy, there will always be an infinite number of possible true reservoirs that are consistent with our data. Since the true sub-surface remains to a large extent unknown throughout the production history, our predictions can never, and should never, be more accurate than the spread within this true uncertainty. It is often stated that we will not know what a reservoir will produce until after end of production. Even then we will not know what we could have produced had we selected another production strategy, and how much of the resource remain in the ground.

Many-to-One statistical spread

Current reservoir simulation models can have millions of gridblocks, but they are still very coarse grained representations of often highly heterogenous geology. Theoretically, with sufficient separation of scales, proper upscaling will eliminate the spread in coarse scale behavior. In practice this is never obtained, and for each coarse scale model there will be an infinite number of consistent fine scale models, each with a different response. This many-to-one statistical spread should be addressed both in model prediction and model updating.

Bias

Coarse graining is also a source of bias. Bias are systematic errors due to deficiencies in modeling or modeling algorithms, and can never be totally eliminated. Known biases can be, and should be, corrected for, but in general there will always be an element of unknown bias which must be accounted for.

Based on a simplified model of normal-distributed errors, the total uncertainty can be expressed through standard deviations σ as

Total uncertainty

$$\sigma^2_{\text{total}} = \sigma^2_{\text{true}} + \sigma^2_{\text{mto}} + \sigma^2_{\text{bias}} \quad . \tag{13.1}$$

13.1 Uncertainty modeling workflow

Estimates for the many-to-one uncertainty and bias are to a large extent set at the discretion of the reservoir team based on experience and expertise, while the true uncertainty is typically represented by *an ensemble of reservoir model realizations*. Each ensemble member (realization), Λ_n, represents a possible reservoir, and has its own dynamic behavior. The ensemble is generated such that it can be used to calculate the expected value $\langle \cdot \rangle$ of a reservoir response R, for instance oil production in a specific year, through

Ensemble of realizations

$$\langle R \rangle = \sum_n w_n R(\Lambda_n) \quad , \tag{13.2}$$

where w_n is a weight given to the ensemble member. Note that (13.2) also applies to uncertainty measures such as variance and probability quantiles. In most cases each ensemble member is given equal weight $w_n = \frac{1}{N}$, where N is the ensemble size. This is often called an ensemble of equiprobable members, which actually is misleading since strictly speaking the members have equal weight and not equal probability.

Ensemble of equiprobable members

In reservoir modeling, uncertainty is often introduced on three distinct levels. This is the *uncertainty hierarchy* illustrated in Fig. 13.1. On the top of the hierarchy we have a set of distinct sce-

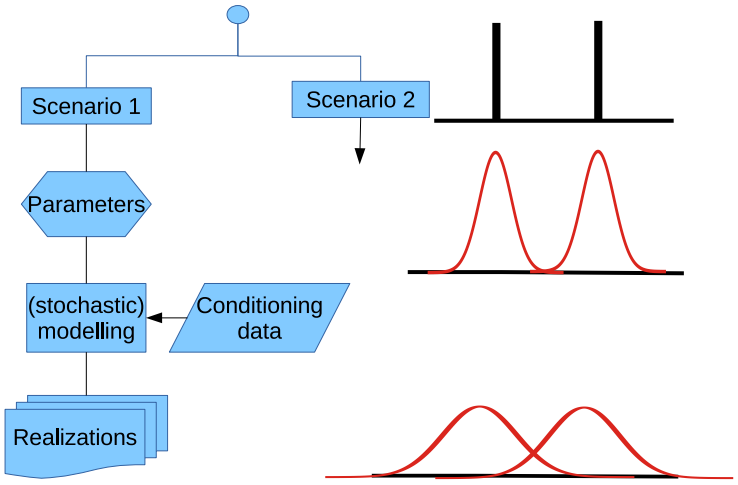


Figure 13.1: The uncertainty hierarchy

narios, each scenario has a set of uncertain model parameters, and for each parameter set a number of realizations is drawn using a stochastic algorithm.

Scenario uncertainty is the spread of outcomes over a set of reasonably plausible, but *qualitatively different*, models. Each scenario is typically a different conceptual model, but any discrete selection

Scenario

may in general define a scenario. Scenario uncertainty is often the main contribution to overall uncertainty, especially early in field life. Designing experiments, and interpreting data, with the goal of discarding a scenario is a crucial part of reservoir characterization. Each scenario is often assigned a different probability. This translates to different weights for ensemble members stemming from different scenarios, or to drawing a different number of ensemble members from each scenario.

Models for each scenario are built in a reservoir modeling tool where a representation of the geology is built using stochastic algorithms which are constrained by a set of *parameters*. Parameters are numbers that characterize model elements of the reservoir, such as zone thicknesses, sand body size and direction, and the variation in these properties. The parameter values are not known a-priori, and static data that constrain the uncertainty are scarce. Often the main constraint is prior knowledge, which is based on outcrop analogies and insight into geological processes. Thus, generating model realizations using fixed parameter values will not result in a correct uncertainty span.

Parameter uncertainty

The algorithms are also constrained by observations such as well-logs and seismic. These constraints are called data-conditioning, and are also typically uncertain as they are not direct observations of reservoir properties. Conditioning data should thus not be treated as hard data.

Conditioning data

The final step is to run a stochastic algorithm that generate realizations with fixed parameters and conditioning data. The total process of varying scenario, parameters, and conditioning data, and generating stochastic realizations, will give a, hopefully statistically correct, ensemble of model realizations. As a rule of thumb: an ensemble size of at least 10 is needed in order to say anything reasonable about expected values, while ensembles with size in the order of 100 are needed for uncertainty evaluation.

Stochastic generation of realizations

13.2 Updating the model

The reservoir model should be consistent with all available data, and as more data is collected the model must be updated in order to accommodate these data. The new data will reduce the number of consistent reservoirs, and in most cases this will also reduce the uncertainty in predictions. Applying measured data in order to update the model is in general called *conditioning to data*. With reference to the uncertainty hierarchy (Fig. 13.1) the data can be applied in at least four ways: Eliminate, or reduce the probability of, scenarios, update model parameter statistics, add as direct conditioning data in geo-statistical algorithms, or post process an existing ensemble of realizations. Complete updating of the whole model is called *big loop* updating, while working with changing realizations and realization-weights only is called *small loop* updating.

Conditioning to data

Small loop vs. big loop updating

Few quantitative methods exist for reducing scenario uncertainty

Reducing scenario uncertainty

in terms of updating scenario probabilities. The most important part of work relating to uncertainty is identifying all of the possible scenarios, and not discarding scenarios too early in the process. In practice scenario uncertainty is updated by identifying data that are inconsistent with certain scenarios, thus eliminating these, and plan measurements that produce such data. Repeated seismic and well test data are well suited to discriminate between structural scenarios, while cored wells and well-logs are important for discriminating conceptual models.

Updating model parameter statistics

Model parameter statistics can be updated based on well-test data, production data, and analysis of new well-logs and cores.

Spatially localized data

New spatially localized data, such as well logs, inverted seismic, and interpreted permeability from well-tests, should be added to the conditioning data in order to make the model locally consistent. Note that in situations where fast model updates are required, such as for geo-steering, these data may also be applied in a "small loop" at the realization level.

13.3 General theory for updating

Bayes theorem

The statistical theory for updating incomplete knowledge from observations is built on *Bayes theorem*, which states that the updated probability (posterior), $p(\lambda|M)$, of a property, λ, after we have made measurements, M, is proportional to the product of the probability assigned to the property before the measurement (prior knowledge), $p_0(\lambda)$, and the probability that a measurement will give the measured values given the property, $p(M|\lambda)$, (likelihood).

$$p(\lambda|M) = \frac{p(M|\lambda)p_0(\lambda)}{p(M)} \propto p(M|\lambda)p_0(\lambda) \tag{13.3}$$

Bayes theorem is applicable to all types of parameter estimation problems, and in the context of reservoir modeling the property λ is a possible reservoir model realization, and the prior and posterior probabilities are typically represented by ensembles. For many data types, the application of Bayes theorem is built into the algorithms that generate the ensemble as conditioning data. A typical example is porosity estimates from well logs. In these cases, model updating means running the modeling software again with more conditioning data. Bayesian updating can in many cases be applied directly on the ensembles by changing weights or creating new ensemble members by mixing properties from members of the prior ensemble. The simplest of these methods is reweighting, where ensemble member weights in (13.2) are scaled by the likelihood

Reweighting

$$w_n \propto w_{n0}\, p(M|\Lambda_n) \quad . \tag{13.4}$$

Simple reweighting is almost never used in practice since it can only be applied in cases where the new measurements carry little information, so that the likelihood is similar for most of the ensemble members. If this is not the case, the number of effective

ensemble members will be significantly reduced. This phenomenon is called ensemble collapse, and can also be a problem in other more sofisticated ensemble updating methods. More applicable are the ensemble Kalman methods, which seek an updated ensemble with members in the space of linear combinations of prior ensemble members.[1] The main limitation of the ensemble Kalman methods is that one has to give mathematical meaning to the concept of "linear combinations of reservoirs". Also a linearity assumption is made when calculating the likelihood from this linear combination, and the likelihood model must be Gaussian[2]. Kalman methods only involve linear algebra, and are very fast. This makes them very attractive in cases where fast updates are required. An example is the updating of local properties around a well path in a well with logging-while-drilling measurements used for geo-steering. The properties are in these cases often also additive, such as porosity, sand fraction, and saturation (see Chap. 12.1), and the measurements near linear in these properties, making Kalman ensemble updating feasible also from a theoretical standpoint.

ensemble Kalman methods

[1] See for instance *"Data Assimilation The Ensemble Kalman Filter"* (Evensen, 2009) for a detailed description.

[2] The Gaussian likelihood model is discussed in more detail later in this chapter. The form assumed by the ensemble Kalman methods is basically (13.12)

13.3.1 Monte Carlo sampling

The practice of generating ensembles representing statistical distributions and using them to calculate properties using (13.2) stems back to the seminal paper "*Equation of State Calculations by Fast Computing Machines*" (Metropolis et al., 1953). In our context, the methods now known as Markov Chain Monte Carlo sampling are random walks through the space of possible reservoirs $\{\Lambda\}$. At each point, the walk has two steps: a proposal step proposing a new reservoir Λ' with probability $p_{\text{prop}}(\Lambda'|\Lambda)$, and an acceptance step where the proposal is accepted with probability $p_{\text{acc}}(\Lambda'|\Lambda)$. In order to get samples from the probability $p(\Lambda)$, the acceptance probability must satisfy the following property[3]

$$p_{\text{acc}}(\Lambda'|\Lambda) = \frac{p(\Lambda')}{p(\Lambda)} \frac{p_{\text{prop}}(\Lambda|\Lambda')}{p_{\text{prop}}(\Lambda'|\Lambda)} \quad . \tag{13.5}$$

[3] The attentive reader will note that the acceptance probability defined here may be larger than one. Proposals where $p_{\text{acc}}(\Lambda'|\Lambda) > 1$ are always accepted.

