Delft Advanced Research Terra Simulator

General Purpose Reservoir Simulator with Operator-Based Linearization

Khait, Mark

DOI
10.4233/uuid:5f0f9b80-a7d6-488d-9bd2-d68b9d7b4b87

Publication date
2019

Document Version
Final published version

Citation (APA)
Khait, M. (2019). *Delft Advanced Research Terra Simulator: General Purpose Reservoir Simulator with Operator-Based Linearization*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:5f0f9b80-a7d6-488d-9bd2-d68b9d7b4b87

Delft Advanced Research Terra Simulator

General Purpose Reservoir Simulator with Operator-Based Linearization

Delft Advanced Research Terra Simulator

General Purpose Reservoir Simulator with Operator-Based Linearization

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. dr. ir. T.H.J.J. van der Hagen,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op
dinsdag 10 december 2019 om 12:30 uur

door

Mark KHAIT

Master of Science in Information Security,
Ufa State Aviation Technical University, Oefa, Rusland,
geboren te Oefa, Rusland.

Dit proefschrift is goedgekeurd door de promotoren.

Samenstelling promotiecommissie bestaat uit:

Rector Magnificus,	voorzitter
Dr. D.V. Voskov	TU Delft, promotor
Prof. dr. ir. J.D. Jansen	TU Delft, promotor
Onafhankelijke leden:	
Prof. dr. ir. C. Vuik	TU Delft
Prof. dr.-ing. R. Helmig	University of Stuttgart
Prof. dr. ing. K.A. Lie	NTNU
Dr. J.P. O'Sullivan	University of Auckland
Dr. H. Hajibeygi	TU Delft
Prof. dr. W.R. Rossen	TU Delft, reservelid

Keywords: operator-based linearization, multiphase flow, compositional formulation, mass and energy transport, unstructured grids, GPU

Printed by: Ipskamp Printing

Cover design: Alina Khait. Picture credit: Stephan Timmers, Total Shot Productions. Invitation picture credit: David Khait.

ISBN 978-94-6366-229-1

An electronic version of this dissertation is available at
http://repository.tudelft.nl/.

Science is a wonderful thing
if one does not have to earn one's living at it.

Albert Einstein

To Alina, David and Liia

Contents

Preface

This story has started in August of 2014. It was a beautiful warm summer evening, which we spent out of town with the whole family. David and Liia were already 8 months old. We were enjoying dinner outside and having a nice usual chat just about everything. We had already been lightly thinking about moving somewhere abroad for some time, but by the end of that calm and quiet evening Alina and I suddenly realized: it is now or never. So, I picked up the phone and called the only person abroad that I knew well - Denis. Until our determination was gone, I wanted to ask if he has anything to suggest - and of course he did.

A year later, we got off a plane and set foot on Dutch soil, as I accepted a PhD candidate position in Delft University of Technology under Denis's supervision. My experience in the development of reservoir simulators and high-performance computing provided certain confidence in the success, creating high pressure on the level of achievements to reach on the other hand. Denis helped to define the right mindset - I desired to use the unique opportunity of working under his guidance for 4 years to create something really interesting, useful, and of course fast.

More than 4 years, 15.000 cycled kilometres, and 100.000 written lines of code later, at this very moment, I can say that the results have exceeded my wildest expectations. The book you are holding now is a mere compilation of the published materials over these years. Could it be written better including more thorough and consistent investigations? Definitely, yes. However, a lot of effort has been deliberately put in flexibility, re-usability, conciseness, and modularity of architecture of the reservoir simulation platform that we created. DARTS is already used by many MSc and PhD students, postdocs in our department for their research. Moreover, my faith in its both academic and industrial potential is so strong, that I decided to carry on as a postdoc at TU Delft to be able to continue this work.

Given the complexity of the algorithms and software that we design, I share the opinion that corresponding source code should be made available for the entire community, complementing the published materials for the sake of reproducible research. Therefore, I set myself a goal to make DARTS publicly available and hereby complete my scientific contribution. I am impatient to see how it will evolve over the next few years. And another calm yet fateful family evening may follow...

Mark KHAIT
Delft, November 2019

Summary

The modern development of subsurface resources requires numerical reservoir simulation. It is used to predict and compare the performance of different reservoir development schemes as well as to reduce uncertainties in parameters estimation and associated field management risks. Growing computational power available in the high-performance computing market spawned a higher demand for more complex, accurate, and larger models. However, these complexities often challenge the performance of the simulation process.

The opening Chapter 1 briefly introduces the current context of general purpose reservoir simulation. Non-isothermal multiphase compositional simulation is based on the solution of governing equations describing mass and energy transfer in the subsurface. Usually, a Newton-based method is used to solve a strongly nonlinear system of residual equations robustly and efficiently. It requires linearization stage consisting of Jacobian assembly for a fully coupled system of equations. This, in turn, requires the determination of the derivatives of all the residual equations with respect to independent variables. Linearization completely defines the modelling capabilities of a reservoir simulator and often represents the most specific, complicated, and volumetric part of its source code.

A novel Operator-Based Linearization (OBL) method for multiphase compositional fluid flow and transport in the subsurface is established in Chapter 2 and extended to support thermal and buoyancy effects in Chapter 3. The approach simplifies the Jacobian assembly introducing discretization of the physical description of fluid and rock in addition to space and time discretizations. The trade-off between accuracy and performance of a reservoir simulator receives thereby a new degree of freedom. Coarse resolutions of discretization of physical properties, delivering satisfiable solution accuracy, can substantially improve the non-linear convergence of the complex thermal-compositional problems. Besides, OBL decouples physical description of fluid and rock from the main simulator core creating unique opportunities in the architecture of a reservoir simulator.

The applicability of the OBL approach to the general purpose reservoir simulation problems is demonstrated for different physical kernels including dead-oil, black-oil, isothermal compositional fluids with 4 and 6 components, low- and high-enthalpy geothermal, and thermal-compositional multiphase formulations. As a rule of thumb, a resolution of 64 uniformly distributed points along each of parameter space axes within the required range is sufficient for an accurate representation of fluid and rock properties. On the other hand, the limited coarsening of parameter space improves the nonlinear convergence in most cases. The performance of the OBL approach benefits from the simplified assembly of Jacobian of the simulation problem and almost complete bypass of phase behaviour calculations (except supporting points).

Chapter 4 describes and compares several prototype implementations of the OBL approach, revealing the difference in computational performance depending on chosen software and hardware platforms. The linearization approach was initially designed within Automatic Differentiation General Purpose Research Simulator (AD-GPRS) framework and then implemented in stand-alone simulators in MATLAB, C++ (for CPU platform), and C++/CUDA (for GPU platform). The latter prototype was implemented for both CPU and GPU platforms. Compared to conventional AD-based linearization, the single-threaded CPU prototype performs the Jacobian construction up to 19x times faster, while the GPU prototype boosts the linearization by a factor of 260x.

The Delft Advanced Research Terra Simulator (DARTS) is introduced and described in Chapter 5. It combines the experience and knowledge obtained during previous iterations of OBL implementation. Having kept all performance-critical parts of simulator core in C++, DARTS exploits physical description decoupling to the full extent, providing Python-based plugin interface to customize fluid and rock properties. DARTS demonstrates how the architecture of a reservoir simulator can reveal the performance potential of OBL in three independent levels: improved non-linear performance - algorithmic level; actual performance of linearization stage - software level; portability to alternative computing architectures including GPU - hardware level.

Finally, Chapter 6 discusses current DARTS applications and future developments. Two different approaches of non-uniform parameterization of physical space are investigated. A significant increase in parameterization accuracy was confirmed compared to uniform parameterization with a similar amount of supporting points in most cases. In addition, DARTS platform can be easily and efficiently used to create different kinds of proxy models. For example, Multi-Scale Compositional Transport (MSCT) approach approximates the compositional description of a multi-component fluid with a specially built binary system. The resulting proxy model is straightforwardly constructed within DARTS simply by substituting restricted fractional flow curves into operators. Another possibility to construct a proxy model in DARTS is to coarsen the space and time discretization resolution of the full model.

Chapter 7 concludes this dissertation recapitulating the main points. Additional accuracy-performance trade-off provided by OBL, simplified manipulation of a simulation model via Python, and exceptional computational performance make DARTS an ultimate platform for both forward and inverse modelling. Its architecture allows to change existing formulations and even introduce new physical descriptions with minimal efforts not sacrificing computational performance. Furthermore, the complete transition of the main simulation loop to GPU, along with the implementation of adjoint gradients will allow taking the inverse modelling performance on a new level.

Samenvatting

De moderne ontwikkeling van ondergrondse bronnen vereist numerieke reservoir simulatie. Het wordt gebruikt om de prestaties van verschillende reservoir ontwikkeling plannen te voorspellen en te vergelijken, en om onzekerheden in de schatting van parameters en bijbehorende veldbeheer risico's te verminderen. Toenemende rekenkracht die beschikbaar is gekomen binnen de hoge-prestatie computatie markt heeft tot een hogere vraag naar complexere, nauwkeurigere en grotere modellen geleidt. Deze complexiteit daagt echter vaak de prestaties van het stimulatie proces uit. Hoofdstuk 1 geeft een korte introductie over wat de huidige context van reservoir simulatie voor algemene doeleinden is. Niet-isotherme meerfasige compositie simulatie is gebaseerd op de oplossing van vergelijkingen die de massa- en energieoverdracht in de ondergrond beschrijven. Normaal gesproken wordt een op Newton gebaseerde methode gebruikt om een sterk niet-lineair systeem van rest-vergelijkingen robuust en efficiënt op te lossen. Het vereist een linearisatie fase bestaande uit Jacobiaanse constructie voor een volledig gekoppeld stelsel vergelijkingen. Dit vereist op zijn beurt de bepaling van de afgeleiden van alle rest-vergelijkingen met betrekking tot onafhankelijke variabelen. Linearisatie definieert volledig de modellering mogelijkheden van een reservoir simulator en vertegenwoordigt vaak het meest specifieke, gecompliceerde en het grootste deel van de broncode. Een nieuw uitgevonden Operator-Based Linearisation (OBL) methode voor meerfasige compositie vloeistofstroming en transport in de ondergrond is vastgesteld in Hoofdstuk 2 en uitgebreid om thermische en drijfvermogen effecten in Hoofdstuk 3 te ondersteunen. De benadering vereenvoudigt de constructie van de Jacobian die discretisatie van de fysieke beschrijving van vloeistof en gesteente introduceert naast ruimte- en tijd-discretisaties. De afweging tussen nauwkeurigheid en prestaties van een reservoir simulator krijgt daardoor een nieuwe graad van vrijheid. Een grove resolutie in de discretisatie van fysische eigenschappen, die een tevreden nauwkeurigheid van de oplossing leveren, kan aanzienlijk de niet-lineaire convergentie van de complexe thermische compositie problemen verbeteren. Bovendien ontkoppelt OBL de fysieke beschrijving van vloeistof en gesteente van de hoofd simulator-kern en creëert het unieke kansen in de architectuur van een reservoir simulator. De toepasbaarheid van de OBL-benadering op de reservoir simulatie problemen voor algemene doeleinden wordt aangetoond voor verschillende fysieke kernen waaronder dode-olie, zwarte-olie, isothermische vloeistoffen met meervoudige componenten met 4 en 6 componenten, geothermische met lage en hoge enthalpie, en thermische formuleringen voor vloeistof met meervoudige componenten. Als vuistregel is een resolutie van 64 uniform verdeelde punten langs elk van de parameter ruimte-assen binnen het vereiste bereik voldoende voor een nauwkeurige weergave van vloeistof- en gesteente-eigenschappen. Anderzijds verbetert de beperkte vergroting van de parameter ruimte in de meeste gevallen de

