An Investigation of Second-Order Finite Volume Methods for Field-Scale Reservoir Simulation

by

Anna Kvashchuk

Thesis submitted in fulfillment of
the requirements for degree of
PHILOSOPHIAE DOCTOR
(PhD)

University
of Stavanger

Faculty of Science and Technology
Department of Energy Resources
2024

University of Stavanger
N-4036 Stavanger
NORWAY
www.uis.no

ISBN: 978-82-8439-229-5
ISSN: 1890-1387
PhD thesis UiS No. 752

Acknowledgements

It took me eight years to reach this point, and I strongly believe I would not have arrived at the finish line without the care, support, and guidance of the many people I thank below. I feel fortunate that so many people helped me in many ways, and I am grateful to all of you!

First, I would like to thank my advisors, Robert Klöfkorn, Tor Harald Sandve, and Steinar Evje, for the countless hours they put into my education and for sharing their knowledge and wisdom with me. I am thankful to Robert Klöfkorn for believing in me and hiring me for the PhD position in the first place. Your guidance and expertise were enormously helpful to me, especially in those first years when you were the only person who could provide insight into the intricacies of DUNE and OPM frameworks. I thank Tor Harald Sandve for agreeing to become my co-advisor in 2018 and for all the positivity, expert guidance, and help he has generously provided me with ever since. I want to thank Steinar Evje for stepping in when it was most needed and convincing me to bring this project home.

I want to express my gratitude to Randi Valestrand for her support and guidance. You have helped me to overcome so many personal and bureaucratic challenges, and I cannot thank you enough!

I would also like to thank my colleagues from Norce (IRIS formally) and the National IOR Center of Norway for fruitful discussions, excellent conferences together, and many coffee breaks and lunches. I would especially like to thank two fellow Ph.D. candidates, Anders Matheson and Yiteng Zhang, who have been an enormous mental support for me in those first challenging years of being a Ph.D. candidate and have become my friends.

I thank Trine Mykkeltvedt for her timely advice and the many books I borrowed from her.

I would like to thank Alexey Khrulenko for many inspiring discussions, his expert reservoir engineer tips, and for refining the grid of the Norne model.

I want to thank my colleagues at Equinor for their support and understanding of the difficulties of finishing a Ph.D. while having a full-time job. I thank Markus Fanebust Dregi and Andrea Brambilla for encouraging me to finish the Ph.D. and allowing me to do so by approving my "Ph.D. leave".

My deepest gratitude goes to my family and friends for their constant support and belief in me. In particular, I would like to thank Oleksandr and Valentyna Kazymyrovs for encouraging me not to quit when I really wanted to and Guzel Shamsutdinova for many coffee chats, which helped me stay sane.

I would like to express my sincere gratitude to my parents for their love and support. You both are an example for me, and I am forever grateful to you for instilling in me the importance of education. Thank you, Mama and Papa!

Last but not least, I would like to thank my dear husband, Sergey Alyaev. You have helped me in numerous ways, starting from taking care of our kids so that I get uninterrupted time to work and finishing with the most useful and up-to-the-point comments and ideas on improving my writing, figures, code, you name it. I admire your ability to think outside the box, your endless pursuit of perfection of any graph or figure (even when it means I will redo my graphs several times), and your ability to read my work and provide valuable feedback even when kids run screaming around you. I love you, and I know for a fact that I never would have finished this work without you! *On the other hand, most probably, I would not have started this work without you either, but that would have been a different story:)*

Finally, I thank Kate and Michael for simply being in my life. You were not there when I started this work, but you appeared and showed me the value of prioritization and work-life balance.

I acknowledge Equinor ASA for its support during the preparation of the last paper. I acknowledge the Research Council of Norway and the industry partners, ConocoPhillips Skandinavia AS, Aker BP ASA, Eni Norge AS, Equinor ASA, Neptune Energy Norge AS, Lundin Norway AS, Halliburton AS, Schlumberger Norge AS, Wintershall Norge AS, and DEA Norge AS, of The National IOR Centre of Norway for support.

Abstract

This thesis focuses on improving the accuracy of numerical simulation of flow in subsurface reservoirs by using second-order finite volume methods. The first-order finite volume method, commonly used in reservoir simulation, is known to suffer from excessive numerical diffusion, leading to inaccuracies in the front position, breakthrough time, etc. Refining the computational grid is one way to increase accuracy; however, it significantly increases computational costs.

Higher-order numerical methods offer a more accurate representation of the solution without the need for grid refinement. This study compares three second-order finite volume methods with respect to their accuracy, convergence rate, and computational efficiency for modeling immiscible fluid displacement.

Two second-order finite volume methods were implemented in the open-source reservoir simulator OPM Flow. This allowed us, first, to perform a study in the realistic reservoir modeling setting and, second, to make it readily accessible to anyone interested in improving reservoir simulations' accuracy.

The implemented methods were validated against synthetic benchmarks and realistic field model Norne to critically evaluate the methods' applicability to complex reservoir flow modeling, with a focus on enhanced oil recovery application. The evaluation shows that our best method, the second-order method with linear programming reconstruction, outperformed other tested second-order methods and provided sharper fluid front resolution compared to the traditionally used first-order method. Moreover, our best method's solution on a coarse grid was on par with the refined-grid solution from the first-order method across the benchmarks.

List of Papers

Paper A Robert Klöfkorn, Anna Kvashchuk, Martin Nolte. (2017) Comparison of linear reconstructions for second-order finite volume schemes on polyhedral grids. *Computational Geosciences*, volume 21, pages 909–919.

Paper B Anna Kvashchuk, Robert Klöfkorn, Tor Harald Sandve. (2019) Comparison of Higher Order Schemes on Complicated Meshes and Reservoirs. In *SPE Reservoir Simulation Conference, Galveston, Texas, USA*.

Paper C Anna Kvashchuk, Robert Klöfkorn, Tor Harald Sandve. (2023) A Second-Order Finite Volume Method for Field-Scale Reservoir Simulation. *Transport in Porous Media*, volume 150, pages 109-129.

List of Repositories

To reproduce the main results of the thesis, we include the following open-source repositories. Repository I contains the installation instructions and the scripts needed for reproducibility. Repository II extends the OPM Flow reservoir simulator with the implementation of second-order finite volume methods.

Repository I Second-Order Finite Volume Method for Field-Scale Reservoir Simulation. https://github.com/kvashchuka/second-order-opm-tests Contributors: Anna Kvashchuk, Alexey Khrulenko.

Repository II opm-models: second-order methods extension. https://github.com/kvashchuka/opm-models/ Contributors: Anna Kvashchuk, Robert Klöfkorn, Tor Harald Sandve, and contributors to https://github.com/OPM/opm-models.

Additional papers

For completeness, we include the following two papers published during the PhD period. They are not considered part of this thesis:

Paper D is a conference paper that was later extended and included in Paper A.

Paper D Robert Klöfkorn, Anna Kvashchuk, Martin Nolte. (2016) Comparison of Linear Reconstructions for Second Order Finite Volume Schemes on Polyhedral Grids. In *ECMOR XV-15th European Conference on the Mathematics of Oil Recovery.*

Paper E is based on the candidate's Master's thesis and lies outside the scope of this work.

Paper E Anna Kvashchuk, Florin Adrian Radu. (2019) A fully-implicit, iterative scheme for the simulation of two-phase flow in porous media. *Lecture Notes in Computational Science and Engineering*, volume 126, pages 625-633.

Contents

List of Figures

Chapter 1

Introduction

The numerical modeling of subsurface flow is crucial for addressing a range of technological challenges, including sustainable management of freshwater reserves, reducing greenhouse gas emissions through CO_2 storage, ensuring safe storage of nuclear waste, and optimizing petroleum recovery. According to the Norwegian OG21[1] strategy report, the efficient use of existing petroleum resources is vital for meeting energy demands and reducing exploration. Software technologies, including high-fidelity numerical simulators of oil extraction processes, should be among the prioritized developments for efficient resource management [OG21, 2021]. Improved Oil Recovery (IOR) includes many techniques that allow the extraction of additional oil from producing fields and ranges from production optimization to infill wells and chemical and heat flow stimulation [Chierici, 1992]. Enhanced Oil Recovery (EOR) is the group of IOR methods that are used to recover oil at later production stages. EOR methods aim at changing the properties of reservoirs and liquids to improve and direct the hydrocarbon flow. They include injection of water alternating with gas (including CO_2), hot water or steam injection, and polymer flooding [Gao, 2011, Mykkeltvedt, 2014].

To comprehend and forecast the movement of fluids such as water, CO_2 gas, and oil in underground rock formations, robust numerical methods for modeling subsurface flow are essential. These methods play an indispensable role in making reliable decisions related to optimizing oil and gas production and managing CO_2 storage effectively. Consequently, the development of accurate and robust numerical techniques for modeling multi-phase multi-component flow stands as a critical area of research that can significantly impact subsurface engineering.

In subsurface reservoir simulation, traditionally, finite volume methods are favored over finite element methods [Lie, 2019]. On the one hand, they have conservation properties and simplicity, which are useful for achieving adequate solutions for large coarse models in a reasonable time. On the other hand, their robustness on general polyhedral grids is of great help in resolving complicated geological structures. These requirements define what we will call **practical reservoir simulation**.

The first-order finite volume method is currently considered an industry-standard approach, and it is the default choice for both commercial and open-source reservoir simulation software. This includes ECLIPSE [SLB, 2020], TOUGH3 [Jung et al., 2017], OPM (Open Porous Media) [Rasmussen et al., 2021], the Matlab Reservoir Simulation Toolbox (MRST) [Lie, 2019], DuMu^x [Flemisch et al., 2011, Koch et al., 2021], PFLOTRAN [Lichtner et al., 2019], and a closed source research simulator, Automatic Differentiation General Purpose Research Simulator (ADGPRS) [Voskov and Tchelepi, 2012]. Unfortunately, the

[1] OG21 – Oil and Gas in the 21st Century

first-order finite volume method is known to suffer from excessive numerical diffusion, leading to smeared fronts and incorrect species concentrations, front position, breakthrough time, etc. [Lie, 2019].

There are several options for improving the accuracy of the numerical simulation of flow in subsurface reservoirs. One of the most straightforward ways to increase accuracy is to refine the computational grid. A finer grid can capture more flow details and show a more accurate representation of the solution. However, fine grids are not feasible for large, full-scale reservoir model simulations due to memory and run-time constraints. Simultaneously, the refinement of field models is a complex and time-consuming task due to discontinuities caused by the geological complexity. Thus, grid refinement is not common for practical reservoir simulation.

An alternative approach is to use higher-order numerical methods to improve the accuracy of the simulation, resulting in better resolution of fronts without the need for grid refinement. This thesis is focused on the use of second-order finite volume methods for practical reservoir simulation. We aim to provide insight, comparison, and strategies for implementing a second-order method in the open-source reservoir simulation framework called the Open Porous Media initiative.

The concept of using higher-order methods in reservoir simulation is not new and has been discussed in the literature since the 1980s. Bell and Shubin [Bell and Shubin, 1985] presented a higher-order Godunov scheme for one- and two-dimensional five-spot problems, while the authors in [Rubin and Blunt, 1991, Blunt and Rubin, 1992, Rubin and Edwards, 1993] discussed the use of higher-order total variation diminishing (TVD) schemes in one and two-dimensional simplified reservoir simulation. Chen et al. [Chen et al., 1993] applied second-order TVD and third-order essentially non-oscillatory schemes to improve front resolution in the 2D five-spot model. A weighted-ENO (WENO) method was introduced in [Jiang and Shu, 1996, Liu et al., 1994], where a weighted combination of the stencils is used, favoring smoother approximations.The discontinuous Galerkin (DG) method was introduced for transport problems in [Cockburn and Shu, 1989] and can also be applied for generating arbitrarily high-order schemes for hyperbolic conservation laws. May and Berger [May and Berger, 2013] proposed using constraint optimization with linear programming to compute higher-order reconstructions. Similarly, Chen et al. [Chen and Li, 2016] proposed an improved linear programming scheme that did not require an initial gradient computation.

Despite the continued research into higher-order methods in reservoir simulation, most implementations are still done in academic codes with Cartesian or simplex meshes [Durlofsky et al., 1992, Harten, 1997, Geiger et al., 2009, Lamine and Edwards, 2015, Contreras et al., 2016, Mykkeltvedt et al., 2017]. While industry-reference reservoir simulator ECLIPSE offers the capability to control numerical diffusion when modeling solvent using the two-point upstream projection technique, this feature is restricted to grids without non-neighbor connections, which means that it cannot handle, for example, faults [SLB, 2020]. Few researchers have applied these methods to implementation-intensive corner-point

grids that capture the complex geometries of subsurface reservoirs. In recent studies, Lie et al. [Lie et al., 2020] applied a weighted-ENO method to simplified test cases on reservoir-type grids, while Klemetsdal et al. [Klemetsdal et al., 2020] used a discontinuous Galerkin method for compositional flow on realistic reservoir meshes. While DG methods show promise for solving such problems, their implementation in industrial codes can be even more challenging than reconstruction-based higher-order FV methods, as most commercial simulators use a data layout corresponding to first-order FV methods.

This work aims to test the applicability of second-order finite volume methods to practical reservoir simulation. The thesis presents a comparative analysis of two second-order finite volume schemes implemented in a full-scale reservoir simulator. The main contributions of this work are listed below.

1. Paper A tests three second-order finite volume methods combined with several slope-limiting techniques on general polyhedral and corner-point grids typically used in reservoir simulation. It verifies the methods' accuracy, convergence rate, and computational efficiency in modeling immiscible fluid displacement in simple-shaped domains. The summary of paper A is presented in Section 6.1.

2. Paper B describes the implementation of the best of the tested methods in the open-source reservoir simulator, OPM Flow, and the conducted convergence studies. The summary of paper B is presented in Section 6.2.

3. The methods' implementation in an industry-adopted OPM Flow makes them readily accessible for reservoir engineers through our open-source repository with detailed installation instructions github.com/kvashchuka/second-order-opm-tests. Moreover, this thesis also provides important implementation details that may help adoption in other simulators.

4. Paper C validates the implementation against industry-standard benchmarks and realistic field studies for IOR and EOR. It conducts WAG and CO_2 injection scenarios on synthetic reservoirs and the publicly available Norne field. The summary of paper C is presented in Section 6.3.

5. To solve the challenge of verification for complex reservoirs, a refined Norne field model was prepared. The first-order method's results on the refined grid verify the improvement the second-order simulation provides [Paper C]. All the benchmark models are published in a github.com/kvashchuka/second-order-opm-tests.

6. Through extensive testing, Paper C critically evaluates the benefits and limitations of higher-order transport methods in the practical reservoir simulation; see Section 6.3 for details.

Organization of thesis

This main part of the thesis provides the necessary background theory and summarizes the Papers A-C comprising the PhD work. Additional included Papers D-E were published during the PhD period but are not considered part of this thesis.

The rest of the main part of the thesis is organized as follows. Chapter 2 introduces the mathematical model for simulating fluid flow in a subsurface reservoir. Initially, we discuss a simple single-phase flow model and then introduce a two-phase flow model. Subsequently, we present the black-oil model, which is the industry-standard three-phase three-component model. Later, we extend the model with the solvent component, which allows us to model the EOR processes, namely WAG and CO_2 injection. Next, Chapter 3 introduces finite volume methods for conservation laws, as a necessary base for an overview of the second-order finite volume methods used in our study, which are presented in Chapter 4. Chapter 5 details the implementation of the proposed second-order numerical methods in the OPM Flow simulator. A summary of the included papers is presented in Chapter 6, followed by the conclusion and future work in the final chapter.

Chapter 2

Mathematical models of flow in porous media

The porous media refers to a vast collection of materials characterized by a solid skeleton (the matrix) and void spaces (the pores) in between. There are various natural and artificial porous media, such as human skin, sponges, wood, subsurface water reservoirs (aquifers), and oil and gas reservoirs. Improving flow modeling through porous media in subsurface reservoirs is the main objective of this study.

In this chapter, we introduce the physical properties of porous media and the mathematical models commonly used for modeling fluid flow through them. Section 2.3 presents a simple single-phase model. We extend it to a two-phase model in Section 2.4 and eventually introduce a black oil model - a three-phase, three-component model widely used in reservoir simulations in Section 2.5. In Section 2.6, we extend the black oil model with the solvent component to enable the CO_2 injection modeling.

2.1 Physical properties

Porous media consists of a solid matrix and voids, so it cannot be represented point-wise. Instead, *representative element volume (REV)* is commonly used. The size of REV is defined as the smallest volume, which has a representative amount of pore spaces and matrix. One of the characteristics of porous media is *porosity*: the fraction of void space in the REV accessible to the flowing fluids. Isolated pores, which an injected fluid cannot reach, are not accounted for in the porosity. Porosity is denoted as ϕ, a function of space and/or time. However, changes in porosity with time are usually minimal and can be neglected.

The next important property of porous media is *absolute permeability* k. It is mainly derived experimentally as the measure of a fluid's ability to flow through the media when only one fluid is present. The SI units of k is $[m^2]$, but commonly used unit is *Darcy* or milliDarcy, 1 darcy $\sim 10^{-12}\text{m}^2$. If the reservoir properties do not change in space (e.g., $k = const$), then the porous medium is called *homogeneous*. However, most real-life reservoirs are *heterogeneous*, meaning that in the general case, the permeability varies spatially. If the permeability is a tensor with only diagonal elements, which are all equal to each other, the porous media is called *isotropic*. Otherwise, it is *anisotropic*.

Let us now introduce the fluid properties in play. Fluid *density* is a mass of the fluid per unit volume; SI units are $[kg/m^3]$. The fluid is called *incompressible* if the fluid density does not change in time; it is not influenced by the change in pressure, temperature, etc. Otherwise, the fluid is *compressible*. Fluid *viscosity*

is a measure of fluid's resistance to the deformations, SI units $[Pa\,s] = [kg/ms]$. Usually, in fluids, the viscosity corresponds to the so-called *thickness* (resistance to pouring). For example, oil is considered thick, while water is thin, meaning that oil is more viscous than water and has a higher viscosity. In general, the fluid's viscosity can also depend on pressure, temperature, etc.

2.2 Darcy's law

Darcy's law is the central part in modeling the fluid flow through the porous media. The law was derived experimentally by French engineer Henry Darcy in the 19th century.

In the original experiment, Darcy measured the water flow through the sand column and noticed a linear dependence between the flow rate and the pressure drop. Darcy's law can be written in many ways, but the most commonly used in reservoir engineering is the following differential form:

$$\mathbf{u} = -\frac{k}{\mu}(\boldsymbol{\nabla} p - \rho \mathbf{g}), \tag{2.1}$$

where $\mathbf{u}$ is the Darcy velocity, k is the absolute permeability, μ and ρ are the fluid viscosity and density, and $\mathbf{g} = -g\mathbf{e}_z = (0, 0, -g)^T$ is the gravity vector. As was stated before, 1 Darcy $\sim 10^{-12}\mathrm{m}^2$, and it represents a 1 cm^3 of fluid with a viscosity of 1 cP (centipoise) flowing through a 1 cm^2 cross-sectional area of the medium in 1 second under a pressure difference of 1 atmosphere per centimeter of length in the direction of flow, which corresponds to Darcy's law.

2.3 Single-phase flow

In this section, we will derive the single-phase model for flow in porous media, starting with the mass balance equation. The change of mass within the domain Ω can be caused only by the mass transfer through the boundary or sinks or sources:

$$\frac{\partial}{\partial t}\int\limits_{\Omega} \eta dV = -\oint\limits_{\partial\Omega} \mathbf{f}\cdot\mathbf{n} ds + \int\limits_{\Omega} \boldsymbol{\psi} dV, \tag{2.2}$$

where η is the mass of fluid within the porous volume, $\mathbf{f}$ is the mass flux, $\mathbf{n}$ is the unit normal vector to the surface $\partial\Omega$ of the domain, pointing in the outward direction, and $\boldsymbol{\psi}$ is the mass source/of accessible to the fluid flow sink term, measured in mass per volume per time.

For single-phase flow, η is equal to the fluid density multiplied with porosity $\eta = \rho\phi$, the mass flux vector can be represented as $\mathbf{f} = \rho\mathbf{u}$, where $\mathbf{u}$ is a volumetric flow rate per area. It represents how much fluid goes through a column in a unit of time. After substituting the above quantities into (2.2) and applying the divergence theorem and the Leibniz integral rule, assuming Ω is a time-invariant domain, we get:

$$\int\limits_{\Omega} \left(\frac{\partial \phi\rho}{\partial t} + \boldsymbol{\nabla}\cdot(\rho\mathbf{u}) - \boldsymbol{\psi} \right) dV = 0. \tag{2.3}$$

Under the condition of sufficient smoothness of the involved functions, we can get the conservation equation in the differential form:

$$\frac{\partial \phi \rho}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \mathbf{u}) = \boldsymbol{\psi}, \tag{2.4}$$

which is also commonly used for the brevity of notation.

Darcy's law (2.1), together with the mass balance equation (2.2) form the mathematical model for single-phase flow in porous media. An appropriate boundary and initial conditions should also be introduced to close the system. In the case of oil reservoirs, the most common boundary condition is a no-flow Neumann boundary, as the reservoir is usually surrounded by impermeable rock. The description of other boundary conditions can be found in many classical books (e.g., [Chen et al., 2006]).

2.4 Two-phase immiscible flow

Single-phase fluid systems are not the most common. In most cases, there is not one but multiple fluids present. Consequently, in order to be able to model multi-phase flow through porous media, we must introduce the *fluid phase saturation*, S_α, which is defined as a fraction of pore space occupied by the fluid α. Naturally, the sum of all present fluid saturations must add up to one $\sum_\alpha S_\alpha = 1$.

In this section, we derive the model for a system with two *immiscible* fluids, meaning that mass transfer between the two phases does not occur. An example of such a system is the oil-water system. The immiscibility assumption will be relaxed in the next section.

In the oil-water system, the water phase is also referred to as a *wetting fluid*, meaning it is preferentially attracted to the rock, while the oil phase is called *non-wetting fluid*, as it generally exhibits the opposite behavior. The mathematical definition of a wetting fluid is one in which the contact angle with the surface is less than 90°, while for the non-wetting phase, the contact angle is between 90° and 180°. This implies that certain forces act between the fluid and the rock, resulting in a pressure difference across the fluid's interface, commonly represented as *capillary pressure*. In the context of two-phase flow, capillary pressure can be written as

$$p_c = p_n - p_w, \tag{2.5}$$

where p_n and p_w are pressure functions of non-wetting and wetting fluids, respectively. In this work, we disregard known capillary pressure non-equilibrium extensions [Joekar-Niasar et al., 2010], which are not yet widely used in reservoir simulation.

Similarly to (2.4), we can write a mass balance equation for each phase α:

$$\frac{\partial \phi \rho_\alpha S_\alpha}{\partial t} + \boldsymbol{\nabla} \cdot (\rho_\alpha \mathbf{u}_\alpha) = \boldsymbol{\psi}_\alpha, \quad \alpha = w, n, \tag{2.6}$$

where ρ_α, S_α, $\mathbf{u}_\alpha$ and $\boldsymbol{\psi}_\alpha$ denote the density, saturation, volumetric flow rate (also called Darcy velocity) and source/sink for each phase α that could be either wetting $\alpha = w$ or non-wetting phase $\alpha = n$.

To determine the Darcy velocity $\mathbf{u}_\alpha$, we must extend Darcy's law (2.1) to describe the fluid flow in the two-phase flow system. Since there are two fluids present, the amount of pore space available for each fluid is reduced. To account for this reduction, *relative permeability* $k_{r,\alpha}$ is introduced. It is defined as a function (typically non-linear) of fluid saturation S_α: $k_{r,\alpha} = k_{r,\alpha}(S_\alpha)$ and is different for each phase α. Relative permeability, like absolute permeability, may be modeled as either a tensor- or a scalar-valued function, depending on the properties of the porous media. However, it is typically anisotropic and determined experimentally. Darcy's law for two-phase flow is then written as:

$$\mathbf{u}_\alpha = -\frac{k_{r,\alpha} k}{\mu_\alpha} \left(\boldsymbol{\nabla} p_\alpha - \rho_\alpha \mathbf{g}\right), \tag{2.7}$$

where μ_α and p_α are phase viscosity and pressure respectively. Let us now introduce *phase mobility*, λ_α, which is defined as

$$\lambda_\alpha = \frac{k_{r,\alpha}}{\mu_\alpha}. \tag{2.8}$$

Then Darcy's law for two-phase flow takes the following form:

$$\mathbf{u}_\alpha = -\lambda_\alpha k \left(\boldsymbol{\nabla} p_\alpha - \rho_\alpha \mathbf{g}\right) \tag{2.9}$$

The full system of equations describing two-phase flow in porous media is written as follows:

$$\begin{aligned}
&\frac{\partial \phi \rho_\alpha S_\alpha}{\partial t} + \boldsymbol{\nabla} \cdot (\rho_\alpha \mathbf{u}_\alpha) = \boldsymbol{\psi}_\alpha, \\
&\mathbf{u}_\alpha = -\lambda_\alpha k \left(\boldsymbol{\nabla} p_\alpha - \rho_\alpha \mathbf{g}\right), \\
&S_w + S_n = 1, \\
&p_n - p_w = p_c(S_w), \\
&S_\alpha^0 = S_\alpha(x, t_0), \quad p_\alpha^0 = p_\alpha(x, t_0), \\
&S_\alpha|_{\partial\Omega} = S_\alpha^\Gamma(x, t), \quad p_\alpha|_{\partial\Omega} = p_\alpha^\Gamma(x, t),
\end{aligned} \tag{2.10}$$

where $\alpha = \{w, n\}$ and two last lines represent initial and boundary conditions.

Capillary pressure and relative permeability functions are typically modeled as algebraic functions of saturation based on laboratory experiments. However, such functions exhibit a history-dependent behavior [Joekar-Niasar et al., 2010, Nordbotten and Celia, 2011]. When considering an immiscible fluid displacement process, the behavior differs depending on whether a non-wetting fluid displaces a wetting fluid (also called *imbibition*) or vice versa (also called *drainage*). As a result, the same saturation value corresponds to two different states, leading to different capillary pressure and relative permeability values. Therefore, depending on whether the saturation increases or decreases, slightly different values for capillary pressure and relative permeabilities must be selected. Experimental

data has shown that capillary pressure depends on hysteresis and the saturation change rate [Hassanizadeh et al., 2002, Holm et al., 2008], making it a significant research question to select the right parametrization. The most commonly used parametrization is the van Genuchten [Van Genuchten, 1980] and Brooks-Corey [Brooks and Corey, 1964], chosen for their simplicity.

2.5 Black-oil equations

The *black oil model* [Chen, 2000, Ghoreishian Amiri et al., 2013, Lie, 2019] is a widely used three-phase, three-component model that effectively represents the behavior of gas, oil, and water in the reservoir. The two-phase flow model (2.10) is derived under the assumption of the immiscibility of the involved fluids. However, in a real reservoir, there could be a mass transfer between the phases under certain conditions, such as gas mixing with oil or oil mixing with gas. This implies that the mass conservation equation for each phase will not hold, and instead, it will be valid within each component. Thus, the black oil model is formulated in terms of components to account for the mass transfer between the phases. To distinguish between phases and components, we will use upper case letters for components $\kappa \in \{W, G, O\}$ ("water", "gas," and "oil") and lower letters for phases $\alpha \in \{w, g, o\}$ (aqueous, gaseous, and oleic). The derivations below follow [Chen et al., 2006].

On the surface conditions produced hydrocarbons can form two pseudo-components: heavy hydrocarbons will form oil, and light will produce gas, but at the reservoir condition, they both can mix and be present both in a liquid oleic phase and in a gaseous phase. Two components in each phase share the same velocity, temperature, etc., and, therefore, should be modeled together. The third component, water, forms a separate aqueous phase, which in this model does not mix with two other phases and consists only of water.

Let us introduce *formation volume factors* for each phase (B_α for phase α). It is determined as a ratio between the phase volume measured at the reservoir conditions V_α to the component volume measured at standard conditions $V_\kappa^{\mathbf{s}}$:

$$B_\alpha := \frac{V_\alpha}{V_\kappa^{\mathbf{s}}}. \tag{2.11}$$

The superscript $\mathbf{s}$ here and later indicate that quantity was measured at the standard condition, while the absence of superscript corresponds to the reservoir condition.