An important aspect with (13.5) is that the acceptance probability only depend on probability *ratios*. Thus, with respect to Bayes theorem (13.3), we don't need to specify the probability of measurement, $p(M)$, or indeed any other normalizing factors on the likelihood or prior probability. Note that if the proposal function propose a sample from the sought probability, i.e. $p_{\text{prop}}(\Lambda|\Lambda') = p(\Lambda)$, the acceptance probability is 1 as it should. Also, if we insert Bayes theorem (13.3) into (13.5) we get

$$p_{\text{acc}}(\Lambda'|\Lambda) = \frac{p(M|\Lambda')p_0(\Lambda')}{p(M|\Lambda)p_0(\Lambda)} \frac{p_{\text{prop}}(\Lambda|\Lambda')}{p_{\text{prop}}(\Lambda'|\Lambda)} \quad , \tag{13.6}$$

and we see that if the proposal function samples from the prior, i.e. $p_{\text{prop}}(\Lambda|\Lambda') = p_0(\Lambda)$, the acceptance probability is only determined

by the likelihood

$$p_{\text{acc}}(\Lambda'|\Lambda) = \frac{p(M|\Lambda')}{p(M|\Lambda)} \quad . \tag{13.7}$$

The uncertainty in reservoir modeling is specified in a hierarchy where a set of parameters, $\{\lambda\}$, are specified with explicit probability distributions. The ensemble of reservoir realizations thus represent the joint probability

$$p(\{\lambda\}, \Lambda) = p(\{\lambda\})\, p(\Lambda|\{\lambda\}), \tag{13.8}$$

where ensemble members, Λ, are sampled from the conditional distribution $p(\Lambda|\{\lambda\})$ using a mixture of Markov Chain Monte Carlo simulation (typically for the object model) and more direct stochastic methods such as sequential Gaussian simulation (typically for the property model).

13.4 Conditioning to dynamic data

Production data has historically been used to tune the properties of typically a single model realization in a process called *history matching*. The resulting modified realization is called a history matched model and is used as a base case model representing some sort of best guess reservoir. In order to say something about uncertainty, which is required in most situations,[4] it is added to the base case prediction from separate uncertainty estimates. These uncertainty estimates can be based on the spread in predictions in an original, not history-matched, ensemble, or calculated by running a parameter based sensitivity analysis around the base case. None of these approaches take dynamical data into account in an appropriate manner, and are not really statistically sound. A number of improved approaches have been developed, however all of these have their own, often severe, limitations. Methods for conditioning to dynamic data remain an active field of research.

History matching

[4] As an example: Norwegian authorities require p10, p50, and p90 estimates in all resource reporting and production forecasting.

Dynamic data are all measurements that are related to the dynamic behavior of the reservoir. This includes production and injection data, bottom-hole and top-side pressures in wells, saturation and pressure data from observation wells, and repeated seismic or gravimetric data. The model used for observation likelihood is invariably based on the difference between observed, M, and simulated, S, behavior, often assuming a Gaussian model

Dynamic data

$$p(M|\lambda) \propto \exp\left(-\frac{(M - S(\lambda))^2}{\sigma^2}\right) \quad . \tag{13.9}$$

Standard Monte Carlo methods for generating a representative ensemble requires evaluation of the likelihood $S(\lambda)$ at least tens of thousands of times. These methods can thus not be directly applied in practice for real reservoir models since evaluation of $S(\lambda)$ for dynamic data involves running the reservoir simulator, which represents a significant CPU and wall-clock running time, typically in

the order of hours for a reservoir simulation. On the other hand, Ensemble Kalman methods only require the evaluation of the likelihood for each member of the prior ensemble. This makes them very attractive in the context of conditioning to dynamic data, but they are unfortually difficult to apply correctly for many reservoir models. As mentioned above the main limitation of the ensemble Kalman methods is that one has to give mathematical meaning to the concept of "linear combinations of reservoirs". Also a linearity assumption is made when calculating the likelihood of this linear combination. The impact of the latter limitation can however be reduced by applying an iterative method, conceptually similar to Newtons method (see Chap.10.1.3), at he cost of running the reservoir simulator again for each ensemble member at each iteration.

Historically the ensemble Kalman methods for conditioning to dynamic data was introduced in the form of the ensemble Kalman filter, which is a method closely coupled to the simulator. An updated ensemble is created at regular intervals over the historic period, and each member carried forward in time to the next update. The updated ensemble members contain both an update of the static model and the dynamic state, such as saturations and pressure. The filter will in this way ideally correct for any modelling errors in the simulator and give an improved picture, e.g., of remaining oil, than just running an updated static model ensemble through history and into prediction. This ideal picture is however usually not obtained in practice. The resulting saturation patterns are often unphysical. Due to the unlinearity in the reservoir-to-dynamic-response function, the ensemble will also experience ensemble collapse at certain points in history where observations that are improbable given the current ensemble are made. As a consequence of these shortcomings it is now more common to employ the simpler ensemble Kalman smoother methods. In these methods the Kalman update is made once on the static model alone using *all observations*[5].

ensemble Kalman filter

ensemble Kalman smoother

An alternative approach is to replace the reservoir simulator with a fast *surrogate function*. The surrogate can be based on simple functions, such as polynomials, or more elaborate machine learning algorithms. In all cases the surrogate is used to calculate the reservoir response corresponding to measurements for model realizations, in order to evaluate the likelihood in Markov Chain Monte Carlo sampling. When simple functions are used, there is a separate surrogate for each measurement, often called a *response surface*.

Just like the ensemble Kalman methods the surrogate function methods can be applied on the simulation model level. However, while the Kalman methods can work with full property fields with porosity, permeability, etc. in each grid block, the surrogate methods are limited to working with a much coarser description such as average permeability in a zone, or parameters that describe relative permeability. This is similar to traditional history matching workflows. The parameters in the coarse description must be assigned

[5] See Evensen (2018) and references therein for details on the ensemble Kalman smoother methods

Surrogate function

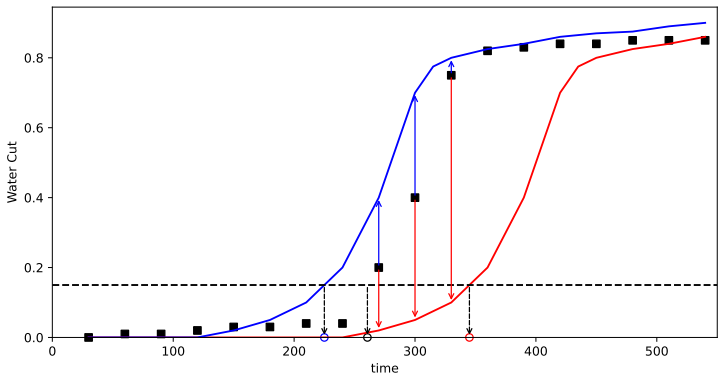


Figure 13.2: Example of the measured water cut in a well (Black squares). Two simulated water cuts in red and blue. We see that the errors are highly correlated between consecutive points as illustrated by the red and blue arrows. An alternative characteristic measurement, break through time, here defined as time when the water cut reaches 0.15, with corresponding simulated values, are shown as circles on the time axis.

prior probability. This should preferably be derived from an analysis of proper prior ensembles, but is in practice often to a large extent based on the engineers gut feeling.

Unlike for Kalman methods the likelihood model used in surrogate methods need not be Gaussian (13.9). This is however usually the case. One reason for this is that it is easier to combine various sources of error in a Gaussian error model since variances are additive. The error model standard deviation σ, represents to which degree a surrogate-predicted value can be expected to match a corresponding measured value. The three main error sources are measurement error σ_{meas}, model error σ_{model}, and surrogate error $\sigma_{\text{surrogate}}$:

$$\sigma^2 = \sigma^2_{\text{meas}} + \sigma^2_{\text{model}} + \sigma^2_{\text{surrogate}} \quad . \tag{13.10}$$

Measurement error

The measurement error represents the actual accuracy of the measurement. Note that many measurements are not direct, but involve an interpretation model. The accuracy of this interpretation model must be accounted for in the measurement error. The model error is mainly the many-to-one statistical spread discussed earlier, but may also include a contribution from an unknown bias. Note that the surrogate model descriptions are much more coarse grained than the actual reservor simulation model realizations. This corresponds to a larger model error in the surrogate based methods than in ensemble Kalman updating that work directly with these realizations. An additional surrogate error correspond to the difference between the actual simulator response and the not perfect surrogate function.

Model error

Surrogate error

We see from (13.3) that we, for a set of independent measurements, can can apply Bayes theorem sequentially and get a product form for the likelihood. For a Gaussian likelihood model this corresponds to

$$\ln p(\{M\}\,|\Lambda) = -\sum_n \frac{(M_n - S_n(\Lambda))^2}{\sigma_n^2} \quad . \tag{13.11}$$

Correlated model errors

For many dynamic measurements the model errors are highly correlated. As an exampe, if the error in simulated water cut in a well is positive one day, it is highly probable that it will be too high

also a month later, as illustrated in Figure 13.2. The form (13.11) is not applicable to such correlated measurements, and its direct use will lead to severe overfitting and underestimated uncertainty. In theory the correlations can be accounted for in the Gaussian model by replacing the variances σ_n^2 with a the covariance matrix C, which gives

$$\ln p(\{M\}\,|\Lambda) = -\sum_{nm}(M_m - S_m(\Lambda))C^{\text{inv}}_{mn}(M_n - S_n(\Lambda)) \quad , \quad (13.12)$$

where C^{inv} is the inverse covariance matrix. There are two main reasons why this formulation is usually not applicable in practice: it is very difficult to get good estimates for the covariance, and even with very good covariance estimates the estimated *inverse* covariance will be very inaccurate. A full multivariate Gaussian is also usually not a good error model. It is better to select a drastically reduced number of measurements that characterize the behavior of the data type, and use the uncorrelated error model as given by Eq. (13.11). For the water cut measurements shown in figure 13.2, we could for example replace the direct measurements with the break through time and the final water cut.

Select measurements that characterize the data type

The surrogate methods are very suitable for big- loop updating. What is meant by big loop here is an update of geo model parameters based on dynamic data. See for instance Slotte and Smorgrav (2008), where among other things fluvial reservoir channel directions are updated, for an early example. In this case the surrogate represent the total geo modelling process followed by reservoir simulation. Note that, since the geo modelling is stochastic, an extra error term, that represent the stochastic spread given a fixed parameter set, must be added to the error in Eq. (13.10) in big loop updating.

Surrogate methods in big loop updating

14
Full field reservoir simulation model

The hammer of the gods
Will drive our ships to new lands
To fight the horde, sing and cry
Valhalla, I am coming

Led Zeppelin - Immigrant Song

In this chapter we will present a full field reservoir simulation model, as it is used in the industry. We will present how to run the model, how to analyze the simulation results, and how to modify the model. Throughout we will use the Norne model, a model of an oil and gas reservoir situated in the Norwegian Sea.