niet-lineaire convergentie(this sentence might need to restructuring, not sure what you exactly want to say here). De prestatie van de OBL-benadering is gebaat bij de vereenvoudigde constructie van de Jacobian van het simulatie probleem en via een bijna volledige vermijding van de fasegedrag berekeningen (behalve ter plaatse van ondersteunende punten). Hoofdstuk 4 beschrijft en vergelijkt verschillende prototype-implementaties van de OBL-aanpak. Vervolgens wordt het verschil in rekenprestaties onthuld welke afhankelijk is van de gekozen software en hardware-platforms. De linearisatie benadering werd oorspronkelijk ontworpen binnen het AD-GPRS (Automatic Differentiation General Purpose Research Simulator) kader en is vervolgens geïmplementeerd in zelfstandige simulators in MATLAB, C ++ (voor CPU-platform) en C ++ / CUDA (voor GPU-platform). Het laatste prototype werd geïmplementeerd voor zowel CPU- als GPU-platforms. Vergeleken met conventionele AD-gebaseerde linearisatie, voert het single-threaded CPU-prototype de Jacobian constructie tot 19x keer sneller uit, terwijl het GPU-prototype de linearisatie met een factor van 260x versneld. De Delft Advanced Research Terra Simulator (DARTS) wordt geïntroduceerd en beschreven in Hoofdstuk 5. Het combineert de ervaring en kennis die is opgedaan tijdens eerdere iteraties van de OBL-implementatie. Alle prestatie-kritische delen van de simulator kern zijn ontwikkeld in C ++ en wordt een op Python gebaseerde interface om vloeistof- en gesteente-eigenschappen aan te passen aangeboden. Hierdoor maakt DARTS optimaal gebruik van de ontkoppeling van de fysische beschrijving en het numerieke rekenwerk. DARTS laat zien hoe de architectuur van een reservoir simulator het prestatiepotentieel van OBL op drie onafhankelijke niveaus kan onthullen: verbeterde niet-lineaire prestaties - algoritmisch niveau; werkelijke prestaties van linearisatie fase - software niveau; draagbaarheid naar alternatieve computerarchitecturen inclusief GPU - hardware niveau. Ten slotte bespreekt Hoofdstuk 6 huidige DARTS-toepassingen en toekomstige ontwikkelingen. Twee verschillende methodes van niet-uniforme parameterisatie van fysieke ruimte worden onderzocht. In de meeste gevallen werd een significante toename van de nauwkeurigheid van de parametrering bevestigd in vergelijking met een uniforme parametrering met een vergelijkbaar aantal ondersteunende punten. Bovendien kan het DARTS-platform eenvoudig en efficiënt worden gebruikt om verschillende soorten proxy-modellen te maken. De benadering van Multi-Scale Compositional Transport (MSCT) benadert bijvoorbeeld de beschrijving van een vloeistof met meerdere componenten met een speciaal gebouwd binair systeem. Het resulterende proxy-model is eenvoudig geconstrueerd binnen DARTS door beperkte fractionele stroomcurves te vervangen door operators. Een andere mogelijkheid om een proxy-model in DARTS te bouwen, is om de ruimte- en tijd-discretisatie resolutie van het volledige model te vergroten. Hoofdstuk 7 concludeert dit proefschrift waarin de belangrijkste punten worden samengevat. Extra afweging van nauwkeurigheid en prestaties verwezenlijkt met OBL, vereenvoudigde manipulatie van een simulatiemodel via Python en uitzonderlijke rekenprestaties maken DARTS een ultiem platform voor zowel voorwaartse als inverse modellering. De architectuur maakt het mogelijk om bestaande formuleringen te veranderen en zelfs nieuwe fysieke beschrijvingen te introduceren met minimale inspanning zonder de rekenprestaties op te offeren. Bovendien zal de volledige overgang van de hoofd

simulatielus naar GPU, samen met de implementatie van aanvullende gradiënten, de inverse modellering prestaties naar een nieuw niveau brengen.

Nomenclature

Linearization Operators

α_c	physical term of component mass accumulation operator
α_e	physical term of energy accumulation operator
β_c	physical term of component mass convection operator
β_e	physical term of energy convection operator
β_{cg}	physical term of component gravity operator
β_{cp}	physical term of component in phase mass convection operator
β_{eg}	physical term of energy gravity operator
β_{ep}	physical term of energy in phase mass convection operator
δ_c	physical term of component gravity operator
δ_p	physical term of phase gravity operator
γ_e	physical term of energy conductive operator
θ_c	component mass influx/outflux term
θ_e	energy influx/outflux term
ζ_p	physical term of phase rate operator
a	spatial term of mass accumulation operator
a_e	spatial term of energy accumulation operator
b	spatial term of mass convection operator
b_e	spatial term of energy convection operator
b_p	spatial term of phase mass convection operator
b_{ep}	spatial term of energy in phase convection operator
b_g	spatial term of gravity operator
c_e	spatial term of energy conductive operator

Other Symbols

$\boldsymbol{\xi}$	spatial coordinate
Γ^l	geometrical part of transmissibility over interface
Γ_c^l	conductive transmissibility over interface
Γ_p^l	phase transmissibility over interface
$\boldsymbol{J}$	Jacobian
$\boldsymbol{\omega}$	physical state
$\boldsymbol{\omega}^{SC}$	physical state at surface conditions
$\boldsymbol{r}$	residual
$\boldsymbol{u}$	well control variables
g	gravitational constant
I	interpolation operator
l	interface between control volumes
n	parameterization resolution
n_c	number of components
n_p	number of phases

Physics Relations

$\boldsymbol{K}$	effective permeability tensor
κ	thermal conduction
κ_p	phase thermal conduction
κ_r	rock thermal conduction
λ_p	phase mobility
μ_p	phase viscosity
ν_p	phase molar fraction
ϕ	effective rock porosity
ϕ_0	initial rock porosity
Φ_p	phase potential between control volumes i and j
ρ_c	component density
ρ_p	phase molar density

$\tilde{q_p}$	phase in/outflux
$\mathbf{U}_p$	phase internal energy
$\mathbf{U}_r$	rock internal energy
$\vec{u_p}$	phase velocity
c_r	rock compressibility
D	depth
f_{cp}	component fugacity in phase
h_p	phase enthalpy
k_{rp}	phase relative permeability
p^l	pressure difference across interface
p_p	phase pressure
p_{ref}	reference pressure
T	temperature
t	time
T^l	temperature difference across interface
V	volume
x_{cp}	component mole fraction in a phase
z_c	component overall molar fraction
γ_p	phase vertical pressure gradient
s_p	phase saturation

1

Introduction

Numerical simulations are essential for the modern development of subsurface reservoirs [1–3]. They are widely used for the evaluation of oil recovery efficiency, performance analysis, and various optimization problems. Due to the complexity of the underlying physical processes and considerable uncertainties in the geological representation of reservoirs, there is a persistent demand for more accurate models.

To increase the accuracy of a model, one can apply a more refined computational grid in space or time, or use a more detailed description of the fluids, such as in a thermal-compositional model. However, an improvement in the accuracy of numerical models is usually counterbalanced by a reduction in the overall performance of the simulation. Besides, more refined space and time approximations can increase the nonlinearity of governing equations, which need to be resolved numerically.

Depending on the formulation, different types of nonlinear unknowns and strategies can be used to perform nonlinear update [4]. The most frequently used approaches for reservoir simulation are the natural [5] and molar formulations [6, 7]. For natural formulation, phase behaviour computations include equally important phase split calculation and stability test. Compared to the straightforward algorithm, the performance of both can be improved by 1-2 orders of magnitude by using physically-based heuristics and bypassing techniques [8] or by employing parameterization idea and approximating the calculations with desired accuracy [9]. The main difference of (overall) molar formulation is that the set of variables remains constant in the course of simulation independently on the number of phases in the given grid cell, avoiding the need of variable substitution. Noting that it is difficult to compare different formulations directly and fairly, the two formulations show comparable performance in terms of nonlinear iterations [10]. However, the molar formulation does not allow to avoid or bypass phase equilibrium calculations hence is likely to be slower in terms of CPU time at a single nonlinear iteration. Confirming that, [11] also show that for miscible displacement regimes the molar

formulation requires significantly less nonlinear iterations outrunning its counterpart in terms of CPU time. Also, the authors demonstrate that the parametrization technique applies to the molar formulation, speeding up phase behaviour computations. Finally, it was shown recently that some specific treatments of phase appearance or disappearance may help to improve the nonlinear behaviour of the natural formulation in miscible regimes [12].