Since the water phase consists only of water component, the water formation volume factor is enough to represent the aqueous phase density:

$$\rho_w = \frac{\rho_W^{\mathbf{s}}}{B_w}. \tag{2.12}$$

However, since vaporized oil can be present in the gas phase and gas, in turn, can be dissolved in the oil phase, we need to introduce two more quantities: *gas*

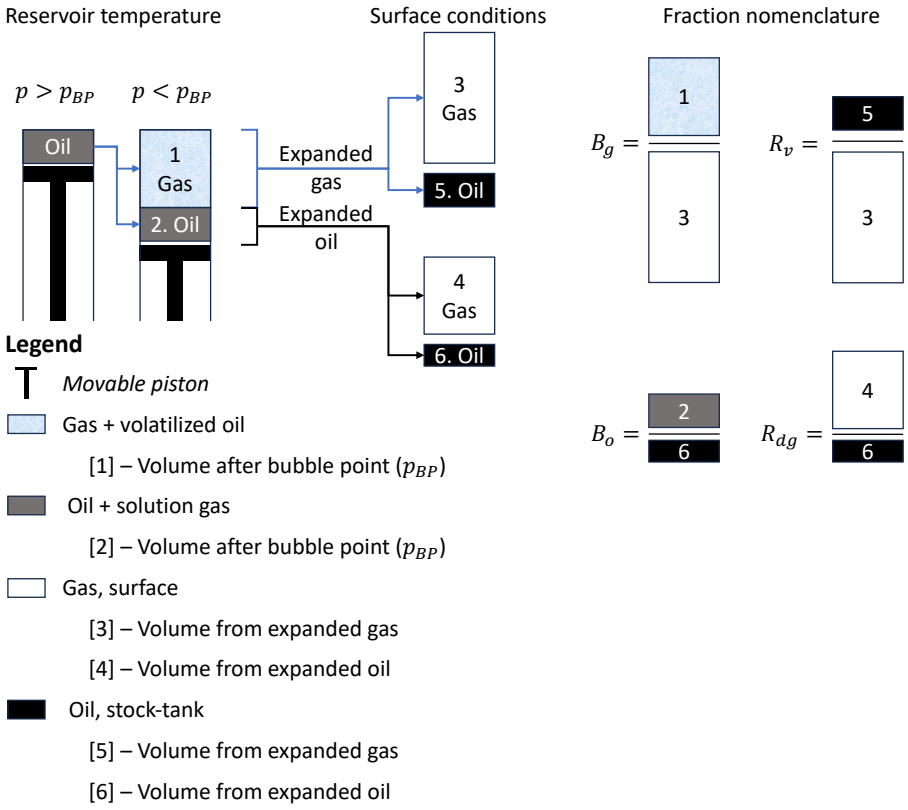

Figure 2.1: Visual explanation of formation volume factors and ratios. Adapted from [Walsh et al., 2003].

solubility, R_{dg} (also referred to as *dissolved gas-oil ratio*) and oil volatility R_v:

$$R_{dg} := \frac{V_{G^d}^{\mathbf{s}}}{V_O^{\mathbf{s}}}, \tag{2.13}$$

$$R_v := \frac{V_{O^v}^{\mathbf{s}}}{V_G^{\mathbf{s}}}. \tag{2.14}$$

The formula above states that gas solubility is equal to the ratio of the volume of surface gas to the volume of stock-tank oil, both measured at the standard conditions and given that they were obtained from some amount of oleic phase at reservoir conditions. Oil volatility in the gas phase is a ratio of the volume of oil to the volume of gas component, both measured at the standard conditions and given that they were both obtained from some amount of gaseous phase at reservoir conditions. All introduced above quantities are visually explained in Figure 2.1.

Let us start by writing down the mass balance equation for each component:

- the water component

$$\frac{\partial(\phi\rho_w S_w)}{\partial t} + \boldsymbol{\nabla}\cdot(\rho_w \mathbf{u}_w) = q_W, \tag{2.15}$$

- the oil component

$$\frac{\partial}{\partial t}\left(\phi(\rho_{O_o} S_o + \rho_{O_g} S_g)\right) + \boldsymbol{\nabla}\left(\rho_{O_o}\mathbf{u}_o + \rho_{O_g}\mathbf{u}_g\right) = q_O, \tag{2.16}$$

- the gas component

$$\frac{\partial}{\partial t}\left(\phi(\rho_{G_g} S_g + \rho_{G_o} S_o)\right) + \boldsymbol{\nabla}\left(\rho_{G_g}\mathbf{u}_g + \rho_{G_o}\mathbf{u}_o\right) = q_G, \tag{2.17}$$

where ρ_{O_o} and ρ_{O_g} are the densities of oil components in oleic and gaseous phases, respectively; and similarly ρ_{G_g} and ρ_{G_o} are the densities of the gas component in gaseous and oleic phases respectively. Note that $\rho_{O_g} = \rho_v$ is the vaporised oil density and $\rho_{G_o} = \rho_{dg}$ is dissolved gas density.

The mass of the oleic phase at reservoir condition is equal to the mass of oil and gas components measured at the standard conditions (see Figure 2.1):

$$\rho_o V_o = \rho^{\mathbf{s}}_O V^{\mathbf{s}}_O + \rho^{\mathbf{s}}_{G^d} V^{\mathbf{s}}_{G^d}, \tag{2.18}$$

After dividing the equation above by V_o and rearranging the terms, we get:

$$\rho_o = \rho^{\mathbf{s}}_O \frac{V^{\mathbf{s}}_O}{V_o} + \rho^{\mathbf{s}}_{G^d}\frac{V^{\mathbf{s}}_{G^d}}{V_o}\frac{V^{\mathbf{s}}_O}{V^{\mathbf{s}}_O} = \underbrace{\frac{\rho^{\mathbf{s}}_O}{B_o}}_{\rho_{O_o}} + \underbrace{\frac{\rho^{\mathbf{s}}_{G^d} R_{dg}}{B_o}}_{\rho_{dg}}. \tag{2.19}$$

We need to perform the same operation for the gas phase density:

$$\rho_g V_g = \rho^{\mathbf{s}}_{O^v} V^{\mathbf{s}}_{O^v} + \rho^{\mathbf{s}}_G V^{\mathbf{s}}_G, \tag{2.20}$$

$$\rho_g = \rho^{\mathbf{s}}_{O^v}\frac{V^{\mathbf{s}}_{O^v}}{V_g}\frac{V^{\mathbf{s}}_G}{V^{\mathbf{s}}_G} + \rho^{\mathbf{s}}_G\frac{V^{\mathbf{s}}_G}{V_g} = \underbrace{\frac{\rho^{\mathbf{s}}_{O^v} R_v}{B_g}}_{\rho_v} + \underbrace{\frac{\rho^{\mathbf{s}}_G}{B_g}}_{\rho_{G_g}}. \tag{2.21}$$

Now, we can substitute introduced phase densities into the mass balance equations for components, and we will get the conservation equations on standard volumes:

$$\frac{\partial}{\partial t}\left(\frac{\phi\rho^{\mathbf{s}}_W}{B_w} S_w\right) + \boldsymbol{\nabla}\cdot\left(\frac{\rho^{\mathbf{s}}_W}{B_w}\mathbf{u}_w\right) = q_W, \tag{2.22}$$

$$\frac{\partial}{\partial t}\left[\phi\left(\frac{\rho^{\mathbf{s}}_O}{B_o} S_o + \frac{\rho^{\mathbf{s}}_O R_v}{B_g} S_g\right)\right] + \boldsymbol{\nabla}\cdot\left(\frac{\rho^{\mathbf{s}}_O}{B_o}\mathbf{u}_o + \frac{\rho^{\mathbf{s}}_O R_v}{B_g}\mathbf{u}_g\right) = q_O, \tag{2.23}$$

$$\frac{\partial}{\partial t}\left[\phi\left(\frac{\rho^{\mathbf{s}}_G}{B_g} S_g + \frac{\rho^{\mathbf{s}}_G R_{dg}}{B_o} S_o\right)\right] + \boldsymbol{\nabla}\cdot\left(\frac{\rho^{\mathbf{s}}_G}{B_g}\mathbf{u}_g + \frac{\rho^{\mathbf{s}}_G R_{dg}}{B_o}\mathbf{u}_o\right) = q_G. \tag{2.24}$$

The phase Darcy velocities $\mathbf{u}_\alpha$, as before, are determined by Darcy's law:

$$\mathbf{u}_\alpha = -\lambda_\alpha k \left(\boldsymbol{\nabla} p_\alpha - \rho_\alpha \mathbf{g}\right). \tag{2.25}$$

Same as for the two-phase flow model, we need constitutive relation:

$$S_w + S_o + S_g = 1, \tag{2.26}$$

and capillary pressures

$$p_{cow} = p_o - p_w, \quad p_{cog} = p_g - p_o. \tag{2.27}$$

The third capillary pressure can be determined by p_{cow} and p_{cog}.

Together with appropriate boundary and initial conditions equations (2.22)-(2.27) form the extension of the black-oil model to include volatile oil.

2.6 The black oil model extended with the solvent component

This section will extend the presented black oil model with an additional solvent component, which is commonly used to model CO_2 flow. This extension was originally proposed by [Todd et al., 1972] and further developed in [Chase Jr et al., 1984]. An alternative method for CO_2 modeling is compositional simulations. The compositional model provides an accurate representation but is considerably more computationally expensive than the black-oil models [Lie and Møyner, 2021]. The extended black-oil model provides a good trade-off between accuracy and simulation time, making it a widely preferred option for practical reservoir simulations. The derivations in this section follow [Sandve et al., 2018].

To create the extended model, we add a conservation equation for the solvent component. The presence of solvent changes the relative permeability, viscosity, capillary pressure, residual saturation, and density of the hydrocarbons. These quantities for the water component stay unchanged. In OPM, the properties of fully miscible and fully immiscible cases are computed, and then a miscibility function M that depends on pressure and solvent saturation is used to interpolate between these two limit points [Rasmussen et al., 2021]. The Todd-Longstaff model [Todd et al., 1972] is used to model the process on a coarse scale. The model does not account for the viscous fingering effect, as it is not practical to resolve individual fingers with fine grid resolution in field-scale simulations. The mixing parameter ω and new effective properties (effective viscosity, effective permeability, effective density) that depend on it are introduced.

The conservation equation for the solvent component is written as:

$$\frac{\partial}{\partial t}\left(\frac{\phi \rho_s}{B_s} S_s\right) + \boldsymbol{\nabla} \cdot \left(\frac{\rho_s}{B_s} \mathbf{u}_s\right) = q_s, \tag{2.28}$$

where B_s is the formation volume factor for solvent, ρ_s, S_s, $\mathbf{u}_s$ is solvent density, saturation and Darcy velocity respectively. As before, Darcy's velocity is given

by Darcy's law (2.9). As we have a new component, the solvent saturation should be added in (2.26)[Rasmussen et al., 2021]:

$$S_w + S_g + S_g + S_s = 1. \tag{2.29}$$

The presence of solvent changes many quantities. Let us start with the residual saturation:

$$S_{ro} = S^m_{ro} M + S^i_{ro}(1 - M), \tag{2.30}$$

where S^m_{ro} is the fully miscible residual oil saturation, while S^i_{ro} - the immiscible. Similarly, for the solvent plus gas, the new residual saturation is defined as:

$$S_{rsg} = S^m_{rsg} M + S^i_{rsg}(1 - M), \tag{2.31}$$

where S^m_{rsg} is again the miscible residual solvent plus gas saturation, and S^i_{rsg} - the immiscible.

The new effective relative permeability is again modeled as an interpolation between fully miscible and fully immiscible relative permeability:

$$k_r = k^m_r M + k^i_r(1 - M), \tag{2.32}$$

where k^m_r is the relative permeability in the fully mixed case and k^i_r - relative permeability for the immiscible case and M is a user-defined miscibility parameter, $M \in [0, 1]$ (keyword MISC is used to define this parameter in the input file, see OPM manual [Baxendale, 2023] for details).

In the fully immiscible case, relative permeability for gas and solvent components are fractions of the new total (gas plus solvent) relative permeability $k_{r,gt}$ of the gas phase:

$$k^i_{r,s} = \frac{S_s}{S_g + S_s} k_{r,gt}, \quad k^i_{r,g} = \frac{S_g}{S_g + S_s} k_{r,gt} \tag{2.33}$$

The other relative permeabilities for immiscible case are:

$$\begin{aligned} k^i_{r,w} &= k_{r,w}(S_w), \\ k^i_{r,o} &= k_{r,o}(S_w, S_g), \\ k^i_{r,gt} &= k_{r,g}(S_g + S_s). \end{aligned} \tag{2.34}$$

The relative permeabilities for the fully miscible case are defined as:

$$\begin{aligned} k^m_{r,w} &= k_{r,w}(S_w), \\ k^m_{r,o} &= \frac{S_g - S_{ro}}{S_n - S_{gc} - S_{ro}} k_{r,n}(S_n), \\ k^m_{r,s} = k^m_{r,g} = k^m_{r,gt} &= \frac{S_g + S_s - S_{gc}}{S_n - S_{gc} - S_{ro}} k_{r,n}(S_n), \end{aligned} \tag{2.35}$$

where $S_n = S_g + S_g + S_s$ is the total hydrocarbon saturation, $k_{r,n}$ is the relative permeability of hydrocarbon to water, S_{ro} is the residual oil saturation and S_{gc} is the critical gas saturation.

Now we are ready to write down the effective relative permeability of oil, gas, and solvent (water relative permeability is not influenced by the solvent):

$$\begin{aligned} k^{\xi}_{r,o} &= M\frac{S_g - S_{ro}}{S_n - S_{gc} - S_{ro}}k_{r,n}(S_n) + (1-M)k_{r,o}(S_w, S_g),\\ k^{\xi}_{r,g} &= M\frac{S_g + S_s - S_{gc}}{S_n - S_{gc} - S_{ro}}k_{r,n} + (1-M)\frac{S_g}{S_g + S_s}k_{r,gt},\\ k^{\xi}_{r,s} &= M\frac{S_g + S_s - S_{gc}}{S_n - S_{gc} - S_{ro}}k_{r,n} + (1-M)\frac{S_s}{S_g + S_s}k_{r,gt}. \end{aligned} \tag{2.36}$$

Water-blocking effects are accounted for in the model by further scaling the normalized saturations by effective mobile saturations; see [Chase Jr et al., 1984] for further reference.

The solvent/oil capillary pressure is needed to be set to zero for the solvent to able to mix with oil [Todd et al., 1972]. The capillary pressure is interpolated between the fully miscible and fully immiscible conditions with the help of pressure miscibility function M_p:

$$p_{cog} = M_p p^m_{cog} + (1 - M_p)p^i_{cog}, \tag{2.37}$$

where

$$\begin{aligned} p^m_{cog} &= p_{cog}(S_g),\\ p^i_{cog} &= p_{cog}(S_g + S_s). \end{aligned} \tag{2.38}$$

Effective viscosities are modeled using the Todd-Longstaff mixing parameter ω:

$$\begin{aligned} \mu_{\xi,o} &= \mu_o^{1-\omega} \cdot \mu^{\omega}_{m,os},\\ \mu_{\xi,s} &= \mu_s^{1-\omega} \cdot \mu^{\omega}_{m,gos},\\ \mu_{\xi,g} &= \mu_g^{1-\omega} \cdot \mu^{\omega}_{m,gs}, \end{aligned} \tag{2.39}$$

where μ_o, μ_g, μ_s are the viscosity of the oil, gas, and solvent components respectively, $\mu_{m,os}$ is the fully mixed viscosity of oil and gas, $\mu_{m,gos}$ is the fully mixed viscosity of gas plus oil plus solvent and $\mu_{m,gs}$ is the fully mixed viscosity of gas and solvent. The fully mixed viscosities are defined by 1/4-power fluidity mixing rule [Todd et al., 1972, SLB, 2020]:

$$\mu_{m,os} = \frac{\mu_o\mu_s}{\left(\frac{S'_o}{S'_o + S'_s}\mu_s^{1/4} + \frac{S'_s}{S'_o + S'_s}\mu_o^{1/4}\right)^4}, \tag{2.40}$$

$$\mu_{m,gs} = \frac{\mu_g\mu_s}{\left(\frac{S'_g}{S'_g + S'_s}\mu_s^{1/4} + \frac{S'_s}{S'_g + S'_s}\mu_g^{1/4}\right)^4}, \tag{2.41}$$

$$\mu_{m,gos} = \frac{\mu_g\mu_o\mu_s}{\left(\frac{S'_o}{S'_n}\mu_s^{1/4}\mu_g^{1/4} + \frac{S'_s}{S'_n}\mu_o^{1/4}\mu_g^{1/4} + \frac{S'_g}{S'_n}\mu_s^{1/4}\mu_o^{1/4}\right)^4}, \tag{2.42}$$

where

$$\begin{aligned} S'_o &= S_g - S_{ro}, \\ S'_s &= S_s - S_{gc}, \\ S'_g &= S_g - S_{gc}, \\ S'_n &= S'_o + S'_g + S'_s \end{aligned} \tag{2.43}$$

and S_{ro} is the residual oil saturation, S_{gc} is a critical gas saturation.

After the effective viscosities are calculated, we can derive the effective densities under the assumption of ideal mixing as described in [Todd et al., 1972, SLB, 2020]. We start with the derivation of effective saturation fraction:

$$\left(\frac{S_g}{S_n}\right)_{\xi,o} = \frac{\mu_o^{1/4}\left(\mu_{\xi,o}^{1/4} - \mu_s^{1/4}\right)}{\mu_{\xi,o}^{1/4}\left(\mu_o^{1/4} - \mu_s^{1/4}\right)} \tag{2.44}$$

$$\left(\frac{S_g}{S_n}\right)_{\xi,g} = \frac{\mu_s^{1/4}\left(\mu_{\xi,g}^{1/4} - \mu_g^{1/4}\right)}{\mu_{\xi,g}^{1/4}\left(\mu_s^{1/4} - \mu_g^{1/4}\right)} \tag{2.45}$$

$$\left(\frac{S_s}{S_n}\right)_{\xi,s} = \frac{\mu_s^{1/4}\left(S_{fg}\mu_o^{1/4} + S_{fo}\mu_g^{1/4}\right) - \mu_o^{1/4}\mu_g^{1/4}\left(\frac{\mu_s^{1/4}}{\mu_{\xi,s}^{1/4}}\right)}{\mu_s^{1/4}\left(S_{fg}\mu_o^{1/4} + S_{fo}\mu_g^{1/4}\right) - \mu_o^{1/4}\mu_g^{1/4}}, \tag{2.46}$$

where $S_{fo} = \dfrac{S'_o}{S'_{og}}$, $S_{fg} = \dfrac{S'_g}{S'_{og}}$, $S'_{og} = S'_o + S'_g$.

The densities of the partially mixed fluids can be derived using the effective fractional saturations (2.44)-(2.45) as:

$$\rho_{\xi,o} = \rho_o \left(\frac{S_g}{S_n}\right)_{\xi,o} + \rho_g \left[1 - \left(\frac{S_g}{S_n}\right)_{\xi,o}\right], \tag{2.47}$$

$$\rho_{\xi,o} = \rho_o \left(\frac{S_g}{S_n}\right)_{\xi,g} + \rho_g \left[1 - \left(\frac{S_g}{S_n}\right)_{\xi,g}\right]. \tag{2.48}$$

$$\rho_{\xi,s} = \rho_s \left(\frac{S_s}{S_n}\right)_{\xi,s} + \rho_g S_{f,g} \left[1 - \left(\frac{S_s}{S_n}\right)_{\xi,s}\right] + \rho_o S_{f,o} \left[1 - \left(\frac{S_s}{S_n}\right)_{\xi,s}\right]. \tag{2.49}$$

For the fully mixed case ($\omega = 1$, $\mu_{\xi,o} = \mu_{\xi,g} = \mu_m$) the effective saturations are equal

$$\left(\frac{S_g}{S_n}\right)_{\xi,o} = \left(\frac{S_g}{S_n}\right)_{\xi,g} = \frac{\mu_o^{1/4}\left(\mu_m - \mu_g^{1/4}\right)}{\mu_m\left(\mu_o^{1/4} - \mu_g^{1/4}\right)}, \tag{2.50}$$

and the effective densities are equal to $\rho_{\xi,o} = \rho_{\xi,g} = \rho_m$:

In the fully unmixed (segregated) case, $\omega = 0$, which means that $\mu_{\xi,o} = \mu_o$ and $\mu_{\xi,g} = \mu_g$, and the effective franctional saturations are equal to:

$$\left(\frac{S_g}{S_n}\right)_{\xi,o} = 1, \quad \left(\frac{S_g}{S_n}\right)_{\xi,g} = 0, \quad \left(\frac{S_s}{S_n}\right)_{\xi,s} = 1. \tag{2.51}$$

The effective densities, in turn, are equal to the standard "pure" phase densities $\rho_{\xi,o} = \rho_o$, $\rho_{\xi,g} = \rho_g$, $\rho_{\xi,s} = \rho_s$.

Another special case is when $\mu_o = \mu_s$ or $\mu_g = \mu_s$, because in this case equations (2.45) and (2.44) are singular and instead the following equations should be used:

$$\begin{aligned} \rho_{\xi,o} &= (1-\omega)\rho_o + \omega\rho_m, \\ \rho_{\xi,g} &= (1-\omega)\rho_g + \omega\rho_m, \\ \rho_{\xi,s} &= (1-\omega)\rho_s + \omega\rho_m, \end{aligned} \tag{2.52}$$

where ρ_m is a fully mixed density:

$$\rho_m = \rho_o \frac{S_g}{S_n} + \rho_g \frac{S_g}{S_n} + \rho_s \frac{S_s}{S_n}. \tag{2.53}$$

The black oil model has been now extended to include the influence of solvent on all relevant quantities.

Chapter 3

Finite volume methods for conservation laws

Conservation laws are fundamental in science and engineering, describing the behavior of physical systems. Some of the most important phenomena in fluid dynamics, acoustics, electromagnetism, materials science, and many more disciplines can be modeled using conservation laws. However, finding numerical solutions to these equations can be challenging, and finite volume methods have emerged as a popular approach.

This chapter covers the fundamentals of the finite volume method for solving conservation laws. In the first section, we derive the numerical method formulation. In the second section, we introduce properties the numerical method must satisfy to be practically relevant. The later section provides examples of centered methods. The upwind method, commonly used in subsurface reservoir simulations, is introduced afterward. Finally, we discuss how high-resolution methods can be constructed and how the slope limiters help avoid oscillations in them.

3.1 General finite volume formulation for conservation laws

General conservation law

The rate of change of the conserved quantity u within a certain domain Ω is equal to the flow $\mathbf{f}$ through the boundary $\partial\Omega$ of the domain plus/minus any quantity generated/subtracted by the sources/sinks inside:

$$\frac{d}{dt}\int_{\Omega} u dV + \int_{\partial\Omega} \mathbf{f}\cdot\mathbf{n}\, dS = \int_{\Omega} Q\, dV. \tag{3.1}$$

In case of absence of sources or sinks in the domain ($Q = 0$), the quantity u stays conserved locally over time within the domain Ω, and the equation (3.1) is referred to as a **conservation law**.

This section's derivations closely follow the classic finite-volume methods book by Randall LeVeque [LeVeque, 2002].

The space domain is subdivided into grid elements, which are also referred to as finite or control volumes in two- and three-dimensional space. The shapes of these elements vary, from cubes in Cartesian grids over tetrahedrons in more unstructured grids to very general polygonal or polyhedral grids. The word "element" is used as it is agnostic to the grid's geometry. Later in this chapter, we will discuss the interface between the two elements, which in one-dimensional space is just a dot, in two-dimensional space - an edge, and in three-dimensional

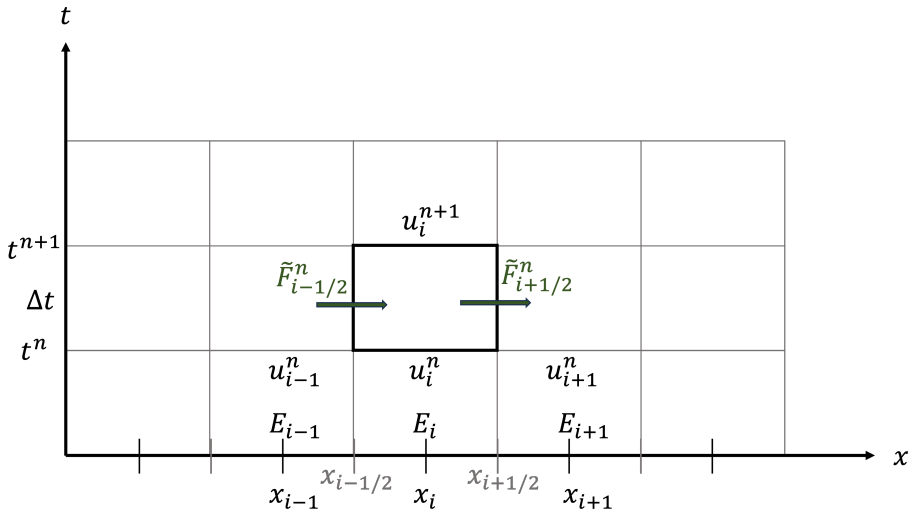

Figure 3.1: Illustration of the 1D cell E_i, its neighboring cells and the update from value u_i^n to u_{i+1}^n using the fluxes through the boundary according to formula (3.8).

space - a polygon. To accommodate all dimensions, we will use the word "interface" as a dimension-neutral word.

We will focus on a 1D domain to simplify the description. However, it's worth noting that the methods used in this section can be applied to any domain with a straightforward extension, including higher dimensions. The domain element is defined as E_i, as shown in Figure 3.1, which in 1D is simply a cell with its left and right points being $x_{i-1/2}$ and $x_{i+1/2}$ respectively, see Figure 3.1:

$$E_i = (x_{i-1/2},\, x_{i+1/2}). \tag{3.2}$$

The general conservation law in the integral form (3.1) in the absence of sources and sinks for a single cell $\Omega = E_i$ can be rewritten as:

$$\frac{d}{dt}\int\limits_{E_i} u dV = -\int\limits_{\partial E_i} \mathbf{f}\cdot\mathbf{n}\, dS. \tag{3.3}$$

For the 1D case, the integral over the boundary becomes a simple substitution:

$$\frac{d}{dt}\int\limits_{E_i} u(x,t)dx = \quad f(u(x_{i-1/2},t)) - f(u(x_{i+1/2},t)), \tag{3.4}$$

The above equation expresses again the conservation principle in 1D: the conserved quantity u between two given points $x_{i-1/2}$ and $x_{i+1/2}$ can only change due to the flow f through the endpoints. If we now integrate equation (3.4) over

the time interval (t^n, t^{n+1}), we get:

$$\int_{E_i} u(x,t^{n+1})dx - \int_{E_i} u(x,t^n)dx = \int_{t^n}^{t^{n+1}} f(u(x_{i-1/2},t))dt - \int_{t^n}^{t^{n+1}} f(u(x_{i+1/2},t)dt. \tag{3.5}$$

After rearranging the terms and dividing by the element size $\Delta x \equiv |E_i|$ we get:

$$\frac{1}{\Delta x}\int_{E_i} u(x,t^{n+1})dx = \frac{1}{\Delta x}\int_{E_i} u(x,t^n)dx - \frac{1}{\Delta x}\left[\int_{t^n}^{t^{n+1}} f(u(x_{i+1/2},t))dt - \int_{t^n}^{t^{n+1}} f(u(x_{i-1/2},t)dt\right]. \tag{3.6}$$

To derive the numerical method, let us introduce a discrete variable u_i^n, which approximates the average value of the unknown over the cell E_i at the time t^n:

$$u_i^n \approx \frac{1}{\Delta x}\int_{E_i} u(x,t^n)dx. \tag{3.7}$$

The formula (3.6) provides a way to update the cell average of u in a single timestep. However, the integrals on the right-hand side cannot be evaluated exactly in the general case, as in situations where u is non-smooth, it can have multiple values at the interface, rendering the exact evaluation of the integrals difficult. This motivates us to study numerical methods in the following form:

$$u^{n+1} = u_i^n - \frac{\Delta t}{\Delta x}(\tilde{F}_{i+1/2} - \tilde{F}_{i-1/2}), \tag{3.8}$$

where $\tilde{F}_{i+1/2}$ and $\tilde{F}_{i-1/2}$ are the approximations of the average flux trough the cell interface $x_{i+1/2}$ and $x_{i-1/2}$, respectively [LeVeque, 2002, Lie, 2019]:

$$\tilde{F}_{i\pm1/2} \approx \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} f(u(x_{i\pm1/2},t))dt. \tag{3.9}$$

The flux function is evaluated at the interface of each pair of cells, while the solution $u(x_{i\pm1/2},t)$, represented by the cell average (3.7), differs on each side of the interface; see Figure 3.1. However, we know that information in a hyperbolic problem propagates with a final speed, which is why we can assume that the flux through the interface depends on the neighboring cell averages:

$$\tilde{F}_{i-1/2} = \mathcal{F}(u_{i-1}^n, u_i^n). \tag{3.10}$$

Function $\mathcal{F}$ is called *numerical flux function*. The specific method is defined, among other things, by the choice of the numerical flux function $\mathcal{F}$. Equation (3.8) with the numerical flux function $\mathcal{F}$ forms a numerical method:

$$u^{n+1} = u_i^n - \frac{\Delta t}{\Delta x}(\mathcal{F}(u_i^n, u_{i+1}^n) - \mathcal{F}(u_{i-1}^n, u_i^n)). \tag{3.11}$$

The numerical method of a form (3.8) is *conservative* [Lie, 2019], meaning that if we sum up over any set of cells $\Omega_{IJ} = \{E_I, \ldots, E_J\}$ the flux differences will cancel out except for the outer boundary edges:

$$\Delta x \sum_{i=I}^{J} u_i^{n+1} = \Delta x \sum_{i=I}^{J} u_i^n - \frac{\Delta t}{\Delta x}\left(\mathcal{F}_{J+1/2} - \mathcal{F}_{I-1/2}\right). \tag{3.12}$$

It is an important feature of the finite volume method, which aligns with the physical properties of fluid flow. When fluid flows through the domain, whatever one cell loses, the other gains, and the change in the overall fluid quantity can happen only at the domain boundary.