14.1 Norne

Figure 14.1: A picture of the Norne FPSO; Norneskipet. The ship was build in Singapore, and has a storage capacity of 115 thousand ton. The gas flare can be seen at the stern of the ship. (Foto: Equinor.)

Norne is a oil and gas producing field situated of the coast of mid-Norway, north-west of Trondheim. The Norne field is approximately 80km north of the larger Heidrun field, and all the fields in this area are operated from Stjørdal. The Norne field is owned by Petoro (54%), Equinor (former Statoil, 39%) and Vår Energi (7%), and the field is operated by Equinor.

The Norne field is named after the *Norns* from Norse mythology. The Norns are goddesses that control the destiny of both humans and æsirs (i.e. a type of Norse gods, including Odin and Thor).

Norne was discovered in 1992, and brought on stream on 6th November 1997. The field has been developed with a production,

storage and off-loading vessel (FPSO), connected to seven subsea wellhead templates. The FPSO, Norneskipet, is seen in Fig. 14.1. The ship is moored to a rotating turret at the seebead, with risers and umbilicals connected to the same turret. The FPSO has a processing plant, and can store the processed oil. The oil is loaded onto tankers for export, while gas is exported through pipeline to the Åsgard field, and further to the Kårstø terminal in Rogaland for processing, before most is exported to Germany.

The water depth at Norne is 380m, while the reservoir lies at a depth of 2500m. The reservoir consist of Jurassic sandstone of good quality. An outline of the reservoir is shown in Fig. 14.2. The reservoir is divided into the Tilje, Tofte, Ile, Not and Garn formations (where Tilje is the deepest, while Garn is the shallowest part). Oil is mainly found in the Ile and Tofte Formations, and gas in the Not and Garn Formation. The main recovery strategy is water injection. Earlier, produced gas was re-injected to avoid expansion of the gas cap due to pressure depletion. Gas export started in 2001, and gas injection stopped in 2005. From then all gas was exported.

The oil production reached 11MSm^3 in 2001, and has been declining ever since. The original plan was to shut down the field in 2014, but the planed production period has been extended several times. The latest extension was granted in 2019. Gas blowdown in the Not formation started in 2019, and two production wells are planned for 2020. The satellite fields Alve, Urd, Skuld and Marulk are all tied-back to the Norne FPSO.

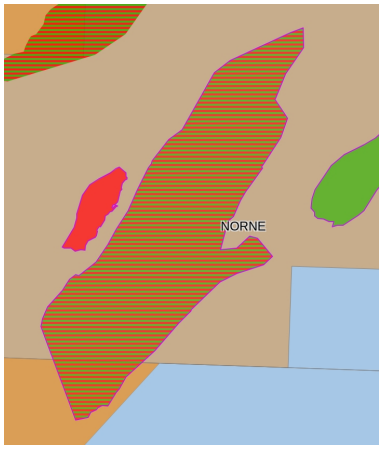


Figure 14.2: A snapshot from the NPD website showing the outline of the Norne field. The green color indicates oil, while the red color indicates gas. Thus the green-red stripes of Norne indicates that this is an oil and gas reservoir.

14.2 Reservoir simulation model

A version of the Norne full field simulation model was released by Equinor through the NTNU program Center for Integrated Operations in the Petroleum Industry. This NTNU program has expired, but the Norne model is still available through the OPM github pages under an open license. The full data set can be found in the Norne subfolder at `https://github.com/OPM/opm-data`. The full OPM data set can be cloned to your computer by the following command:

```
git clone https://github.com/OPM/opm-data.git
```

An alternative to using `git` is to download the files from the webpage.

The Norne model is a reservoir model of the Norne field described above, which has been history matched up to December 2006. The simulation grid has $46 \times 112 \times 22 = 113344$ cells, of which 44927 cells are active. The grid is a faulted corner-point grid, and the permeability field is heterogeneous with directional-dependent permeability. The 22 layers in the vertical direction represent different formations, as shown in Table 14.1.

The Norne model is a black oil model, which contains water, oil, gas, dissolved gas and vaporized oil. The model use end-point scal-

Layer	Formation	Layer	Formation
1	Garn 3	12	Tofte 2.2
2	Garn 2	13	Tofte 2.1.3
3	Garn 1	14	Tofte 2.1.2
4	Not	15	Tofte 2.1.1
5	Ile 2.2	16	Tofte 1.2.2
6	Ile 2.1.3	17	Tofte 1.2.1
7	Ile 2.1.2	18	Tofte 1.1
8	Ile 2.1.1	19	Tilje 4
9	Ile 1.3	20	Tilje 3
10	Ile 1.2	21	Tilje 2
11	Ile 1.1	22	Tilje 1

Table 14.1: Table relating the layers in the Norne model to the different formations.

ing and hysteresis for the saturation functions (i.e. for the relative permeability and capillary pressure curves). Oil is mainly in the Ile and Tofte formations, while gas is mostly in the Garn formation. The Not formation is inactive, thus it represents a barrier to flow upwards.

The model is divided into 16 fluid in place regions, defined using the `FIPNUM` keyword. These fluid in place regions give the intersection of the Garn Ile, Tofte and Tilje formations and different segments of the reservoir. The distribution of the fluid in place regions are shown in Fig. 14.3.

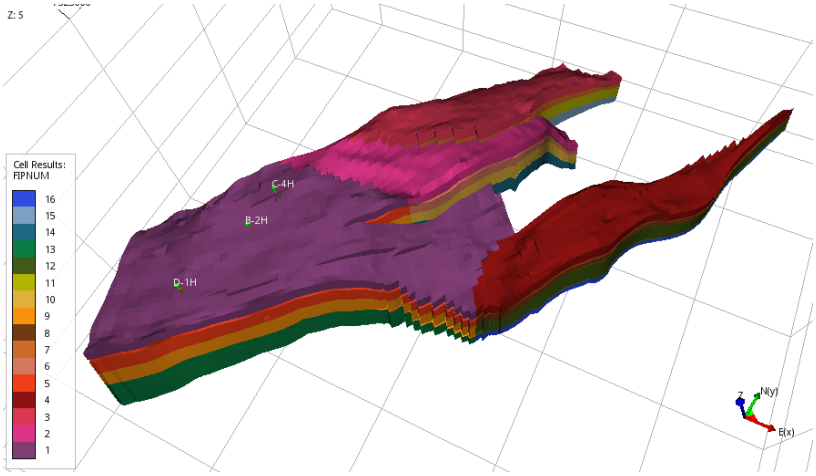


Figure 14.3: The `FIPNUM` keyword in the Norne reservoir model.

The reservoir model has five equilibrium regions where the oil-water contact and gas-oil contact are defined. These regions are defined using the keyword. This data is used to initialize the model; from the given fluid-contacts the saturation distribution and capillary pressure are calculated for all grid cells. Note that the different regions have different height for the fluid contacts; the oil-water contact varies between 2585m and 2692m for the different equilibrium regions. The distribution of the equilibrium regions are shown in Fig. 14.4.

Additionally, the reservoir model has transmissibility multipliers

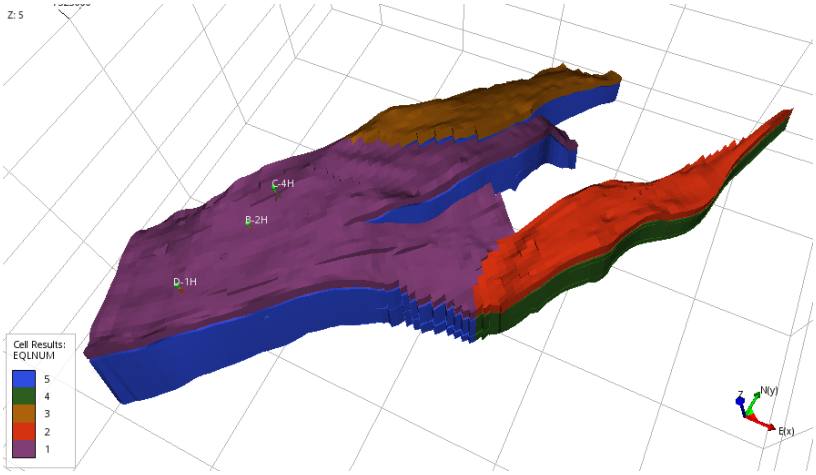


Figure 14.4: The EQLNUM keyword in the Norne reservoir model.

(for faults), pressure-dependent porosity and transmissibility, end-point scaling for relative permeability and capillary pressure and history-matched production well controls that change throughout the simulation schedule. The model is thus quite complex.

There are very few examples of actual full-field reservoir models that have been released, as most companies consider the models as information that is valuable to keep proprietary. For that reason, this Norne model has been used in a wide range of reservoir simulation studies in academia, and is featured in a number of scientific papers.

15
Optimization

Things are going great, and they're only
getting better
I'm doing all right, getting good grades
The future's so bright, I gotta wear shades

Timbuk 3 - The Future's So Bright

In this chapter we will give a brief introduction to optimization routines for reservoir models. The chapter is centered around possibilities in the software FieldOpt, developed at NTNU. For a general introduction on FieldOpt, please confer (Baumann et al., 2020).

15.1 Optimization

We have already touched upon optimization several times in this book. As an example, we used the Python library `optimize.fmin` to find the front saturation for our Buckley-Leverett derivations in Sec. 9.3, and we used Newton's method in our fully-implicit solution to incompressible flow in Sec. 10.1.3. Optimization is a mathematical discipline on finding the best element from a set of available alternatives. For our purpose, it is about finding the set of input variables $\mathbf{x}_0$ that will maximize or minimize a function $f(\mathbf{x})$. Here f is called the objective function.

Objective function

The simplest optimization problems are to find a minimal or maximal value of a real-valued function $f(x) \in \mathbb{R}$ for a scalar $x \in \mathbb{R}$. For the minimization problem, we then want to find the value x_0 such that

The symbol $\forall$ means *for all*.

$$f(x_0) \leq f(x) \quad \forall x \in \mathbb{R} \quad . \tag{15.1}$$

More generally, the input is not necessarily a scalar $x \in \mathbb{R}$, but a set of variables $\mathbf{x} = \{x_0, x_1, \ldots, x_n\}$ where these variables can be of any type, e.g., logical values (Boolean data), integers etc., in addition to scalar values. Examples of logical values are well type (producer versus injector) or well status (open versus shut), an example of integer value is well placement in a indexed grid (the index value $(i, j, k) \in \mathbb{N}^3$ for a perforated part of the well), while scalars could be well rate or bottom hole pressure.

We usually denote the space of possible variables as X, so $\mathbf{x} \in X$. Further, we may only use variables inside a restricted domain, such as a subset of the integers or real numbers, so we only allow for $\mathbf{x} \in S \subset X$. We say that the arguments of the maxima are the points in S that maximize the objective function $f : X \to \mathbb{R}$:

Arguments of the maxima: arg max

The $\wedge$ symbol means *and*.

$$\underset{\mathbf{x} \in S \subset X}{\arg\max} f(\mathbf{x}) = \{\mathbf{x} \mid \mathbf{x} \in S \wedge f(\mathbf{y}) \leq f(\mathbf{x}) \, \forall \, \mathbf{y} \in S\} \quad . \tag{15.2}$$

From this definition, there might exist several elements in $\arg\max f$.