Fully implicit methods are conventionally used in reservoir simulation because of their unconditional stability [1]. On the other hand, after discretization is applied to governing Partial Differential Equations (PDE) of a problem, the resulting nonlinear system represents different tightly coupled physical processes, which is difficult to solve. Usually, a Newton-based iterative method is applied, which demands an assembly of the Jacobian and the residual for the combined system of equations (i.e., linearization) at every iteration forming a linear system of an equal size (often ill-conditioned). Precisely the solution of such systems takes most of the simulation time in most practical applications. Alternatively, localized nonlinear solving strategies can be used. They exploit the fact that the transport mechanism of fluid phases is in practice mostly unidirectional, exhibiting countercurrent flow due to buoyancy and capillary forces only in local areas of the computational domain. Therefore, it is possible to apply flux-based reordering (see Cascade ordering for cocurrent flow defined by [13] and generalized to address countercurrent flow by [14]) and solve a series of nonlinear systems of reduced size localized along the upwind direction. This strategy may involve a rearrangement of blocks of the entire nonlinear system to exhibit lower triangular form, which is then efficiently solved on a cell-by-cell basis. This approach was consistently improved over the years by [15–17], including generalizations for an unstructured grid and compositional formulation. Alternatively, the influence of every source of mass imbalance can be limited to a certain neighbourhood, leading to a collection of localized nonlinear problems which superpositioned solutions reproduce Newton update of the full system [18, 19]. These strategies, however, require sequentially coupled solution of flow and transport equations for total velocity field construction, hence inheriting splitting errors.

Conventionally used in most practical applications Newton-based nonlinear solvers require linearization. Several conventional linearization approaches exist, though neither of them is robust, flexible, and computationally efficient all at once. Numerical derivatives provide flexibility in the nonlinear formulation (see [20], for example), but a simulation based on numerical derivatives may lack robustness and performance [21]. Straightforward hand-differentiation is the state-of-the-art strategy in modern commercial simulators [22, 23]. However, this approach requires an introduction of a complicated framework for storing and evaluating derivatives for each physical property, which in turn reduces the flexibility of a simulator to incorporate new physical models and increases the probability for potential errors.

The development of Automatic Differentiation (AD) technique allows preserving both flexibility and robustness in derivative computations. In reservoir simulation, the Automatically Differentiable Expression Templates Library (ADETL) was introduced by [24]. Using the capabilities of ADETL, an Automatic Differentiation Gen-

eral Purpose Research Simulator (AD-GPRS) was developed [10, 25]. Later, the AD technique becomes more demanded in research frameworks for reservoir simulation [26]. Certain frameworks even allow any of the mentioned linearization approaches to be used in a particular formulation [27]. Being attractive from the perspective of flexibility, the AD technique by design inherits computational overhead, which affects the performance of reservoir simulation [28].

Once the linear system with Jacobian and right-hand side is constructed, it needs to be solved. Since the dimensionality of a typical reservoir simulation problem is rather high, iterative linear solvers are usually used with effective preconditioning schemes. Widely used two-stage preconditioning scheme addresses mixed elliptic-hyperbolic behaviour of underlying nonlinear equations applying Constrained Pressure Residual (CPR) [29–31] to decouple an elliptic system with only pressure unknowns. Then it can be efficiently solved using algebraic multigrid methods (AMG, see [32, 33]) concluding the first stage of preconditioning. At the second stage, the entire linear system is processed with fine-scale smoother to address high-frequency errors, and incomplete LU-factorization with zero fill-in (ILU(0)) is the standard choice [34].

Once the solution to the linear system with predefined tolerance is found, we need to update the nonlinear unknowns and repeat the nonlinear iteration. The nonlinear solution may require several nonlinear iterations to converge depending on the nonlinearity of a problem. Even if a fully-implicit scheme is chosen, a standard Newton solver can fail to converge within a reasonable number of iterations, especially for large timesteps. In such cases, all computations related to that timestep are discarded, and the Netwon process is repeated with a smaller timestep. In order to address this problem, a sophisticated analysis of Newton updates can be made to loosely follow the solution path. For example, Continuation Newton method assigns every nonlinear update to an inner timestep and therefore avoids discarded nonlinear iterations [18].

The number of nonlinear iterations for transport problems can be sufficiently reduced (implying also that more timesteps will not be discarded) by controlling the nonlinear update of saturation variable. Several heuristic algorithms were designed for black-oil models to prevent a rapid change of phase mobility properties, as well as that of the magnitude of saturations themselves (e.g., Appleyard chop) [22]. The generalization of such control of saturation update, based on a thorough analysis of the shape of flux functions, has led to a new family of trust-region based nonlinear solvers, established by [35]. Later, it was enhanced by [36] improving convergence on large timesteps for viscous dominated flows, and by [37] with the focus on strong capillary forces.

All these techniques can be straightforwardly and effectively applied to compositional models with the natural formulation where saturation unknown is present (e.g., see [38]). The Negative Saturation method is another extension of the natural formulation helping to avoid variable substitution and apply corrections to discontinuous changes in derivative, usually related to phase appearance and disappearance [39, 40]. On the other hand, there is a lack of efficient advanced strategies for the overall molar formulation, where saturation unknown is not present. A version of

a trust region correction was developed for the molar formulation [41], but it still lacks robustness in comparison to the natural formulation. This can be explained by the more complicated nonlinear update procedure, which requires performing an exact flash for every block at a two-phase state in each nonlinear iteration.

This problem can be avoided by using parameterization in compositional space instead. A strategy, based on the Compositional Space Parameterization idea [9], was designed by [11]. The nonlinear solver based on a special point correction along the fractional flow curve has proved to be robust and efficient [11]. However, this approach requires reformulation of a nonlinear problem in a tie-line space and formally cannot be applied to the conventional molar formulation [42].

Another approach for the molar formulation called Operator-Based Linearization (OBL), was proposed recently in [43]. It could be seen as an extension of the idea to abstract the representation of properties from the governing equations, suggested in [11] and [44]. In the OBL approach, the parameterization is performed based on the conventional molar variables. A similar approach can be designed for the natural formulation, but it requires dealing with several parameter spaces and switching between them.

In the OBL approach, all properties involved in the governing equations are lumped in a few operators, which are parameterized in the physical space of the simulation problem either in advance or adaptively during the simulation process. The control on the size of parameterization hyperrectangle helps to preserve the balance between the accuracy of the approximation and the performance of nonlinear solver [45]. Note, that the OBL approach does not require the reduction in the number of unknowns, and only employs the fact that physical description (i.e., fluid properties) is approximated using piecewise linear interpolation.

1.1. Research Objectives

Numerical simulation is based on space and time discretization, which provides a trade-off between accuracy and computational performance. The OBL approach can be viewed as an attempt to build such a discretization for the physical description of fluid and rock. The research objectives addressed by this work are:

- Investigate the applicability of the OBL approach to general purpose reservoir simulation based on thermal-compositional description.
- Develop a general purpose simulation framework targeting to exploit the advantages of OBL to the full extent.
- Investigate how OBL can be coupled with modern software and hardware architectures to improve flexibility and performance of reservoir simulation.

1.2. Thesis Outline

The dissertation consists of seven chapters, including this introductory Chapter 1. First, we show the applicability of the OBL approach to multiphase multi-component mass transport and demonstrate the numerical convergence of physical solutions

based on the OBL technique for problems with up to 6 components in Chapter 2. Chapter 3 describes the extension of the OBL method to account for energy balance and buoyancy. We investigate several approaches for the robust treatment of gravity forces and demonstrate its applicability for challenging thermal-compositional problems including a full-field example.

Then, we describe the details of OBL implementation in different prototypes in Chapter 4. Based on several numerical models with various physical descriptions, the computational performance of prototypes is compared, including an entirely GPU-based implementation. Chapter 5 presents a detailed description of the Delft Advanced Research Terra Simulator (DARTS). It provides a simulation framework built around the OBL approach in an attempt to maximize its flexibility and performance. We show how this attempt affects the architecture of the reservoir simulation framework and what advantages it allows to achieve. Sensitivity to the resolution of the OBL representation is investigated. Benchmarks comparing the accuracy of the numerical solution and computational performance with other simulators are also provided.

Finally, Chapter 6 shows several applications of DARTS and describes how the OBL method can be advanced further. We demonstrate two approaches for a non-uniform OBL parameterization and describe proxy-modelling within the DARTS framework. Chapter 7 concludes the work and defines perspectives of further development of OBL and DARTS.

2

Operator-Based Linearisation (OBL)

2.1. Governing Equations

First, we describe one of the conventional nonlinear formulations for a general purpose compositional model. This formulation was implemented in the Automatic Differentiation General Purpose Research Simulator (AD-GPRS)[10] and is used to obtain the reference solution. Mass transport for a system with n_p phases and n_c components is considered. For this model, the n_c component molar mass conservation equations can be written as

$$\frac{\partial}{\partial t}\left(\phi \sum_{p=1}^{n_p} x_{cp}\rho_p s_p\right) + \operatorname{div} \sum_{p=1}^{n_p} x_{cp}\rho_p \vec{u_p} + \sum_{p=1}^{n_p} x_{cp}\rho_p \tilde{q_p} = 0, \quad c = 1, \ldots, n_c. \tag{2.1}$$

Here, t is time, ϕ is effective rock porosity, x_{cp} is component c concentration in phase p, ρ_p denotes phase p molar density, s_p is saturation of phase p, $\vec{u_p}$ is velocity of phase p, and $\tilde{q_p}$ denotes source of phase p.

Phase flow velocity is assumed to follow the Darcy law:

$$\vec{u_p} = -\left(\boldsymbol{K}\frac{k_{rp}}{\mu_p}(\nabla p_p - \gamma_p \nabla D)\right), \tag{2.2}$$

where $\boldsymbol{K}$ is the effective permeability tensor, k_{rp} is phase p relative permeability, μ_p is phase p viscosity, p_p is phase p pressure, γ_p is vertical pressure gradient, and D is depth.

Parts of this chapter have been published in Journal of Petroleum Science and Engineering **157**, 990 (2017) [46]

Equation 2.1 is then approximated in time using a Fully Implicit Method (FIM). The method suggests that the convective flux term depends on the values of non-linear unknowns at the current time step. After application of a finite-volume discretization on a general unstructured mesh and a backward Euler approximation in time we get

$$V\left(\left(\phi\sum_{p=1}^{n_p}x_{cp}\rho_p s_p\right)-\left(\phi\sum_{p=1}^{n_p}x_{cp}\rho_p s_p\right)^n\right)$$
$$-\Delta t\sum_l\left(\sum_{p=1}^{n_p}x_{cp}^l\rho_p^l\Gamma_p^l\Delta\psi^l\right)+\Delta t\sum_{p=1}^{n_p}x_{cp}\rho_p\mathbf{q}_p \quad = \quad 0, \quad c=1,\dots,n_c, \tag{2.3}$$

where V is volume of mesh grid block and $q_p = \tilde{q_p}V$ is a source of phase p over the control volume. Here, we have neglected capillarity, gravity and used a Two-Point Flux Approximation (TPFA) with an upstream weighting. Therefore, $\Delta\psi^l$ becomes a simple difference in pressures over an interface l. In addition, $\Gamma_p^l = \Gamma^l k_{rp}^l/\mu_p^l$ is a phase p transmissibility over interface l, with Γ^l assumed to be the constant geometrical part of transmissibility, including rock permeability and geometry of the control volumes connected by interface l. All terms of the equations are defined at $n+1$ timestep, except the second part of accumulation term denoted by n superscript.