3.2 Important properties of the numerical methods

The conservative numerical methods defined by the formula (3.11) vary based on the numerical flux function used. This section will introduce crucial properties of these methods, such as accuracy, stability, and convergence, which are essential for evaluating their reliability and usefulness in practice. Note that the theoretical background of numerical methods is extensive, and we recommend referring to classic literature such as [Kröner, 1997, LeVeque, 2002, Trangenstein, 2009] for a more comprehensive discussion.

3.2.1 Convergence of the numerical method

To be useful in practice, the numerical method should be **convergent**, meaning that the numerical solution $u^{N,\Delta x,\Delta t}(x,t)$ at the final step N approaches the exact solution $u(x,t)$ as the reference grid size Δx and time step Δt approach 0 [LeVeque, 2002]. However, knowing the exact solution is unusual in practical applications, so we rely instead on the method being **consistent** and **stable** as the fundamental theorem of numerical analysis states that consistency and stability of numerical methods implies convergence [LeVeque, 2002].

The numerical flux function $\mathcal{F}$ should approximate the integral in (3.9). If the function $u(x,t) \equiv \overline{u}$ is constant, the integral in (3.9) is reduced to $f(\overline{u})$, therefore, as part of the **consistency** condition, we require that for any $\overline{u}$ the following is true:

$$\mathcal{F}(\overline{u}, \overline{u}) = f(\overline{u}). \tag{3.13}$$

Typically, some requirement of Lipschitz **continuity** of the numerical flux function is made[LeVeque, 2002]. It means that there exists a real positive

constant L such that

$$\mathcal{F}(u_{i-1}^n, u_i^n) - f(\overline{u}) \leq L \max(|u_i^n - \overline{u}|, |u_{i-1}^n - \overline{u}|), \tag{3.14}$$

holds for all u_{i-1}^n, u_i^n.

3.2.2 Stability of the numerical method

For the numerical method to be stable and convergent as the grid is refined, the Courant-Friedrichs-Lewy (CFL) condition must be satisfied. It is important to note that it is *necessary* condition, but not *sufficient* [LeVeque, 2002].

The CFL condition was introduced in an early paper on finite-difference methods [Courant et al., 1928, Courant et al., 1967]. It guarantees that information travels at the physical rate, meaning that any information that may influence the solution at element E_i will have enough time to reach it. The condition can be expressed as:

$$\frac{\Delta t}{\Delta x} \max_j |\lambda_j| \leq 1, \tag{3.15}$$

Here, $\lambda_1 \leq \cdots \leq \lambda_n$ refer to the eigenvalues of the Jacobian matrix of the flux function f.

Another way to interpret (3.15) is that the domain of dependence of the exact solution should be within the domain of dependence for the discrete equation [LeVeque, 2002].

3.2.3 Order of the numerical method

Numerical methods can be classified based on their order, which represents their convergence rate to the true solution as the grid is refined. It is identified as the **order of accuracy** of the numerical method and indicates the speed at which the error diminishes as the cell size decreases.

For a method to be k**-order numerical method**, a norm of its error $||E||$ is expected to be bounded by the reference grid cell size Δx to the power k [LeVeque, 2002]:

$$\|u^{N,\Delta x,\Delta t} - u\| \equiv \|E\| \leq C(\Delta x)^k + \text{higher-order terms}, \tag{3.16}$$

given that $\Delta x \to 0$ and C is some constant that depends on the particular solution being computed and the time.

The order of convergence can be approximated numerically:

$$\text{EOC} = \frac{\log(e_{new}/e_{old})}{\log(\Delta x_{new}/\Delta x_{old})}, \tag{3.17}$$

where EOC stands for **Experimental Order of Convergence**. Here, e_{new} and e_{old} represent the L_1-norm of the error, i.e., the difference between the numerical and analytical solutions obtained on the grid with the reference cell sizes Δx_{new} and Δx_{old}, respectively. We use this measure to verify the implementation of the proposed methods.

3.2.4 Total variation diminishing numerical method

While first-order numerical methods are oscillation-free, it is important to control and minimize oscillations that occur when using second-order methods. A standard metric for measuring oscillations is total variation. The **total variation** [Trangenstein, 2009] of a function $u(x,t)$ for a given t is

$$TV(u) = \limsup_{\epsilon \to 0} \frac{1}{\epsilon} \int_{-\infty}^{\infty} |u(x+\epsilon,t) - u(x,t)| dx, \tag{3.18}$$

Note that for all continuously-differentiable functions $u \in \mathbb{C}^1(-\infty,\infty)$ the total variation is $TV(u) = \int_{-\infty}^{\infty} |u'(x)| dx$. The total variation of a piecewise-constant function w is defined as:

$$TV(w) = \sum_{\forall i} |w_{i+1} - w_i|, \tag{3.19}$$

where w_i are cell values of w.

When a numerical method introduces oscillations, the total variation of the discrete function is expected to increase over time. To prevent oscillations in hyperbolic equations, [Harten, 1997] suggests using a method that does not increase the total variation. A numerical method is said to be **total variation diminishing (TVD)** if the total variation (3.19) of the solution at any next time step u^{n+1} is not larger than the total variation of the solution on current time step u^n:

$$TV(u^{n+1}) \leq TV(u^n). \tag{3.20}$$

It was proved in [Harten, 1997] that a TVD method is monotonicity-preserving, meaning that if the initial data is monotone, the solution will be monotone as well. If, on the other hand, initial data contain discontinuity, it will be smeared out with time, but no oscillation will be introduced.

3.3 Centered methods

This section introduces several first- and second-order centered numerical methods. As the name suggests, these methods draw information from both sides of the cell interface, which means that these methods are agnostic to the type of problem we want to solve. The next section will discuss an alternative class of methods called "upwind," which only draws information from one side of the cell interface.

3.3.1 First-order centered methods

Unstable centered method

The most straightforward method for approximating the numerical flux is to calculate the arithmetic average of the flux values on the adjacent cells:

$$\mathcal{F}^n_{i\pm1/2} = \tfrac{1}{2}\left[f(u^n_{i\pm1}) + f(u^n_i)\right]. \tag{3.21}$$

The numerical method is then read as follows:

$$u^{n+1} = u^n_i - \frac{\Delta t}{2\Delta x}\left(f(u^n_{i+1}) - f(u^n_{i-1})\right). \tag{3.22}$$

Unfortunately, the method above is generally unstable for hyperbolic problems [LeVeque, 2002].

Lax-Friedrichs method

A common approach for stabilizing the method is to add an artificial diffusion term $\frac{\Delta x^2}{\Delta t}u_{xx}$ and, after applying the standard central difference, we get the following formula for the numerical flux:

$$\mathcal{F}^n_{i+1/2} = \frac{1}{2}\left[f(u^n_{i+1}) + f(u^n_i)\right] - \frac{\Delta x}{2\Delta t}(u^n_i - u^n_{i+1}). \tag{3.23}$$

The numerical method (3.11) is than written as:

$$u^{n+1} = \frac{1}{2}(u^n_{i+1} + u^n_{i-1}) - \frac{\Delta t}{2\Delta x}\left[f(u^n_{i+1}) - f(u^n_{i-1})\right]. \tag{3.24}$$

The method is named after Peter Lax and Kurt O. Friedrichs and is forward in time and centered in space. The method is formally first-order accurate and stable under the CFL condition, introduced in Section 3.2.2.

3.3.2 Second-order centered methods

Lax-Wendroff method

The accuracy can be improved by using the mid-point rule to evaluate the average flux through the cell interface (3.9) [Lie, 2019], with the mid-point values obtained by the Lax-Friedrichs method. The resulting method is a second-order, predictor-corrector method, which is called the Richtmyer two-step Lax–Wendroff method and is written as follows:

$$\begin{aligned}
u^{n+1/2}_{i+1/2} &= \frac{1}{2}(u^n_{i+1} + u^n_i) - \frac{\Delta t}{2\Delta x}\left[f(u^n_{i+1}) - f(u^n_i)\right], \\
u^{n+1/2}_{i-1/2} &= \frac{1}{2}(u^n_i + u^n_{i-1}) - \frac{\Delta t}{2\Delta x}\left[f(u^n_i) - f(u^n_{i-1})\right], \\
u^{n+1} &= u^n_i - \frac{\Delta t}{\Delta x}\left(f(u^{n+1/2}_{i+1/2}) - f(u^{n+1/2}_{i-1/2})\right).
\end{aligned} \tag{3.25}$$

The method is effective at accurately preserving the smooth parts of the solution. However, it tends to introduce oscillations near areas where discontinuities are present; see Figure 3.2.

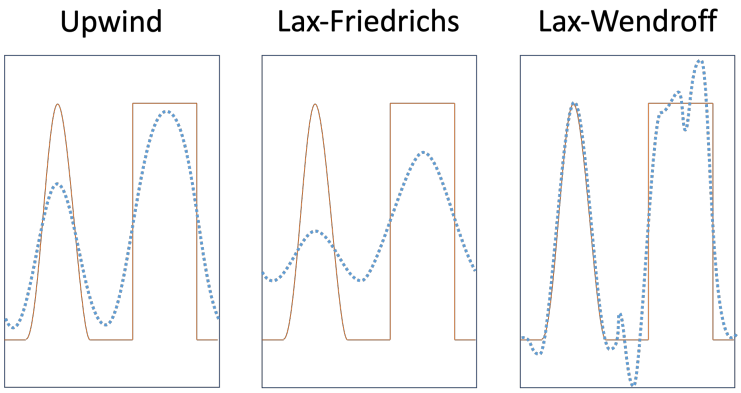

Figure 3.2: A visual comparison of three numerical methods' results on a simple test case. Adapted from [LeVeque, 2002].

Beam-Warming method

This second-order implicit method was introduced by Richard M. Beam and R. F. Warming in [Warming and Beam, 1976] and is defined as follows:

$$\begin{aligned} u_{i+1/2}^{n+1/2} &= u_i^n + \frac{1}{2}(u_i^n - u_{i-1}^n) - \frac{\Delta t}{2\Delta x}\left[f(u_i^n) - f(u_{i-1}^n)\right], \\ u_{i-1/2}^{n+1/2} &= u_{i-1}^n + \frac{1}{2}(u_{i-1}^n - u_{i-2}^n) - \frac{\Delta t}{2\Delta x}\left[f(u_{i-1}^n) - f(u_{i-2}^n)\right], \\ u^{n+1} &= u_i^n - \frac{\Delta t}{\Delta x}\left(f(u_{i+1/2}^{n+1/2}) - f(u_{i-1/2}^{n+1/2})\right). \end{aligned} \tag{3.26}$$

As one can see, the scheme has a very similar structure to the Lax–Wendroff scheme, but the evaluations are done on one side only.

3.4 The upwind method

The previously discussed methods are considered centered because they utilize information from both sides of the grid interface when computing the flux function. However, for partial differential equations that model a wave-like phenomenon (or hyperbolic type PDE), we can anticipate how the solution behaves across the interface. The quantity is propagated along the direction of the flow. This idea has led to the development of the **upwind scheme**. A detailed description and derivation of the upwind method can be found in many books about numerical methods, including [LeVeque, 2002].

The upwind method only uses a one-sided difference in the "upwind" direction or, in other words, aligned with the propagation of waves/information in the domain. For a case of a conservation law, the value $f'(u)$ will determine the direction of the wave propagation. The upwind scheme, in this case, is written

as follows:

$$\begin{aligned} u_i^{n+1} &= u_i^n - \frac{\Delta t}{\Delta x}[f(u_i^n) - f(u_{i-1}^n)], \quad \text{if } f'(u) > 0, \\ u_i^{n+1} &= u_i^n - \frac{\Delta t}{\Delta x}[f(u_{i+1}^n) - f(u_i^n)], \quad \text{if } f'(u) < 0. \end{aligned} \tag{3.27}$$

See Figure 3.2 for the illustration of the method's behavior on a simple test case and a comparison of its results with the two methods presented before, Lax-Friedrichs and Lax-Wendroff.

The upwind method is widely used in reservoir modeling and is considered the standard method in many reservoir simulators, including OPM Flow [Baxendale, 2023, Rasmussen et al., 2021]. In OPM Flow, the upwind direction of fluid flow is determined by the pressure difference between the adjacent cells near the interface, modified by the gravity component, which naturally agrees with the physical model of fluid flow in the reservoir.

The second-order finite volume methods discussed in this work are built on top of the upwind method. As the upwind method is standard in OPM Flow, this choice eases the implementation and makes it least invasive to the code base.

3.5 High-resolution methods

Second-order methods, like Lax-Wendroff or Beam-Warming, provide better accuracy on smooth parts of the solution than the upwind method. However, such methods generate oscillations near discontinuities.

Numerical methods that are at least second-order accurate on smooth parts of the solution and resolve discontinuities with high accuracy and no spurious oscillations are called **high-resolution methods** [LeVeque, 2002, Mykkeltvedt, 2014]. High-resolution methods strive to provide second-order accuracy or higher whenever possible and relax this condition near a discontinuity by introducing some amount of numerical dissipation to avoid oscillations.

3.5.1 Flux-limiter methods

The numerical flux of such methods can be represented as a combination of high-order and low-order fluxes [Lie, 2015]:

$$\mathcal{F}_{i+1/2}^n = \mathcal{F}_{i+1/2}^{low} + \psi_i \left(\mathcal{F}_{i+1/2}^{high} - \mathcal{F}_{i+1/2}^{low} \right), \tag{3.28}$$

where low-order flux can be, for example, Lax-Friedrichs flux (3.23) or upwind (3.27), and Lax-Wendroff (3.25) can be used as the high-order flux. The *flux limiter* function ψ_i represents a switch between high-resolution methods ($\psi_i \approx 1$) on smooth solution regions and a low-resolution method ($\psi_i \approx 0$) on areas with discontinuities.

3.5.2 Slope-limiter methods

In order to introduce slope-limiter methods, let us first introduce the REA (reconstruct-evolve-average) algorithm [LeVeque, 2002]:

1. **Reconstruct** a piecewise polynomial function $\tilde{u}(x, t^n)$ for all x from the known cell averages u_i^n. If the reconstruction is constant, meaning that the function $\tilde{u}(x, t^n)$ is a piecewise constant function that takes the cell average value u_i^n in each element E_i, we get a first-order scheme. Linear reconstruction gives a second-order method, quadratic gives a third-order, and so on.

2. **Evolve** the hyperbolic equation, exactly or approximately, with this initial data $\tilde{u}(x, t^n)$ to obtain $\tilde{u}(x, t^{n+1})$ at the next time $t^{n+1} = t^n + \Delta t$.

3. **Average** the obtained $\tilde{u}(x, t^{n+1})$ function over each element of the grid to get the new cell averages u_{i+1}^n.

Solving the hyperbolic equation in Step 2 is essential to implement the algorithm above. This can be achieved by utilizing the theory of Riemann problems, as the initial data is a piecewise polynomial function $\tilde{u}(x, t^n)$. You can find more comprehensive coverage on Riemann problems in the book authored by LeVeque [LeVeque, 2002]. Moreover, utilizing this approach for the advection equation will result in the upwind algorithm.

As for the third step in the algorithm above, here are two fundamentally different approaches for the averaging. For upwind methods ($x = x_i$), the temporal integrals in (3.5) are evaluated at points $x = x_{i\pm1/2}$, where $\tilde{u}(x, t)$ is discontinuous. Therefore, standard integration and extrapolation techniques cannot be applied, and resolving the wave structures that emerge due to the discontinuity is necessary. This is usually accomplished by solving a Riemann problem or similar generalizations [Lie, 2015]. In contrast, central methods ($x = x_{i+1/2}$) use sliding averages that are computed over a staggered grid cell $[x_i, x_{i+1}]$. If a CFL condition of one-half is maintained, the integrand remains smooth, allowing for the application of standard integration and extrapolation techniques.

The evolving and averaging steps cannot increase the total variation, therefore, if the reconstruction step does not increase the total variation (3.20), the resulting method would be total variation diminishing (TVD) [LeVeque, 2002], see Section 3.2.4 for the definition. If we use a linear reconstruction in the form:

$$\tilde{u}(x, t^n) = u_i^n + \psi(x - x_i), \quad x_{i-1/2} \le x \le x_{i+1/2}, \tag{3.29}$$

where x_i is the center of the element E_i ($x_i = \frac{1}{2}(x_{i-1/2} + x_{i+1/2})$ in 1D case), and ψ is the slope on the E_i [LeVeque, 2002]. Note that the linear function defined in (3.29), is equivalent to u_i^n in the center x_i. Moreover, the cell average is also equivalent to u_i^n no matter the slope ψ. This property is crucial in creating conservative methods for conservation laws. And, since the step 2 and 3 in

the REA algorithm above are conservative in general, the resulting method is conservative [LeVeque, 2002].

To obtain a second-order numerical method, we want to choose the slope ψ in a way that provides second-order accuracy on the smooth parts of the solution and does not introduce oscillations near discontinuities. Therefore, ψ is a possibly non-linear function that considers the solution's behavior near the cell E_i. For example, when the solution is smooth, the Lax-Wendroff slope is suitable for ψ. However, near a discontinuity, a smaller magnitude value must be used to limit the slope and avoid oscillations. This approach is known as the **slope-limiter method**, which was first introduced in [Van Leer, 1974, Van Leer, 1977, Van Leer, 1979].

The limiter $\psi(r_{E_i})$ is defined as a function of a ratio of successive gradients [Van Leer, 1974, Sweby, 1984]:

$$\psi(r_{E_i}) = \psi\left(\frac{u_i^n - u_{i-1}^n}{u_{i+1}^n - u_i^n}\right). \tag{3.30}$$

[Sweby, 1984] performed a graphical analysis of the limiter functions and presented a visualization for the second-order TVD region, where the limiter function must fall for the method to be second-order TVD, see the shaded area on Figure 3.3. The second-order region is bounded by the presented earlier second-order Lax-Wendroff method (3.25), $\psi^{LW}(r) = 1$, and Warming and Beam method (3.26), $\psi^{WB}(r) = r$. The TVD region is defined by the following inequality [LeVeque, 2002]:

$$0 \leq \psi(r) \leq \text{minmod}(2, 2r). \tag{3.31}$$

The second-order TVD region is, therefore, an intersection of the TVD and the second-order regions. Its lower boundary is equivalent to the minmod limiter, and it's upper to the Superbee limiter, both presented below.

The minmod limiter

The lower line of the second-order TVD region from Figure 3.3 represents the *minmod limiter* [LeVeque, 2002, Sweby, 1984]. It is a simple yet robust limiter that is commonly used and is defined as:

$$\psi_{\text{minmod}}(r) := \text{minmod}(1, r) = \begin{cases} 1 & \text{if } |r| > 1 \text{ and } r > 0, \\ r & \text{if } |r| < 1 \text{ and } r > 0, \\ 0 & \text{if } r \leq 0. \end{cases} \tag{3.32}$$

The superbee limiter

The least restrictive limiter is the *superbee limiter* [Roe, 1985]. On the Figure 3.3 it is on the top of the second-order TVD region and it is defined as follows:

$$\psi_{\text{superbee}}(r) = max(0, min(1, 2r), min(2, r)). \tag{3.33}$$

This limiter lies on the upper bound of the second-order TVD region and generates sharp gradients.

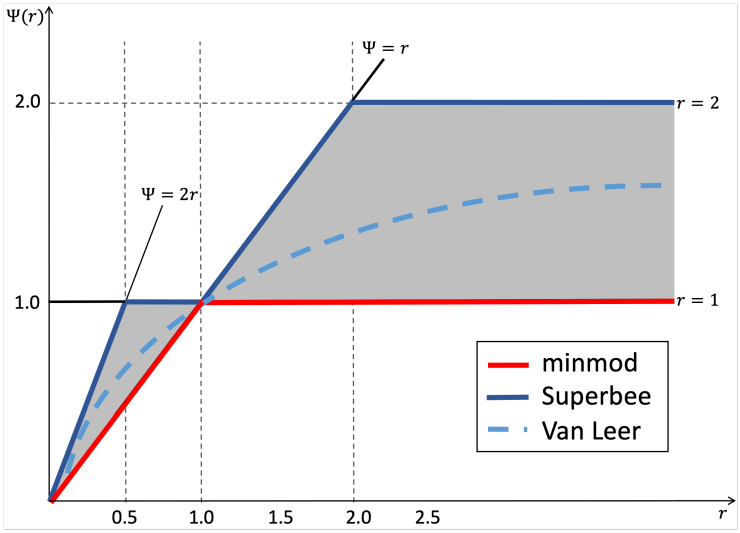

Figure 3.3: Second-order TVD region and three limiters: minmod, superBee, and van Leer.

The van Leer limiter

The two limiters presented above are piece-wise constant limiters. A smooth limiter was introduced by van Leer in [Van Leer, 1974] and is defined as:

$$\psi_{\text{vanLeer}} = \frac{r + |r|}{1 + |r|}. \tag{3.34}$$

The literature contains various limiters that can be used to construct higher-order schemes [Van Leer, 1974, Sweby, 1984, Harten, 1997, LeVeque, 2002].

Chapter 4

Second-order finite volume methods for reservoir simulation

In the previous chapter, we introduced the finite volume method concept in 1D. Reservoir simulation requires applying these methods to the black-oil problems in multi-dimensions. We first introduce the generalization of the second-order FV method on more complex grids. Then, we present three second-order finite-volume methods focusing on reservoir modeling applications.

4.1 General second-order finite volume method for black-oil model

In this section, we present a numerical scheme for the cell-centered finite volume method, as detailed in Section 3.1, and apply it to the black-oil model equations from Chapter 2. Our derivations will utilize an integral formulation of the black-oil model equations, as derived in Section 2.5. To simplify the discussion, we will focus solely on the water component:

$$\frac{\partial}{\partial t}\left(\frac{\phi S_w}{B_w}\right) - \boldsymbol{\nabla}\cdot\frac{\lambda_w k}{B_w}\left(\boldsymbol{\nabla} p_w - \rho_w \mathbf{g}\right) = Q_W. \tag{4.1}$$

It is worth noting that extending the derivations to other components is a straightforward process.

Writing the equation above for each element E_i of the domain and applying the implicit Euler method for the time discretization, we get:

$$\left(\frac{\phi S_w}{B_w}\right)^{n+1} = \left(\frac{\phi S_w^n}{B_w}\right)^{n} + \frac{dt}{|E_i|}\oint_{\partial E_i}\frac{\lambda_w^{n+1} k}{B_w}\left(\boldsymbol{\nabla} p_w^{n+1} - \rho_w \mathbf{g}\right)\cdot\mathbf{n}dx + Q_W^{n+1}, \tag{4.2}$$

where n and $n+1$ denote the current and the next time steps, respectively, Q_W^{n+1} is the time-averaged integral over the element of the right-hand side. We assume a no-flow boundary condition as reservoirs generally have an impermeable boundary with the surrounding rocks.

The first two terms, which are linear, can be computed easily by using averaged element values. While the right-hand side integral can be approximated through various quadrature rules, we are using the midpoint rule:

$$\oint_{\partial E_i}\frac{\lambda_w^{n+1} k}{B_w}\left(\boldsymbol{\nabla} p_w^{n+1} - \rho_w \mathbf{g}\right)\cdot\mathbf{n}dx \approx \sum_{e_{ij}\in\partial E_i}|e_{ij}|\frac{\lambda_w^{n+1}(x_{ij})k(x_{ij})}{B_w}\left(\boldsymbol{\nabla} p_w^{n+1}(x_{ij}) - \rho_w(x_{ij})\mathbf{g}(x_{ij})\right)\cdot\mathbf{n}_{ij}, \tag{4.3}$$

where e_{ij} is the segment of the element's boundary ∂E_i between the current element E_i and its neighbour E_j, $\mathbf{n}_{ij}$ is the outer normal to the said segment, and x_{ij} is the midpoint at which the integral is evaluated. The standard two-point flux approximation [Aavatsmark, 2002, Lie, 2019, Alyaev et al., 2014] is used to determine the pressure gradient, while the arithmetic mean is used to estimate the gravitational acceleration.

Let us now focus on the computation of the mobility function $\lambda_w^{n+1}(x_{ij})$. For conciseness, we will use the following notation:

$$\lambda_w^{n+1}(x_{ij}) = \lambda_w^{ij}. \tag{4.4}$$

The most common approach to compute phase mobility in reservoir simulation is the upwind method, which was discussed in Section 3.4:

$$\lambda_w^{ij} = \begin{cases} \lambda_{ij}^-, & \text{if } \left(\boldsymbol{\nabla} p_w^{n+1} - \rho_w \mathbf{g}\right) \cdot \mathbf{n} \geq 0, \\ \lambda_{ij}^+, & \text{otherwise,} \end{cases} \tag{4.5}$$

where λ_{ij}^- is the mobility function on the element E_i and λ_{ij}^+ - on the neighboring element E_j. The boundary condition value is used instead of the neighboring value for the boundary elements. In the typical case of the "no-flow" boundary condition, the interface is not considered.

For second-order methods, the upstream and downstream values of the phase mobility function are calculated differently, meaning that the upwind scheme gets values L_{ij}^- and L_{ij}^+ from the reconstructed linear function for every element instead of λ_{ij}^- and λ_{ij}^+. The linear reconstruction function has to satisfy the following requirements:

$$\begin{aligned} L_{E_i}(x) :&= \lambda_{E_i} + \nabla L_{E_i} \cdot (x - \mathbf{w}_{E_i}), \\ L_{E_i}(\mathbf{w}_{E_j}) &= \lambda_{E_j}, \forall (E_i, E_j) \in \partial E_i, \end{aligned} \tag{4.6}$$

where $\mathbf{w}_{E_i}$ is the barycenter coordinates of the element E_i, and ∇L_{E_i} is the gradient that needs to be computed. Figure 4.1 shows an illustration of how linear reconstructions can be built in a simple 1D case. The blue color refers to the values used in the first-order method, while in the orange color, we see the local reconstruction function L_{E_i} for the element and its neighbors.

The second-order methods differ in how the linear reconstruction and the gradient ∇L_{E_i} are computed. In the next sections, we will discuss in detail three different approaches to reconstructing the function L_{E_i}. Since all derivations must be applied to all phases, we will drop the water phase index.

4.2 Least squares reconstruction

We want to utilize information from all the neighboring cells, see Figure 4.2. Therefore, we want our reconstructed function L_{E_i} on the element E_i to go through all neighboring cell-averaged values, aka satisfy:

$$L_{E_i}(\mathbf{w}_{E_j}) = \lambda_{E_j}, \forall (E_i, E_j) \in \partial E_i, \tag{4.7}$$

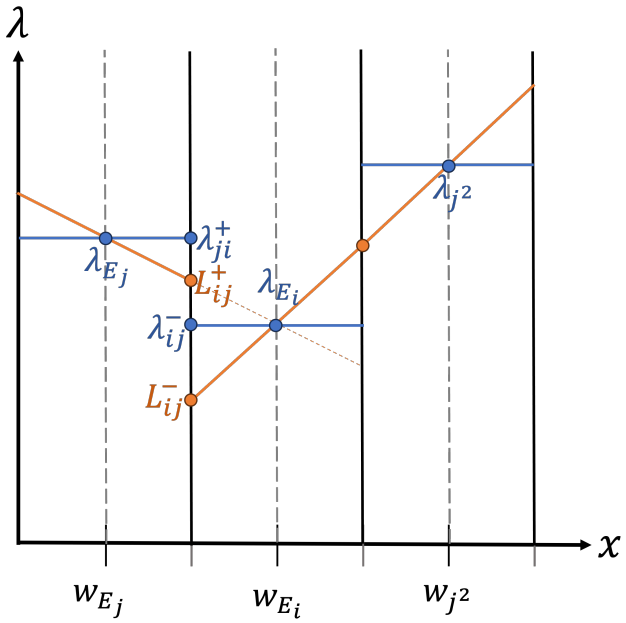

Figure 4.1: An example of the second-order reconstructions $L_{E_i}(x)$ for the one-dimensional grid cell E_i.