Reservoir simulations main purpose is input to decision-making processes for field development. These decisions can require an optimal drainage strategy (together with uncertainty), thus they push for optimization of the simulation model. The objective function for field development decisions is usually the net present value (V_{NP}) of the drainage scenario for the simulation model. Such net present value calculations can be complex; they might account for cost of water injection, water treatment, CO_2 tax, etc., in addition to the prize of oil which is dependent on an uncertain oil price. The net present value can be calculated as

Net present value

$$V_{NP} = \sum_{\omega=1}^{N_\omega} \sum_{\tau=1}^{N_\tau} \frac{C_\omega(t(\tau)) c_\omega(t(\tau))}{(1 + d_\omega)^{t(\tau)}} \quad , \tag{15.3}$$

where C_ω is the amount of property ω, $t(\tau)$ is the time at time-step τ, c_ω is the earnings or costs of property ω, while d_ω is the *discount rate* of property ω. The properties ω we are summing over could be produced oil, injected water, produced water etc. If ω is the property produced oil, then C_ω will be the amount of produced oil, while c_ω will be the price of oil. A discount rate of about 8% is common, and it is common to use the same discount rate d_r for all properties, thus $d_\omega = d_r$ for all ω.

Discount rate

The Newton's method, as presented in Sec. 10.1.3, use the derivative of the function in each iteration step when searching for the point where the function is zero. The derivative is often used for effective optimization methods. One such method is gradient descent, which is an iterative algorithm for finding a minimum of the objective function. The gradient descent is iteratively given as

Gradient descent

The step size in the gradient descent could change between the iterations. For certain functions there exist conditions on the step size to ensure convergence, e.g., the Wolfe condition.

$$\mathbf{x}^{(m+1)} = \mathbf{x}^{(m)} - \gamma \nabla f(\mathbf{x}^{(m)}) \quad . \tag{15.4}$$

For a single variable, the gradient descent method simplifies to

$$x^{(m+1)} = x^{(m)} - \gamma f'(x^{(m)}) \quad . \tag{15.5}$$

When an analytical expression for the gradient (derivative) of the objective function is available, this is what will be used. When the analytical gradient is not available, one can use a finite difference quotient instead.

However, there are several issues with methods employing the gradient. The first is that they tend to find a local minimum instead

Convergence to a local minimum

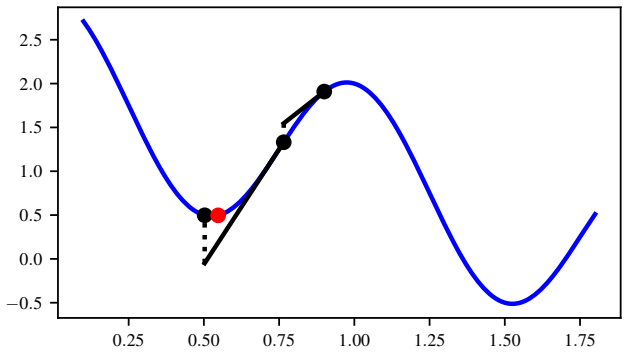


Figure 15.1: Application of the gradient descent method to a function with several minima.

of a global minimum. As an example, look at the function

$$f(x) = \cos(2\pi x) - x + 2 \quad , \tag{15.6}$$

which is plotted as the blue line in Fig. 15.1. Starting with $x^{(0)} = 0.9$ and a step size $\gamma = 0.05$, the three first iterations will be as indicated in the figure. We see that the solution converges to a local minimum. This is a general problem with methods based on the gradient of the objective function.

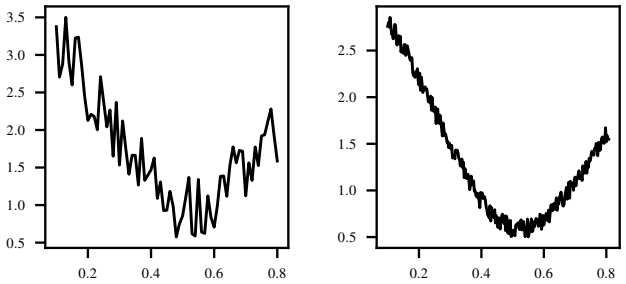


Figure 15.2: Plot of the objective function with different level of smoothness.

Another general problem with gradient based methods is that they require smooth objective functions. Assume we applied the gradient descent method to the leftmost figure in Fig. 15.2. The derivative would be highly erratic, which will impair the gradient descent method (and any other method that is based on the gradient). When the erratic nature of the objective function occurs at a smaller scale, such as in the rightmost figure in Fig. 15.2, the finite difference quotients could work as an approximation for the gradient of a smoothed version of the objective function.

15.2 Derivative-free optimization

When the objective function is non-smooth and computationally costly to compute, then methods that rely on the gradient of the objective function can be problematic, as indicated in the previous section. For such cases it is therefore common to use so-called

derivative-free optimization methods (aka. direct search methods, i.e., methods that do not use the gradient to find the optimal solution. In other words, derivative-free methods do not use the derivatives of the objective function f with respect to the set of input variables $\mathbf{x}$.

For complex reservoir models we do not have analytical derivatives for the objective function with respect to input variables. As an example, assume the objective function is the net present value and the input variable is bottom hole pressure for a producer. Then we do not have an analytical expression for the derivative of the net present value with respect to changes in bottom hole pressure for the producer. Since reservoir models are heavy to run, calculating the finite difference quotient for the net present value with respect to the bottom hole pressure is computationally costly, and therefore challenging for a large number of finite difference quotient evaluations. While changing the operational conditions for a well could be assumed to give fairly smooth response to the net present value, moving the well around in the reservoir for optimization of well placement could be expected to be highly non-smooth. Thus optimization with reservoir simulation models are ill suited for optimization methods that rely on derivatives.

Conversely, with modern day computer clusters, one can run a set of reservoir models in parallel. This favors optimization methods which allows for many objective function evaluations simultaneously. As the reservoir models might have significant different running times, it also favors methods where the objective function can be evaluated at different times.

There exist a large range of derivative-free optimization methods. The most famous might be the Nelder–Mead method, which is one of the methods implemented in FieldOpt. Pattern search (also known as compass search) is a method that shares similarities with Nelder-Mead, and is also implemented in FieldOpt. While Nelder-Mead use $n + 1$ objective function values in each iteration step (where n is the dimension of the problem, i.e., the dimension of $\mathbf{x}$), the pattern search method use $2n + 1$ values. Asynchronous parallel pattern search (APPS) is a version of pattern search that allows for the function evaluations a different times, and is also included in FieldOpt (Hough et al., 2001). Further, the particle swarm optimization (PSO) (Floreano and Mattiussi, 2008; Onwunalu and Durlofsky, 2010), genetic algorithm and a model-based derivative-free trust-region algorithm (DFTR) (Silva et al., 2022) are also included in FieldOpt. We will give a brief intro to the pattern search method in the following.

15.2.1 Pattern search

Pattern search is a simple method for searching for the optimal solution. If the parameter space $\mathbf{x} \in \mathbb{R}^n$ is n-dimensional in the real numbers, then the pattern search method calculates the objective

function in all directions in the n-dimensional space.

Let $\mathcal{D} = \{d_0, d_1, d_2, \ldots, d_{2n}\}$ be a set where $d_0 = 0$ is the zero vector, while d_i for $i > 0$ are the $2n$ directions in the n-dimensional space. Here $\|d_i\| = 1$ for all $i > 0$. Further, let $f\colon \mathbb{R}^n \to \mathbb{R}$ be the objective function, Δ_t a tolerance and γ a step-length contraction factor. Let $\mathbf{x}^{(0)} \in \mathbb{R}^n$ be the starting point, and an initial step-length is given by $\Delta^{(0)}$.

For each iteration m, there exists a $i \in [0, 2n]$ such that $f(\mathbf{x}^{(m)} + d_i\Delta^{(m)})$ is maximal, i.e. $f(\mathbf{x}^{(m)} + d_i\Delta^{(m)}) \geq f(\mathbf{x}^{(m)} + d_j\Delta^{(m)})$ for all $j \neq i$. If $i = 0$, then the largest function value is the current point. We will then stay in the same point, while we reduce the step-length by a factor γ, thus we let $\Delta^{(m+1)} = \gamma\Delta^{(m)}$ and $\mathbf{x}^{(m+1)} = \mathbf{x}^{(m)}$. When $i \neq 0$, then the function is larger at the point in direction d_i. We will then move to this new point with higher function evaluation, while we keep the step-length the same: $\Delta^{(m+1)} = \Delta^{(m)}$ and $\mathbf{x}^{(m+1)} = \mathbf{x}^{(m)} + d_i\Delta^{(m)}$.

This can be implemented in Python as:

```
## Pattern search (Compass search)
# Initial point, and step length
afCentralPoint=(-1.0,1.0)
fStepLength=0.5
# Tolerance, and contraction factor
fTolerance=0.1
fContraction=0.5
# Search directions
aafDirections=np.array
    ↪ ([[0.0,0.0],[-1.0,0.0],[0.0,-1.0],[1.0,0.0],[0.0,1.0]])

while (fStepLength>fTolerance):
  plotCross(afCentralPoint,fStepLength,aafDirections)
  afTestPoints=np.copy(aafDirections)*fStepLength
  afTestPoints[:,0]=afTestPoints[:,0]+afCentralPoint[0]
  afTestPoints[:,1]=afTestPoints[:,1]+afCentralPoint[1]
  iMaxDir=np.argmax(hObjectiveFunction(afTestPoints[:,0],
    ↪ afTestPoints[:,1]))
  if iMaxDir>0:
    print('Move central point',iMaxDir,afCentralPoint,(
    ↪ afTestPoints[iMaxDir,0],afTestPoints[iMaxDir,1]))
    afCentralPoint=(afTestPoints[iMaxDir,0],afTestPoints[iMaxDir
    ↪ ,1])
  else:
    print('Reduce step size',fStepLength,fStepLength*fContraction)
    fStepLength=fStepLength*fContraction
```

For visualization purposes, assume we are in a two-dimensional space, $n = 2$, and consider the function:

$$f(x, y) = -x\|(3 - x)x - 3y + 1\|^{\frac{7}{3}} - \|(3 - y)y - x + 1\|^{\frac{7}{3}} \tag{15.7}$$

In the code snippet above we have started at the point $\mathbf{x}^{(0)} = (-1.0, 1.0)$, and we are using a contraction factor of $\gamma = 0.5$. Running the code gives a development as shown in Fig. 15.3.