This choice of a fully-coupled approach introduces nonlinearity into the system, which is further increased by the closure assumption of instantaneous thermodynamic equilibrium. In this formulation, an exact thermodynamic equilibrium is required at every nonlinear iteration. Hence, for any grid block that contains a multiphase (n_p) multi-component (n_c) mixture we solve the following system:

$$F_c = z_c - \sum_{p=1}^{n_p}\nu_p x_{cp} \quad = \quad 0, \tag{2.4}$$

$$F_{c+n_c} = f_{c1}(p,T,\boldsymbol{x}_1) - f_{cp}(p,T,\boldsymbol{x}_p) \quad = \quad 0, \tag{2.5}$$

$$F_{p+n_c\times n_p} = \sum_{c=1}^{n_c}(x_{c1}-x_{cp}) \quad = \quad 0, \tag{2.6}$$

$$F_{n_p+n_c\times n_p} = \sum_{p=1}^{n_p}\nu_p - 1 \quad = \quad 0. \tag{2.7}$$

In this procedure, which is usually called multiphase flash [47], the overall molar composition z_c of component c is defined as:

$$z_c = \frac{\sum_p x_{cp}\rho_p s_p}{\sum_p \rho_p s_p}. \tag{2.8}$$

Note, that overall molar composition is computed across all existing phases in the mixture, unlike composition in the chemical sense meaning the relative amounts of chemicals making up a single phase. Further in this work, by composition we will assume overall molar composition. Next, $f_{cp}(p, T, x_{cp})$ is the fugacity of component c in phase p. By solving the system of Equation 2.4-Equation 2.7, we obtain mole fractions for each component x_{cp}, phase molar fractions ν_p, and consequently phase saturations S_p for the given state.

After obtaining the solution of the multiphase flash, we determine partial derivatives with respect to nonlinear unknowns using the inverse theorem (see [10] for details) and assemble the Jacobian and the residual. This step, often referred to as linearization, is required by the Newton-Raphson method, which solves the following linear system of equations on each nonlinear iteration:

$$\boldsymbol{J}(\boldsymbol{x}^k)(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k) + \boldsymbol{r}(\boldsymbol{x}^k) = 0, \tag{2.9}$$

where $\boldsymbol{J}(\boldsymbol{x}_k)$ and $\boldsymbol{r}(\boldsymbol{x}_k)$ are the Jacobian and the residual defined at a nonlinear iteration k. The conventional approach assumes the numerical representation of rock and fluid properties and their derivatives with respect to nonlinear unknowns. This may require either table interpolation (e.g., for relative permeability) or the solution of a highly nonlinear system Equation 2.4-Equation 2.7 for properties defined by an Equation of State (EoS) in combination with the chain rule and the inverse theorem. As a result, a nonlinear solver has to resolve all of the small features of the property descriptions, which can be quite challenging and is often unnecessary due to the numerical nature and uncertainties in property evaluation.

2.2. Physical State and Spatial Coordinate

According to the Operator Based Linearization (OBL) method proposed in [43], all variables in the discretized form of Equation 2.1 are introduced as functions of a physical state $\boldsymbol{\omega}$ and/or a spatial coordinate $\boldsymbol{\xi}$. The physical state represents a unification of all state variables (i.e., nonlinear unknowns: pressure, temperature/enthalpy, saturations/compositions, etc.) of a single control volume. In the overall molar formulation, the nonlinear unknowns are p and z_c, therefore the physical state $\boldsymbol{\omega}$ is completely defined by these variables. The spatial coordinate defines the location of a given control volume which reflects the distribution of heterogeneous rock properties (e.g., porosity) and elements of space discretization (e.g., transmissibility). Besides, well control variables $\boldsymbol{u}$ are introduced to represent various well management strategies.

Now, all terms of Equation 2.1 and Equation 2.2 can be characterized as functions of the spatial coordinates $\boldsymbol{\xi}$, physical state $\boldsymbol{\omega}$, and well control variables $\boldsymbol{u}$ as follows:

- $\phi(\boldsymbol{\xi}, \boldsymbol{\omega})$ – effective rock porosity,
- $x_{cp}(\boldsymbol{\omega})$ – component concentration in phase,
- $\rho_p(\boldsymbol{\omega})$ – phase molar density,

- $s_p(\boldsymbol{\omega})$ – phase saturation,
- $\vec{u_p}(\boldsymbol{\xi}, \boldsymbol{\omega})$ – phase velocity,
- $\tilde{q_p}(\boldsymbol{\xi}, \boldsymbol{\omega}, \boldsymbol{u})$ – source of phase,
- $\boldsymbol{K}(\boldsymbol{\xi})$ – effective permeability tensor,
- $k_{rp}(\boldsymbol{\omega})$ – phase relative permeability,
- $\mu_p(\boldsymbol{\omega})$ – phase viscosity,
- $p_p(\boldsymbol{\omega})$ – phase pressure,
- $\gamma_p(\boldsymbol{\omega})$ – vertical pressure gradient,
- $D(\boldsymbol{\xi})$ – depth.

2.3. Operator Form of Conservation Equations

First, we introduce notions of state-dependent and space-dependent operators. A state-dependent operator is defined as a function of the physical state only. Therefore, it is independent of spatial position and represents physical properties of fluids and rock. A space-dependent operator is defined as a function of both physical state $\boldsymbol{\omega}$ and spatial coordinate $\boldsymbol{\xi}$.

Next, we rewrite Equation 2.3 neglecting buoyancy and capillary forces, and represent each term as a product of state-dependent and space-dependent operators [43]. The resulting mass conservation equation reads

$$\begin{aligned} a(\boldsymbol{\xi})\left(\alpha_c(\boldsymbol{\omega}) - \alpha_c(\boldsymbol{\omega}_n)\right) &+ \sum_l b(\boldsymbol{\xi}, \boldsymbol{\omega})\beta_c(\boldsymbol{\omega}) \\ &+ \theta_c(\boldsymbol{\xi}, \boldsymbol{\omega}, \boldsymbol{u}) = 0, \quad c = 1, \ldots, n_c. \end{aligned} \tag{2.10}$$

Here

$$a(\boldsymbol{\xi}) = \phi_0 V(\boldsymbol{\xi}), \tag{2.11}$$

$$\alpha_c(\boldsymbol{\omega}) = (1 + c_r(p - p_{ref})) \sum_{p=1}^{n_p} x_{cp}\rho_p s_p, \tag{2.12}$$

$$b(\boldsymbol{\xi}, \boldsymbol{\omega}) = \Delta t \Gamma^l(\boldsymbol{\xi}) p^l, \tag{2.13}$$

$$\beta_c(\boldsymbol{\omega}) = \sum_{p=1}^{n_p} x_{cp}^l \rho_p^l \frac{k_{rp}^l}{\mu_p^l}, \tag{2.14}$$

$$\theta_c(\boldsymbol{\xi}, \boldsymbol{\omega}, \boldsymbol{u}) = \Delta t \sum_{p=1}^{n_p} x_{cp}\rho_p q_p(\boldsymbol{\xi}, \boldsymbol{\omega}, \boldsymbol{u}). \tag{2.15}$$

In the equations above, ϕ_0 - rock porosity at reference pressure, c_r is rock compressibility, p_{ref} - reference pressure, p^l - pressure difference between the mesh grid blocks connected by interface l, while $\boldsymbol{\omega}$ and $\boldsymbol{\omega}_n$ are nonlinear unknowns at the current and previous time step respectively.

The physical meaning of mass accumulation operator α_c is the molar mass of component c per unit pore volume of uncompressed rock under physical state $\boldsymbol{\omega}$. The physical meaning of the mass flux operator for component c is the total mobile molar mass of that component in all phases of the mixture under physical state $\boldsymbol{\omega}$ per unit time, pressure gradient, and constant geometrical part of transmissibility.

This representation allows us to identify and distinguish the physical state-dependent operators - α_c, β_c in mass conservation Equation 2.3. The source/sink term can also be processed in a similar manner, see Section 5.4.

2.4. Approximation of Fluid and Rock Properties

The proposed approach simplifies the description of fluid and rock properties by building approximation interpolants for the operators α_c, β_c within the parameter space of a simulation problem. For a general isothermal compositional problem with n_c components and n_p phases with neglected buoyancy and capillary effects, the method requires $[2n_c]$ operators: one accumulation and one flux operator per component. If fluid properties change spatially and several regions of pressure-volume-temperature (PVT) or special core analysis (SCAL) properties are employed, several sets of operators need to be introduced accordingly (see Section 5.5). The values of the operators are fully determined by the set of $N = [n_c]$ independent variables $\{p, z_1, \dots, z_{n_c-1}\}$. The range of pressure variable in the compositional parameter space can usually be determined by the conditions specified for wells, or inferred from permissible pump operation regimes, while the overall composition is naturally bounded by the interval [0,1].

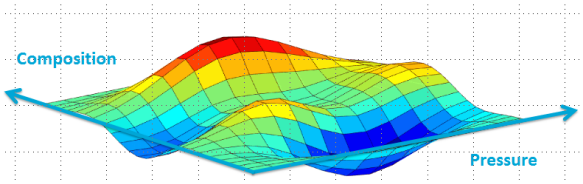


Figure 2.1: 2D parameterization of an abstract operator

Next, we parametrize the interval of each state variable using, for simplicity, the same number $n = n_1 = \cdots = n_N$ of uniformly distributed points on the intervals of parameters, according to Equation 2.16. This results in a set of *supporting points* $(p_i, z_{1_i}, \dots, z_{n_c-1_i}) : i = 1, \dots, n$, which can be interpreted as a discrete representation of physical space in the simulation. At the pre-processing stage, or adaptively, we can evaluate the operators α_c, β_c at every point in the discrete parameter space and store them in n_c-dimensional tables A_c and B_c. Figure 2.1 illustrates an example for an abstract operator, parametrized in two-dimensional space describing a

binary system. During the simulation, we interpolate both the values and the partial derivatives of all state-dependent operators, using tables created for each grid block. This provides a continuous description based on the interpolation operator whose accuracy is controlled by the resolution of discretization in parameter space.