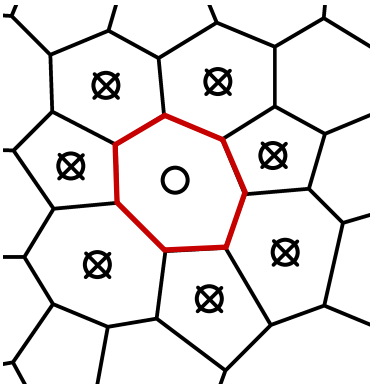

Figure 4.2: The stencil for the least-square method. The least-square method simultaneously utilizes the information from all the neighbors to find the resulting reconstruction.

which is equivalent to:

$$(\mathbf{w}_{E_j} - \mathbf{w}_{E_i})\nabla L_{E_i} = (\lambda_{E_j} - \lambda_{E_i}),\ \forall (E_i, E_j) \in \partial E_i. \tag{4.8}$$

However, since the number of neighbors is much larger than the space dimension, the system (4.8) is an over-determined linear system and can only be solved in the least squares sense [Trefethen and Bau, 1997, Feistauer et al., 2003].

Let us define

$$\begin{aligned} \boldsymbol{A}_{E_i} &:= (\mathbf{w}_{E_j} - \mathbf{w}_{E_i})_{i=1\ldots N_{E_i}} \in \mathbb{R}^{N_{E_i} \times \mathcal{N}}, \\ \boldsymbol{b}_{E_i} &:= (\lambda_{E_j} - \lambda_{E_i})_{i=1\ldots N_{E_i}} \in \mathbb{R}^{N_{E_i}}. \end{aligned} \tag{4.9}$$

The system (4.8) can be written as

$$\boldsymbol{A}_{E_i} \nabla L_{E_i} = \boldsymbol{b}_{E_i}, \tag{4.10}$$

And the least squares solution of the above system can be computed by solving the following equation:

$$(\boldsymbol{A}_{E_i}^T \boldsymbol{A}_{E_i}) \nabla L_{E_i} = \boldsymbol{A}_{E_i}^T \boldsymbol{b}_{E_i}, \tag{4.11}$$

Note that the linear reconstruction near a no-flow boundary requires special treatment. We need to ensure that the directional derivative of the reconstructed saturation normal to the boundary is zero. A ghost cell behind the no-flow boundary that mirrors the opposite neighbor can be used as suggested in [Mykkeltvedt et al., 2017].

As it was stated before, a gradient obtained from solving (4.11) should be limited in order to avoid spurious oscillations. One can apply any slope limiter to the method, such as the ones described in Section 3.5.2. Paper A studied the performance of the presented above numerical method combined with the slope limiters described in Section 3.5.2. A brief overview of the study can be found in Section 6.1.

Later on, we will refer to the described method as the second-order LS method.

4.3 Selective linear reconstruction

In order to construct a linear function in $\mathcal{N}$-dimensional space, $\mathcal{N}+1$ points are needed. Therefore, another possible approach to find L_{E_i} is to compute all possible linear functions using the current element's value and $\mathcal{N}$ neighboring values. Then, after possible limiting and checking that the approximate solution is physical, choose one of the reconstructed functions for later computations, for example, the one with the steepest gradient. This approach was introduced in [Durlofsky et al., 1992] for 2D triangular grids. A similar approach but for the Discontinuous Galerkin method was used in [Dedner and Klöfkorn, 2011], where it was generalized for two- and three-dimensional non-conforming grids.

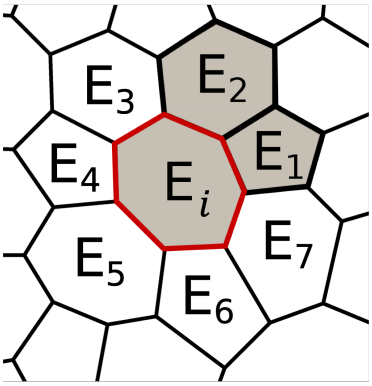

Figure 4.3: The stencil for the selective reconstruction method. The selective reconstruction method computes a set of all possible linear reconstructions based on the value in the element and $d = \mathcal{N} - 1$ neighbors; in the figure, you can see an example of one reconstruction that uses the values in neighboring element 1 and 2.

To introduce this method, we will use the following notation. Let us consider an element E_i. We define a set of all neighboring elements of E_i as $\mathcal{Y}(E_i)$, which is the collection of all elements that share a boundary with E_i. A $\mathcal{N}$-combination of the set $\mathcal{Y}(E_i)$ is denoted as $\mathcal{J}_{E_i}$ and presents a complete set of all possible neighbor combinations that can be used for the reconstruction. Figure 4.3 shows a stencil of the element E_i and $\mathcal{N} = 2$ chosen neighboring elements $j = \{1, 2\}$ that are used for the linear reconstruction. The size of the set $\mathcal{J}_{E_i}$ depends on the number of neighbors and can be calculated as $\frac{N_{E_i}!}{\mathcal{N}!(N_{E_i} - \mathcal{N})!}$. Therefore, as the number of neighboring elements increases, the size of $\mathcal{J}_e$ also increases, making computation time-consuming, especially in complex multi-dimensional grids where each element has many neighbors.

The algorithm for computing L_{E_i} then goes as follows:

1. Compute a complete set of all possible linear reconstructions L_E^j for each possible combination of $\mathcal{N} - 1$ neighbours $j = E_1, \ldots, E_{\mathcal{N}-1}$
2. Out of the above set of all possible functions, we compute a set of admissible linear functions. It means that limiter is applied and only functions that satisfy the physicality condition are allowed in this set.
3. Out of the set of admissible linear functions, the one with the steepest gradient is chosen.

In the algorithm above, the term "physicality condition" refers to a set of criteria based on the physics of the problem. For example, when dealing with saturation, the physicality condition requires that the calculated values remain within the physically possible range from 0 to 1.

On the first step, for each j a linear $\mathcal{N} \times \mathcal{N}$ system is solved:

$$\boldsymbol{A}_j \nabla L^j_{E_i} = \boldsymbol{b}_j, \quad \forall j \in \mathcal{J}_{E_i}, \tag{4.12}$$

where

$$\begin{aligned} \boldsymbol{A}_j &:= (\mathbf{w}_{E_j} - \mathbf{w}_{E_i}) \in \mathbb{R}^{\mathcal{N} \times \mathcal{N}}, \\ \boldsymbol{b}_j &:= (\lambda_{E_j} - \lambda_{E_i}) \in \mathbb{R}^{\mathcal{N}}. \end{aligned} \tag{4.13}$$

It should be noted here that the inverse of the matrix $\boldsymbol{A}_j$ can be easily computed using Cramer's rule because its size only depends on the spatial dimension $\mathcal{N}$. However, special treatment is needed for the system in the case of a Cartesian grid, as the matrix becomes singular when the considered barycenters align. The special treatment is explained in [Dedner and Klöfkorn, 2011]. The idea is to add one more neighbor to the system (4.12) that is not already a member of j and solve the new system using a least squares method. Since matrix $\boldsymbol{A}_j$ only depends on the grid's geometry, this information needs to be calculated only once.

The limiting is also done slightly differently here compared to the least squares second-order method 4.2. In this scheme, only neighbors who are not part of the reconstruction are used for limiting. For example, if in Figure 4.3 neighbours $j = 1, 2$ are used for the reconstruction than $\tilde{j} := \{3, 4, 5, 6, 7\}$ are considered in limiter. We select the one with the steepest gradient from the set of admissible limited linear reconstructions.

Similar to the previous numerical method, any slope limiter can be applied to this method as well. In Paper A, Section 6.1, the performance of the method combined with the slope limiters from Section 3.5.2 was studied.

4.4 Linear programming reconstruction

In the two second-order methods presented above, the final gradient was computed in two steps: first, it was initially computed using either least squares (4.2) or selective linear reconstruction (4.3), and second, a limiter (for example, from Section 3.5.2) was applied. [May and Berger, 2013] proposed to use linear programming instead of the limiter. They were still computing a gradient ∇L first, but after the "limited" gradient $\nabla \tilde{L}$ was obtained by solving a small linear programming problem, which minimized the difference $\nabla L - \nabla \tilde{L}$ under the monotonicity conditions. Each gradient component was limited by a separate scalar, which means that the resulting gradient could be of a different length and direction. Later, [Chen and Li, 2016] noticed this and proposed a new linear programming scheme that did not require an initial gradient computation. It is important to note here that the two schemes give different optimal solutions. However, [Chen and Li, 2016] proves that the obtained gradient is sufficiently close to the unlimited least squares gradient. This approach will be described below.

Linear programming is a mathematical modeling technique that can find an extremum of a linear objective function under certain linear constraints [Nocedal

and Wright, 2006]. In our case, we want to minimize the total gaps between the reconstructed values and the cell-averaged values at all neighboring cells:

$$\delta(L) := \sum_{\forall(E_i,E_j)\in\partial E_i} |\lambda_{E_j} - L_{E_i}(\mathbf{w}_{E_j})|. \tag{4.14}$$

The constraints are the following monotonicity conditions:

$$\min\{\lambda_{E_i}, \lambda_{E_j}\} \leq L_{E_i}(\mathbf{w}_{E_j}) \leq \max\{\lambda_{E_i}, \lambda_{E_j}\}, \forall(E_i, E_j) \in \partial E_i. \tag{4.15}$$

The monotonicity condition can be stated as follows: when the reconstructed function L_{E_i} on an element E_i is evaluated on each neighboring element E_j, the resulting values should be bounded by the cell-averaged values on both E_i and E_j. This constraint is equivalent to the minmod limiter on a one-dimensional domain, as shown in [Chen and Li, 2016].

The equation (4.14) is equal to

$$\begin{aligned}
&\sum_{\forall(E_i,E_j)\in\partial E_i} |\lambda_{E_j} - L_{E_i}(\mathbf{w}_{E_j})| \\
&\qquad = \sum_{\forall(E_i,E_j)\in\partial E_i} |\lambda_{E_j} - \lambda_{E_i} - \nabla L_{E_i} \cdot (\mathbf{w}_{E_j} - \mathbf{w}_{E_i})| \\
&\qquad = \sum_{\forall(E_i,E_j)\in\partial E_i} |v_{E_j} - \tilde{v}_{E_j}|,
\end{aligned} \tag{4.16}$$

where v and $\tilde{v}$ are defined as

$$v_{E_j} := \lambda_{E_j} - \lambda_{E_i} \quad \text{and} \quad \tilde{v}_{E_j} = \nabla L_{E_i} \cdot (\mathbf{w}_{E_j} - \mathbf{w}_{E_i}). \tag{4.17}$$

Under the monotonicity constraints (4.15), the difference $v_{E_j} - \tilde{v}_{E_j}$ has the same sign as v_{E_j}. This observation allows us to go from the non-linear formulation (4.16) to linear

$$\begin{aligned}
&\max \sum_{\forall(E_i,E_j)\in\partial E_i} \operatorname{sgn}(v_{E_j})(\mathbf{w}_{E_j} - \mathbf{w}_{E_i}) \cdot \nabla L_{E_i} \\
&\text{subject to } v^-_{E_j} \leq (\mathbf{w}_{E_j} - \mathbf{w}_{E_i}) \cdot \nabla L_{E_i} \leq v^+_{E_j},
\end{aligned} \tag{4.18}$$

where

$$\begin{aligned}
v^-_{E_j} &= 0 \\
v^+_{E_j} &= |\lambda_{E_i} - \lambda_{E_j}|.
\end{aligned} \tag{4.19}$$

The linear programming problem takes the following form:

$$\max c^T \mathbf{x} \text{ subject to } \mathbb{A}\mathbf{x} \leq b, \tag{4.20}$$

where

$$c = \begin{bmatrix} \sum_{\forall(E_i,E_j)\in\partial E_i} \operatorname{sgn}(v_{E_j})(x_{E_j} - x_{E_i}) \\ \sum_{\forall(E_i,E_j)\in\partial E_i} \operatorname{sgn}(v_{E_j})(y_{E_j} - y_{E_i}) \\ \sum_{\forall(E_i,E_j)\in\partial E_i} \operatorname{sgn}(v_{E_j})(z_{E_j} - z_{E_i}) \end{bmatrix}, \tag{4.21}$$

$$\mathbb{A} = \begin{bmatrix} x_{E_1} - x_{E_i} & y_{E_1} - y_{E_i} & z_{E_1} - z_{E_i} \\ \vdots & \vdots & \\ x_{E_\eta} - x_{E_i} & y_{E_\eta} - y_{E_i} & z_{E_\eta} - z_{E_i} \\ -(x_{E_1} - x_{E_i}) & -(y_{E_1} - y_{E_i}) & -(z_{E_1} - z_{E_i}) \\ \vdots & \vdots & \\ -(x_{E_\eta} - x_{E_i}) & -(y_{E_\eta} - y_{E_i}) & -(z_{E_\eta} - z_{E_i}) \end{bmatrix}, \quad b = \begin{bmatrix} v^+_{E_1} \\ \vdots \\ v^+_\eta \\ -v^-_{E_1} \\ \vdots \\ -v^-_\eta \end{bmatrix}. \tag{4.22}$$

In the equations above, η is the number of neighbouring elements E_j of the element E_i. The unknown vector $\mathbf{x}$ is the transposed gradient vector of the linear reconstruction $\mathbf{x} = [\nabla L^x_{E_i}, \nabla L^y_{E_i}, \nabla L^z_{E_i}]^T$.

An admissible solution to our LP problem is the zero gradient, given by $\mathbf{x} = [0, 0, 0]^T$, which corresponds to a first-order scheme. Therefore, we initialize the LP with this solution. The LP starts with $\mathcal{N}$ active constraints, and similar to previous works [May and Berger, 2013, Chen and Li, 2016], we adopt an all-inequality simplex method to solve the LP. This method can be applied if all constraints are inequalities, which is the case here. Furthermore, the all-inequality simplex method is much faster than the standard simplex method when the number of constraints exceeds the number of unknowns, which again is the case here. We refer the interested reader to [May and Berger, 2013] and Section 5.8 for a more detailed explanation of the all-inequality simplex algorithm. Later on, we will refer to this method as the second-order LP method.

Chapter 5

Implementation in OPM

This chapter covers the implementation details of second-order methods integration in the reservoir simulator OPM Flow (Open Porous Media Flow) opm-project.org.

The chapter consists of two parts. The first part serves as an introduction to OPM Flow (Section 5.1) and provides build instructions for the OPM Flow with the second-order methods option (Section 5.2).

The second part describes the implementation details of the second-order method within a reservoir modeling software infrastructure and is intended to help possible reimplementations in other simulators. The second-order implementation has minimal impact on the existing code since it requires modifications in only a few specific places.

Starting with the high-level overview of a reservoir simulator in Section 5.3, we will go through the necessary changes to enable the second-order method to function effectively in Sections 5.4 - 5.6. In Sections 5.7 - 5.8 we present specifics of the implementation of methods with least-squares reconstruction and linear programming reconstruction, respectively. These two methods had the best performance on smaller problems, according to Paper A (see Section 6.1), and therefore were chosen for the implementation in OPM Flow.

5.1 The Open Porous Media Initiative

The OPM initiative encourages open innovation and reproducible research in the field of modeling flow in porous media. It started more than ten years ago as a collaboration between groups at Equinor (formerly Statoil), SINTEF, the University of Stuttgart, and the University of Bergen. The community grew over time as more groups and individuals joined. It mainly has been developed by SINTEF, NORCE, Equinor ASA, Ceetron Solutions and OPM-OP [Rasmussen et al., 2021].

Open-source projects, such as OPM, offer transparency and accessibility that attract both individual developers and companies. This accessibility allows interested parties to inspect code, report bugs, provide constructive feedback, suggest new features, and contribute directly to the project's development, consequently building a community around the project. In OPM, the development is done through reviewed GitHub pull requests, ensuring the resulting software's quality. Only maintainers are allowed to merge new pieces of code into the main branch.

OPM is a suite of packages designed for reservoir simulation, comprising various tools and modules. It provides a powerful open-source tool for reservoir simulation called OPM Flow, which is used in both industry and research commu-

nities. OPM includes other packages, such as upscaling tools and experimental modules.

Our implementation extends the 2020.04 release of OPM Flow and is available on GitHub. OPM Flow 2020.04 consists of five modules: `opm-common`, `opm-material`, `opm-grid`, `opm-models`, `opm-simulators`.These modules are available under the standard open-source license **GNU General Public License v3.0** at the GitHub OPM page https://github.com/OPM. There are also two data modules – `opm-data` and `opm-tests` for benchmarking and testing. While the modules are separate, they depend on each other, and as the OPM project is an actively developing software, their connections may change over time. The structure of the modules and their interdependences are shown in Figure 5.1. The `opm-common` module, which does not rely on any DUNE modules, can be built separately. Other modules depend on one or more DUNE modules, as the OPM Flow simulator is dependent on `dune-common`, `dune-geometry`, `dune-grid`, and `dune-istl`.

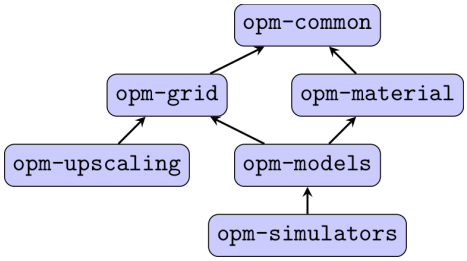

Figure 5.1: OPM modules structure as of 2020.04 release. Adopted from OPM project website https://opm-project.org/ (GPL3.0).

For 3D visualization and post-processing of the reservoir simulations, we use **ResInsight** https://github.com/OPM/ResInsight developed by Ceetron Solutions in collaboration with Equinor ASA. It is a powerful, open-source, 3D visualization and post-processing tool for data in the standard output file format from the reservoir simulation. Most of the resulting figures presented in this thesis are done with the help of either ResInsight or ParaView, which is also an open-source visualization tool suitable for .vtk and .pvd file formats [Ahrens et al., 2005].

This work aims to test the applicability of higher-order finite volume methods to practical reservoir simulation. Therefore, OPM Flow is a natural choice for the implementation framework for many reasons, including:

- Easy to contribute; open-source.
- Fast and robust; has comparable performance with commercial reservoir simulators both for serial and parallel simulations.
- Supports different grids, including those commonly used in practical reservoir simulation - corner point grids.

- Supports industry-standard input and output file formats, allowing testing on realistic and industry-relevant models.

The following sections list the most important implementation details on extending any reservoir simulator with a second-order finite volume method. The described implementation builds on top of OPM Flow 2020.04 – the latest at the time of preparation of **Paper C**. With minor adjustments, the code would be compatible with the latest version of OPM Flow. To ensure the reproducibility of the results, the relevant versions of OPM Flow are frozen in the personal GitHub "forks" with installation instructions in https://github.com/kvashchuka/second-order-opm-tests.

5.2 Instructions of building and running second-order methods in OPM Flow

This section contains detailed instructions for building and running second-order methods in OPM Flow. The instructions have been divided into two subsections: in the first subsection, the build instructions can be found, and in the second, explanations on how to run OPM with second-order methods. The aim of this section is to provide readers with a step-by-step guide to successfully building and running second-order methods in OPM Flow.

5.2.1 Build instructions for OPM Flow with second-order methods

The build instructions below are written for a Linux system and tested on Ubuntu 2020.04.6 LTS virtual machine on Windows 10.

1. Make sure you have installed all prerequisites, available at the OPM website opm-project.org, under prerequisites. To ease this process, you can use Makefile from github.com/kvashchuka/second-order-opm-tests repository, put it to the directory, where OPM Flow will be built, and simply run `make` in the Terminal from this directory. Note that this command requires `make` to be installed.

2. Clone and build dune modules `dune-common, dune-geometry, dune-grid, dune-istl`, all in version 2.7.0:

```
1 for module in common geometry grid istl
2 do   git clone -c http.sslVerify=false https://gitlab.dune-project.org/
       core/dune-$module.git --branch v2.7.0
3 done
4 for module in common geometry grid istl
5 do   ./dune-common/bin/dunecontrol --only=dune-$module cmake -
       DCMAKE_DISABLE_FIND_PACKAGE_MPI=1
6      ./dune-common/bin/dunecontrol --only=dune-$module make -j5
7 done
```

Listing 5.1: Example of how one can clone and build dune modules

3. Clone opm modules `opm-common, opm-material, opm-grid, opm-models, opm-simulators` from https://github.com/kvashchuka and checkout branch **thesis-build**:

```
for repo in common material grid models simulators
do
    git clone -c http.sslVerify=false -b thesis-build git@github.com:
     kvashchuka/opm-$repo.git
done
```

Listing 5.2: Example of how one can clone opm modules.

4. Build OPM modules using `cmake` command. Remember to specify the path to all built modules in CMake flags, otherwise, the build process will not be completed due to the modules not being found. An example of how one can build necessary OPM modules:

```
CURRENT_DIRECTORY="$PWD"
mkdir build
for repo in common material grid models opm-simulators
do
    rm -rf build/opm-$repo
    mkdir build/opm-$repo
    cd build/opm-$repo

    cmake -DUSE_MPI=0 -DCMAKE_BUILD_TYPE=Release -DCMAKE_PREFIX_PATH="
     $CURRENT_DIRECTORY/dune-common/build-cmake;$CURRENT_DIRECTORY/dune-
     grid/build-cmake;$CURRENT_DIRECTORY/dune-geometry/build-cmake;
     $CURRENT_DIRECTORY/dune-istl/build-cmake;$CURRENT_DIRECTORY/build/opm
     -common;$CURRENT_DIRECTORY/build/opm-grid;$CURRENT_DIRECTORY/build/
     opm-models" $CURRENT_DIRECTORY/opm-$repo
    make -j5
    cd ../..
done
```

Listing 5.3: Example of how one can build opm modules.

The complete build script and the building instructions are also available in github.com/kvashchuka/second-order-opm-tests repository.

5.2.2 Running OPM Flow with second-order methods

A large list of tests is available in the github.com/opm/opm-tests repository. The test cases used in paper C, discussed in section 6.3, are available in the https://github.com/kvashchuka/second-order-opm-tests repository.

To run a test case with the second-order method, all you need to do is enable certain flags:

```
    ./*path_to_the_build_folder_of_opm-simulators*/bin/flow CASE_NAME --enable-
     higher-order=1 --enable-local-reconstruction=1 --reconstruction-scheme-id=3
      --only-reconstruction-for-solvent-or-polymer=false
```

Listing 5.4: Example of how to run OPM Flow with second-order method.

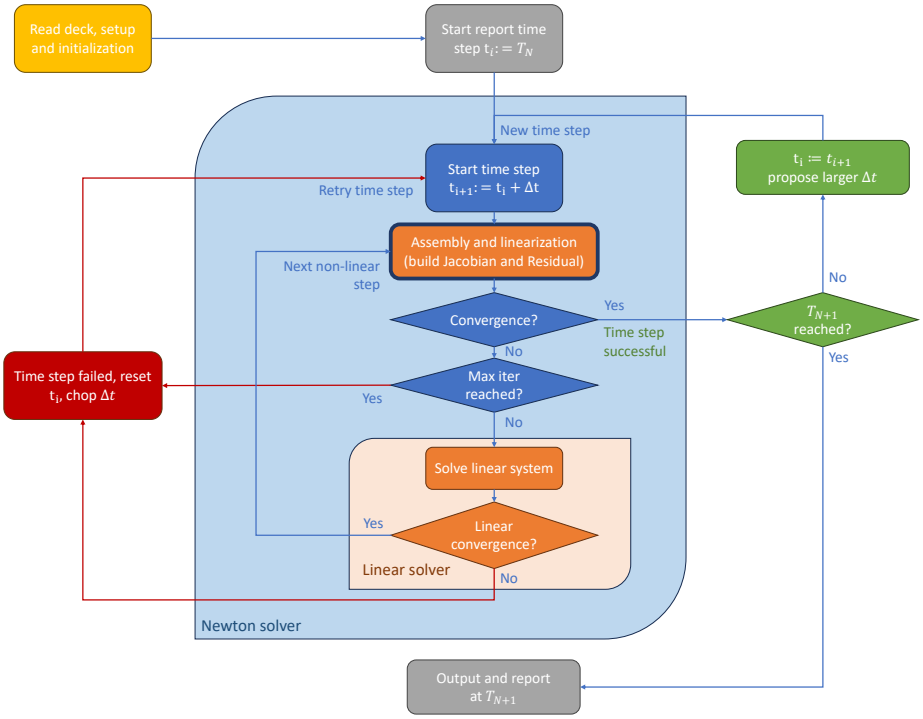

Figure 5.2: An overview of the reservoir simulator on the example of OPM Flow.

Here, `reconstruction-scheme-id` allows the user to choose between the implemented second-order methods: **2** for the second-order method with least-squares reconstruction (4.2), **3** for the second-order method with linear programming reconstruction (4.4). Flag `only-reconstruction-for-solvent-or-polymer` enables the use of higher-order methods only for solvent or polymer equations. The default value is **false**. This option was tested and discussed in paper B; see Section 6.2.

5.3 High-level overview of OPM Flow

This section presents a structure of the OPM Flow reservoir simulator, version 2020.04. A reservoir simulation solves a large, fully implicit system of nonlinear equations, which results from discretizing the governing equations (presented in Section 2.5), coupled with the well model, boundary conditions, etc. Solving such a system efficiently is a non-trivial task. In OPM, it is solved with a Newton–Raphson method. The table 5.1 and Figure 5.2 contain a high-level structure of OPM Flow.

Time loop	Takes care of the adaptive time step. It is the outermost loop that is started at each time step.
Nonlinear solver (The Newton-Raphson solver)	Builds residual and Jacobian, checks if the convergence is reached. It calls the discretization scheme to linearize the system and, finally, calls the linear solver itself.
Discretization Schemes	Implementation of the finite volume discretization. **This is the part that changes for the second-order finite volume implementation.**
Linear solver together with precondition	If linear solver converged, we return to the non-linear solver loop. If the linear convergence condition is not satisfied and the maximum number of linear iterations is not reached, we chop the time step and try the linear solver again.

Table 5.1: High-level overview of a reservoir simulator with the example of OPM Flow.

5.4 Newton iteration and automatic differentiation

The Newton-Rapson method has the second order of convergence. However, it requires the initial guess to be sufficiently close to the solution. Initially, we start with the large system of equations in the form:

$$\mathbf{K}\mathbf{u} = \mathbf{f}, \tag{5.1}$$

which is equal to

$$\mathbf{r}(\mathbf{u}) = \mathbf{K}\mathbf{u} - \mathbf{f} = 0, \tag{5.2}$$

where $\mathbf{u}$ is the primary variables vector. Let's assume ξ is the point where the residual achieves zero, and we have an initial guess $\mathbf{u}_0$. Under the assumption of sufficient smoothness in the neighborhood $N(\xi)$ of the solution, we can write a Taylor expansion of the residual function:

$$0 = \mathbf{r}(\xi) = \mathbf{r}(\mathbf{u}_0 + \Delta\mathbf{u}^{i+1}) \approx \mathbf{r}(\mathbf{u}_0) + \mathbf{J}(\mathbf{u}_0)\Delta\mathbf{u}^{i+1}. \tag{5.3}$$

In the formula above, we disregarded all the members of the Taylor expansion that contain the derivative of order two and higher. Now, the Newton method can be written as:

$$\begin{aligned} \mathbf{J}(\mathbf{u}_0)\Delta\mathbf{u}^{i+1} &= -\mathbf{r}(\mathbf{u}_0), \\ \mathbf{u}^{i+1} &= \mathbf{u}^i + \Delta\mathbf{u}^{i+1}. \end{aligned} \tag{5.4}$$

The efficiency of the Newton method is highly dependent on the accuracy of the Jacobian matrix $\mathbf{J}$. The system of equations describing flow in the reservoir

is complex and nonlinear. The key to efficiently and accurately computing the Jacobian matrix of such a system is the *automatic differentiation (AD)*.

The Evaluation class in AD implemented in OPM is designed to store not only the function's value but also its corresponding derivative(s). Any time the object of this class undergoes an arithmetic operation, the derivative is updated accordingly using the chain rule of differentiation. This allows for avoiding expensive and inaccurate derivative computations using perturbation techniques. Our implementation of the second-order methods in OPM uses the Evaluation class, thus inheriting the AD benefits for the computation of the Jacobian matrix. We refer the reader to [Lauser et al., 2018, Rasmussen et al., 2021] for the AD implementation details.

5.5 Variables reconstructed when using second-order scheme.

The extended black oil model equations were described in Chapter 2, Section 2.6. In principle, we are solving a system of nonlinear hyperbolic equations. The primary variables in the immiscible case in OPM, as described in [Rasmussen et al., 2021], are oil phase pressure and saturation of the water, gas, and solvent. The miscible case is more complicated because at certain system states, either oil or gas will not be present - gas can fully dissolve in the oleic phase, and oil could vaporize into the gaseous phase. Therefore, the composition of the phase that is present is tracked instead.