We see that in the first four iterations the point $\mathbf{x}$ is moved while the step-length is kept constant, while in the fifth iteration the step-

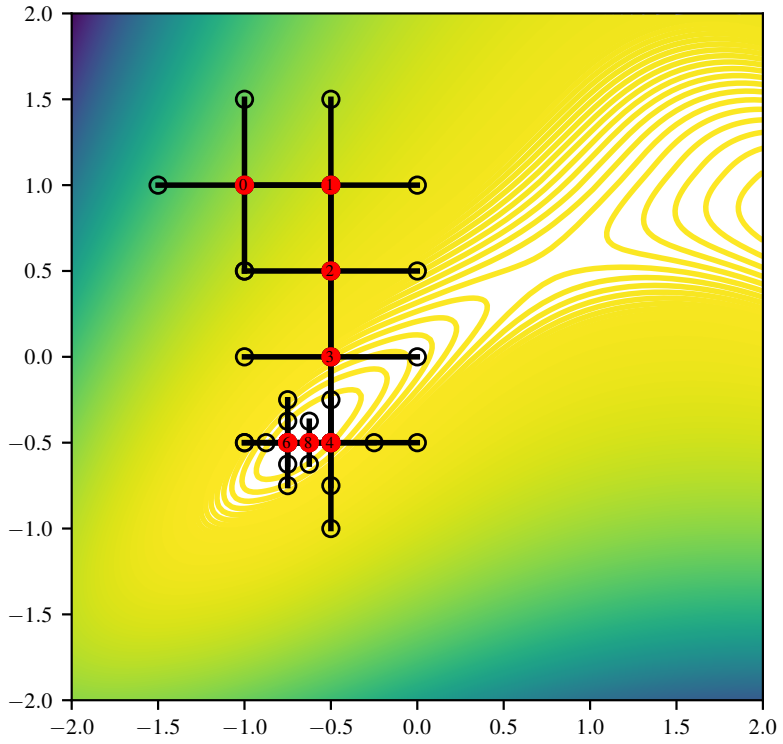


Figure 15.3: Application of the pattern search method with the objective function given by Eq. (15.7). The red circles are the points $\mathbf{x}^{(m)}$, where the numbers represent the iteration step m.

length is reduced by a factor $\gamma = 0.5$. Thus, in Fig. 15.3 there are no number $m = 5$, as this iteration is not moving the point $\mathbf{x}$. At iteration step 6 the point is moved left, while step 7 is a reduction of step-length size again. At step 8 the point moves right. Finally, at step 9 the step-length is reduced to a length smaller than the tolerance, which stops the iterative process.

As reservoir simulations are heavy, it is the computational time of the object function evaluations that is the limiting step for our optimization. However, it is possible to evaluate all the different directions in $\mathcal{D}$ in parallel. This will speed up the optimization significantly. Unfortunately, there might be large differences in simulation time for the different models, some might not even converge. We therefore want an optimization method that can continue without obtaining the function evaluation in all directions. Such a method is asynchronous parallel pattern search (APPS) (Hough et al., 2001). Whenever a lower objective function value is found, the APPS method will start calculating new directions from this new lower function value point. The remaining directions in the previous point are still calculating, and could still yield lower function values which would initiate new trial directions. This way the APPS method can reach the optimal solution faster than the standard pattern search method. It will however run more function evaluations in parallel.

15.2.2 Nelder-Mead

The Nelder-Mead method (aka. downhill simplex method) is a numerical method to find the extrema of a real-valued multi-variable function. It is based on moving a simplex around by several simple operations based on the function value in the corner points of the simplex.

The method was introduced by Nelder and Mead (1965).

A simplex is a generalization of a triangle to arbitrary dimensions. A k-simplex is a polytope of $k+1$ vertices in k-dimensional space. It is a line in 1D, and a triangle in 2D.

Consider the variable space $\mathbb{R}^n$, and a real-valued function $f\colon \mathbb{R}^n \to \mathbb{R}$ that we want to maximize. Further, assume an initial simplex described by the points $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$. The Nelder-Mead method is given by iterating the following steps, where α, γ, ρ and σ are reflection, expansion, contraction and shrink coefficients, respectively:

Common values for the reflection, expansion, contraction and shrink coefficients are $\alpha = 1, \gamma = 2, \rho = 1/2$ and $\sigma = 1/2$.

1. *Order:* Order the test points (the corners of the simplex) according to their function values, i.e., such that $f(\mathbf{x}_0) \geq f(\mathbf{x}_1) \geq f(\mathbf{x}_2) \geq \cdots \geq f(\mathbf{x}_n)$. Check whether to stop the algorithm by a convergence criteria, e.g., if the standard deviation of the value $f(\mathbf{x}_i)$ is smaller than a tolerance value.

2. *Centroid:* Calculate the centroid $\mathbf{x}_m$ of all points except $\mathbf{x}_n$:

$$\mathbf{x}_m = \frac{1}{n} \sum_{i=0}^{n-1} \mathbf{x}_i \tag{15.8}$$

3. *Reflection:* Find the *reflection point* $\mathbf{x}_r = \mathbf{x}_m + \alpha\,(\mathbf{x}_m - \mathbf{x}_n)$, where $\alpha > 0$ is the reflection coefficient. If $f(\mathbf{x}_0) \geq f(\mathbf{x}_r) > f(\mathbf{x}_{n-1})$, i.e., the function value at the reflection point is larger than the second to smallest, but not larger than the best point, then we replace the worst point with the reflection point, $\mathbf{x}_n = \mathbf{x}_r$. We have thus obtained a simplex with better points, and return back to the ordering in step 1. Else, if the reflection point is also better than the best point, then we should consider moving even further in the direction of the reflection point, so we continue with the next point; expansion.

4. *Expansion:* If $f(\mathbf{x}_r) > f(\mathbf{x}_0)$, i.e., the reflection point is the best point, then we want to check if we should move even further in the direction of the reflection point. For this end we do the following:

   - Find the *expansion point* $\mathbf{x}_e = \mathbf{x}_m + \gamma\,(\mathbf{x}_r - \mathbf{x}_m)$, where $\gamma > 1$ is the expansion coefficient.
   - If $f(\mathbf{x}_e) > f(\mathbf{x}_r)$, i.e., the expansion point is better than the reflection point, then we replace the worst point with the expansion point, $\mathbf{x}_n = \mathbf{x}_e$, and return to the ordering in step 1.
   - Else if $f(\mathbf{x}_e) \leq f(\mathbf{x}_r)$, i.e., the reflection point is better than the expansion point, then we replace the worst point with the reflection point, $\mathbf{x}_n = \mathbf{x}_r$, and go to step 1.

5. *Contraction:* If we arrive here, then we have $f(\mathbf{x}_r) \leq f(\mathbf{x}_{n-1})$, thus the reflection point is smaller than the second to smallest point. If $f(\mathbf{x}_r) > f(\mathbf{x}_n)$, i.e., the reflection point is better than the worst point, then:

   - Compute the contraction point on the outside as $\mathbf{x}_c = \mathbf{x}_m + \rho(\mathbf{x}_r - \mathbf{x}_m)$, where the contraction coefficient is $0 < \rho \leq 0.5$
   - If $f(\mathbf{x}_c) > f(\mathbf{x}_r)$, i.e., the contraction point is better than the reflection point, then let $\mathbf{x}_n = \mathbf{x}_c$, and go to step 1.
   - Else, if $f(\mathbf{x}_c) \leq f(\mathbf{x}_r)$, go to step 6.

   If $f(\mathbf{x}_r) \leq f(\mathbf{x}_n)$, i.e., the reflection point is worse than the worst point:

   - Compute the contraction point on the inside as $\mathbf{x}_c = \mathbf{x}_m + \rho(\mathbf{x}_n - \mathbf{x}_m)$.
   - If $f(\mathbf{x}_c) > f(\mathbf{x}_n)$, i.e., the contraction point is better than the worst point in the current simplex, then let $\mathbf{x}_n = \mathbf{x}_c$, and go to step 1.
   - Else, if $f(\mathbf{x}_c) \leq f(\mathbf{x}_n)$, go to step 6.

6. *Shrink:* In this case we have not been able to find any better point through all the steps above. This indicates that the optimal point is close to the current best point. We typically arrive at this shrink part of the algorithm at late iterations, when we are getting closer to convergence. As the current best point is probably close to the optimal point, we want to shrink the simplex towards the current best point, i.e., towards the point $\mathbf{x}_0$. We do this by moving all other points than the best point a factor σ towards the best point: $\mathbf{x}_i = \mathbf{x}_0 + \sigma(\mathbf{x}_i - \mathbf{x}_0)$ for all $i \in [1, n]$, where $\sigma < 1$ is the shrink coefficient. After shrinking towards the best point we then go to step 1.

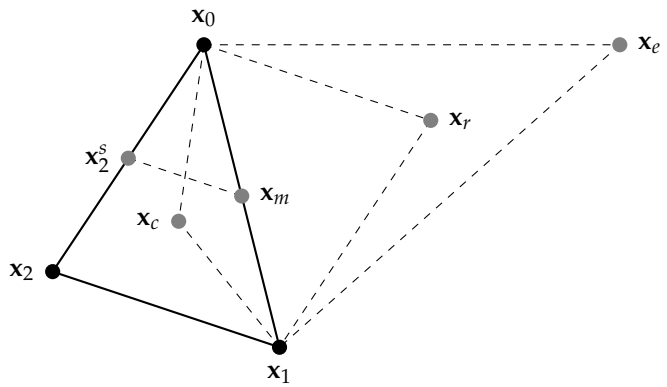


Figure 15.4: A sketch of the Nelder-Mead algorithm.