Note, that this approach is different from the numerical derivatives often used in reservoir simulation [20, 48], since the nonlinear physics is fully defined by interpolated properties $f = \{\alpha_c, \beta_c\}$ and consistent with their derivatives. Due to piecewise interpolation, the approximated operators may not be differentiable at the supporting points (i.e., are piecewise differentiable). However, such functions do not cause any problems for numerical simulation partly because of discrete computer representation of floating point numbers [49]. Piecewise differentiable functions are widely used in industry-grade simulators (e.g., majority of PVT and SCAL properties are tabulated).

This representation significantly simplifies the implementation of complex simulation frameworks. Instead of keeping track of each property and its derivatives with respect to nonlinear unknowns, we can construct a linear system of equations with abstract algebraic operators representing the complex physics. The performance of this formulation benefits from the fact that all expensive evaluations can be performed using a limited number of supporting points. Finally, the performance of the nonlinear solver can be improved since the Jacobian is constructed based on a combination of piece-wise linear operators directly dependent on the nonlinear unknowns.

2.5. Computation of Partial Derivatives During Multilinear Interpolation

The key difference of the proposed approach is the replacement of conventional property computations by an interpolation procedure. Specifically, we use a piecewise multilinear generalization of piecewise bilinear interpolation for an N-dimensional space at the linearization stage. We chose this approach for its relative application simplicity in comparison with the approach proposed in [11] for compositional systems with a large number of components. Both methods have comparable accuracy and performance when applied to systems with a limited number of degrees of freedom; see [50] for details.

An interpolant approximation $A(x_1, \dots, x_N)$ to a function $\alpha(x_1, \dots, x_N)$ can be built using interpolation table values of α:

$$\{\alpha(X_{i_1}, X_{i_2}, \dots, X_{i_N}) : i_1 = 1, \dots, n_1, \dots, i_N = 1, \dots, n_N\}, \tag{2.16}$$

where $n_1, \dots, n_N$ are the numbers of points along interpolation axes. The first step of the method is to find table intervals $(X_{I_1}, X_{I_1+1}), \dots, (X_{I_N}, X_{I_N+1})$ such that

$$X_{I_1} \le x_1 \le X_{I_1+1}, \dots, X_{I_N} \le x_N \le X_{I_N+1}. \tag{2.17}$$

In order to further simplify the description, we scale each of the intervals to $(0, 1)$. That allows us to reformulate the problem to finding an approximation $\Pi(y_1, \dots, y_N)$

for a function $\pi(y_1, \dots, y_N)$ defined over the unit N-cube, described as

$$0 \le y_1 \le 1, \dots, 0 \le y_N \le 1, \tag{2.18}$$

where

$$y_i = \frac{x_i - X_{I_i}}{X_{I_i+1} - X_{I_i}}, \tag{2.19}$$

using the table values

$$\{\pi(j_1, \dots, j_N) = \alpha(X_{i_1+j_1}, , \dots, X_{i_N+j_N}) : j_1 = 0 \text{ or } 1, \dots, j_N = 0 \text{ or } 1\}. \tag{2.20}$$

The piecewise multilinear approximation is here represented via recursion just to provide a clear description. Implementation-wise, however, it is performed via a nested loop, which is far more efficient in this case. First, we define

$$\begin{aligned} \Pi_1^i &= \Pi(j_1, \dots, j_{i-1}, 1, y_{i+1}, \dots, y_N), \\ \Pi_0^i &= \Pi(j_1, \dots, j_{i-1}, 0, y_{i+1}, \dots, y_N). \end{aligned} \tag{2.21}$$

Then,

$$A = \Pi(y_1, \dots, y_N), \tag{2.22}$$

$$\Pi(j_1, \dots, j_i, y_{i+1}, \dots, y_N) = \Pi_0^i + y_i(\Pi_1^i - \Pi_0^i), \quad i = 1, \dots, N, \tag{2.23}$$

where the table values are

$$\Pi(j_1, \dots, j_N) = \pi(j_1, \dots, j_N). \tag{2.24}$$

The partial derivatives are determined in a similar way. First,

$$\begin{aligned} \Pi_1^{ki} &= \Pi^k(j_1, \dots, j_{i-1}, 1, y_{i+1}, \dots, y_N), \\ \Pi_0^{ki} &= \Pi^k(j_1, \dots, j_{i-1}, 0, y_{i+1}, \dots, y_N), \end{aligned} \tag{2.25}$$

and then

$$\frac{\delta\alpha}{\delta x_k} = \frac{\delta A}{\delta x_k} = \Pi^k(y_1, \dots, y_N), \tag{2.26}$$

$$\Pi^k(j_1, \dots, j_i, y_{i+1}, \dots, y_N) = \begin{cases} \Pi_0^{ki} + y_i(\Pi_1^{ki} - \Pi_0^{ki}), & i = 1, \dots, k, \\ \dfrac{\Pi_1^i - \Pi_o^i}{X_{I_i+1} - X_{I_i}}, & i = k, \\ \Pi_0^i + y_i(\Pi_1^i - \Pi_0^i), & i = k+1, \dots, N. \end{cases} \tag{2.27}$$

2.6. Adaptive Parameterization

The total size of the interpolation tables is defined by the number of dimensions N and the number of interpolation points n as n^N. While the dimensionality depends on the number of components and thermal assumptions in a problem of interest,

the number of interpolation points corresponds to the desired accuracy of the physical representation. Therefore, parameterization at the pre-processing stage would require a substantial amount of memory for the multi-component systems modelled at a high interpolation precision. Furthermore, the necessity of searching supporting points (i.e., operator values) for every interpolation in a huge array of data affects the performance of the simulation. Notice that due to the hyperbolic nature of some variables (e.g., overall compositions), the vast majority of parameter space remains unused [9, 11].

The adaptive parameterization approach avoids these disadvantages by removing the need for the entire pre-processing stage [11]. In this approach, supporting points are computed only when they are required by the current physical state of a control volume. The obtained operator values are then employed in the interpolation process and stored for future use.

Consequently, the method adds a new supporting point and computes appropriate operators, if the supporting point was not evaluated before, as shown in Figure 2.2. On the left, an example of a two-dimensional parameter space is shown at the moment, when the simulation occupies rectangle 2, while rectangle 1 was already used. Each rectangle has 4 vertices (for a n-dimensional space there will be hyperrectangles, or n-orthotopes, with 2^n vertices each), depicted as coloured circles. Each circle represents a supporting point with the set of corresponding operator values required to perform interpolation within the rectangle. Since rectangles share vertices, and a simulation process is likely to spread continuously over the parameter space, in most cases many operator values can be re-used.

An efficient implementation of adaptive parameterization includes two storages - hyperrectangle and vertex - which are associative containers of key-value(s) pairs with unique hash-based keys. This choice was made to ensure fast data access for high-dimensional cases. In the hyperrectangle storage, all vertices of each occupied hyperrectangle are kept together to maximize interpolation performance. The vertex storage is used when a new hyperrectangle is requested by the simulation process. If the new hyperrectangle shares some vertices with already visited hyperrectangles, then those vertices will be simply copied to the first storage, as shown by black arrows on the right in Figure 2.2. Missing supporting points will be calculated and added to both storages (shown with green arrows). This approach is crucial in high-dimensional cases when each vertex is shared among many hyperrectangles.

At the end of the simulation, the resulting sparse multi-dimensional table of stored operators represents an actual subspace of physical parameters being used in the process. For example, Figure 2.3 shows an adaptive parameterization in

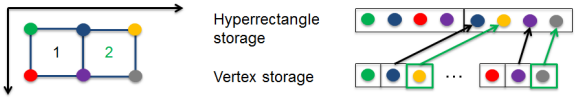


Figure 2.2: Representation of adaptive OBL storage

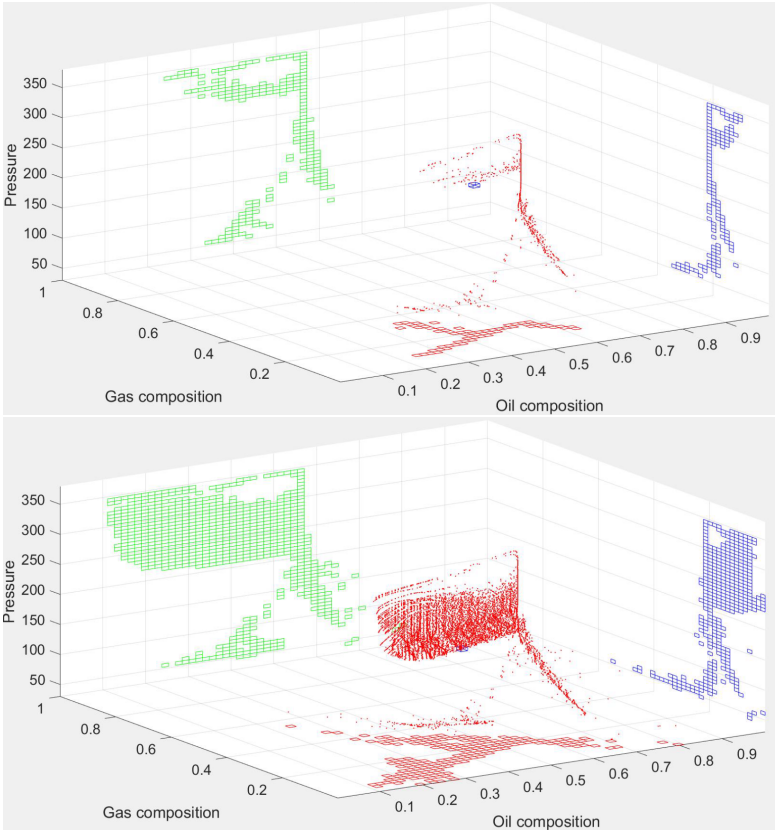


Figure 2.3: Snapshot of adaptive parameterized points in the OBL for black-oil physical kernel after 10 and 1900 days respectively

the parameter space for a black oil simulation at two different timesteps (at the beginning and the end of the simulation). The adaptive approach reproduces the exact numerical results of the pre-processing method used in [45] with greatly improved overall performance, especially for multi-component systems.