The second-order method is especially useful for front capture; therefore, we want to apply it to the flow transport part of the problem. That is why, among the primary variables, fluid saturation is the obvious candidate for second-order reconstruction. However, we can reconstruct the phase mobility function λ_α, defined in (2.8), as it depends non-linearly on the phase saturation. Our study showed that the result does not change whether we reconstruct saturation or phase mobility. For ease of implementation, we reconstruct the phase mobility function.

5.6 Common changes required for second-order FV methods

This section presents the common implementation details for both implemented second-order finite volume methods.

5.6.1 Second-order stencil

A numerical stencil is a geometric arrangement of a nodal group used in finite differences/finite volumes methods to discretize differential operators. Any second-order finite volume method utilizes information from the neighboring cells. Therefore, the stencil has to be extended. We create a new second-order finite volume stencil class `class SofvStencil` in the new file **opm-**

models/opm/models/discretization/sofv/sofvstencil.hh. It has the same methods as `class EcfvStencil` (element-center finite volume method's stencil implementation), with the straightforward update that the element and its neighbors are now part of the stencil. To extend the stencil, we loop over all element's intersections and check if the current intersection has a neighbor. If it does, we add a degree of freedom and an internal face, else, we add a boundary face. Another difference is that the number of degrees of freedom is now not one, but the number of sub-control volumes, which is equal to one plus the number of neighboring elements:

5.6.2 Second-order discretization

For the second-order implementation, a new class `SofvDiscretization` has been created to hold and handle specific functions. It is stored in new file **opm-models/opm/models/discretization/sofv/sofvdiscretization.hh** This class inherits from `FvBaseDiscretization` class.

The flux is calculated in **opm-simulators/ebos/eclfluxmodule.hh**. In that file, there is a function `calculateGradients_`, which updates the required gradients for interior faces. The function takes as an argument the element context `const ElementContext elemCtx`, stencil-local finite volume geometry index `scvfIdx`, and time index `timeIdx`. Upstream mobility is calculated in this function and used for the volume flux computations on each element. In the case of the second-order method, this function calls `evaluateReconstruction`, a function of class `SofvDiscretization` that will provide `upstreamMobilityLocal` - the upstream mobility.

Through some more abstraction layers, we finally call `updateLocal` function from **reconstruction.hh** file. The function `updateLocal` calls the specific implementation of the second-order method to compute the gradient. The implementations are stored in files **leastsquaresreconstruction.hh** and **lpreconstruction.hh** in the following folder of opm-models module **opm-models/dune/fv/**. In the next sections, we will discuss how the computation for each second-order method is carried out.

`void computeError()` function is implemented for computing the error of the method after the end of the simulation, given that the exact solution is known. The method was used in the EOC studies.

5.7 Implementation of the second-order method with least-squares reconstruction

opm-models/dune/fv/leastsquaresreconstruction.hh

The `LimitedLeastSquaresReconstructionLocal` class uses a second-order least-squares reconstruction scheme to fit a polynomial to the solution values at the center and neighbors of a given element.

For each element, we run the least-squares reconstruction algorithm, presented below.

1. loop over current element's intersections with neighbors end compute "discrete gradient" pairs $(\delta x, \Delta\lambda)$, where δx is the distance vector between the elements barycenters and $\Delta\lambda$ is the difference of solutions (4.9). Simultaneously count the total number of such intersections.

```
for( auto iit = gridView().ibegin( element ); iit != iend; ++iit ) {
    const auto intersection = *iit;

    if (intersection.neighbor()) {
        const auto neighbor = intersection.outside();
        const size_t neighborIdx = stencil.globalToLocal(neighbor);
        const GlobalCoordinate dx = neighbor.geometry().center() -
    elCenter;
        const GlobalCoordinate iNormal = intersection.
    centerUnitOuterNormal();
        const StateVector uBv = u[neighborIdx];

        const StateVector du1 = u[neighborIdx] - u[elIndex];
        negGradient.axpy(du1, dx);
        differences.emplace_back(dx, du1);
        neighbours.emplace_back(uBv, iNormal);
        neighboursCounter++;
    }
}
```

2. if there are unaccounted intersections, they belong to the domain external boundary. For each of them, we create a "ghost element" to simulate no-flow boundary conditions. To implement it, we apply a symmetric negative discrete difference for all the actual neighbors "opposing" the boundary. "Opposing" elements are those for which the projection of their normal to the ghost normal is negative.

```
if ( (stencilSize - neighboursCounter) > 1.0 ) {
    const auto iend = gridView().iend( element );
    for( auto iit = gridView().ibegin( element ); iit != iend; ++iit ) {
        const auto intersection = *iit;
        const GlobalCoordinate iCenter = intersection.geometry().center();

        if( intersection.boundary() ){
            const GlobalCoordinate iNormal = intersection.
    centerUnitOuterNormal();
            GlobalCoordinate iNormalOpposite;
            for (auto ii=0; ii < iNormal.size(); ++ii)
                iNormalOpposite [ii] = (-1.0) * iNormal[ii];
            for( const auto &neighbour : neighbours ) {
                double product = neighbour.second[0]*iNormalOpposite[0] +
    neighbour.second[1]*iNormalOpposite[1] + neighbour.second[2]*
    iNormalOpposite[2];
                if (product < 0.0) {
                    const StateVector du1 = neighbour.first - u[elIndex];
                    const GlobalCoordinate dx = (iCenter - elCenter) + (
    iCenter - elCenter);
                    negGradient.axpy(du1, dx);
                    differences.emplace_back(dx, du1);
                }
            }
```

```
        }
    }
}
```

3. the sum of the "discrete gradient" pairs form the negative flow gradient on the element

```
hessianInverse.mv( negGradient, du );
```

4. if the limiter shall be applied to the chosen limiter function based on all the "discrete gradient" pairs.

```
    du = limit( differences, du, factor );
```

5.8 Implementation of the second-order method with linear programming reconstruction. All-inequality simplex method.

opm-models/dune/fv/lpreconstruction.hh

When second-order LP method is used, the function `updateLocal` from **reconstruction.hh** calls

```
linProgLocal_[ ThreadManager::threadId() ]( elemCtx, upstreamDofIdx, timeIdx, U
    , gradient );
```

```
typedef Dune::FV::LPReconstructionLocal< GridViewType, Evaluation,
    BoundaryValue > LinearProgrammingLocal;
    ...
std::vector< LinearProgrammingLocal > linProgLocal_;
```

Below are the key steps that are performed in the `operator() (const ElementContext& elemCtx, unsigned elIndex, unsigned timeIdx, const Vector & u, Jacobian & du )` of a class `LPReconstructionLocal`, which is the implemented class for linear programming reconstruction.

1. Initialize an empty vector of pairs `differences` that will store pairs $< dx, du >$, and an empty vector of type `Constraint` called `constraints ()`. This type is defined in the file **<dune/optim/constraint/linear.hh>**.

2. Loop over all intersections in the element's grid view and populate differences, taking into account whether the intersection is on a boundary or with a neighbor:

```
if( intersection.boundary() )
{
    const GlobalCoordinate iCenter = intersection.geometry().center();
    differences.emplace_back( iCenter - elCenter, StateVector(0) );
}
else if( intersection.neighbor() )
```

```
7       {
8          const auto neighbor = intersection.outside();
9          const GlobalCoordinate nbCenter = neighbor.geometry().center();
10         const size_t neighborIdx = stencil.globalToLocal( neighbor );
11         differences.emplace_back( nbCenter - elCenter, u[ neighborIdx ] -
       u[ elIndex ] );
12      }
13
```

3. prepare variables for the linear programming optimization according to formulas (4.21) - (4.22). In the code below, `constraints[*].normal()` corresponds to the matrix A and `constraints[*].rhs()` corresponds to the vector b from (4.22). `negGradient_` corresponds to the vector c from (4.21).

```
GlobalCoordinate negGradient( 0 );
for( std::size_t i = 0u; i < numConstraints; ++i )
{
    const double sign = (differences[ i ].second >= 0 ? 1 : -1);

    negGradient.axpy( sign, differences[ i ].first );

    constraints[ 2*i ].normal() = differences[ i ].first;
    constraints[ 2*i ].normal() *= sign;
    constraints[ 2*i ].rhs() = sign*differences[ i ].second;

    constraints[ 2*i+1 ].normal() = constraints[ 2*i ].normal();
    constraints[ 2*i+1 ].normal() *= -1;
    constraints[ 2*i+1 ].rhs() = 0;
}
```

4. Choose active constraints

5. Solve the linear programming problem using lp_ with negGradient, constraints, du, and active as arguments

dune/optim/lp.hh

The efficiency of the second-order linear programming method, presented in Section 4.4, largely depends on how effectively we solve the linear programming problem for each cell. Even though each cell's LP problem is very small, in the reservoir simulation, typically, the number of cells is very large - hundreds of thousands. Therefore, efficiency is the key. Following [May and Berger, 2013] and [Chen and Li, 2016], we use an all-inequality simplex algorithm for solving LP problems. As the name suggests, it can be applied only to problems where all constraints are inequalities, which is the case here. What is more, it works particularly well when the number of constraints is much larger than the number of variables, which again is the case for most of the grids.

Same as in the usual simplex method, an optimal solution is assumed to be reached on one of the vertices of the feasible region, and the search is an iterative process. We have described what the problem and constraints look

like in Section 4.4, and here we will briefly discuss the all-inequality simplex algorithm.

Simplex algorithm

Let us start with an LP problem with only inequality constraints in a standard form:

$$\min_x c^T x \text{ subject to } Ax \geq b, \tag{5.5}$$

where $c, x \in \mathcal{R}^{\mathcal{N}}$, $A \in \mathcal{R}^{m \times \mathcal{N}}$, $b \in \mathcal{R}^m$, and $\text{rank}(A) = \mathcal{N}$. $\mathcal{N}$ here denotes the dimension of the space and m is the number of constraints. Let us reformulate (5.5) with Lagrange multipliers:

$$\min_{x,\lambda} L(x, \lambda) = c^T x - \lambda^T (Ax - b), \tag{5.6}$$

where L is the Lagrangian.

The all-inequality simplex algorithm is an iterative algorithm, where at each iteration k we perform the following steps [May and Berger, 2013].

1. First, we need to choose $\mathcal{N}$ active constraints and set a new matrix A_k, which is an $\mathcal{N} \times \mathcal{N}$ subset of the initial matrix A that contains only rows with currently active constraints. In the same fashion, we choose b_k. We denote the set of active constraints as $\mathcal{W}_k = w_1, \ldots w_{\mathcal{N}}$. On the first step we compute x_k as a solution to $\frac{\partial L}{\partial \lambda} = 0$, which simplifies to $A_k x_k = b_k$.

2. Calculate the Lagrange multipliers that satisfy $\frac{\partial L}{\partial x} = 0$. That is, find $\lambda_k \in \mathcal{R}^{\mathcal{N}}$ by solving $A_k^T \lambda_k = c$.

3. If all $\lambda_k \geq 0$, the iterations stop as the found x_k is optimal.

4. Otherwise we find index q that corresponds to the smallest Lagrange multiplier $\lambda_k^q = min(\lambda_k)$. The corresponding constraint will be removed from the active constraints.

5. The descent direction p_k is calculated as $Ap_k = e_q$, where e_q is the q-th coordinate vector.

6. Now it is time to calculate the step length α_k.

   - Find indices of the constraints that are decreasing along the descent direction p_k:

     $$D := \{i : a_i p_k < 0\},$$

     where a_i is the i-th row of A_k. Note that it means that we are not choosing from the active constraints, as for all the active constraints $a_i^T p_k = 0$.
   - If $D = 0$ we stop as the problem is unbound. However, it cannot happen for our LP problem.

- Calculate the maximum step length that does not violate the constraints:

$$\alpha_k = \max \frac{a_i^T x_k - b_i}{-a_i^T p_k}. \tag{5.7}$$

7. $x_{k+1} = x_k + \alpha_k p_k$.

8. Replace q-th row of active matrix A_k with the row of A to which the largest step length α_k correspond. If there are multiple, choose the one with the lowest index and update the set of active constraints accordingly.

9. Return to step 2 and repeat.

Chapter 6

Summary of the included papers

This chapter summarizes the included papers in the thesis that investigate the use of second-order finite volume methods in subsurface reservoir modeling. The primary objective of this research is to explore the potential of second-order methods in improving the accuracy and efficiency of reservoir simulations, thereby advancing the field of subsurface reservoir modeling.

The papers included in this thesis address different aspects of using second-order finite volume methods. The first paper compares various second-order methods against each other and the commonly used first-order finite volume method on simply shaped domains with grids relevant to practical reservoir simulations. The second paper takes a step towards incorporating second-order methods into an open-source reservoir simulator called OPM Flow. Finally, the last paper investigates the reliability of the implemented second-order method's results and analyzes its benefits and limitations on realistic reservoir models.

Overall, the papers provide a comprehensive overview and analysis of various aspects related to using second-order finite volume methods in reservoir modeling. These papers emphasize the need for more sophisticated simulation tools that reduce numerical diffusion and improve accuracy in predictions for successful reservoir modeling and management.

6.1 Testing second-order FV methods and slope-limiters on general polyhedral and corner-point grids [Paper A]

Title:	Comparison of linear reconstructions for second-order finite volume schemes on polyhedral grids
Authors:	Robert Klöfkorn, Anna Kvashchuk, Martin Nolte
Published in:	Computational Geosciences 21, pages 909–919 (2017)

This paper is the first step toward using higher-order methods in reservoir modeling. We explore the behavior of three second-order finite volume methods combined with several slope limiters on general polyhedral and corner-point grids, which are standard in reservoir simulation. By doing so, we aim to compare these methods against each other and the traditionally used first-order finite volume method and discuss their respective advantages and disadvantages. The numerical methods implementation is based on the DUNE framework [Bastian et al., 2008b, Bastian et al., 2008a, Bastian et al., 2021, Dedner et al., 2010, Alkämper et al., 2016], and the corresponding code is available within the DUNE module at https://gitlab.dune-project.org/anna.kvashchuk/polygonal-fv. Specifically, we use `PolygonGrid` for 2D computations, which is a standalone implementation,

and `PolyhedralGrid` for 3D computations, which is a component of the Open Porous Media (OPM) Initiative [Rasmussen et al., 2021].

This study focuses on the second-order finite volume method with least-squares reconstruction (introduced in Section 4.2), the second-order finite volume method with selective linear reconstruction (introduced in Section 4.3), and the second-order finite volume method with an optimization-based reconstruction (introduced in Section 4.4). As discussed in the Sections 4.2 - 4.4, the least squares and the selective linear reconstruction second-order FV methods require limiters. We use three limiters with each of these two methods - minmod (3.32), van Leer (3.34), and Superbee (3.33) The optimization-based second-order finite volume method has a limiter embedded in it.

This study aims to verify the experimental convergence properties of selected second-order methods on arbitrary polyhedral meshes. The test cases use 2D and 3D domains with simple shapes (a square and a cube, respectively). We choose transport problems that have known analytical solutions, enabling us to measure and compare the experimental order of convergence (EOC) (3.17) for all the methods.

The 2D solid body rotation test [Leveque, 1996] is performed on triangular and polygonal grids. In this classical test case, we rotate three shapes - a cone, a cylinder, and a cylinder with a part missing, with a constant velocity around the center of the unit square. When conducting an EOC study, the results of all methods are computed and compared on a series of grids. Each grid has a reference size twice as small as the previous grid. This test case allows us to see to which extent the methods can preserve discontinuity presented in the initial state and how much smearing they introduce. All second-order approaches outperform the first-order method by orders of magnitude in terms of accuracy vs. run time. The second-order method with the linear reconstruction combined with Van Leer or Superbee limiters together with the optimization-based second-order method shows the best performance on both triangular and polygonal grids.

The second 2D test case studies the Buckley-Leverett type problem, again on a series of triangular and polygonal grids. It represents a simplified model for water-oil displacement in a reservoir. This is a non-linear scalar problem with a shock and a rarefaction wave, where a quasi-exact solution can be computed, which provides an interesting setup for a convergence study. In this test case, the second-order methods are more accurate and faster than the first-order method. The first-order method would require a much finer grid and two orders of magnitude more run time to achieve the accuracy of the second-order methods. This behavior is consistent across different meshes for this test case.

In the two three-dimensional test cases, we use a series of unit cubes with hexagonal prisms and tetrahedral decreasing in size. The grids are part of the 3D Benchmark on Discretization Schemes for Anisotropic Diffusion Problems on General Grids presented in [Eymard et al., 2011]. A simple tracer transport is modeled, together with the Buckley-Leverett equation. In both test cases, the second-order schemes outperform the first-order scheme by orders of magnitude on coarse grids, with the first-order scheme only achieving similar accuracy on much finer grids. However, the least squares approach fails to achieve the

second-order method performance on polyhedral grids and shows accuracy and CPU time almost identical to the first-order method. The first-order method and second-order method with least-squares reconstruction on 3D polyhedral grids achieve comparable accuracy on the finest grid as the second-order method with selective linear reconstruction and linear programming reconstruction on the coarsest grid in the series.

Our major findings showed that second-order methods outperform first-order method in terms of accuracy and run time in all test cases, indicating the potential for improving reservoir simulation tools to avoid numerical diffusion. While second-order methods require more CPU time, for the first-order method to achieve the same accuracy, a much finer grid has to be used, leading to a considerably larger increase in the run time. Of the second-order methods, the fastest was the least squares approach. However, the method deteriorated to the first-order accuracy on the 3D polyhedral grid. On the other hand, the selective reconstruction showed better accuracy at the expense of around 50% more CPU time than the least squares reconstruction. The second-order finite volume method with selective linear reconstruction slows down as the number of neighbors per cell increases. This is why it takes longer to execute in 3D despite showing a comparable CPU time to other second-order methods in 2D. The LP reconstruction scheme provided a good compromise between the two approaches in terms of complexity and accuracy.

These findings show the potential of second-order finite volume schemes for reservoir simulation with complex geometries.

6.2 Convergence study of second-order FV implementation in OPM Flow [Paper B]

Title:	Comparison of Higher Order Schemes on Complicated Meshes and Reservoirs
Authors:	Anna Kvashchuk, Robert Klöfkorn, Tor Harald Sandve
Published in:	SPE Reservoir Simulation Conference, Galveston, Texas, USA, April 2019

This paper builds upon the previous study by extending the integration of second-order numerical schemes into the open-source framework - the Open Porous Media (OPM) initiative. The study focuses on implementing two second-order finite volume methods, the least-squares and linear programming reconstructions, in the OPM Flow simulator. The accuracy, performance, and ability to handle nonlinearities of the second-order methods are compared to the standard first-order method using several benchmark cases relevant to practical reservoir simulation. Overall, the paper provides valuable insights into enhancing the accuracy and efficiency of fluid transport modeling in reservoir simulations using more sophisticated numerical schemes.

To verify the implementation, the experimental order of convergence is checked on a simple transport test case on two series of grids - Cartesian and corner-point grids typically used in reservoir modeling. The results indicate that both second-order methods produce more accurate solutions than the first-order method on Cartesian grids. However, the second-order method with linear programming reconstruction outperforms the second-order least-squares method on the 3D unstructured corner-point grid, indicating it is a preferred choice for complex reservoir grids.

In the next test case, we compare the methods' performance on a well-known benchmark test case for the solvent injection - SPE5. The water alternating gas injection (WAG) with a one-year cycle scenario is employed. The most noticeable difference between the first- and second-order methods is observed in how well they could capture a sharp gas wave formed in the reservoir. Both second-order methods predict a half-a-year delay in the arrival of the gas wave and render the front sharper and with less smearing.

The use of a Cartesian grid in this test case allows for straightforward refinement, facilitating the comparison of second-order methods' predictions with the results from the first-order method on a finer grid, i.e., the reference solution. The reference solution agrees with the front positioning obtained by the two second-order methods. However, the amplitude of the gas wave was slightly overestimated.

While second-order numerical methods produce more accurate results than the first-order method, they can be computationally expensive. It may be beneficial to selectively apply the second-order method to equations with linear fluxes to reduce computational time while providing an accurate solution. Later, in this test case, we show that applying the second-order method only to the solvent equation provides good predictions of the position and the amplitude of the gas wave and reduces computational time while still providing an accurate solution that agrees well with the first-order method on a refined grid.

In the last test case, we study the impact of using a higher-order numerical method in the field-scale CO_2-EOR study. The openly available Norne field reservoir model is used for this study. The study employs the solvent model, an extended black-oil model that distinguishes between the injected and formation gas. We observe that the first and second-order methods provide different results when predicting the solvent production rate. Unfortunately, since it is a field-scale example with a full complexity of the used grid and fluid interactions, we do not know the true solution and cannot verify which of the methods provides a more accurate solution.

6.3 Validation of OPM Flow's second-order methods for practical EOR simulations [Paper C]

Title:	A second order finite volume method for field-scale reservoir simulation
Authors:	Anna Kvashchuk, Robert Klöfkorn, Tor Harald Sandve
Published in:	Transport in Porous Media, volume 150, pages 109-129, 2023.

This paper explores the potential of the second-order linear programming method in fluid transport modeling for reservoir simulations. We aim to investigate the accuracy and efficiency of this method in complex and realistic reservoir examples. Our objective is to verify the results' reliability, given the absence of "true" solutions, and to analyze the limitations and benefits of the method.

Our tests use two realistic reservoirs: a medium-sized realistic reservoir with an unstructured corner point grid and an openly available Norne field model. As a reference solution for the result verification, we use the first-order method on the refined grid, and a new refined Norne model has been produced for this study. All mentioned models are available in the second-order-opm-tests repository, https://github.com/kvashchuka/second-order-opm-tests.

The first test case in this study presents a complex setup with regard to fluid interaction and uses a corner-point grid. However, the realistic features present in most reservoirs, such as faults and regions, are absent here, which makes interpreting the results more accessible. The domain consists of a rectangular hexahedron with a single production and an injection well placed in opposing corners. In this test case, we set up a water-alternating gas injection (WAG) scenario, modeling a three-phase flow with four components. We analyze production rate curves for solvent, water, oil, and gas components. Results show an agreement between the second-order method on the coarse grid and the reference solution provided by the first-order method on the refined grid. As expected, the second-order method tends to show later arrival times of the fluid fronts because of the reduced smearing. Furthermore, proper interpretation of the results requires careful consideration of the nature of the fluid fronts. When pressure changes are driving the fluid fronts, the implemented second-order method fails to improve the results of the first-order method, as pressure calculations remain unchanged. However, when the transport phenomena dominate the fluid flow, the front positioning is improved, and smearing is reduced. This behavior agrees with the first-order method result on the refined grid.

The study provides three test cases on the Norne field. The first test is a piston-type injection with homogeneous and heterogeneous porosity and permeability. In order to focus on fluid displacement effects, the two fluids have the same properties, and any regions and faults in the reservoir are neglected.

While the differences between the first- and second-order methods are more pronounced in the homogeneous test case, the second-order method reduces the smearing effect in both setups.

To verify the performance of the second-order method, the first-order method is run on the refined Norne model. The standard Norne field model has a grid block size of approximately 100m, and the refined model used in this study - 50 m in X and Y directions (since we are interested in fluid displacement, the grid cells are not refined in the Z direction). The test case shows that the second-order method on the standard grid successfully produces a sharper resolution of the front, with minimal spread, compared to the first-order method. This outcome agrees with the results from the first-order method on the refined Norne field model.

The last test case provides the most complex setup as we model a CO_2 injection on the Norne field with its full complexity, meaning we have two wells, heterogeneity, regions, faults, etc. It is worth noting that while the Norne field is not a primary target for CO_2 injection, this test case provides valuable insight into the methods' performance in the EOR/IOR scenario on a realistic field model. Our objective is to determine when the effect of the second-order method is most pronounced. That is why we vary the placement of the injector well: first, it is placed in the same compartment as the producer well; second, the two wells are separated by one fault; and third, the injector is in the corner surrounded by faults. In all three cases, the second-order method predicts the later arrival of the injected CO_2. However, a smaller difference is seen when there are faults between the wells. It shows that while we observe a relevant difference in the front positioning when using the higher-order method, when the complexity of the reservoir increases, it can overshadow the effects gained by it.

Overall, this paper provides an important contribution to the field of subsurface reservoir modeling and addresses the limitations of the first-order method, potentially enabling more informed decision-making in various applications.

Chapter 7

Conclusions and future work

Throughout this work, several higher-order numerical methods for fluid transport in reservoir simulation were implemented, tested, and compared. Consequently, the implementation of second-order finite volume methods in OPM Flow, an open-source reservoir simulator, was published. This also serves as an implementation guide for further developments.

The studies presented here investigated and provided valuable insights into the benefits and limitations of second-order numerical methods. We compared the performance of several second-order methods and the traditional first-order method on different grid types, various injection/production scenarios, and reservoir models of different complexity levels.

Based on the findings of the study, the second-order method with linear programming reconstruction (the second-order LP method) outperformed the other second-order methods that were tested. On 3D grids, where elements have more neighbors, the LP method showed faster performance than the second-order method with selective reconstruction. The third second-order method, with the least square reconstruction, did not improve accuracy over the first-order method in complex reservoir settings. During our final testing, the LP method showed good agreement with the reference solution obtained on a refined grid of a complex reservoir Norne field (the reference solution used the standard first-order method).

Let us highlight the benefits and limitations of the second-order LP method for practical reservoir simulation based on our findings (compared to standard methods).

Benefits

- Improved accuracy of fluid transport modeling in reservoir simulations.
- Sharper front representation, with reduced smearing.
- Potential to enhance decision-making in various applications.

Disadvantages

- Increased computational time.
- Benefits may be overshadowed by reservoir model complexity.

We see several future research possibilities for improved reservoir simulation that can mitigate the listed disadvantages of the second-order LP method.

To increase the computational efficiency of the second-order method while preserving accuracy, one option could be to explore machine-learning reconstructions on the local stencils. Such a reconstruction based on a neural network

might reduce the number of function calls and be optimized with respect to the Newton solver. Very recently, [Keim et al., 2023] explored the utilization of reinforcement learning (RL) in numerical modeling. The authors present a new slope limiter for a two-dimensional finite volume scheme derived through the use of RL techniques.

The complex nature of the reservoir simulation problem does not allow us to say with certainty that using the second-order methods will always give enough benefit to compensate for the significant increase in simulation time. Although accuracy is important in transport modeling, the complexity and inherent uncertainty of the subsurface reservoirs, coupled with numerous fluid interactions and pressure effects, can overshadow the benefits gained from using second-order methods for transport. However, using more advanced numerical methods is not the only way to improve the prediction accuracy simulations. For instance, ensemble methods are steadily gaining more and more popularity in closed-loop reservoir modeling. Using an ensemble method means running multiple realizations of a reservoir model, each with slightly different parameters, and then adjusting those parameters to fit the available data in a closed loop. This allows to capture of uncertainties associated with the complex geologic and fluid properties of subsurface systems. While an ensemble method does not improve simulation results directly, its ability to capture uncertainty and possibly compensate for modeling errors enables better results with existing simulation tools. With the increase of computational power, one can combine the second-order finite volume methods with the ensemble methods in the future.

Lately, there has been an increased focus on data-driven approaches and machine learning algorithms, which can be easily integrated with ensemble methods to accelerate and sometimes improve reservoir characterization, modeling, and forecasting. However, for problems with relatively scarce data, such as reservoir modeling, the field data shall be combined with simulated data during training. To that end, second-order methods can excel in providing more accurate simulated data during slow-pace 'offline' training of the ML models.

Bibliography

[Aavatsmark, 2002] Aavatsmark, I. (2002). An introduction to multipoint flux approximations for quadrilateral grids. *Computational Geosciences*, 6:405–432.

[Ahrens et al., 2005] Ahrens, J., Geveci, B., and Law, C. (2005). ParaView: An end-user tool for large data visualization. In *Visualization Handbook*. Elesvier. ISBN 978-0123875822.

[Alkämper et al., 2016] Alkämper, M., Dedner, A., Klöfkorn, R., and Nolte, M. (2016). The dune-alugrid module. *Archive of Numerical Software*, 4(1):1–28.

[Alyaev et al., 2014] Alyaev, S., Keilegavlen, E., and Nordbotten, J. M. (2014). Analysis of control volume heterogeneous multiscale methods for single phase flow in porous media. *Multiscale Modeling & Simulation*, 12(1):335–363.

[Bastian et al., 2008a] Bastian, P., Blatt, M., Dedner, A., Engwer, C., Klöfkorn, R., Kornhuber, R., Ohlberger, M., and Sander, O. (2008a). A generic grid interface for parallel and adaptive scientific computing. part II: implementation and tests in dune. *Computing*, 82:121–138.

[Bastian et al., 2008b] Bastian, P., Blatt, M., Dedner, A., Engwer, C., Klöfkorn, R., Ohlberger, M., and Sander, O. (2008b). A generic grid interface for parallel and adaptive scientific computing. part I: abstract framework. *Computing*, 82:103–119.