This can be implemented in Python as:

```
#Coefficients
fAlpha=1
fGamma=2
fRho=0.5
```

```
fSigma=0.5

def updateFuncVal(aaSimplex):
  for ii in range(0,3):
    aaSimplex[ii,2]=hObjectiveFunction(aaSimplex[ii,0],aaSimplex[
    ↪ ii,1])
  return aaSimplex

def updateFuncValSingle(aPoint):
  aPoint[2]=hObjectiveFunction(aPoint[0],aPoint[1])
  return aPoint

def sortPoints(aaSimplex):
  iChange=1
  while iChange>0:
    iChange=0
    if aaSimplex[0,2]<aaSimplex[1,2]:
      aPointTemp=np.copy(aaSimplex[0,:])
      aaSimplex[0,:]=np.copy(aaSimplex[1,:])
      aaSimplex[1,:]=np.copy(aPointTemp)
      iChange+=1
    if aaSimplex[1,2]<aaSimplex[2,2]:
      aPointTemp=np.copy(aaSimplex[1,:])
      aaSimplex[1,:]=np.copy(aaSimplex[2,:])
      aaSimplex[2,:]=np.copy(aPointTemp)
      iChange+=1
  return aaSimplex

def hNedlerMead(aaSimplex):
  #Calculate centroid
  aCentroid=np.array([np.sum(aaSimplex[:-1,0])/2.0,np.sum(
    ↪ aaSimplex[:-1,1])/2.0,0.0])
  aCentroid=updateFuncValSingle(aCentroid)
  #Find reflection point
  aReflection=aCentroid+fAlpha*(aCentroid-aaSimplex[-1,:])
  aReflection=updateFuncValSingle(aReflection)
  if aReflection[2]>aaSimplex[1,2]:
    if aReflection[2]<aaSimplex[0,2]:
      #Replace the worst point with the reflection point
      aaSimplex[2,:]=np.copy(aReflection)
    else:
      #Find expansion point
      aExpansion=aReflection+fGamma*(aReflection-aCentroid)
      aExpansion=updateFuncValSingle(aExpansion)
      if aExpansion[2] > aReflection[2]:
        aaSimplex[2,:]=np.copy(aExpansion)
      else:
        aaSimplex[2,:]=np.copy(aReflection)
  else:
    #Calculate contraction point inside or outside
    if aReflection[2]>aaSimplex[2,2]:
      aContraction=aCentroid+fRho*(aReflection-aCentroid)
      aContraction=updateFuncValSingle(aContraction)
    else:
      aContraction=aCentroid+fRho*(aaSimplex[2,:]-aCentroid)
      aContraction=updateFuncValSingle(aContraction)
    if (aReflection[2]>aaSimplex[2,2] and aContraction[2]>
```

```
    ↪ aReflection[2]) or (aReflection[2]<=aaSimplex[2,2] and
    ↪ aContraction[2]>aaSimplex[2,2]):
      aaSimplex[2,:]=np.copy(aContraction)
    #Shrink
    else:
      for jj in range(1,3):
        aaSimplex[jj,:]=aaSimplex[0,:]+fSigma*(aaSimplex[jj,:]-
    ↪ aaSimplex[0,:])
  aaSimplex=updateFuncVal(aaSimplex)
  return aaSimplex

aaSimplex=np.zeros((3,3))
aaSimplex[0,:-1]=[0.0,0.0]
aaSimplex[1,:-1]=[1.2,0.0]
aaSimplex[2,:-1]=[0.0,0.8]

aaSimplex=updateFuncVal(aaSimplex)
aaSimplex=sortPoints(aaSimplex)

fEps=1E-2
iMaxIt=20
fMaxError=np.sqrt((aaSimplex[0,2]-aaSimplex[2,2])**2)
fError=fMaxError
plotSimplex(aaSimplex)

ii=0
while ii<iMaxIt and fError>fEps:
  ii+=1
  aaSimplex=hNedlerMead(aaSimplex)
  sortPoints(aaSimplex)
  fError=np.sqrt((aaSimplex[0,2]-aaSimplex[2,2])**2)
  print('Iteration␣',ii,'␣Error:␣',fError)
  plotSimplex(aaSimplex)
```

For visualization purposes, assume we are in a two-dimensional space, $n = 2$, and consider the same function as was used in the pattern search example, Eq. (15.7)

In the code snippet above we have started with the simplex given by the points $(0, 0)$, $(1.2, 0)$ and $(0, 0.8)$, and we are using the standard coefficients. Running the code gives a development as shown in Fig. 15.5.

15.2.3 *Particle swarm optimization*

The particle swarm optimization is an evolutionary algorithm for optimization inspired by biological systems, e.g., a flock of bird or a shoal of fish searching for food or avoiding predators. When searching for food (or any other collective task), the swarm is sharing information and cooperating in their search. The algorithm was first developed to simulate social behavior (Eberhart and Kennedy, 1995), but is today applied to a wide range of applications.

The algorithm optimize a problem by iteratively testing out new possible positions, where new test positions are determined by using a combination of local and global information. The *swarm*

The version of PSO implemented in FieldOpt is inspired by Floreano and Mattiussi (2008). PSO in FieldOpt has been applied for well placement optimization in (Floreano and Mattiussi, 2008).

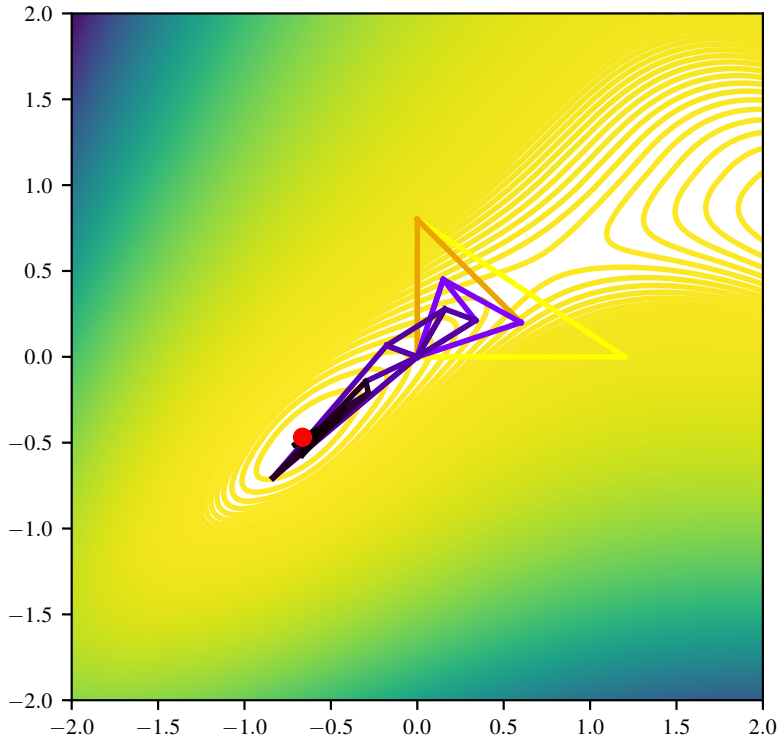


Figure 15.5: Application of the Nelder-Mead algorithm to the objective function given by Eq. (15.7). The triangles are given darker color when the algorithm progress, while the red circle indicates the optimal point after convergence.

consists of the set of candidate solutions, often called particles, and these particles are moved around in the search-space hunting for the optimal solution. The movement between each iteration is individual for each particle, where the speed and direction is determined by the current speed (inertia), the best position encountered by the given particle (memory), and the best position encountered by any of the particles in the swarm (collaboration).

As before, let f be the objective function that we want to maximize, and let $\mathbf{x}_i^k$ be particle $i \in [1, N]$ at iteration k, where N is the number of particles in the swarm. For each particle i and each iteration t, we find the best historic position for this particle as

$$\mathbf{p}_i^t = \{\mathbf{x}_i^k \mid f(\mathbf{x}_i^k) \geq f(\mathbf{x}_i^l) \, \forall \, l \in [1, t]\} \quad . \tag{15.9}$$

The best position of all particles, i.e., the global best position, is then found as

$$\mathbf{g}^t = \{\mathbf{p}_i^t \mid f(\mathbf{p}_i^t) \geq f(\mathbf{p}_j^t) \, \forall \, j \in [1, N]\} \quad . \tag{15.10}$$

All the particles are initialized with a velocity $\mathbf{v}_i^0$ (possibly a zero velocity). This velocity is updated in each iteration to guide the particles towards the *best* position, where by *best* we mean a combination of the particle p_i^t and global g^t best positions. The updated velocity at each iteration is then given by the equation

$$\mathbf{v}_i^{t+1} = \mathbf{v}_i^t + c_1(p_i^t - \mathbf{x}_i^t) + c_2(g^t - \mathbf{x}_i^t) \quad , \tag{15.11}$$

where c_1 and c_2 are factors weighting the relative importance of the particle and global best position, typically called learning factors

(also known as acceleration factors). It is also common to add some randomness by multiplying the learning factors by random numbers, where the random numbers are updated for each iteration. After obtaining the velocity vector, we move the particle as follows:

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1} \quad . \tag{15.12}$$

In effect, the particle then moves towards a combination of the best position for the particle and the best position for the swarm. This mix of a particle and global best position has the benefit that the swarm of particles explore the search space before converging to the global best position. It is possible to tune this process by changing the learning factors; if c_2 is increased relative to c_1 the particles will converge towards one position earlier, thus speed up the convergence, on the expense of a less throughout exploration of the search space.

16
Good advice from Khalid Aziz

Khalid Aziz, an undisputed Nestor in the field of of reservoir simulation, has given the following good advise (Aziz, 1989).

Ten golden rules of reservoir simulation

Khalid Aziz, SPE
Stanford University

With the proliferation of reservoir simulators and extensive use of this powerful technology by the industry, there is increasing danger that inexperienced users will misuse sophisticated models available to them. To minimize this danger, one should keep certain basic rules in mind. The following rules reflect my own experience and what I have gained by reading papers and through discussions with colleagues in industry and academia.

1. **Understand Your Problem and Define Your Objectives.** Before you do any simulation, understand the geological characteristics of your reservoir, the fluids it contains, and its dynamic behavior. Also clearly state the objectives of your study on paper before you start. Ask yourself if the objectives are realistic. These considerations will help you choose the most appropriate model for your study.
2. **Keep it Simple.** Start and end with the simplest model that is consistent with the nature of the reservoir, the objective of your study, and the availability of data. Classic reservoir engineering, simple analytical models, or single-block simulations are often all you need. At other times, the most sophisticated model available to you may not serve your needs. Understand model limitations and capabilities.
3. **Understand Interaction Between Different Parts.** Remember that a reservoir is not an isolated entity. It may connect with an aquifer and, through it, even to other reservoirs. Furthermore, reservoirs are connected through wells to the surface facilities. The isolation of different components of this system for separate study often may lead to inappropriate results by neglecting interaction between different parts of the system. However, when appropriate, don't be afraid to break a big problem into its smaller components. This can lead not only to substantial savings but to greater understanding of the mechanisms involved.
4. **Don't Assume Bigger Is Always Better.** Always question the size of a study that is limited by the computer resources or the budget. Simulation engineers often believe that no computer is big enough for what they want to do and tend simply to increase the size of their models to fit the computer. More blocks and components do not automatically translate into greater accuracy and reliability. In fact, in some situations the reverse is true. Insist on seeing appropriate justification for the number of blocks used in a given study.
5. **Know Your Limitations and Trust Your Judgment.** Remember that simulation is not an exact science. All models are based on assumptions and provide only an approximate answer to the real problem. Hence, a good understanding of both the problem and the model is essential for success. Numerical approximations may introduce "pseudophysical" phenomena like numerical dispersion. Use and trust your judgment, especially if it is based on

your analysis of field or laboratory observations. Be careful to check your input and output. Do simple material-balance calculations to check simulation results. Pay particular attention to such things as negative compressibilities and permeabilities.

6. **Be Reasonable in Your Expectations.** Don't try to get from the simulator what it is incapable of producing. Often the most you can get from a study is some guidance on the relative merits of choices available to you. At other times you have the right to demand a lot more. But remember that if you exclude a mechanism during model development, you cannot study its effect with that model.
7. **Question Data Adjustment for History Matching.** Always question data adjustments during history matching. Remember that this process does not have a unique solution. The most reasonable solution will result from paying close attention to physical and geological reasonableness. A "good" history match with inappropriate adjustments to the data will lead to poor predictions. Don't be lulled into false security by a "good" or "close" match.
8. **Don't Smooth Extremes.** Pay attention to extremes in permeability (barriers and channels). Be careful in the process of averaging to avoid losing essential information when averaging the extremes. Never average out extremes.
9. **Pay Attention to the Measurement and Use Scales.** Measured values at the core scale may not apply directly at the larger block scale, but measurements do influence values at other scales. Remember that averaging may change the nature of the variables you average. For example, permeability may be a scalar at some small scale, and a tensor at a larger scale. Even the meaning of capillary pressure and relative permeability can be different at different scales. Also, the dispersive terms in our equations are a result of the process of averaging.
10. **Don't Skimp on Necessary Laboratory Work.** Models do not replace good laboratory experiments that are designed to gain an understanding of the nature of the process being modeled or to measure essential parameters of the equations being solved by your simulator. Plan your laboratory work with the end use of this information in mind. Learn how to scale data.