2.7. Numerical Results

In this section, we present the results of modelling with the OBL approach, implemented in the AD-GPRS simulator [10, 51]. A performance study and an error analysis are provided for different resolutions of the physical parameter space, using the results of the conventional approach as a reference solution. The improvement in the performance of OBL-based simulations is achieved by a smaller number of nonlinear iterations, the absence of iterative phase behaviour computations in the OBL method, and avoidance of derivative computations in ADETL [24], which is the underlying automatic differentiation library used by AD-GPRS for construction

and assembly of the Jacobian. However, the necessity of artificial data (values and derivatives obtained by interpolation) injection back to AD-based data structures negatively impacts the performance. That can be avoided if a stand-alone simulator is implemented entirely from the OBL perspective, as shown in Chapter 4.

For all simulations, we used two-dimensional heterogeneous reservoir based on a 7-th layer of SPE10 model shown in Figure 2.4. An injection well is placed in the middle of the reservoir, with four producers set at the corners. We applied TPFA discretization and coupled this model with different physical kernels to demonstrate the applicability of the OBL method for a general purpose simulation.

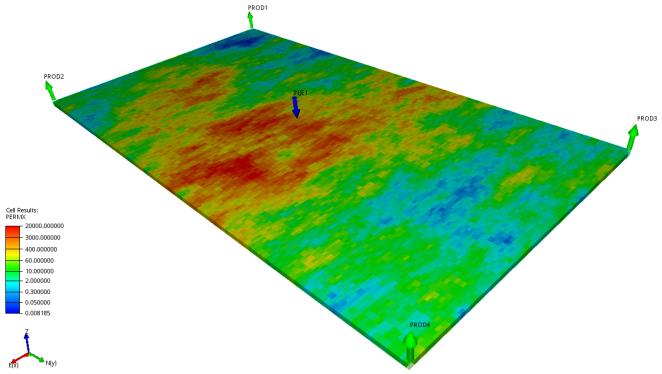


Figure 2.4: Reservoir permeability map used for all simulations

2.7.1. Isothermal Black-Oil Kernel

Here, we used a standard black-oil model, where only the gas component can dissolve in the oil phase and most of the properties are described as table-based correlations. The water injection well operates at Bottom Hole Pressure (BHP) control at a pressure $P_i = 350$ bar, and the producer well operates at $P_p = 270$ bar for the first 2000 days and then switched to $P_p = 170$ bar for the rest of simulation. The reservoir was initialized uniformly with pressure $P_0 = 300$ bar, water saturation $S_w = 0.2$, gas saturation $S_g = 0$ and bubble pressure $P_{bub} = 270$ bar. All simulations were run for 6000 days with a maximum timestep of $\Delta t = 10$ days, which corresponds to average Courant–Friedrichs–Lewy (CFL) number = 5.3.

In order to estimate the error between the reference solution and the solution obtained with OBL, the following error estimation was introduced for each of primary variables:

$$E = \frac{\sum_{i=0}^{n} \left| x_{obl}^{i} - x_{ref}^{i} \right|}{n \left(max(\boldsymbol{x_{ref}}) - min(\boldsymbol{x_{ref}}) \right)} \tag{2.28}$$

Here, n is the number of grid blocks in the model, $\boldsymbol{x_{obl}}$ and $\boldsymbol{x_{ref}}$ are solution vectors for OBL and reference simulations respectively, and x^i is a particular solution value at grid block i. This error was determined at the end of simulation (65 years) for

pressure and composition variables.

The PVT properties and relative permeabilities were used from the SPE 9 test case [52]. The obtained performance results are shown in Table 2.1. The resolution of parameter space, defined by the number of interpolation points n, is shown in the first column. The total number of nonlinear iterations for each test case is presented in the second column. The next two columns show the error in pressure and compositions (average for both components). The fifth column shows the percentage of points used for the adaptive parameterization of parameter space by the OBL approach. The sixth column reflects the CPU time required for a serial run on an Intel Xeon E5-1620 @ 3.5 GHz processor. Finally, the last two columns show the percentage of CPU time spent on generation and interpolation of all operators respectively.

Table 2.1: Results of black oil simulation

Resolution	Iters.	E_p, %	E_z, %	Space,%	CPU, sec.	Gen., %	Interp., %
Std.	6404	-	-	-	1217.2	-	-
64	4206	1.12	2.60	1.7447	659.4	< 0.1	19.7
32	3544	1.60	3.18	3.8681	555.9	< 0.1	19.7
16	3303	1.69	4.09	9.3506	542.2	< 0.1	18.8
8	2916	2.26	7.53	22.5586	482.2	< 0.1	18.8

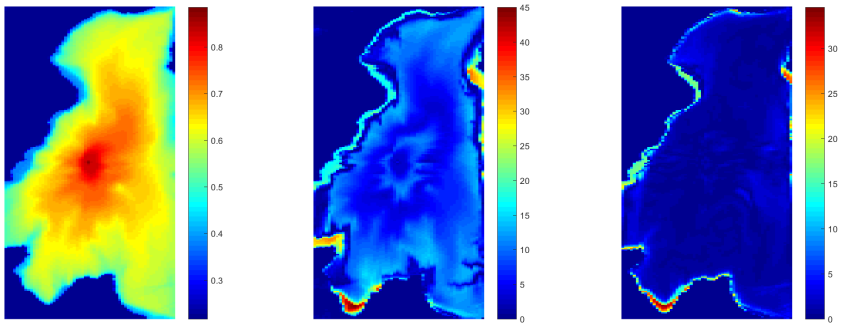
Table 2.1 demonstrates that a smaller number of interpolation points results in a simpler nonlinear system since it requires fewer Newton iterations to be solved. Note, that the number of Newton iterations performed in the OBL method is significantly lower than that for the standard simulation. Based on this and other improvements provided by the OBL approach (e.g., simplified Jacobian assembly), the corresponding CPU time is significantly reduced in comparison to the conventional approach implemented in AD-GPRS. For a black-oil kernel, the generation stage is very cheap and does almost not require any extra time. The time spent on interpolation of operators is almost independent of the resolution of parameterization space and represents the time spent for complete Jacobian and residual assembly including property and derivatives evaluation.

In this test, the overall composition of the water component introduces the largest error with respect to the rest of the unknowns. Maps of water overall composition and distribution of errors (in %) after 6000 days of the simulation are shown in Figure 2.5. It is clear that the error is distributed near the displacement fronts and is comparable with the time and space truncation error typical for reservoir simulation [53, 54].

2.7.2. Isothermal Compositional Kernel

4 Components

Next, we demonstrate the applicability of the OBL technique for an isothermal process of carbon dioxide and methane injection into the oil with composition from [55]. The initial oil was made of 4 components CO_2, C_1, C_4, and C_{10} at correspond-



Reference solution OBL error (n = 8), % OBL error (n = 64), %

Figure 2.5: Composition of water and error distribution for two OBL resolutions after 6000 days of simulation

ing compositions: 1% of carbon dioxide, 11% of methane, 38% of n-butane, and 50% of decane. We injected a mixture of 80% of CO_2 and 20% of C_1 at a BHP $P_i = 120$ bar. The production wells operated at BHP $P_p = 60$ bar. The initial pressure was $P_0 = 90$ bar and temperature $T_0 = 80^o$ C. The simulation period was 4000 days with a maximum timestep $\Delta t = 50$ days that corresponds to an average CFL = 110. A description of phase behaviour and properties based on the Peng-Robinson Equation of State [56] and Lohrenz-Bray-Clark (LBC) correlations for viscosity [57] was used in this kernel.

Table 2.2 shows the main results of the isothermal compositional simulation. The difference in the number of Newton iterations between the standard and operator-based linearization simulations is less than in the previous case, but the trend is similar with an exception for 8 points where the number of nonlinear iterations is slightly larger than for 16 points. This reflects the fact that the location of interpolation supporting points in the current version of the approach was blindly determined by uniform distribution without any analysis of nonlinearity.

At the same time, the performance of simulation with the OBL approach was improved even more significantly in comparison with the conventional simulation, than it was in the black-oil kernel. It can be explained by more expensive phase behaviour, usually required for conventional compositional simulation, in comparison with black oil. Notice that these phase behaviour calculations are almost completely absent in the OBL approach, which explains an additional CPU gain. On the other hand, the interpolation kernel still performs effectively (see Gen. and Interp. columns) since the dimensionality of parameter space is relatively low.

In this test, the overall composition of CO_2 component generates the largest error. The distribution of CO_2 composition and error maps (in %) after 2000 days of the simulation are shown in Figure 2.6. Again, the error is distributed near the

Table 2.2: Results of compositional (4 comp.) simulation

Resolution	Iters.	E_p, %	E_z, %	Space,%	CPU, sec.	Gen., %	Interp., %
Std.	626	-	-	-	562.2	-	-
64	561	0.33	0.71	0.1262	152.6	11.6	20.3
32	531	0.34	1.00	0.4392	129.8	3.0	21.7
16	498	0.34	1.73	1.7006	119.5	0.8	22.0
8	509	0.54	3.96	7.4463	123.7	0.2	21.5

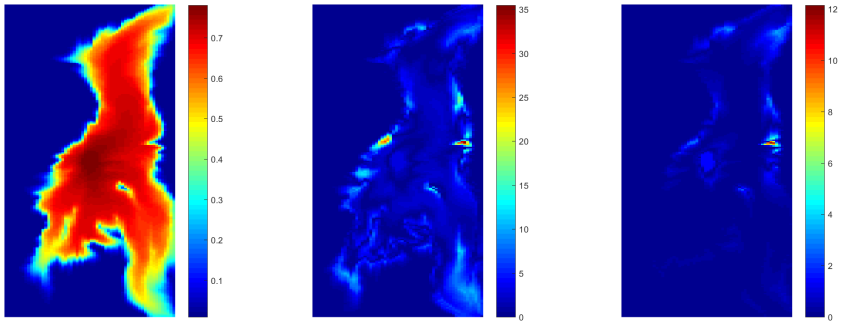


Figure 2.6: Composition of CO_2 and error distribution for two OBL resolutions after 2000 days of simulation

displacement fronts and is comparable to the typical time and space truncation error.