[Bastian et al., 2021] Bastian, P., Blatt, M., Dedner, M., Dreier, N.-A., Engwer, Ch. Fritze, R., Gräser, C., Grüninger, C., Kempf, D., Klöfkorn, R., Ohlberger, M., and Sander, O. (2021). The dune framework: Basic concepts and recent developments. *Computers & Mathematics with Applications*, 81:75–112.

[Baxendale, 2023] Baxendale, D. (2023). *OPEN POROUS MEDIA OPM Flow Reference Manual (2023-04)*.

[Bell and Shubin, 1985] Bell, J. B. and Shubin, G. R. (1985). Higher-Order Godunov Methods for Reducing Numerical Dispersion in Reservoir Simulation. In *SPE Reservoir Simulation Symposium, Dallas, Texas*. Society of Petroleum Engineers.

[Blunt and Rubin, 1992] Blunt, M. and Rubin, B. (1992). Implicit flux limiting schemes for petroleum reservoir simulation. *Journal of Computational Physics*, 102(1):194–210.

[Brooks and Corey, 1964] Brooks, R. H. and Corey, A. T. (1964). *Hydraulic properties of porous media*, volume 3. Colorado State University.

[Chase Jr et al., 1984] Chase Jr, C. A., Todd, M. R., et al. (1984). Numerical simulation of CO_2 flood performance. *Society of Petroleum Engineers Journal*, 24(06):597–605.

[Chen and Li, 2016] Chen, L. and Li, R. (2016). An integrated linear reconstruction for finite volume scheme on unstructured grids. *Journal of Scientific Computing*, 68(3):1172–1197.

[Chen et al., 1993] Chen, W., Durlofsky, L., Engquist, B., and Osher, S. (1993). Minimization of grid orientation effects through use of higher order finite difference methods. *SPE Advanced Technology Series*, 1(02):43–52.

[Chen, 2000] Chen, Z. (2000). Formulations and numerical methods of the black oil model in porous media. *SIAM Journal on Numerical Analysis*, 38(2):489–514.

[Chen et al., 2006] Chen, Z., Huan, G., and Ma, Y. (2006). *Computational methods for multiphase flows in porous media*. SIAM.

[Chierici, 1992] Chierici, G. L. (1992). Economically improving oil recovery by advanced reservoir management. *Journal of Petroleum Science and Engineering*, 8(3):205–219.

[Cockburn and Shu, 1989] Cockburn, B. and Shu, C.-W. (1989). TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws. II. General framework. *Mathematics of computation*, 52(186):411–435.

[Contreras et al., 2016] Contreras, F., Lyra, P., Souza, M., and Carvalho, D. d. (2016). A cell-centered multipoint flux approximation method with a diamond stencil coupled with a higher order finite volume method for the simulation of oil–water displacements in heterogeneous and anisotropic petroleum reservoirs. *Computers & Fluids*, 127:1–16.

[Courant et al., 1928] Courant, R., Friedrichs, K., and Lewy, H. (1928). Über die partiellen differenzengleichungen der mathematischen physik. *Mathematische annalen*, 100(1):32–74.

[Courant et al., 1967] Courant, R., Friedrichs, K., and Lewy, H. (1967). On the partial difference equations of mathematical physics. *IBM journal of Research and Development*, 11(2):215–234.

[Dedner and Klöfkorn, 2011] Dedner, A. and Klöfkorn, R. (2011). A generic stabilization approach for higher order discontinuous galerkin methods for convection dominated problems. *Journal of Scientific Computing*, 47(3):365–388.

[Dedner et al., 2010] Dedner, A., Klöfkorn, R., Nolte, M., and Ohlberger, M. (2010). A generic interface for parallel and adaptive scientific computing: Abstraction principles and the dune-fem module. *Computing*, 90(3–4):165–196.

[Durlofsky et al., 1992] Durlofsky, L. J., Engquist, B., and Osher, S. (1992). Triangle based adaptive stencils for the solution of hyperbolic conservation laws. *Journal of Computational Physics*, 98(1):64–73.

[Eymard et al., 2011] Eymard, R., Henry, G., Herbin, R., Hubert, F., Klöfkorn, R., and Manzini, G. (2011). 3D benchmark on discretization schemes for anisotropic diffusion problems on general grids. In *Finite Volumes for Complex Applications VI Problems & Perspectives: FVCA 6, International Symposium, Prague, June 6-10, 2011*, pages 895–930. Springer.

[Feistauer et al., 2003] Feistauer, M., Felcman, J., and Straškraba, I. (2003). *Mathematical and computational methods for compressible flow*. Oxford University Press, USA.

[Flemisch et al., 2011] Flemisch, B., Darcis, M., Erbertseder, K., Faigle, B., Lauser, A., Mosthaf, K., Müthing, S., Nuske, P., Tatomir, A., Wolff, M., et al. (2011). DuMux: DUNE for multi-{phase, component, scale, physics, ...} flow and transport in porous media. *Advances in Water Resources*, 34(9):1102–1112.

[Gao, 2011] Gao, C. H. (2011). Scientific research and field applications of polymer flooding in heavy oil recovery. *Journal of Petroleum Exploration and Production Technology*, 1:65–70.

[Geiger et al., 2009] Geiger, S., Matthäi, S., Niessner, J., and Helmig, R. (2009). Black-oil simulations for three-component, three-phase flow in fractured porous media. *SPE journal*, 14(02):338–354.

[Ghoreishian Amiri et al., 2013] Ghoreishian Amiri, S., Sadrnejad, S., Ghasemzadeh, H., and Montazeri, G. (2013). Application of control volume based finite element method for solving the black-oil fluid equations. *Petroleum Science*, 10:361–372.

[Harten, 1997] Harten, A. (1997). High resolution schemes for hyperbolic conservation laws. *Journal of computational physics*, 135(2):260–278.

[Hassanizadeh et al., 2002] Hassanizadeh, S. M., Celia, M. A., and Dahle, H. K. (2002). Dynamic effect in the capillary pressure–saturation relationship and its impacts on unsaturated flow. *Vadose Zone Journal*, 1(1):38–57.

[Holm et al., 2008] Holm, R., Van Dijke, M. I. J., Geiger, S., and Espedal, M. (2008). Consistent capillary pressure and relative permeability for mixed-wet systems in macroscopic three-phase flow simulation. In *ECMOR XI-11th European Conference on the Mathematics of Oil Recovery*, pages cp–62. European Association of Geoscientists & Engineers.

[Jiang and Shu, 1996] Jiang, G.-S. and Shu, C.-W. (1996). Efficient implementation of weighted ENO schemes. *Journal of computational physics*, 126(1):202–228.

[Joekar-Niasar et al., 2010] Joekar-Niasar, V., Hassanizadeh, S., and Dahle, H. (2010). Non-equilibrium effects in capillarity and interfacial area in two-phase flow: dynamic pore-network modelling. *Journal of Fluid Mechanics*, 655:38–71.

[Jung et al., 2017] Jung, Y., Pau, G. S. H., Finsterle, S., and Pollyea, R. M. (2017). TOUGH3: a new efficient version of the tough suite of multiphase flow and transport simulators. *Computers & Geosciences*, 108:2–7.

[Keim et al., 2023] Keim, J., Schwarz, A., Chiocchetti, S., Rohde, C., and Beck, A. (2023). A reinforcement learning based slope limiter for two-dimensional finite volume schemes. In *International Conference on Finite Volumes for Complex Applications*, pages 209–217. Springer.

[Klemetsdal et al., 2020] Klemetsdal, Ø. S., Rasmussen, A. F., Møyner, O., and Lie, K.-A. (2020). Efficient reordered nonlinear gauss–seidel solvers with higher order for black-oil models. *Computational Geosciences*, 24(2):593–607.

[Koch et al., 2021] Koch, T., Gläser, D., Weishaupt, K., et al. (2021). DuMux 3 – an open-source simulator for solving flow and transport problems in porous media with a focus on model coupling. *Computers & Mathematics with Applications*, 81:423–443.

[Kröner, 1997] Kröner, D. (1997). *Numerical Schemes for Conservation Laws*. Verlag Wiley & Teubner, Stuttgart.

[Lamine and Edwards, 2015] Lamine, S. and Edwards, M. G. (2015). Multidimensional upwind schemes and higher resolution methods for three-component two-phase systems including gravity driven flow in porous media on unstructured grids. *Computer Methods in Applied Mechanics and Engineering*, 292:171–194.

[Lauser et al., 2018] Lauser, A., Rasmussen, A., Sandve, T., and Nilsen, H. (2018). Local forward-mode automatic differentiation for high performance parallel pilot-level reservoir simulation. In *ECMOR XVI-16th European Conference on the Mathematics of Oil Recovery*, pages 1–12. European Association of Geoscientists & Engineers.

[Leveque, 1996] Leveque, R. J. (1996). High-resolution conservative algorithms for advection in incompressible flow. *SIAM Journal on Numerical Analysis*, 33(2):627–665.

[LeVeque, 2002] LeVeque, R. J. (2002). *Finite volume methods for hyperbolic problems*, volume 31. Cambridge University Press.

[Lichtner et al., 2019] Lichtner, P. C., Hammond, G. E., Lu, C., Karra, S., Bisht, G., Andre, B., Mills, R. T., Kumar, J., and Frederick, J. M. (2019). PFLOTRAN user manual.

[Lie, 2015] Lie, K.-A. (2015). Hyperbolic conservation laws: Computation. *Encyclopedia of Applied and Computational Mathematics*.

[Lie, 2019] Lie, K.-A. (2019). *An introduction to reservoir simulation using MATLAB/GNU Octave: User guide for the MATLAB Reservoir Simulation Toolbox (MRST)*. Cambridge University Press.

[Lie and Møyner, 2021] Lie, K.-A. and Møyner, O. (2021). *Advanced Modeling with the MATLAB Reservoir Simulation Toolbox*. Cambridge University Press.

[Lie et al., 2020] Lie, K.-A., Mykkeltvedt, T. S., and Møyner, O. (2020). A fully implicit weno scheme on stratigraphic and unstructured polyhedral grids. *Computational Geosciences*, 24(2):405–423.

[Liu et al., 1994] Liu, X.-D., Osher, S., and Chan, T. (1994). Weighted essentially non-oscillatory schemes. *Journal of computational physics*, 115(1):200–212.

[May and Berger, 2013] May, S. and Berger, M. (2013). Two-dimensional slope limiters for finite volume schemes on non-coordinate-aligned meshes. *SIAM Journal on Scientific Computing*, 35(5):A2163–A2187.

[Mykkeltvedt, 2014] Mykkeltvedt, T. S. (2014). *Numerical solutions of two-phase flow with applications to CO2 sequestration and polymer flooding*. PhD thesis, The University of Bergen.

[Mykkeltvedt et al., 2017] Mykkeltvedt, T. S., Raynaud, X., and Lie, K.-A. (2017). Fully implicit higher-order schemes applied to polymer flooding. *Computational Geosciences*, 21(5):1245–1266.

[Nocedal and Wright, 2006] Nocedal, J. and Wright, S. (2006). *Numerical optimization*. Springer Science & Business Media.

[Nordbotten and Celia, 2011] Nordbotten, J. M. and Celia, M. A. (2011). *Geological storage of CO2: modeling approaches for large-scale simulation*. John Wiley & Sons.

[OG21, 2021] OG21 (2021). Og21 a new chapter. oil and gas for the 21st century. https://www.og21.no/siteassets/figurer-og21-strategi-2021/og21-strategi_eng.pdf.

[Rasmussen et al., 2021] Rasmussen, A. F., Sandve, T. H., Bao, K., Lauser, A., Hove, J., Skaflestad, B., Klöfkorn, R., Blatt, M., Rustad, A. B., Sævareid, O., et al. (2021). The open porous media flow reservoir simulator. *Computers & Mathematics with Applications*, 81:159–185.

[Roe, 1985] Roe, P. L. (1985). Some contributions to the modelling of discontinuous flows. *Large-scale computations in fluid mechanics*, pages 163–193.

[Rubin and Blunt, 1991] Rubin, B. and Blunt, M. (1991). Higher-order implicit flux limiting schemes for black oil simulation. In *SPE Symposium on Reservoir Simulation*. Society of Petroleum Engineers.

[Rubin and Edwards, 1993] Rubin, B. and Edwards, M. (1993). Extension of the TVD midpoint scheme to higher-order accuracy in time. In *SPE Symposium on Reservoir Simulation*. Society of Petroleum Engineers.

[Sandve et al., 2018] Sandve, T. H., Rasmussen, A., and Rustad, A. B. (2018). Open reservoir simulator for CO_2 storage and CO_2-EOR. In *14th Greenhouse Gas Control Technologies Conference Melbourne*, pages 21–26.

[SLB, 2020] SLB (2020). Eclipse™2020.3. http://www.software.slb.com/products/eclipse.

[Sweby, 1984] Sweby, P. K. (1984). High resolution schemes using flux limiters for hyperbolic conservation laws. *SIAM journal on numerical analysis*, 21(5):995–1011.

[Todd et al., 1972] Todd, M., Longstaff, W., et al. (1972). The development, testing, and application of a numerical simulator for predicting miscible flood performance. *Journal of Petroleum Technology*, 24(07):874–882.

[Trangenstein, 2009] Trangenstein, J. A. (2009). *Numerical solution of hyperbolic partial differential equations*. Cambridge University Press.

[Trefethen and Bau, 1997] Trefethen, L. N. and Bau, D. (1997). *Numerical Linear Algebra*. Society for Industrial and Applied Mathematics.

[Van Genuchten, 1980] Van Genuchten, M. T. (1980). A closed-form equation for predicting the hydraulic conductivity of unsaturated soils. *Soil science society of America journal*, 44(5):892–898.

[Van Leer, 1974] Van Leer, B. (1974). Towards the ultimate conservative difference scheme. II. Monotonicity and conservation combined in a second-order scheme. *Journal of computational physics*, 14(4):361–370.

[Van Leer, 1977] Van Leer, B. (1977). Towards the ultimate conservative difference scheme III. Upstream-centered finite-difference schemes for ideal compressible flow. *Journal of Computational Physics*, 23(3):263–275.

[Van Leer, 1979] Van Leer, B. (1979). Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method. *Journal of computational Physics*, 32(1):101–136.

[Voskov and Tchelepi, 2012] Voskov, D. V. and Tchelepi, H. A. (2012). Comparison of nonlinear formulations for two-phase multi-component EoS based simulation. *Journal of Petroleum Science and Engineering*, 82:101–111.

[Walsh et al., 2003] Walsh, M. P., Lake, L. W., and Walsh, M. (2003). *A generalized approach to primary hydrocarbon recovery*, volume 4. Elsevier Amsterdam.

[Warming and Beam, 1976] Warming, R. and Beam, R. M. (1976). Upwind second-order difference schemes and applications in aerodynamic flows. *AIAA Journal*, 14(9):1241–1249.

Papers

Paper A

Comparison of linear reconstructions for second-order finite volume schemes on polyhedral grids.

Robert Klöfkorn, Anna Kvashchuk, Martin Nolte
Computational Geosciences, **volume 21, pages 909–919** (2017)
DOI: 10.1007/s10596-017-9658-8

This paper is not included in the repository due to copyright restrictions.

Paper B

Comparison of Higher Order Schemes on Complicated Meshes and Reservoirs.

Anna Kvashchuk, Robert Klöfkorn, Tor Harald Sandve
In *SPE Reservoir Simulation Conference, Galveston, Texas, USA* (2019)
DOI: 10.2118/193839-MS

This paper is not included in the repository due to copyright restrictions.

Paper C

A Second-Order Finite Volume Method for Field-Scale Reservoir Simulation.

Anna Kvashchuk, Robert Klöfkorn, Tor Harald Sandve
Transport in Porous Media, **volume 150, pages 109-129** (2023)

Transport in Porous Media (2023) 150:109–129
https://doi.org/10.1007/s11242-023-01999-1

A Second-Order Finite Volume Method for Field-Scale Reservoir Simulation

Anna Kvashchuk[1,2] · **Robert Klöfkorn**[3] · **Tor Harald Sandve**[4]

Received: 10 June 2022 / Accepted: 7 July 2023 / Published online: 18 August 2023

Abstract

Subsurface reservoirs are large complex systems. Reservoir flow models are defined on complex grids that follow geology with relatively large block sizes to make consistent simulations feasible. Reservoir engineers rely on established reservoir simulation software to model fluid flow. Nevertheless, fluid front position inaccuracies and front smearing on large grids may cause significant errors and make it hard to predict hydrocarbon production efficiency. We investigate higher-order methods that reduce these undesired effects without refining the grid, thus making reservoir simulation more accurate and robust. For this paper, we implemented a second-order finite volume method with linear programming (LP) reconstruction in the open-source industry-grade reservoir simulator OPM Flow (part of the open porous media initiative, OPM). We benchmark it against the first-order method on full-scale cases with standard coarse and refined grids. We prepared open refined-grid models of a synthetic reservoir with an unstructured grid and refined Norne field example. Our results confirm that the LP method predicts front positions as accurately as the first-order method on the refined grid for problems dominated by transport. These include the water alternating gas scenario on the synthetic reservoir and piston-type injection on the Norne field. Moreover, we study the gains from the LP method for CO_2 injection problems on the Norne field with full multi-phase complexity beyond transport. We observe the relevant difference between the first- and the second-order methods in these cases. However, in some configurations, the reservoir complexity overshadows the gains from the second-order methods.

✉ Anna Kvashchuk
kvashchuk.anna@gmail.com

Robert Klöfkorn
robertk@math.lu.se

Tor Harald Sandve
tosa@norceresearch.no

[1] Equinor ASA, Forusbeen 50, 4035 Stavanger, Norway

[2] Faculty of Science and Technology, Department of Energy Resources, University of Stavanger, P.O. Box 8600, 4036 Stavanger, Norway

[3] Center for Mathematical Sciences, Lund University, P.O. Box 117, 221 00 Lund, Sweden

[4] Energy and Technology, NORCE Norwegian Research Centre, Nygårdsgaten 112, 5008 Bergen, Norway

Keywords Second-order method · Reservoir simulation · OPM · Norne · Realistic reservoir

1 Introduction

Modeling multi-phase multi-component flow in porous media requires accurate and robust numerical methods. It is an essential part of the oil and gas production (Trangenstein and Bell 1989; Coats et al. 1995), carbon storage (Class et al. 2009; Celia and Nordbotten 2009; Sandve et al. 2018; Mykkeltvedt et al. 2021), enhanced oil recovery (Lake 1989; Mykkeltvedt et al. 2017) and many other applications.

The first-order finite volume (FV) is the usual method for modeling multi-phase multi-component flow. It is the default option in many standard reservoir simulators, both commercial and open-source, for example, ECLIPSE (2014), OPM (Open Porous Media) (Lauser et al. 2018), the Matlab Reservoir Simulation Toolbox (MRST) (Lie 2019), DuMu^x (Flemisch et al. 2011), PFLOTRAN (Lichtner et al. 2019), an Automatic Differentiation General Purpose Research Simulator (ADGPRS) (Voskov and Tchelepi 2012) or TOUGH3 (Jung et al. 2017). First-order methods are widely used because of their robustness and ease of implementation. Unfortunately, those are known to suffer from numerical diffusion, which leads to smearing of the fluid fronts and, therefore, incorrect computations of the front position, components concentrations, water breakthrough, etc.

To reduce the numerical diffusion and increase the accuracy, there are mainly two options: to refine the grid or increase the order of the numerical method. Grid refinement, while being a valid solution in many cases, is often impractical for reservoir simulation due to the reservoir size and complex grids, faults, etc. It complicates the model and increases the simulation time significantly. On the other hand, increasing the order of the numerical method can provide a more accurate solution with reduced grid-orientation effects and better front resolution on a practical grid block size without modifying the grid (Sammon et al. 2001). However, a practical challenge with the higher-order methods is unphysical oscillations (undershoots and overshoots) around the front, which require the introduction of slope-limiters (LeVeque 2002).

The early papers applying higher-order methods in reservoir simulation date back to the 1980s. Bell and Shubin (1985) presented a higher-order Godunov scheme for one- and two-dimensional five-spot problems with miscible and immiscible displacement flow models. In Rubin and Blunt (1991), Blunt and Rubin (1992) and Rubin and Edwards (1993), the authors show how higher-order total variation diminishing (TVD) schemes can be applied to one- and two-dimensional simplified reservoir simulation. Chen et al. (1993) used second-order TVD- and third-order essentially non-oscillatory (ENO) schemes to minimize grid orientation effects and improve front resolution on the 2D five-spot model. In May and Berger (2013) proposed to use constraint optimization with linear programming to compute the reconstruction in a higher-order method. And Chen and Li (2016) further proposed an improved linear-programming scheme that did not require an initial gradient computation.

Despite continued research of higher-order methods in reservoir simulation (Durlofsky et al. 1992; Harten 1997; Geiger-Boschung et al. 2009; Lamine and Edwards 2015; Contreras et al. 2016; Mykkeltvedt et al. 2017), most implementations are done in academic codes with Cartesian or simplex meshes. Only a few researchers applied them to implementation-intensive corner-point grids that capture complex geometries of subsurface reservoirs. In Lie et al. (2020), a weighted-ENO (WENO, introduced in Jiang and Shu 1996; Liu et al.

1994) method was applied to simplified test cases on reservoir-type grids. In Klemetsdal et al. (2020), a discontinuous Galerkin (DG) method (introduced for transport problems in Cockburn and Shu 1989) was applied for compositional flow on realistic reservoir meshes. DG are very promising for such problems; however, since most industrial simulators rely on a data layout corresponding to that of first-order FV methods, the implementation of DG methods in industrial codes will be even more complicated than reconstruction-based higher-order FV methods.

In Klöfkorn et al. (2017), the authors compared three second-order schemes, each with several limiting techniques on different 2D and 3D grids, including corner-point examples. The study concluded that the second-order method with reconstruction based on optimization with linear programming (second-order LP method) was among the best methods for 3D cases. Moreover, the second-order LP method was most suitable for implementation in the full-scale reservoir simulator OPM Flow.

In this paper, we present and verify the implementation of the second-order LP method in a full-scale reservoir simulator. To show the method's capabilities, we run WAG and CO_2 injection scenarios on a medium-sized synthetic reservoir with an unstructured grid and the openly available Norne field. We constructed experiments that isolate and highlights the effects of components' transport, for which the simulation quality will deteriorate for the classical methods. The implementation is done in OPM Flow, an open reservoir simulator capable of modeling the black-oil model as fast and accurately as conventional commercial reservoir simulators (Rasmussen et al. 2021; Lauser et al. 2018; Sandve et al. 2018). The open-source access allows us to integrate new features directly and verify them on industry-standard test cases and field studies. This means that the second-order method is readily available for practical reservoir simulation. A reservoir engineer wanting to increase the accuracy and improve the fluid front positioning and resolution can start the simulation with the higher order flag `--enable-higher-order` instead of going through the time-consuming and work-intensive process of grid refinement.

The paper is organized as follows: We start by describing the first- and second-order finite volume methods on reservoir discretizations. Section 3 presents the numerical results on realistic cases. We summarise the results in Sect. 4.

2 Discretization

In this section, we discuss discretization used in the fully-implicit numerical method for solving the black oil model extended with a solvent component (22–24).

Let us start by introducing a computational grid $\mathcal{T}$ of the domain $\Omega \subset R^d$ and denoting $\partial\mathcal{T}$ as a tessellation of the boundary $\partial\Omega$. In the classical cell-centered finite volume method the unknown function is approximated by a piece-wise constant function, which takes the average value of the unknowns for each discretization element (Eymard et al. 2000). The finite volume method uses an integral formulation of the model Eqs. (22–24) applied on each element of the computational grid. For the sake of simplicity, all further derivations will be done for the water component equation from (23):

$$\frac{\partial}{\partial t}A_W - \nabla \cdot \frac{\lambda_w k}{B_w}\left(\nabla p_\alpha - \rho_\alpha g\right) = Q_W. \tag{1}$$

As for time discretization we use an implicit Euler method. Following the derivations from LeVeque (2002) and Eymard et al. (2000), we integrate the water component equation over

the element $E \in \mathcal{T}$, apply the Gauss theorem and get the implicit formulation of a classical finite volume scheme:

$$(\phi A_W)^{n+1} = (\phi A_W)^n + \frac{dt}{|E|} \oint_{\partial E} \frac{\lambda_w^{n+1} k}{B_w} \left(\nabla p_w^{n+1} - \rho_w g\right) \cdot \mathbf{n} \, \mathrm{d}x + Q_\kappa^{n+1}, \tag{2}$$

where the superscripts n and $n+1$ denote the time step, Q_κ^{n+1} is the averaged in time integral over the element of the right-hand side, A_W is the accumulation term of the water component, n_{ij} is the outer normal to the intersection e_{ij} and λ_w^{n+1} is the water mobility, which is equal to the ratio of the relative permeability function to the water viscosity. As we are modeling reservoirs, which typically are isolated from the surrounding rocks, a no-flow boundary condition is assumed.

The two accumulation terms are linear and can be easily computed using the averaged values at each element. The integral in the right-hand side can, in principle, be approximated using any quadrature rule, we choose the midpoint rule:

$$\begin{aligned} &\oint_{\partial E} \frac{\lambda_w^{n+1} k}{B_w} \left(\nabla p_w^{n+1} - \rho_w g\right) \cdot \mathbf{n} \, \mathrm{d}x \\ &\qquad \approx \sum_{e_{ij} \in \partial E} |e_{ij}| \frac{\lambda_w^{n+1}(x_{ij}) k(x_{ij})}{B_w} \left(\nabla p_w^{n+1}(x_{ij}) - \rho_w(x_{ij}) g(x_{ij})\right) \cdot \mathbf{n}_{ij}. \end{aligned} \tag{3}$$

To calculate the gradient of the pressure we use the standard two-point flux approximation (Aavatsmark 2007; Alyaev et al. 2014) and the gravitational acceleration is approximated using the arithmetic mean. The mobility function λ_w^{n+1}, on the other hand, is non-trivial to compute. Let us denote it as:

$$\lambda_w^{n+1}(x_{ij}) = \lambda_w^{ij}. \tag{4}$$

The reconstruction of the phase mobility function will determine the type of the numerical method. As discussed in Lie et al. (2020), one can use either primary variables or the phase mobilities to reconstruct the second-order method. We compared the results when reconstructing primary variables and mobilities in the test run and observed that the choice did not influence the results significantly. Also reconstructing mobilities extends the current version of OPM more naturally. Therefore, in this paper, we chose to reconstruct the phase mobilities in the presented second-order method implementation.

In the first-order method, we use a simple upwind scheme for the mobility evaluation and the mid-point rule for the integral:

$$\lambda_w^{ij} = \begin{cases} \lambda_{ij}^- & \text{if } \left(\nabla p_w^{n+1} - \rho_w g\right) \cdot n_{ij} \le 0, \\ \lambda_{ij}^+ & \text{otherwise,} \end{cases} \tag{5}$$

where λ_{ij}^- is the mobility function on the element E and λ_{ij}^+—on the neighboring element E'. For boundary elements, the boundary condition value is used instead of the neighboring value or the same value in the case of a no-flow boundary, which is typical in reservoir modeling. Now, when all the terms in (2) are discretized, we can repeat the same procedure with the obvious modifications for the other components and get a non-linear system of equations. The solution of the system is computed by the Newton method with an automatic differentiation approach for the computation of Jacobian matrices. More details on the use of automatic differentiation in OPM can be found in Lauser et al. (2018).

For second-order methods we use a reconstructed linear function for every element:

$$\begin{aligned} L_E(x) &: = \lambda_E + \nabla L_E \cdot (x - \mathbf{w}_E), \\ L_E(\mathbf{w}_{E'}) &= \lambda_{E'}, \ \forall (E, E') \in \partial E, \end{aligned} \tag{6}$$

where $\mathbf{w}_E$ is the coordinates of the barycenter of the element E, and ∇L_E is the gradient that needs to be computed. For clarity of notation, we are dropping the water component index w in the mobility function. The same procedure is applied to all the mobility functions. Afterward, the same upwind scheme (5) is used, however, instead of λ_{ij}^- and λ_{ij}^+ we use the reconstructed L_{ij}^- and L_{ij}^+, which corresponds to the linear reconstructions functions evaluated on the intersections barycenter on element E and its neighbor E', respectively:

$$\lambda_w^{ij} = \begin{cases} L_{ij}^-(\mathbf{w}_{(E,E')}) & \text{if } \left(\nabla p_w^{n+1} - \rho_w g\right) \cdot n_{ij} \leq 0, \\ L_{ij}^+(\mathbf{w}_{(E,E')}) & \text{otherwise,} \end{cases} \tag{7}$$

In order to reconstruct the linear function we need to compute a gradient ∇L_E on each element. Below we discuss one possible approach to computing the gradient and therefore the linear function.