I would like to thank Aziz Odeh and Roland Horne for their comments on these rules.

17
Mathematical notes

The notation that are used in these lecture notes may be unfamiliar to some. The coordinate free notation that is used is more common in physics texts than in engineering books. Darcys law is for instance written as

$$q = -\frac{1}{\mu} K \cdot \nabla p$$

in coordinate free notation, instead of

$$q_i = -\frac{1}{\mu} \sum_{j=1}^{3} K_{ij} \frac{\partial}{\partial x_j} p$$

in the normal engineering type notation.

17.1 Scalars, vectors, and tensors

The fundamental objects in any continium theory, including fluid flow in porous media, are scalar-fields, vector-fields, and tensor-fields. This text folllows the common lazy tradition of using the terms scalar, vector, and tensor for these fields.

All readers should be familiar with the concepts of scalars and vectors, while tensors may be a less familiar object. In the context of the present text, only tensors of order 2 is encountered[1], and the most prominent is the permeability, K. Permeability is a linear operator that operates on a gradient (derivative of pressure) to produce a vector (volumetric flux). Such operators are called (2,0) tensors, or more simply tensors of order 2. The tensor concept can be viewed as a generalization of a vector, and taking the so called tensor product of two vectors create a tensor of order 2 (see table 17.1).

[1] The term tensor is usually not used for scalars and vectors, but scalars can be viewed as tensors of order 0 and vectors as tensors of order 1.

No special notation is used in order to distiguish between scalars, vecrtors, and tensors. However, lower case letters are typically used for scalars and vectors, and upper case letters for tensors.

Scalars, vectors and tensors can be multipled, either using the dot product or the tensor product as shown in table 17.1.

Coordinate free notation	Index notation
$v = au$	$v_i = au_i$
$a = u \cdot v$	$a = \sum_i u_i v_i$
$v = K \cdot u$	$v_i = \sum_i K_{ij} u_j$
$v = u \cdot K$	$v_j = \sum_i u_i K_{ij}$
$K = L \cdot M$	$K_{ij} = \sum_n L_{in} M_{nj}$
$K = uv$	$K_{ij} = u_i v_j$

Table 17.1: Multiplication of tensors. a is a scalar, u and v are vectors and K, L, and M are tensors.

17.2 Spatial derivatives and the gradient operator

The operator for spatial derivatives is the nabla, or gradient, operator, ∇. In terms of notation, ∇ behaves like a vector, but it must be remembered that ∇ operates on (takes the derivative of) the expression to the right. Some examples of expressions involving the nabla operator can be found in table 17.2. The shorthand notation ∇^2 is used for the second derivative operator $\nabla \cdot \nabla$ (the Laplace operator). The derivation takes presedence over multiplication. Paranthesis are used to group.

Coordinate free notation	Index notation
∇	$\frac{\partial}{\partial x_i}$
$\nabla^2 = \nabla \cdot \nabla$	$\frac{\partial^2}{\partial x_i^2}$
$\nabla\nabla$	$\frac{\partial^2}{\partial x_i \partial x_j}$
$v = \nabla a$	$v_i = \frac{\partial}{\partial x_i} a$
$a = \nabla \cdot v$	$a = \sum_i \frac{\partial}{\partial x_i} a_i$
$K = \nabla\nabla a$	$k_{ij} = \frac{\partial^2}{\partial x_i \partial x_j} a$
$b = \nabla^2 a$	$b = \frac{\partial^2}{\partial x_i^2} a$
$c = \nabla a \cdot K \cdot \nabla b$	$c = \sum_{ij} \frac{\partial a}{\partial x_i} K_{ij} \frac{\partial b}{\partial x_j}$
$c = \nabla \cdot (aK \cdot \nabla b)$	$c = \sum_{ij} \frac{\partial}{\partial x_i} \left(aK_{ij} \frac{\partial b}{\partial x_j} \right)$

Table 17.2: Examples of expressions involving the ∇ operator. a, b, and c are a scalars, u and v are vectors and K is a tensor.

We see in particular that the divergence of a vector field $q = (q_x, q_y, q_z)$ is a scalar given by the dot product $\nabla \cdot q = \frac{\partial q_x}{\partial x} + \frac{\partial q_y}{\partial y} + \frac{\partial q_z}{\partial z}$, that the gradient of a scalar field p is a vector field $\nabla p = \left(\frac{\partial p}{\partial x}, \frac{\partial p}{\partial y}, \frac{\partial p}{\partial z} \right)$, and that the Laplace operator ∇^2 sends a scalar to a scalar.

17.3 The Gauss theorem, and the continuity equation

The Gauss divergence theorem:

Gauss divergence theorem.

$$\int \nabla \cdot v \, dV = \int v \cdot dS \quad . \tag{17.1}$$

v is a vector, the left integral is over a volume, and the right integral is over the enclosing surface. Note that dS is an outward pointing *vector* normal to the surface element.

The continuity equation for some entity involves three quantities:

- ρ: Density, that is the amount per volume.

- j: Flux, that is the amount that flows per area and time (a vector).
- σ: The amount that is created per volume and time.

For any given volume we then have

Continuity equation on integral form.

$$\frac{\partial}{\partial t}\int \rho \, dV = \int \sigma \, dV - \int j \cdot dS \tag{17.2}$$

This is the continuity equation on integral form. Numerical simulation often employ control volume discretisation, and eq. 17.2 is then applied to the each control volume or grid-block.

If we apply the Gauss theorem to the surface integral in eq. 17.2 we get

$$\frac{\partial}{\partial t}\int \rho \, dV = \int \sigma \, dV - \int \nabla \cdot j \, dV \quad , \tag{17.3}$$

and if we let the volume be an infinitely small differential element, eq. 17.3 gives

Continuity equation on differential form.

$$\frac{\partial}{\partial t}\rho + \nabla \cdot j = \sigma \quad , \tag{17.4}$$

which is the continuity equation on differential form.

17.4 Localization theorem

The localization theorem states that for a continuous real-valued function $F(x)\colon \mathcal{X} \to \mathbb{R}$ defined on an open connected set $\mathcal{X}$ (e.g. $\mathcal{X} \subset \mathbb{R}^3$ is an open connected subspace of $\mathbb{R}^3$), then

Localization theorem

$$\int_D F(x)dx = 0 \quad \forall D \subset \mathcal{X} \Rightarrow F(x) = 0 \quad \forall x \in \mathcal{X} \quad . \tag{17.5}$$

In the setting of reservoir simulation, then $\mathcal{X}$ is the reservoir volume, and the localization theorem can be used to obtain the equation for conservation of mass from a reformulation of Eq. (3.34) as:

$$\int_V \nabla \cdot (\rho q) + \frac{\partial}{\partial t}(\phi\rho)dV = 0 \quad \forall V \subset \mathcal{X} \quad , \tag{17.6}$$

where V is an arbitrary control volume inside the reservoir $V \subset \mathcal{X} \subset \mathbb{R}^3$.

Note that in physics it is common to start with the conservation of mass of a moving fluid element, and then use Reynolds transport theorem to obtain the equation above for any control volume, before using the localization theorem to obtain the local form of the conservation of mass equation.

17.5 Big-$\mathcal{O}$

When comparing approximations and the rate of convergence of numerical methods, we will use the big-oh notation, indicated by the symbol $\mathcal{O}$. Let f and g be two functions of x, then we say that

$$f(\delta) = \mathcal{O}(g(\delta)) \text{ as } \delta \to 0 \quad , \tag{17.7}$$

if there exists a constants C such that

$$\left|\frac{f(\delta)}{g(\delta)}\right| < C \quad \forall \delta < \epsilon \quad , \tag{17.8}$$

for a sufficiently small value ϵ. Rearranging, this gives

$$|f(\delta)| < C|g(\delta)| \quad \forall \delta < \epsilon \quad , \tag{17.9}$$

thus $f(\delta)$ goes to zero at least as fast as the function $g(\delta)$ when δ goes to zero.

As an example, we see that $\delta^n = \mathcal{O}(\delta^n)$. Since $|\delta^n/\delta^n| = 1$, let $C > 1$, then for any ϵ we have

$$\left|\frac{\delta^n}{\delta^n}\right| < C \quad \forall \delta < \epsilon \quad . \tag{17.10}$$

We say that an approximation is n-order accurate if the error $E(\delta^n) = \mathcal{O}(\delta^n)$, where δ is the grid step size.

Nomenclature

η	(hydraulic) diffusivity, $\eta = \frac{k}{\mu\phi c_t}$, SI derived unit: m^2/s
λ	mobility, $\lambda = k_r\rho/\mu$
μ	viscosity, SI derived unit: $\mathrm{Pa\,s} = \mathrm{kg}/(\mathrm{m\,s})$
Ω	pore space, the subset of a volume V consisting of the void space in the porous medium, SI derived unit: m^3
Φ	force potential, SI derived unit $\mathrm{m}^2/\mathrm{s}^2$
ϕ	porosity, $\phi = V_c/V_t$, dimensionless
ρ	fluid density, SI derived unit: kg/m^3
ρ_w	water density, SI derived unit: kg/m 3
$\vec{n}$	normal vector
A	surface, e.g. surface area of volume element, $\delta V = A$, SI derived unit: m^2
B	formation volume factor, dimensionless
B_g	gas formation volume factor, $B_g = V_{gr}/V_{gs}$, dimensionless
B_o	oil formation volume factor, $B_o = V_{or}/V_{os}$, dimensionless
c	compressibility, SI derived unit: 1/Pa
c_ϕ	formation compressibility, SI derived unit: 1/Pa
c_f	fluid compressibility
c_t	total compressibility
c_w	water compressibility, SI derived unit: 1/Pa
D	diffusion coefficient, SI derived unit: m^2/s
f_w	fractional flow of the wetting phase (water)
h	pressure head, SI derived unit: m
J	mass rate, SI derived unit: kg/s
j	mass flux, SI derived unit: $\mathrm{kg}/(\mathrm{m}^2\,\mathrm{s})$
k	permeability, SI derived unit: m^2
m	mass, SI derived unit: kg
P	numerical approximation of the pressure
p	pressure, SI derived unit: Pa=$\mathrm{kg}/(\mathrm{ms}^2)$