6 Components

To estimate the performance of the OBL approach for a system with a larger number of components, we ran a similar simulation with 6 components oil made of $\{CO_2(1\%), C_1(10\%), C_2(9\%), C_3(10\%), C_4(15\%), C_{10}(55\%)\}$. The same mixture of $\{CO_2(80\%), C_1(20\%)\}$ was used as an injection stream and the same timestep $\Delta T = 50$ days, which corresponds to an average CFL = 139, was employed in this case. The results of the simulation are presented in Table 2.3. Here, the performance of the OBL approach still improves in comparison to the conventional technique, but the speed-up is lower. This is because the performance of the OBL approach is directly dependent on the performance of a piece-wise multilinear interpolation, which becomes more expensive in the case of a high dimensional parameter space, as discussed in [50].

In Table 2.3 one can see, that both generation and interpolation times significantly increase in comparison to the previous (four component) simulations. Here, the generation of operator tables becomes the slowest procedure for a high-

resolution case due to the larger dimensionality of the parameterization space. In this case, it is more convenient to switch to the simplex-based interpolation which requires less supporting points and was fully utilized in [11] for compositional simulation based on tie-line space parameterization. Another possibility is to improve the generation stage by optimizing flash calculations [58]. Still, the most expensive high-resolution OBL case performs more than 2-times faster than the conventional compositional approach implemented in AD-GPRS. The error distribution in this case is similar to a four-component test case.

Table 2.3: Results of compositional (6 comp.) simulation

Resolution	Iters.	E_p, %	E_z, %	Space,%	CPU, sec.	Gen., %	Interp., %
Std.	577	-	-	-	829.5	-	-
64	466	0.31	0.91	0.0001	393.6	44.0	18.5
32	448	0.31	1.51	0.0017	249.2	15.0	27.4
16	431	0.35	3.02	0.0280	213.8	4.4	30.5
8	416	0.70	7.53	0.4761	202.6	1.0	31.2

3

Extensions of Operator-Based Linearisation

3.1. Thermal Extension

3.1.1. Governing Equations

In this section, we extend the description of multiphase multipcomponent mass transport for the non-isothermal case introducing an energy conservation equation:

$$\frac{\partial}{\partial t}\left(\phi \sum_{p=1}^{n_p} \rho_p s_p U_p + (1-\phi)U_r\right) + \operatorname{div} \sum_{p=1}^{n_p} h_p \rho_p \vec{u_p} + \operatorname{div}(\kappa \nabla T) + \sum_{p=1}^{n_p} h_p \rho_p \tilde{q_p} = 0. \quad (3.1)$$

Here $\mathbf{U}_p$ is phase p internal energy, $\mathbf{U}_r$ is rock internal energy, h_p denotes phase p enthalpy, and κ is thermal conduction. After application of a finite-volume discretization on a general unstructured mesh and backward Euler approximation in time we get

$$V\left[\left(\phi \sum_{p=1}^{n_p} \rho_p s_p U_p + (1-\phi)U_r\right) - \left(\phi \sum_{p=1}^{n_p} \rho_p s_p U_p + (1-\phi)U_r\right)^n\right] - \Delta t \sum_l \left(\sum_{p=1}^{n_p} h_p^l \rho_p^l \Gamma_p^l \Delta \psi^l + \Gamma_c^l \Delta T^l\right) + \Delta t \sum_{p=1}^{n_p} h_p \rho_p \mathbf{q}_p = 0. \quad (3.2)$$

Parts of this chapter have been published in SPE Journal **33**, 522 (2018) [59] and in Geothermics **74**, 7 (2018) [60]

As in Equation 2.3, we neglected capillarity, gravity and used a Two-Point Flux Approximation (TPFA) with an upstream weighting. Therefore, ΔT^l is the temperature difference between neighboring blocks. In addition, Γ_c^l corresponds to conductive transmissibility which includes thermal conduction of all phases (including solid) and geometry as

$$\Gamma_c^l = \Gamma^l \left(\phi(\sum_{p=1}^{n_p} s_p \kappa_p) + (1-\phi)\kappa_r \right). \tag{3.3}$$

Similarly to Equation 2.3, all terms of the equation are defined at $n+1$ timestep, except the second part of accumulation term denoted by n superscript.

Operator form of the conservation equations

For the non-isothermal case, physical state $\boldsymbol{\omega}$ is also defined by T (or h) in addition to p and z_c. All terms in Equation 3.1 can be characterized as functions of the spatial coordinates $\boldsymbol{\xi}$ and physical state $\boldsymbol{\omega}$ as follows:

- $\mathbf{U}_p(\boldsymbol{\omega})$ – phase internal energy,
- $\mathbf{U}_r(\boldsymbol{\xi}, \boldsymbol{\omega})$ – rock internal energy,
- $h_p(\boldsymbol{\omega})$ – phase enthalpy,
- $\kappa(\boldsymbol{\xi}, \boldsymbol{\omega})$ – thermal conduction.

Next, for simplicity we assume that the rock internal energy and thermal conduction are spatially homogeneous, thus

$$U_r = f(\boldsymbol{\omega}), \quad \kappa = f(\boldsymbol{\omega}). \tag{3.4}$$

In order to apply the described approximation method, we rewrite Equation 3.2, representing each term as a product of state-dependent and space-dependent operators [43]. Besides, we assume the initial porosity as a pseudo-physical state variable ($\phi_0 \in \boldsymbol{\omega}$). The modified energy conservation equation becomes

$$\begin{aligned} a_e(\boldsymbol{\xi})(\alpha_e(\boldsymbol{\omega}) - \alpha_e(\boldsymbol{\omega}_n)) \quad &+ \quad \sum_l b_e(\boldsymbol{\xi}, \boldsymbol{\omega}) \beta_e(\boldsymbol{\omega}) \\ &+ \quad \sum_l c_e(\boldsymbol{\xi}, \boldsymbol{\omega}) \gamma_e(\boldsymbol{\omega}) + \theta_e(\boldsymbol{\xi}, \boldsymbol{\omega}, \boldsymbol{u}) = 0, \end{aligned} \tag{3.5}$$

where

$$a_e(\boldsymbol{\xi}) = V(\boldsymbol{\xi}), \tag{3.6}$$

$$\alpha_e(\boldsymbol{\omega}) = \phi(\sum_{p=1}^{n_p} \rho_p s_p U_p - U_r) + U_r, \tag{3.7}$$

$$b_e(\boldsymbol{\xi}, \boldsymbol{\omega}) = b(\boldsymbol{\xi}, \boldsymbol{\omega}), \tag{3.8}$$

$$\beta_e(\boldsymbol{\omega}) = \sum_{p=1}^{n_p} h_p^l \rho_p^l \frac{k_{rp}^l}{\mu_p^l}, \tag{3.9}$$

$$c_e(\boldsymbol{\xi}) = \Delta t \Gamma^l T^l, \tag{3.10}$$

$$\gamma_e(\boldsymbol{\omega}) = \phi(\sum_{p=1}^{n_p} s_p \kappa_p - \kappa_r) + \kappa_r, \tag{3.11}$$

$$\theta_e(\boldsymbol{\xi}, \boldsymbol{\omega}, \boldsymbol{u}) = \Delta t \sum_{p=1}^{n_p} h_p \rho_p q_p(\boldsymbol{\xi}, \boldsymbol{\omega}, \boldsymbol{u}). \tag{3.12}$$

In these derivations, T^l is temperature difference between two mesh grid blocks connected by interface l.

This representation allows us to identify and distinguish the physical state-dependent operators - $\alpha_e, \beta_e, \gamma_e$ in the energy conservation equation.

Approximation of fluid and rock thermal properties

The proposed approach simplifies the description of fluid and rock properties by building approximation interpolants for the operators $\alpha_c, \beta_c, \alpha_e, \beta_e, \gamma_e$ within the parameter space of a simulation problem. For a general non-isothermal compositional problem with n_c components, the method requires $[2n_c + 3]$ operators.

The values of the operators are fully determined by the set of $N = [n_c + 1]$ independent variables $\{p, T, z_1, \dots, z_{n_c-1}\}$. The pressure and temperature ranges in the compositional parameter space can usually be determined by conditions specified for wells, while the overall composition is naturally bounded by the interval [0,1]. As mentioned above, we add the porosity as a pseudo-physical state variable with the corresponding range.

Next, we parametrize the interval of each state variable using, for simplicity, the same number $n = n_1 = \cdots = n_N$ of uniformly distributed points on the intervals of parameters, according to Equation 2.16. This results in a set of vectors $(p_i, T_i, z_{1_i} \dots, z_{n_c-1_i}, \phi_i) : i = 1, \dots, n$, which can be interpreted as a discretization of physical-state space in the simulation. At the pre-processing stage, or adaptively, we can evaluate the operators $\alpha_c, \beta_c, \alpha_e, \beta_e, \gamma_e$ at every point in the discrete parameter space and store them in $(n_c + 2)$-dimensional tables A_e and Γ_e and $(n_c + 1)$-dimensional tables A_c, B_c, B_e.

3.1.2. Numerical Results

In the next few subsections, we introduce numerical results of simulations based on the described approach. First, in Subsection 3.1.3, we present a three-dimensional heterogeneous model describing a realistic reservoir for low-enthalpy geothermal operations. We show a comparison between simulation using the conventional geothermal formulation in AD-GPRS [61] and using the COMSOL simulation platform [62], which was utilized for low-enthalpy geothermal simulations in the past [63]. COMSOL is interactive simulation software, where one can model problems from different application fields (e.g., electrical, mechanical, chemical or fluid flow). The penalty of this generality is the low computational performance of the simulation.

Further, in Subsection 3.1.4, we display the results of a simple sensitivity analysis of the geothermal model, based on the variation of a doublet position. In Subsection 3.1.5, we present a convergence study of the operator-based linearization for the one-component geothermal model based on different resolutions of parameterized tables using the same reservoir. Finally, similar convergence analysis is performed for a geothermal system with natural gas co-production in low-enthalpy (Subsection 3.1.6) and high-enthalpy (Subsection 3.1.7) regimes.