2.1 Linear Programming Reconstruction

One option to compute the gradient for the linear reconstructions is by solving an optimization problem for each cell. The idea of using linear programming (LP) for reconstruction and limiting the gradient was presented in May and Berger (2013). The algorithm for obtaining the linear reconstruction still consists of two steps: computation of the initial gradient and solving a constrained optimization problem, where the goal is to minimize the difference between the initial and the limited gradients while satisfying monotonicity constraints. Each component of the multidimensional gradient is limited separately using scalars, which means that in the end the direction of the initial gradient could be changed (in contrast with scalar limiters, when all gradient's components are multiplied with the same scalar). Chen and Li (2016) addressed this in the paper, where they presented a method that does not require an initial gradient computation. Indeed, since both the direction and the length of the gradient can be changed by the LP algorithm, it would be beneficial computationally to omit the initial gradient computation. As the authors point out in Chen and Li (2016), the two algorithms give different optimal solutions, however, they also present a theorem, where they prove that the obtained gradient is sufficiently close to the unlimited least squares gradient. The approach used in this paper is based on Chen and Li (2016) and will be briefly described below.

Linear programming is an optimization method to find an extremum of the objective function under the linear constraints (Nocedal and Wright 2006). In our case, we want to minimize the difference between reconstructed values and the cell-averaged values without violating the monotonicity conditions:

$$\min\{\lambda_E, \lambda_{E'}\} \leq L_E(\mathbf{w}_{E'}) \leq \max\{\lambda_E, \lambda_{E'}\}, \ \forall (E, E') \in \partial E. \tag{8}$$

This means that the reconstructed function evaluated in the center of every neighboring element should be bounded by minimum and maximum from the cell-averaged values of the current element and the neighbor. The elements center is calculated as an average of the corner coordinates. This will allow better numerical accuracy and reduce the numerical

diffusion without creating spurious oscillations. This monotonicity condition essentially acts like a minmod limiter applied to the gradient of the reconstructed function.

The standard form of the linear programming problem is:

$$\min \tilde{c}^{\mathrm{T}} \tilde{x} \text{ subject to } \tilde{A}\tilde{x} = \tilde{b},\ \tilde{x} \geq 0, \tag{9}$$

where $\tilde{c}$ and $\tilde{x}$ are vectors from $\mathbb{R}^n$, vector $\tilde{b}$ belongs to $\mathbb{R}^m$, and $\tilde{A} \in \mathbb{R}^{(m \times n)}$.

As it was said before, we are minimizing the total gaps between the reconstructed values and the cell-averaged values at the neighboring cells:

$$\delta(L) := \sum_{\forall (E,E') \in \partial E} \mid \lambda_{E'} - L_E(\mathbf{w}_{E'}) \mid, \tag{10}$$

which is equal to

$$\sum_{\forall (E,E') \in \partial E} \mid \lambda_{E'} - L_E(\mathbf{w}_{E'}) \mid = \sum_{\forall (E,E') \in \partial E} \mid \lambda_{E'} - \lambda_E - \nabla L_E \cdot (\mathbf{w}_{E'} - \mathbf{w}_E) \mid \tag{11}$$

$$= \sum_{\forall (E,E') \in \partial E} \mid v_{E'} - \tilde{v}_{E'} \mid, \tag{12}$$

where v and $\tilde{v}$ are variables defined as

$$\tilde{v}_{E'} = \nabla L_E \cdot (\mathbf{w}_{E'} - \mathbf{w}_E) \quad \text{and} \quad v_{E'} := \lambda_{E'} - \lambda_E. \tag{13}$$

As it was pointed out in Chen and Li (2016), the difference $v_{E'} - \tilde{v}_{E'}$ has the same sign as $v_{E'}$, which brings us to the following LP problem:

$$\begin{aligned} &\max \sum_{\forall (E,E') \in \partial E} \operatorname{sgn}(v_{E'})(\mathbf{w}_{E'} - \mathbf{w}_E) \cdot \nabla L_E \\ &\text{subject to } v^-_{E'} \leq (\mathbf{w}_{E'} - \mathbf{w}_E) \cdot \nabla L_E \leq v^+_{E'}, \end{aligned} \tag{14}$$

where

$$\begin{aligned} v^-_{E'} &= \min\{0, \lambda_E - \lambda_{E'}\}, \\ v^+_{E'} &= \max\{0, \lambda_E - \lambda_{E'}\}. \end{aligned} \tag{15}$$

The transition from max to min formulation in (14) is straightforward, so we will just write down how matrix A and vectors b and c from (9) will look like. The unknown vector x is the gradient of the linear reconstruction $x = [\nabla L^x_E, \nabla L^y_E, \nabla L^z_E]^{\mathrm{T}}$. The matrix A and vectors c and b are:

$$c = \begin{bmatrix} \sum_{\forall (E,E') \in \partial E} \operatorname{sgn}(v_{E'})(x_{E'} - x_E) \\ \sum_{\forall (E,E') \in \partial E} \operatorname{sgn}(v_{E'})(y_{E'} - y_E) \\ \sum_{\forall (E,E') \in \partial E} \operatorname{sgn}(v_{E'})(z_{E'} - z_E) \end{bmatrix}, \tag{16}$$

$$A = \begin{bmatrix} x_{E_1} - x_E & y_{E_1} - y_E & z_{E_1} - z_E \\ \vdots & \vdots & \\ x_{E'} - x_E & y_{E'} - y_E & z_{E'} - z_E \\ -(x_{E_1} - x_E) & -(y_{E_1} - y_E) & -(z_{E_1} - z_E) \\ \vdots & \vdots & \\ -(x_{E'} - x_E) & -(y_{E'} - y_E) & -(z_{E'} - z_E) \end{bmatrix}, \quad b = \begin{bmatrix} v^+_{E_1} \\ \vdots \\ v^+_{E'} \\ -v^-_{E_1} \\ \vdots \\ -v^-_{E'} \end{bmatrix}, \tag{17}$$

We start the iteration process with zero gradient

$$x = \left[\nabla L^x_E, \nabla L^y_E, \nabla L^z_E\right]^{\mathrm{T}} = [0,\, 0,\, 0]^{\mathrm{T}}, \tag{18}$$

which corresponds to the first-order scheme. Following May and Berger (2013) and Chen and Li (2016), an all-inequality simplex method is used as a linear programming solver. Compared to the usual simplex method, the advantage of this method is that it is much faster on problems where the number of constraints is bigger than the number of unknowns, which is the case for polyhedral meshes where each cell has many neighbors. A detailed description of the all-inequality simplex algorithm can be found in the appendix of May and Berger (2013). Further on in this paper, we will refer to this method as the second-order LP method.

3 Numerical Results

In this chapter, we test and compare the methods on two realistic reservoirs. In the first test case, we compare the methods on a medium-sized realistic reservoir with an unstructured corner point grid when modeling water-alternating gas injection (WAG), Sect. 3.1. This test case provides a complicated setup in terms of the presented fluids and their interactions; however, the reservoir is relatively small and does not contain complex features like faults, regions, etc. After that, we present the simulation results on the Norne field, an open data set of a real reservoir on the Norwegian Continental Shelf. We run three different experiments on the Norne field. First, we model a piston-type injection on homogeneous and heterogeneous Norne fields to observe reduced smearing for the second-order method, Sect. 3.2. Second, we run the first-order method on a refined Norne model to validate the front resolution of both methods, Sect. 3.2. And finally, we present the carbon dioxide injection test case on a heterogeneous Norne field with its full complexity, Sect. 3.3. We run several setups to compare the accuracy of the front position of the second-order method in less and more complex conditions. We constructed experiments that isolate and highlights effects of components' transport, for which the simulation quality will deteriorate for the classical methods.

3.1 Medium-Sized Realistic Reservoir

In this test case, we model a three-phase flow with four components on a 3D domain with an unstructured grid. The phases are the same as in the standard black-oil model: oleic, gaseous, and aqueous. The components are oil, gas, water, and solvent. The domain is a rectangular hexahedron, which is 7014 m long in X and Y directions and 120 m long in the Z direction, having $14 \times 14 \times 3$ grids cells, each 501 m by 501 m by 40 m, see left part of Fig. 1. It is located 8325 m below the surface and has two wells—an injector in the lower

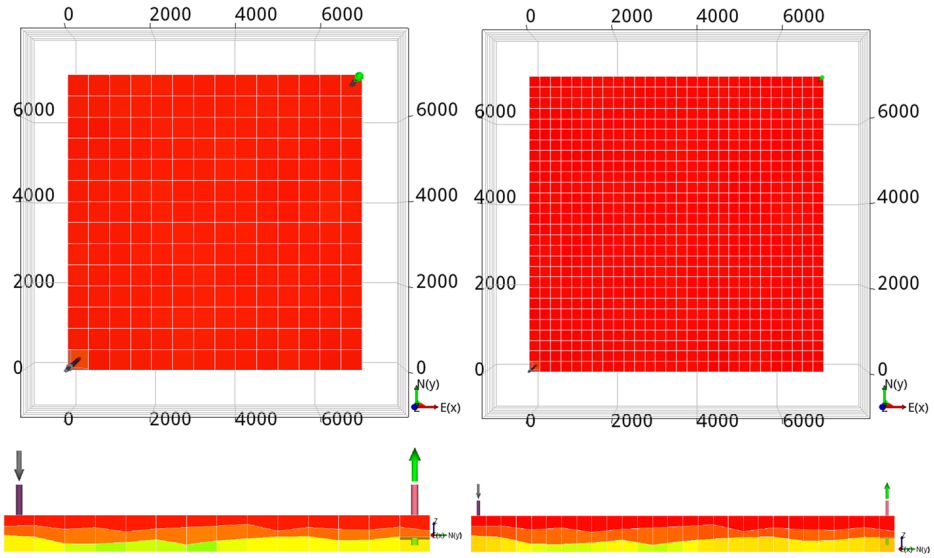

Fig. 1 Top and west views of the coarse and refined medium-sized reservoir, described in the Sect. 3.1

left corner and a producer in the upper right. Both wells are under rate control with a target rate of 12,000 STB/day. The BHP target is set to 1000 psia for the producer and 10000 psia for the injector. The fluid properties are adopted from the well-known SPE5 benchmark (Killough and Kossack 1987). On this reservoir, we will run both first- and second-order methods. As we do not know the "true" solution, we will also run first-order method on a refined grid ($28 \times 28 \times 3$ grid cells) and view it as the reference solution. Note, we refine the grid only in X and Y direction, since fluid flow mostly happening in this direction.

The schedule in this test case emulates WAG (water alternating gas) injection with 1 and 5-year cycles. During the first 2 years, we perform depletion of the reservoir and only produce without injecting. After the first 2 years, we do 20 years of WAG injection with 1 year cycle, meaning that we inject water for one full year and follow it with gas injection for the next full year, both with a constant rate. Afterward, we change the schedule and inject gas for 5 consecutive years and water for the next 5 years. The simulation is finished with 5 more years of gas injection, which gives us 15 years of WAG injection with 5-year cycle. The schedule has two cycle schedules in one, first with the short and second with the long span, which will help us to show how the methods perform in different conditions.

Initially, the reservoir is filled with oil in the top layers and water in the bottom. The oil in the reservoir is light and has a certain fraction of gas mixed into it. The injected gas consists only of solvent, which is a mix of several components, but mostly consists of methane.

Let us now examine the production rate curves for all the components. We will start with the solvent production rate, see Fig. 2. Three curves in the figure correspond to the result produced by the first- and second-order methods on the coarse grid and a reference first-order method on the refined grid. Throughout the whole simulation time, we see an agreement between the second-order method and the reference method. We zoom into three parts of the production rate curve: first, to the time when the solvent first reaches the producer, second, the waves of 1-year WAG injection, and third, to the last wave of the 5-year WAG injection. In the first, we see that both the second-order method and first-order on fine grid method predict later arrival time and sharper front for the solvent production.

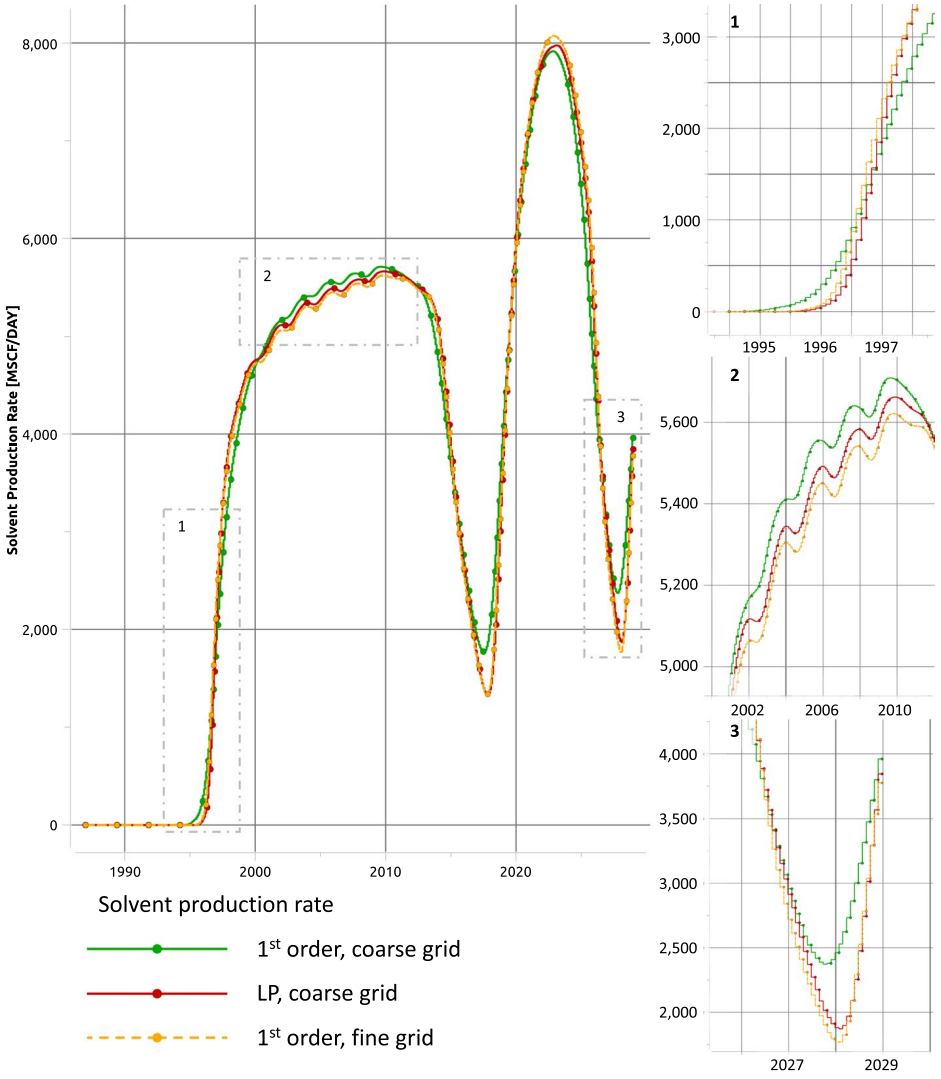

Fig. 2 Solvent production rate: whole simulation time on the left and three zoom-ins on the right

This agrees very well with the results obtained for the simple immiscible fluid displacement cases, presented in Klöfkorn et al. (2017). We see a 3-month difference between the first- and second-order methods' curves reaching 200 MSCF/day level of production. In the two later stages, we also see a noticeable difference in the results: the second-order method tends to show later arrival of the fronts. It also predicts higher local maximums and lower local minimums, which is expected as the second-order method reduces smearing.

Gas and oil production rates are shown in Fig. 3. We plot them together because they share one interesting feature. We observe a gas wave formed due to the pressure drop in the reservoir during the first 2 years of depletion. The gas that was initially dissolved in the oil was realized and formed a separate sharp gas wave. And therefore, around 1995 we observe a pick in gas production and a drop in oil production. The first-order method on the refined grid confirms the front position obtained by the second-order method.

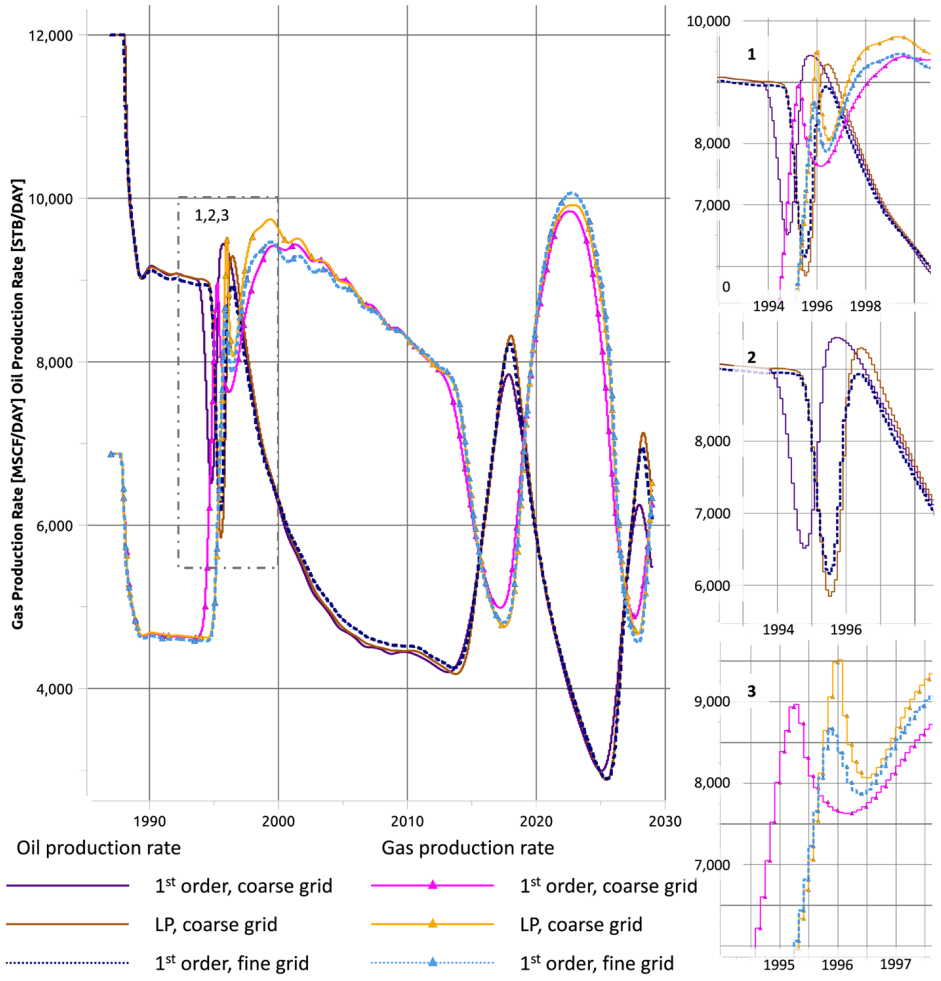

Fig. 3 Gas and oil production rates during the whole simulation time on the right and zoom-ins on the right

The second-order method, therefore, outperforms the first order in the accuracy of the positioning of the sharp front.

Let us examine closely the water production rate, see Fig. 4. Here we see two "waves": first at the beginning of 1988 and second at the beginning of 1995. However, the second-order method gives identical results to the first order on the first wave and predicts a later arrival of the waterfront for the second wave. The reason for it is the nature of those peaks in water production. The water production rate is plotted together with the solvent production rate, such that we know when the injected fluid reaches the producer. And in our case, it happens only in 1995 (May 1995 for the first order and December 1995 for the second to reach 50 STB/day production rate). This means that before that time we produce the reservoir water and its behavior, especially during the first 2 years when there is no injection, is pressure driven. And as we do not improve the pressure calculations, the production curves for the first- and second-order methods are identical. However, when the injected water reaches the producer, we see the transport

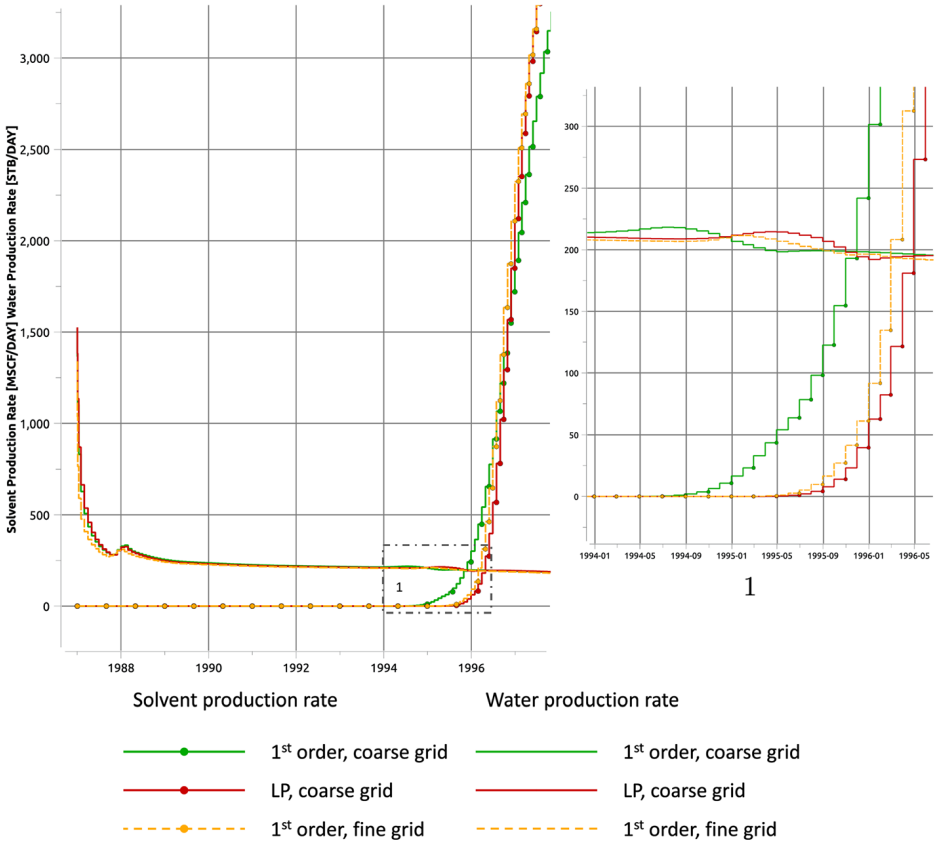

Fig. 4 Production rates of water and solvent during the whole simulation on the right and zoom in on the left

phenomena and the second-order method predicts the later arrival of the waterfront, and the reference method agrees with it.

In summary, whenever the transport phenomena were dominant for all production rate curves, we saw an improved front position and reduced smearing for the second-order method compared to the first-order. The result was verified by the result of the first-order method on the refined grid.

3.2 Simulation on the Norne Field

The Norne field is an oil and gas sandstone reservoir on the Norwegian continental shelf. It is one of the few fields where the simulation model together with production data was published under an open content license. Since then it became a benchmark field model.

In order to compare the performance of the first- and second-order FV methods on a realistic reservoir, we are going to use the benchmark simulation model with a grid block size of approximately 100 m and a refined Norne model with a grid block size of approximately 50 m. Since this is a real reservoir, we do not know the analytical solution even to the simplest production/injection schedule. That is why we will use the Norne field with the refined grid, such that we can compare the results of the second-order method with

the results of the first-order method on the refined grid. The previous study on the simple geometry reservoirs (Kvashchuk et al. 2019) suggests that the second-order method accuracy in the front position detection can be compared with the first-order only on the refined grid.

Following Lie et al. (2020) we construct a simple piston-type injection example on the Norne field. In order to do so, first we use homogeneous porosity and permeability throughout the whole reservoir and neglect regions and fault multipliers, that are used in the original model. Second, we fill the whole reservoir with "red" fluid and inject "blue" fluid through the east side. Red and blue liquids have the same properties such that we can isolate the fluid displacement effects. Third, we put two production wells on the other side such that the injected fluid can flow through the reservoir. Finally, we run simulations for almost 25 years (9000 days) for both the first- and second-order methods. The field after 4.5 years is shown in Fig. 5. The top row corresponds to the homogeneous case and the bottom to the heterogeneous. The difference is more visible for the homogeneous grid, which also agrees with the curves in Fig. 7. The curves in the figure represent the ratio of the red fluid produced in the reservoir to the total fluid volume. Figure 6 shows the actual difference between the results on each grid cell in December 2002, 5 years and 1 month after the simulation started. The curves in Fig. 7 agrees with the results reported in Lie et al. (2020). We see that the second-order finite volume method reduces the smearing effect and that the reduction is more pronounced on the homogeneous media.

We verify the results obtained above by running the same simulation on the refined Norne grid. We refine the original grid block, which is approximately 100 m, in X and Y directions, and get a refined Norne model with a grid block size of approximately 50 m. Since we are focusing on fluid displacement and it is happening in the X–Y plane, we do not refine cells in the Z direction, where mostly gravity effects take place. The results on the refined grid confirm the behavior of the second-order method that we saw in Klöfkorn et al. (2017) and Kvashchuk et al. (2019) and in the previous example—it gives a sharper front and predicts its position closer to what is obtained with the first-order method on the refined grid, see Fig. 8. In Fig. 8 we see very good agreement between the higher order method and the first-order method on the refined grid in predicting the arrival time of the fluid front. However, when it comes to later stages of production (around the year 2020),

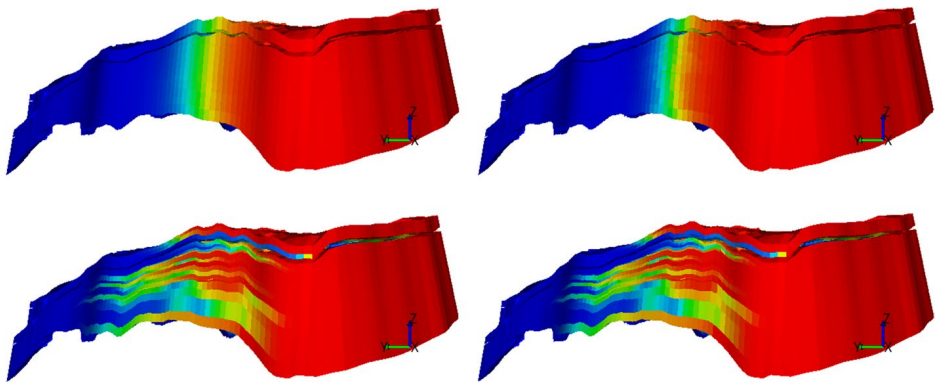

Fig. 5 Homogeneous (top) and heterogeneous (bottom) Norne field in the middle of production, east view. The results for the first order are on the left, and for the second order on the right side. Here we can clearly see the reduced smearing for the second-order method on the homogeneous Norne

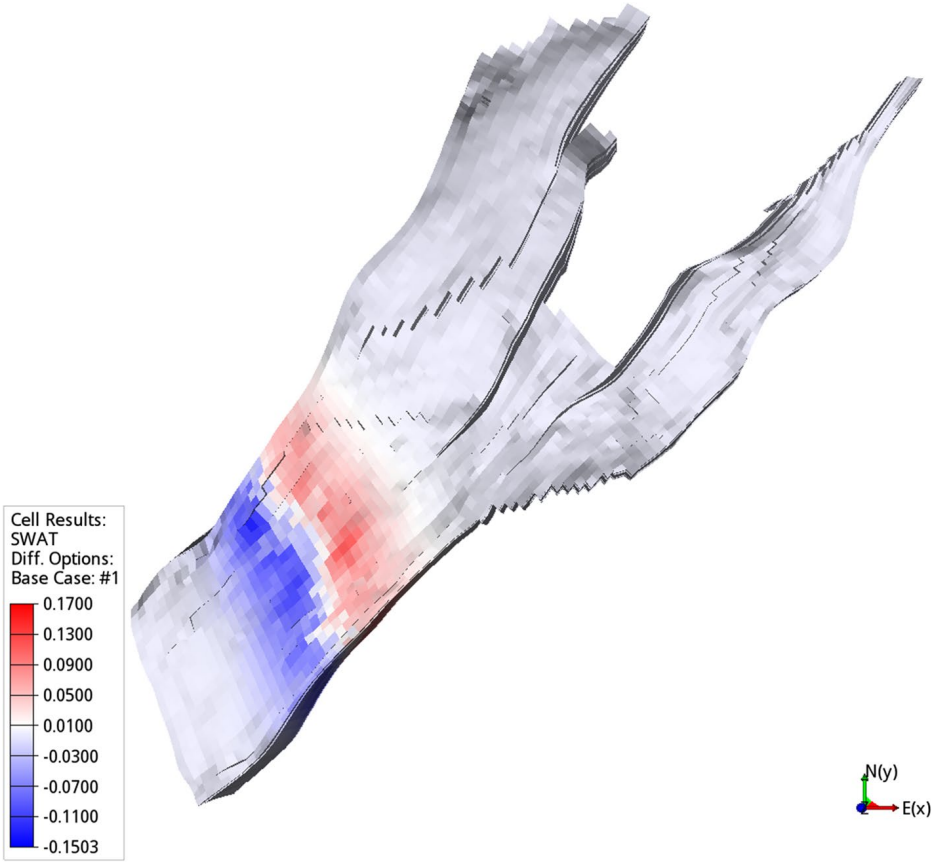

Fig. 6 Difference between the results of the first- and second-order methods

the first- and second-order methods on the coarse grid produce the same result, while the first-order on the refined grid predicts a lower number. The results of the first- and second-order methods on the same grid differ only in how much the front is spread: it is spread more for the first-order method and is sharper (less spread) for the second-order. That is why after the front is resolved they agree and stay on the same level.