p_a atmospheric pressure, 1.01×10^6 Pa

p_b (well) block pressure, SI derived unit: Pa

p_c capillary pressure, SI derived unit: Pa

p_c characteristic pressure, $p_c = \frac{Q\mu}{2\pi kh}$, SI derived unit: Pa

P_i^t numerical approximation of the pressure at spatial point i and time step t

p_r reference pressure, SI derived unit: Pa

p_w well pressure, SI derived unit: Pa

p_{BHP} bottom hole pressure, SI derived unit: Pa

Q volume rate, SI derived unit: m^3/s

q volumetric fluid flux (Darcy velocity), SI derived unit: m/s

q_t total Darcy velocity

Q_w well flow rate, SI derived unit: m^3/s

q_w wetting phase (water) Darcy velocity

R residual (in the Newton's method)

r_e equivalent well-block radius, SI derived unit: m

R_s solution-gas-oil ratio, $R_s = V_{gs}/V_{os}$, dimensionless

R_v solution-oil-gas ratio, $R_v = V_{os}/V_{gs}$, dimensionless

r_w well radius, SI derived unit: m

s_o oil saturation

s_p saturation of phase p

s_w water saturation

T_{wb} well connection transmissibility factor, $T_{wb} = \frac{kh}{\ln\left(\frac{r_e}{r_w}\right)}$, SI derived unit: m^3

u interstitial fluid velocity, SI derived unit: m/s

V volume

V_c connected pore space

V_f fluid volume

V_p pore volume

V_t total volume

V_{gs} volume of of gas with 100% gas component at standard conditions

V_{or} liquid phase (oil) at reservoir conditions

V_{os} volume of liquid with 100% oil component at standard conditions

Subscripts

f fluid

g gas

$i + 1/2$ the boundary between grid cell i and $i + 1$

i	spatial point represented by the grid index number i
o	oil
p	fluid phase, usually represents wetting and non-wetting phase, i.e. $p \in w, n$
r	reference
r	reservoir conditions
s	standard conditions
w	water

Superscripts

C	upscaled coarse grid values
t	time level

18
Bibliography

Jamal H. Abou-Kassem, S. M. Farouq Ali, and Rafiqul Islam. *Petroleum reservoir simulation: A basic approach*. Gulf Publishing Company, 2006.

K. Aziz. Ten Golden Rules for Simulation Engineers. *Journal of Petroleum Technology*, page 1157, 1989.

Khalid Aziz. Reservoir simulation grids: opportunities and problems. *Journal of Petroleum Technology*, 45(07):658–663, 1993. Publisher: Society of Petroleum Engineers.

Khalid Aziz and Antonin Settari. *Petroleum reservoir simulation*. Applied Science Publishers, 1979.

Ilenia Battiato, Daniel O'Malley, Cass T. Miller, Pawan S. Takhar, Francisco J. Valdés-Parada, and Brian D. Wood. Theory and Applications of Macroscale Models in Porous Media. *Transport in Porous Media*, pages 1–72, 2019.

Einar JM Baumann, Stein I. Dale, and Mathias C. Bellout. FieldOpt: A powerful and effective programming framework tailored for field development optimization. *Computers & Geosciences*, 135: 104379, 2020.

Jacob Bear. *Dynamics of fluids in porous media*. Dover, 1988.

Jacob Bear and Yehuda Bachmat. *Introduction to modeling of transport phenomena in porous media*, volume 4. Springer Science & Business Media, 2012.

R. Brooks and T. Corey. Hydraulic properties of porous media. *Hydrology Papers, Colorado State University*, 24:37, 1964.

S.E. Buckley and M.C. Leverett. Mechanism of fluid displacement in sands. *Transactions of the AIME*, 146(01):107–116, 1942.

Zhangxin Chen, Guanren Huan, and Yuanle Ma. *Computational methods for multiphase flows in porous media*. SIAM, 2006.

M. A. Christie and M. J. Blunt. Tenth SPE comparative solution project: A comparison of upscaling techniques. In *SPE Reservoir Simulation Symposium*. Society of Petroleum Engineers, 2001. URL `https://www.onepetro.org/conference-paper/SPE-66599-MS`.

Henry Darcy. *Les fontaines publiques de la ville de Dijon: exposition et application...* Victor Dalmont, 1856.

Russell Eberhart and James Kennedy. Particle swarm optimization. In *Proceedings of the IEEE international conference on neural networks*, volume 4, pages 1942–1948. Citeseer, 1995.

Turgay Ertekin, Jamal H. Abou-Kassem, and Gregory R. King. *Basic applied reservoir simulation*. Society of Petroleum Engineers, 2001.

Geir Evensen. *Data assimilation: the ensemble Kalman filter*. Springer Science & Business Media, 2009.

Geir Evensen. Analysis of iterative ensemble smoothers for solving inverse problems. *Computational Geosciences*, 22(3):885–908, 2018. Publisher: Springer.

Dario Floreano and Claudio Mattiussi. *Bio-inspired artificial intelligence: theories, methods, and technologies*. MIT press, 2008.

R. M. Fonseca, E. Della Rossa, A. A. Emerick, R. G. Hanea, and J. D. Jansen. Overview of the Olympus field development optimization challenge. In *ECMOR XVI-16th European Conference on the Mathematics of Oil Recovery*, volume 2018, pages 1–10. European Association of Geoscientists & Engineers, 2018. Issue: 1.

Sindre T. Hilden and Carl Fredrik Berg. An analysis of unsteady flooding processes: varying force balance and the applicability of steady-state upscaling. *Transport in Porous Media*, 115(1):125–152, 2016.

Patricia D. Hough, Tamara G. Kolda, and Virginia J. Torczon. Asynchronous parallel pattern search for nonlinear optimization. *SIAM Journal on Scientific Computing*, 23(1):134–156, 2001. Publisher: SIAM.

Frederick A. Howes and Stephen Whitaker. The spatial averaging theorem revisited. *Chemical engineering science*, 40(8):1387–1392, 1985. Publisher: Elsevier.

M. King Hubbert. The theory of ground-water motion. *The Journal of Geology*, 48(8, Part 1):785–944, 1940. Publisher: University of Chicago Press.

P. R. King. The use of renormalization for calculating effective permeability. *Transport in porous media*, 4(1):37–58, 1989. Publisher: Springer.

Joseph Kuniansky and J. G. Hillestad. Reservoir simulation using bottomhole pressure boundary conditions. *Society of Petroleum Engineers Journal*, 20(06):473–486, 1980.

Knut-Andreas Lie. An introduction to reservoir simulation using MATLAB: user guide for the Matlab Reservoir Simulation Toolbox (MRST). *SINTEF ICT, Norway*, 2016.

Frode Lomeland, Einar Ebeltoft, and Wibeke Hammervold Thomas. A new versatile relative permeability correlation. In *International Symposium of the Society of Core Analysts, Toronto, Canada*, volume 112, 2005.

T. Manzocchi, J. J. Walsh, P. Nell, and G. Yielding. Fault transmissibility multipliers for flow simulation models. *Petroleum Geoscience*, 5(1):53–63, 1999. Publisher: European Association of Geoscientists & Engineers.

Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953. Publisher: American Institute of Physics.

A. R. Mitchell and D. F. Griffiths. *The Finite Difference Method in Partial Differential Equations*. John Wiley and Sons, New York, 1980.

Rémi Moyen. *Paramétrisation 3D de l'espace en Géologie sédimentaire: Le modèle Geochron*. PhD Thesis, Institut National Polytechnique de Lorraine, 2005.

John A. Nelder and Roger Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965. Publisher: Oxford University Press.

Jérôme E. Onwunalu and Louis J. Durlofsky. Application of a particle swarm optimization algorithm for determining optimum well location and type. *Computational Geosciences*, 14(1):183–198, 2010. Publisher: Springer.

Donald W. Peaceman. *Fundamentals of numerical reservoir simulation*, volume 6. Elsevier, 1977.

Donald W. Peaceman. Interpretation of well-block pressures in numerical reservoir simulation. *Society of Petroleum Engineers Journal*, 18(03):183–194, 1978.

Donald W. Peaceman. Interpretation of well-block pressures in numerical reservoir simulation with nonsquare grid blocks and anisotropic permeability. *Society of Petroleum Engineers Journal*, 23 (03):531–543, 1983.

Atgeirr Flø Rasmussen, Tor Harald Sandve, Kai Bao, Andreas Lauser, Joakim Hove, Bård Skaflestad, Robert Klöfkorn, Markus Blatt, Alf Birger Rustad, and Ove Sævareid. The Open Porous Media Flow reservoir simulator. *Computers & Mathematics with Applications*, 81:159–185, 2021. Publisher: Elsevier.

Philip Ringrose and Mark Bentley. *Reservoir model design*. Springer, 2015. URL `http://link.springer.com/content/pdf/10.1007/978-94-007-5497-3.pdf`.

Tor Harald Sandve, Sarah E. Gasda, Atgeirr Rasmussen, and Alf Birger Rustad. Convective Dissolution in Field Scale Co2 Storage Simulations Using the OPM Flow Simulator. In *TCCS–11. CO2 Capture, Transport and Storage. Trondheim 22nd–23rd June 2021 Short Papers from the 11th International Trondheim CCS Conference*. SINTEF Academic Press, 2021.

Robert Scheichl, R. Masson, and J. Wendebourg. Decoupling and block preconditioning for sedimentary basin simulations. *Computational Geosciences*, 7(4):295–318, 2003.

Antonin Settari and Khalid Aziz. Use of irregular grid in reservoir simulation. *Society of Petroleum Engineers Journal*, 12(02):103–114, 1972.

Thiago L. Silva, Mathias C. Bellout, Caio Giuliani, Eduardo Camponogara, and Alexey Pavlov. Derivative-free trust region optimization for robust well control under geological uncertainty. *Computational Geosciences*, 26(2):329–349, 2022. Publisher: Springer.

S. M. Skjaeveland, L. M. Siqveland, A. Kjosavik, W. L. Hammervold, and G. A. Virnovsky. Capillary pressure correlation for mixed-wet reservoirs. In *SPE India Oil and Gas Conference and Exhibition*. Society of Petroleum Engineers, 1998.

Per Arne Slotte and Eivind Smorgrav. Response surface methodology approach for history matching and uncertainty assessment of reservoir simulation models. In *Europec/EAGE Conference and Exhibition*. Society of Petroleum Engineers, 2008.

Brook Taylor. *Methodus incrementorum directa et inversa*. Innys, 1715.

Salvatore Torquato. *Random heterogeneous materials: microstructure and macroscopic properties*. 2001.

J. E. Warren and H. S. Price. Flow in heterogeneous porous media. *Society of Petroleum Engineers Journal*, 1(03):153–169, 1961. Publisher: Society of Petroleum Engineers.

Stephen Whitaker. Flow in porous media I: A theoretical derivation of Darcy's law. *Transport in porous media*, 1(1):3–25, 1986.

Otto Wiener. Die Theorie des Mischkorpers fur das Feld der stationaren Stromung. *Abhandlungen der Sachsischen Gesellschaft der Akademischen Wissenschaften in Mathematik und Physik*, 32:507–604, 1912.

19
Index