3.3 Carbon Dioxide Injection on Norne

In order to test the performance of the second-order method in more realistic conditions, we set up a CO_2 injection test case. Even though the Norne field is not a primary candidate for the CO_2 injection, it can give valuable insight into how the methods compare to the realistic field scale model.

For ease of simulation, we will have only two wells—one injector and one producer. However, the rest of the complexity of the field is present—we have faults, regions, heterogeneity, etc. Also, we change the initial conditions—in this test case, the reservoir is filled only with oil at the start of the simulation. The CO_2 is injected at a constant rate of 100,000 sm^3/day from the start of the simulation and for consecutive 5000 days. We assume the injected CO_2 is miscible in oil when its gas fraction is more than 0.01. We use

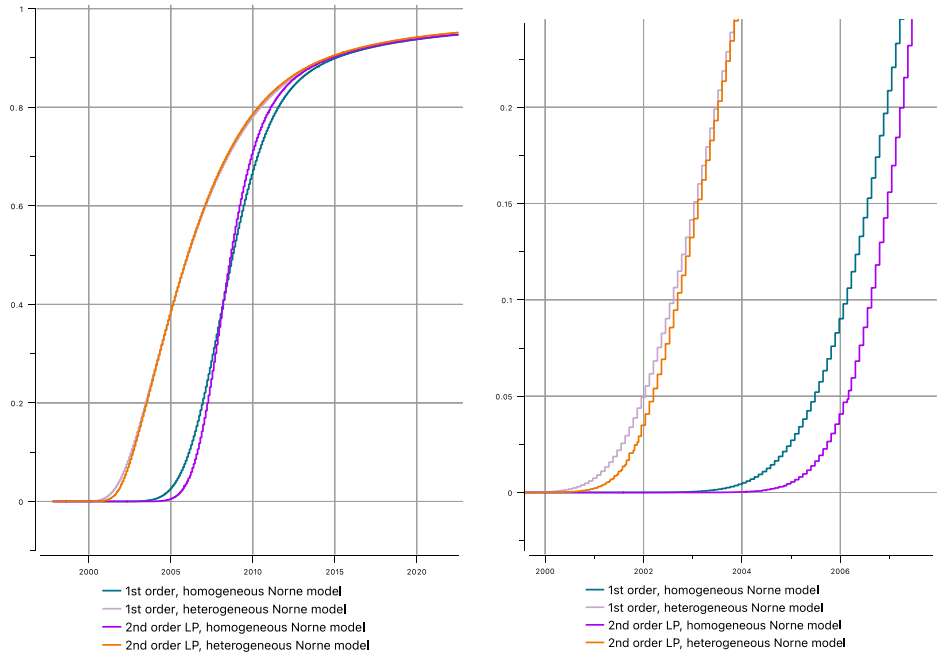

Fig. 7 The ratio of red liquid produced compared to the volume of total liquids produced in the Norne reservoir. The full simulation time is on the left, and on the right we zoom in from 2000 to 2006

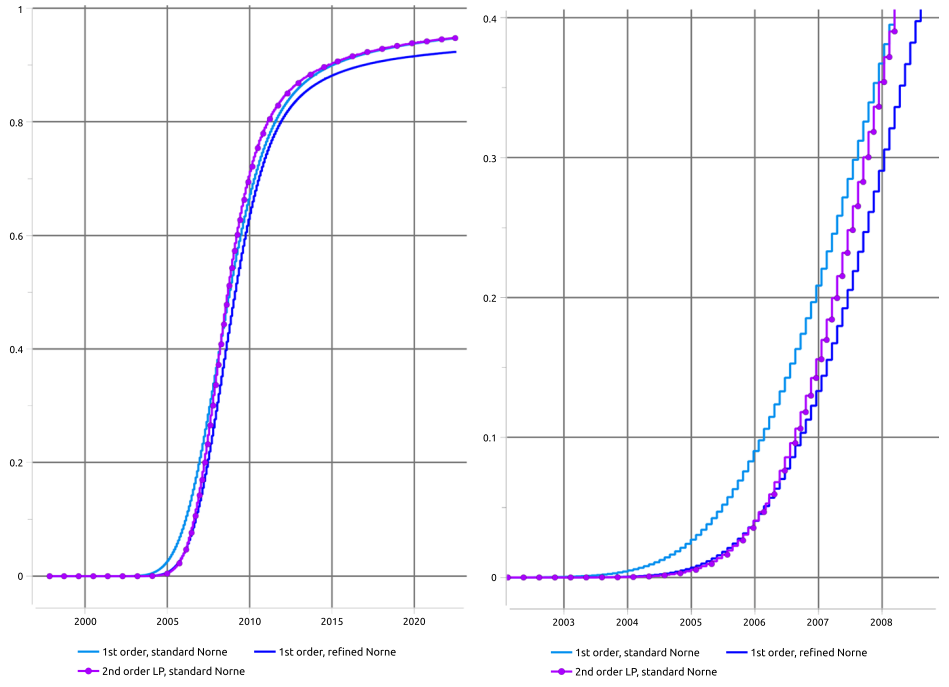

Fig. 8 The ratio of red liquid produced compared to the volume of total liquids produced in the standard and refined Norne reservoir. The full simulation time is on the left and from 2003 to 2008—on the right

the "MISC" keyword (Baxendale 2022) to define that fluid is immiscible at low concentrations and switches to miscible behavior when the concentration increases. For all simulated scenarios, the production well is placed in the middle of the reservoir and is controlled by a liquid rate target of 4000 sm^3/day. It is a horizontal well in layer 13 with wellhead coordinates $I = 13$ and $J = 39$. The injector well will be placed in three different places: first, the two wells are in the same compartment meaning that there are no faults between them, second—the injector and producer are separated by just one fault; and third, the injector well is placed in the corner of the reservoir surrounded by faults, see Fig. 9.

The resulting CO_2 (solvent) production rate curves are presented in Fig. 10. In the first scenario, solvent reaches the producer faster than in the other two as there are no faults between the two wells. We also observe 105 days difference between the first- and second-order methods' prediction of when the solvent production rate reaches 1000 sm^3/day. In two other cases, when we have faults between the injector and producer, we see the same trend—the second-order method predicts the later arrival of the carbon dioxide; however, the difference is less: solvent production rate reaches 1000 sm^3/day with 70 days difference for the "behind one fault" case and 65 days difference for the "corner" case. It shows that the increased complexity of the reservoir can overshadow the effects gained by using a higher-order computational method. However, we still observe a relevant difference in the front positioning when using the higher-order method.

4 Conclusion

We showed that the presented second-order linear programming method predicts the fluid front position more accurately and decreases the numerical diffusion on realistic reservoir models. We also verified the results with the first-order method on the refined grid, both for the medium-sized reservoir and the Norne test case. In all the presented test cases, we saw that the second-order method would predict the position of the fluid front as accurately as the first order on the refined grid.

We also observed that the result depends on the type of fluid flow. When pressure effects dominated the fluid flow, we did not observe significant improvements, which is not surprising as we did not improve the accuracy of the pressure computation. In general, the more complex fluid flow we have, the harder it is to model the process and interpret the result. For example, in the simple scenario of carbon dioxide injection on a complex Norne field, we observed that the difference between the first- and second-order methods results depends on the number of faults between the injector and the producer. And for piston-type injection, the smearing was reduced more for

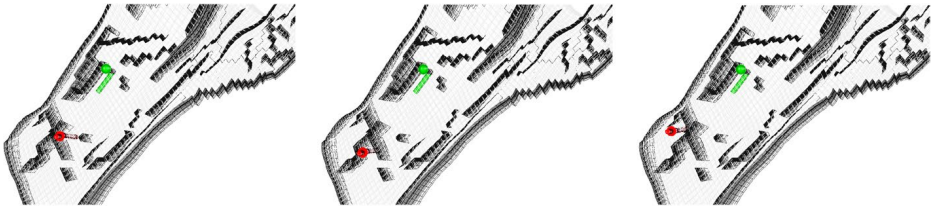

Fig. 9 Positions of the wells in the Norne CO_2 injection scenarios: left for the wells in the same compartment, middle–wells are separated by a fault, right—injection well in the corner

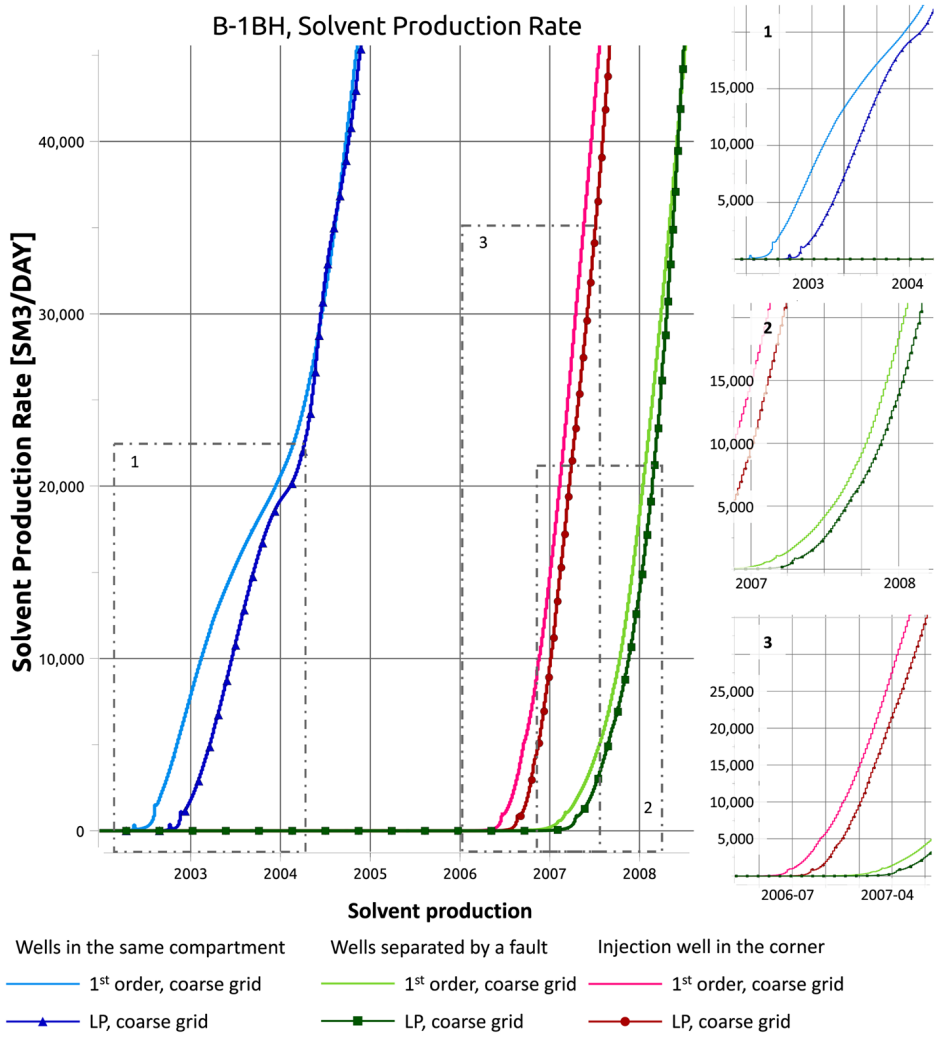

Fig. 10 Solvent production rate for the three scenarios of solvent injection on Norne. The subplots zoom into the times of solvent arrival for each scenario: (1) Wells in the same compartment; (2) Wells separated by a fault; (3) Injection well in the corner

the homogeneous grid than for the heterogeneous. More testing in realistic settings is needed to answer under which conditions the higher-order methods are most helpful.

A significant advantage of the presented second-order method is the fact that it is implemented in the open reservoir simulator OPM Flow (Open porous media), where using the higher order method is a simple task of switching the flag when starting the simulation. This allows for immediate usage and further testing in realistic settings and facilitates future improvements and extensions of the presented method. This means faster learning and more informed decisions when modeling the fluid flow in the reservoir.

Appendix 1. Modeling Equations

Black-Oil Model

The black-oil model is a three-phase three-component model, which is an industry-standard model for the reservoir processes (Chen et al. 2006). Let us state below the assumptions and equations that are governing the model. As phases and components share the same names (gas, water, and oil), we will use different indices in order to distinguish between them: index $\alpha \in \{w, g, o\}$ ("water", "gas" and "oil") for phases and index $\kappa \in \{W, G, O\}$ ("Water", "Gas" and "Oil") for components. The miscibility assumptions are:

- water and gas phases are immiscible,
- the water phase is composed only of water,
- gas phase may contain vaporized oil,
- the oil phase is assumed to be a mixture of gas and oil components.

The densities of the phases are determined by so-called formation volume factors, which is the ratio of the phase volume measured at reservoir conditions ρ_α to the phase volume measured at standard conditions $\rho_{\alpha sc}$:

$$B_\alpha := \frac{\rho_\alpha}{\rho_{\alpha sc}}. \tag{19}$$

Here and later on, we will use subscript *sc* to indicate that quantity was measured at standard condition.

The water density can be computed using just the water volume formation factor. In order to compute oil and gas densities, we need to introduce gas solubility R_s (also called dissolved gas-oil ratio), which is the volume of gas at standard conditions V_{Gsc} divided by the volume of oil at standard conditions V_{Osc} given that they were both obtained from some amount of oil phase at reservoir conditions:

$$R_s := \frac{V_{Gsc}}{V_{Osc}}, \tag{20}$$

and oil volatility R_v, which is the volume of oil at standard conditions V_{Osc} divided by the volume of gas at standard conditions V_{Gsc} given that they were both obtained from some amount of gas phase at reservoir conditions:

$$R_v := \frac{V_{Osc}}{V_{Gsc}}. \tag{21}$$

This allows for calculating all quantities required for the mass-conservation equations for each component. All three equations share the same structure:

$$\frac{\partial}{\partial t}(\phi A_\kappa) + \nabla \cdot v_\kappa = Q_\kappa, \tag{22}$$

where ϕ is the porosity, A_κ—the accumulation term, v_κ and Q_κ is the mass flux and source or sink term of the phase κ respectively. For each component κ the accumulation term and the mass flux are:

$$\begin{aligned} A_W &= \frac{S_w}{B_w}, & v_W &= \frac{u_w}{B_w}, \\ A_O &= \frac{S_o}{B_o} + \frac{R_v S_g}{B_g}, & v_O &= \frac{u_o}{B_o} + \frac{R_v u_g}{B_g}, \\ A_G &= \frac{S_g}{B_g} + \frac{R_s S_o}{B_o}, & v_G &= \frac{u_g}{B_g} + \frac{R_s u_o}{B_o}. \end{aligned} \tag{23}$$

The phase flux u_α is determined by the standard multi-phase Darcy equation (Nordbotten and Celia 2011), i.e.

$$u_\alpha = -\frac{k_{r,\alpha}}{\mu_\alpha} k\big(\nabla p_\alpha - \rho_\alpha g\big), \tag{24}$$

where k is the permeability, g - gravity, and $k_{r,\alpha}$, μ_α, ρ_α, p_α are relative permeability, viscosity, density and pressure of the phase α respectively.

The Model with Solvent

In this study, we use the black-oil model extended with the solvent equation instead of a fully compositional model. This choice has been made in OPM to reduce the computational time and to ease the implementation of the CO_2-EOR injection scenario (Sandve et al. 2018). However, the proposed second-order methods are not limited to this model and in principle can be used in any other formulation. To extend the black-oil model we add the following equation:

$$A_s = \frac{S_s}{B_s}, \qquad v_s = \frac{u_s}{B_s}. \tag{25}$$

Additionally, the presence of solvent influences relative permeability and viscosity, and these become effective relative permeability $k_{r\alpha e}$ and effective viscosity $\mu_{\alpha e}$. The effective properties depend on the miscibility factor M and are defined as follows:

$$k_{roe} = M \cdot \frac{S_o - S_{or}}{S_n - S_{gc} - S_{or}} \cdot k_{rn} + (1 - M) \cdot k_{ro}, \tag{26}$$

$$k_{rge} = M \cdot \frac{S_g + S_s - S_{gc}}{S_n - S_{gc} - S_{or}} \cdot k_{rn} + (1 - M) \cdot \frac{S_g}{S_g + S_s} k_{rgt}, \tag{27}$$

$$k_{rse} = M \cdot \frac{S_g + S_s - S_{gc}}{S_n - S_{gc} - S_{or}} \cdot k_{rn} + (1 - M) \cdot \frac{S_s}{S_g + S_s} k_{rgt}. \tag{28}$$

Here k_{rgt} is the total relative permeability of the gas phase, S_n is the total hydrocarbon saturation and S_{or} and S_{gc} is the residual oil saturation and the critical gas saturation, respectively.

Effective viscosities are calculated using the Todd-Longstaff mixing parameter w (Todd and Longstaff 1972):

$$\mu_{oe} = \mu_o^{1-w} \cdot \mu_{mos}^{w}, \quad \mu_{ge} = \mu_g^{1-w} \cdot \mu_{msg}^{w}, \quad \mu_{se} = \mu_s^{1-w} \cdot \mu_m^{w}. \tag{29}$$

The fully mixed viscosities for the oil and solvent mixture μ_{mos}, the gas and solvent mixture μ_{msg}, and the oil, gas, and solvent mixture μ_m are computed using the standard mixing rule (Todd and Longstaff 1972).

Acknowledgements Anna Kvashchuk acknowledges Alexey Khrulenko for help with the refined Norne model and many fruitful discussions. Anna Kvashchuk also acknowledges Equinor ASA for support. Robert Klöfkorn acknowledges the INTPART project INSPIRE (274883). Anna Kvashchuk and Robert Klöfkorn acknowledge the Research Council of Norway and the industry partners, ConocoPhillips Skandinavia AS, Aker BP ASA, Eni Norge AS, Equinor ASA, Neptune Energy Norge AS, Lundin Norway AS, Halliburton AS, Schlumberger Norge AS, Wintershall Norge AS, and DEA Norge AS, of The National IOR Centre of Norway for support. Tor Harald Sandve acknowledges the CLIMIT program (617115) and Equinor ASA for support. The authors three anonymous reviewers for thoughtful comments that helped improve the manuscript.

Funding Open access funding provided by University of Stavanger & Stavanger University Hospital. Authors Anna Kvashchuk and Robert Klöfkorn received support from the Research Council of Norway and the industry partners ConocoPhillips Skandinavia AS, Aker BP ASA, Eni Norge AS, Equinor ASA, Neptune Energy Norge AS, Lundin Norway AS, Halliburton AS, Schlumberger Norge AS, Wintershall Norge AS, and DEA Norge AS, of The National IOR Centre of Norway (230303). Author Anna Kvashchuk has received research support from Equinor ASA. The contribution of Robert Klöfkorn was also supported by the INTPART project INSPIRE (274883). The contribution of Tor Harald Sandve was supported by the CLIMIT program (617115). The authors have no relevant financial or non-financial interests to disclose. All authors contributed to the study's conception and design. Data collection and analysis were performed by Anna Kvashchuk, Robert Klöfkorn, and Tor Harald Sandve. The first draft of the manuscript was written by Anna Kvashchuk, and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.The datasets generated during and/or analysed during the current study are available in the second-order-opm-tests repository, https://github.com/kvashchuka/second-order-opm-tests.

References

Aavatsmark, I.: Interpretation of a two-point flux stencil for skew parallelogram grids. Comput. Geosci. **11**(3), 199–206 (2007)

Alyaev, S., Keilegavlen, E., Nordbotten, J.M.: Analysis of control volume heterogeneous multiscale methods for single phase flow in porous media. Multiscale Model. Simul. **12**(1), 335–363 (2014)

Baxendale, D.: OPM flow reference manual (2022-10). Open Porous Media Initiative (2022)

Bell, J., Shubin, G.: Higher-order godunov methods for reducing numerical dispersion in reservoir simulation. In: SPE Reservoir Simulation Symposium. OnePetro (1985)

Blunt, M., Rubin, B.: Implicit flux limiting schemes for petroleum reservoir simulation. J. Comput. Phys. **102**(1), 194–210 (1992)

Celia, M.A., Nordbotten, J.M.: Practical modeling approaches for geological storage of carbon dioxide. Groundwater **47**(5), 627–638 (2009)

Chen, L., Li, R.: An integrated linear reconstruction for finite volume scheme on unstructured grids. J. Sci. Comput. **68**(3), 1172–1197 (2016)

Chen, W., Durlofsky, L., Engquist, B., Osher, S.: Minimization of grid orientation effects through use of higher order finite difference methods. SPE Adv. Technol. Ser. **1**(02), 43–52 (1993)

Chen, Z., Huan, G., Ma, Y.: Computational Methods for Multiphase Flows in Porous Media, vol. 2. SIAM, Philadelphia (2006)

Class, H., Ebigbo, A., Helmig, R., Dahle, H.K., Nordbotten, J.M., Celia, M.A., Audigane, P., Darcis, M., Ennis-King, J., Fan, Y., et al.: A benchmark study on problems related to CO_2 storage in geologic formations. Comput. Geosci. **13**(4), 409–434 (2009)
Coats, K.H., Thomas, L., Pierson, R.: Compositional and black oil reservoir simulation. In: SPE Reservoir Simulation Symposium. OnePetro (1995)
Cockburn, B., Shu, C.-W.: TVB Runge–Kutta local projection discontinuous Galerkin finite element method for conservation laws. II. general framework. Math. Comput. **52**(186), 411–435 (1989)
Contreras, F., Lyra, P., Souza, M., Carvalho, Dd.: A cell-centered multipoint flux approximation method with a diamond stencil coupled with a higher order finite volume method for the simulation of oil-water displacements in heterogeneous and anisotropic petroleum reservoirs. Comput. Fluids **127**, 1–16 (2016)
Durlofsky, L.J., Engquist, B., Osher, S.: Triangle based adaptive stencils for the solution of hyperbolic conservation laws. J. Comput. Phys. **98**(1), 64–73 (1992)
ECLIPSE^{TM} 2014.2 (2014). http://www.software.slb.com/products/eclipse
Eymard, R., Gallouët, T., Herbin, R.: Finite volume methods. In: Handbook of Numerical Analysis, vol. 7, pp. 713–1018 (2000)
Flemisch, B., Darcis, M., Erbertseder, K., Faigle, B., Lauser, A., Mosthaf, K., Müthing, S., Nuske, P., Tatomir, A., Wolff, M., et al.: Dumux: Dune for multi-phase, component, scale, physics, flow and transport in porous media. Adv. Water Resour. **34**(9), 1102–1112 (2011)
Geiger-Boschung, S., Matthäi, S.K., Niessner, J., Helmig, R.: Black-oil simulations for three-component, three-phase flow in fractured porous media. SPE J. **14**(02), 338–354 (2009)
Harten, A.: High resolution schemes for hyperbolic conservation laws. J. Comput. Phys. **135**(2), 260–278 (1997)
Jiang, G.-S., Shu, C.-W.: Efficient implementation of weighted ENO schemes. J. Comput. Phys. **126**(1), 202–228 (1996)
Jung, Y., Pau, G.S.H., Finsterle, S., Pollyea, R.M.: Tough3: a new efficient version of the tough suite of multiphase flow and transport simulators. Comput. Geosci. **108**, 2–7 (2017)
Killough, J.E., Kossack, C.A.: Fifth comparative solution project: evaluation of miscible flood simulators. In: SPE Symposium on Reservoir Simulation (1987). https://doi.org/10.2118/16000-MS
Klemetsdal, Ø.S., Rasmussen, A.F., Møyner, O., Lie, K.-A.: Efficient reordered nonlinear gauss-seidel solvers with higher order for black-oil models. Comput. Geosci. **24**(2), 593–607 (2020). https://doi.org/10.1007/s10596-019-09844-5
Klöfkorn, R., Kvashchuk, A., Nolte, M.: Comparison of linear reconstructions for second-order finite volume schemes on polyhedral grids. Comput. Geosci. (2017). https://doi.org/10.1007/s10596-017-9658-8
Kvashchuk, A., Klöfkorn, R., Sandve, T.H.: Comparison of higher order schemes on complicated meshes and reservoirs. In: SPE Reservoir Simulation Conference. OnePetro (2019)
Lake, L.W.: Enhanced oil recovery (1989)
Lamine, S., Edwards, M.G.: Multidimensional upwind schemes and higher resolution methods for three-component two-phase systems including gravity driven flow in porous media on unstructured grids. Comput. Methods Appl. Mech. Eng. **292**, 171–194 (2015)
Lauser, A., Rasmussen, A., Sandve, T., Nilsen, H.: Local forward-mode automatic differentiation for high performance parallel pilot-level reservoir simulation. In: ECMOR XVI-16th European Conference on the Mathematics of Oil Recovery, vol. 2018, pp. 1–12. European Association of Geoscientists andEngineers (2018)
Leveque, R.J.: Finite Volume Methods for Hyperbolic Problems, vol. 31. Cambridge University Press, Cambridge (2002)
Lichtner, P.C., Hammond, G.E., Lu, C., Karra, S., Bisht, G., Andre, B., Mills, R.T., Kumar, J., Frederick, J.M.: PFLOTRAN user manual. Technical report (2019). http://documentation.pflotran.org
Lie, K.-A.: An Introduction to Reservoir Simulation Using MATLAB/GNU Octave: User Guide for the MATLAB Reservoir Simulation Toolbox (MRST). Cambridge University Press, Cambridge (2019). https://doi.org/10.1017/9781108591416
Lie, K.-A., Mykkeltvedt, T.S., Møyner, O.: A fully implicit WENO scheme on stratigraphic and unstructured polyhedral grids. Comput. Geosci. **24**(2), 405–423 (2020)
Liu, X.-D., Osher, S., Chan, T.: Weighted essentially non-oscillatory schemes. J. Comput. Phys. **115**(1), 200–212 (1994)
May, S., Berger, M.: Two-dimensional slope limiters for finite volume schemes on non-coordinate-aligned meshes. SIAM J. Sci. Comput. **35**(5), 2163–2187 (2013)
Mykkeltvedt, T.S., Raynaud, X., Lie, K.-A.: Fully implicit higher-order schemes applied to polymer flooding. Comput. Geosci. **21**(5), 1245–1266 (2017)

Mykkeltvedt, T.S., Gasda, S.E., Sandve, T.H.: CO_2 convection in hydrocarbon under flowing conditions. Transp. Porous Media **139**(1), 155–170 (2021)
Nocedal, J., Wright, S.: Numerical Optimization. Springer, New York (2006)
Nordbotten, J.M., Celia, M.A.: Geological Storage of CO_2: Modeling Approaches for Large-Scale Simulation. Wiley, New Jersey (2011)
Rasmussen, A.F., Sandve, T.H., Bao, K., Lauser, A., Hove, J., Skaflestad, B., Klöfkorn, R., Blatt, M., Rustad, A.B., Sævareid, O., Lie, K.-A., Thune, A.: The open porous media flow reservoir simulator. Comput. Math. Appl. **81**, 159–185 (2021). https://doi.org/10.1016/j.camwa.2020.05.014
Rubin, B., Blunt, M.: Higher-order implicit flux limiting schemes for black oil simulation. In: SPE Symposium on Reservoir Simulation. OnePetro (1991)
Rubin, B., Edwards, M.: Extension of the TVD midpoint scheme to higher-order accuracy in time. In: SPE Symposium on Reservoir Simulation. OnePetro (1993)
Sammon, P.H., Kurihara, M., Jialing, L.: Applying high-resolution numerical schemes in reservoirs described by complex corner-point grids. In: SPE Reservoir Simulation Symposium. OnePetro (2001)
Sandve, T.H., Rasmussen, A., Rustad, A.B.: Open reservoir simulator for CO_2 storage and CO_2-EOR. In: 14th Greenhouse gas control technologies conference Melbourne, pp. 21–26 (2018)
Todd, M., Longstaff, W., et al.: The development, testing, and application of a numerical simulator for predicting miscible flood performance. J. Pet. Technol. **24**(07), 874–882 (1972)
Trangenstein, J.A., Bell, J.B.: Mathematical structure of the black-oil model for petroleum reservoir simulation. SIAM J. Appl. Math. **49**(3), 749–783 (1989)
Voskov, D.V., Tchelepi, H.A.: Comparison of nonlinear formulations for two-phase multi-component EoS based simulation. J. Pet. Sci. Eng. **82**, 101–111 (2012)