3.1.3. Three-Dimensional Realistic Heterogeneous Geothermal Reservoir

Here, we present the results of a geothermal simulation based on the realistic geological model introduced by [64]. This model is one of the realizations of sedimentological simulation for the Nieuwerkerk sedimentary formation in the West Netherlands Basin. These realizations have been generated for an investigation of the performance of a doublet (a pair of injection and production wells) in low-enthalpy geothermal systems. Reservoir dimensions are 1km x 2 km x 50 m and the discretized model contains 50x100x20 grid blocks. Both wells are placed in the middle of the model, along the long side (Y-axis) with a spacing of 1 km (see Figure 3.1). The fluvial sandstone bodies are located along the longer side of the reservoir, with the porosity distributed within the range [0.16, 0.36] and permeability distributed within the range [6, 3360] mD. The boundary conditions along the short sides (X-axis) of the reservoir are set to a constant initial pressure; the boundary conditions at the other sides are set to no-flow. The reservoir in Figure 3.1 is vertically scaled up by a factor of 5 for better visibility.

Both wells operate under a constant water rate control q = 2400 m^3/day. The production well consumes energy from the reservoir, producing hot water at a reservoir temperature T_{prod} = 348 K. The injection well returns cold water to the reservoir at T_{inj} = 308 K, forcing a cold-front propagation to the production well. Both wells are perforated through all layers of the model. Two energy-transfer mechanisms are involved in this process: fluid flow and heat conduction. When the cold front arrives at the production well, the temperature drops below a certain limit (338 K in this study) and the so-called doublet lifetime is reached.

To verify the conventional geothermal formulation in the AD-GPRS framework, we compare our simulation results with the results of a COMSOL simulation described in [64]. For both simulations, we used similar correlations for the proper-

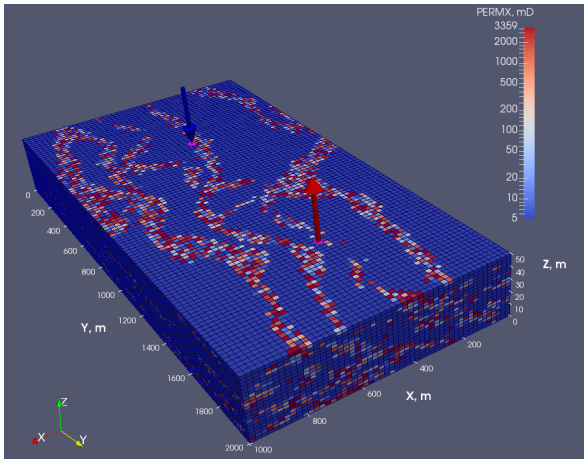


Figure 3.1: Permeability distribution and geothermal doublet configuration of the geothermal reservoir realization

ties of fluid and rock described by [63]. In Figure 3.2, we show the comparison between the temperatures at the production well in both cases.

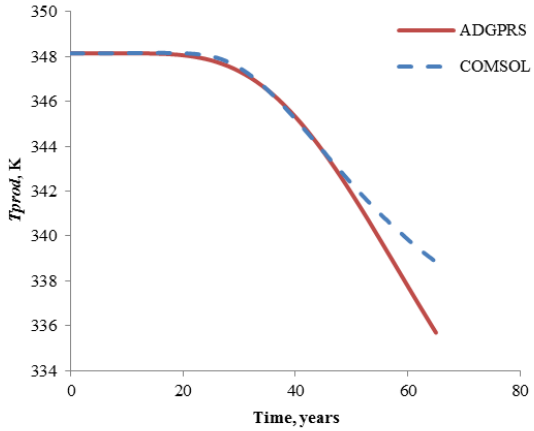


Figure 3.2: Comparison between COMSOL and AD-GPRS realistic heterogeneous reservoir simulation results

It can be seen that the AD-GPRS and COMSOL results are very similar until the time around 50 years when the temperature reduction is already quite significant. These differences can be explained by the differences in the spatial discretization since AD-GPRS is using a conservative Finite Volume discretization while COMSOL supports a general Finite-Element (i.e., non-conservative) discretization. Based on this fact, we believe that the temperature reduction is more realistic in AD-GPRS

simulation; however, further investigation is required. We use the conventional geothermal formulation by AD-GPRS as a reference solution and compare it with the proposed OBL approach.

3.1.4. Sensitivity Analysis of Geothermal Doublet Position

The importance of sensitivity analysis can hardly be overestimated for risk management in geothermal reservoir development. Sometimes, key performance indicators dramatically change with a small variation of, as it might seem, insignificant parameters. To demonstrate it, we ran a series of geothermal simulations, using the described above model as a base case and varying just one parameter – doublet position.

Both wells were simultaneously shifted in the lateral direction from the base grid cell to all neighbouring cells (including diagonal neighbours) so that their mutual arrangement remained unchanged. Wells were controlled by rate $q = 2400$ m^3/day during the whole simulation period of 200 years for each of the models. This light deviation in wells position provoked a large difference in geothermal doublet lifetime (up to 20 years or more), even if we discard three cases with the biggest lifetime (see Figure 3.3). The two numbers in square brackets denote the offsets (in grid cells) of the doublet position from the base case along X and Y axes of the reservoir respectively. The base case is therefore denoted as [0;0].

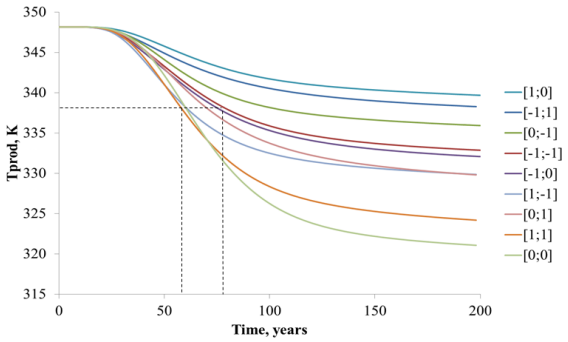


Figure 3.3: Variation of the cold-front arrival time for doublet lateral position deviations of one grid cell from the base case [0;0]

That difference can be explained by different distributions of energy in the reservoir, caused by variation of connectivity between injection and production wells. Thorough sensitivity and uncertainty analyses help to mitigate reservoir development risks but require a large number of simulations. A tradeoff between the number of models to run and available time/computational resources always occurs, that is why the computational performance of reservoir simulation is so important.

3.1.5. One-Component Geothermal Model

Convergence of Operator-Based Linearization

Here, we compare the results of simulation with OBL performed at the different resolutions of interpolation tables and the reference solution based on the conventional (continuous) linearization method. We use a modified variant of the original test case with uniformly distributed thermal properties of rock while rock permeabilities and porosities remain heterogeneous. Both wells work at the same rate control q = 2400 m^3/day. Each simulation was performed with a different number of points in the interpolation table which, for simplicity, was equal for all of the unknowns (p, t and ϕ), while values were uniformly distributed within the range between p_{min}=160 bar and p_{max}=250 bar for pressure, T_{min}=308 K and T_{max}=349 K for temperature, and between 0 and 1 for porosity.

The results of the comparison are presented in Table 3.1. The number of points used for interpolation operators is shown in the first column. The second column contains the number of nonlinear iterations, which are directly proportional to the simulation time. The third and fourth columns represent the error in the temperature and pressure solution (obtained according to Equation 2.28), respectively. The last column shows relative single average linearization cost (in terms of CPU time) of the OBL-based simulator prototype, described in Chapter 4, with respect to the standard AD-GPRS simulator.

Table 3.1: Results of 3D homogeneous simulation

Resolution	Newton iters.	E_p, %	E_T, %	Linearization cost per Newt.
Std.	174	-	-	1
64	182	0.0005	0.002	0.051
32	212	0.001	0.007	0.054
16	231	0.002	0.028	0.051
8	240	0.008	0.116	0.048
4	245	0.039	0.561	0.051
2	195	0.24	3.775	0.045

From Table 3.1, the results based on any parameterization approach with a resolution of 8 and higher show relatively small error, while the linearization is performed about 20 times faster in comparison with AD-based linearization. At the same time, this cost does not change significantly with an increase in resolution in the OBL approach. The error in this simulation is so small because all interpolated properties, based on correlations from [63], have substantially linear behaviour with respect to the nonlinear unknowns. It seems sufficient in this model to perform the operator-based linearization using the resolution of 8 points.

The comparison of the production temperatures and temperature distribution also supports this conclusion, as shown in Figure 3.4 and Figure 3.5 respectively. Figure 3.4 demonstrates a good match between reference and parameterization approach based solution with 8 points, while simulation based on the coarsest table introduces non-physical initial growth of temperature due to a very coarse approximation of operators involved in the energy equation.

In Figure 3.5, the cold front propagates over the top layer of the reservoir. The

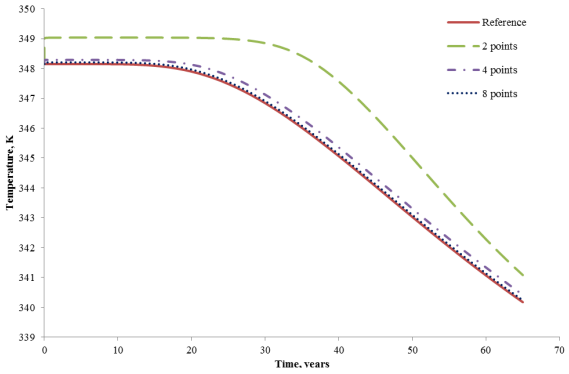


Figure 3.4: Comparison of temperatures at production well based on different linearization approaches

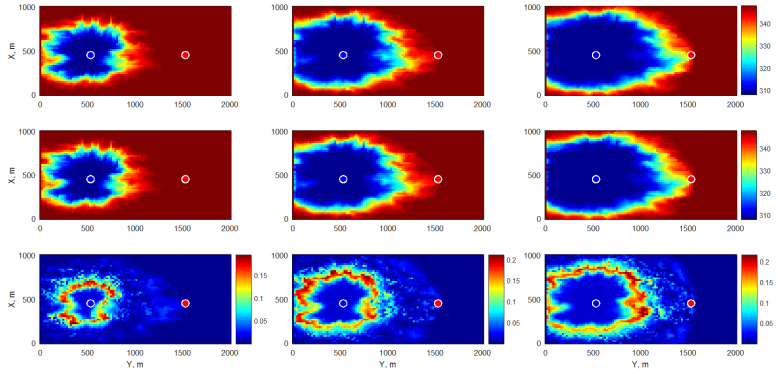


Figure 3.5: Temperature front after 15 (a), 30 (b) and 45 (c) years for the conventional linearization (upper), OBL with 8 point resolution (middle), and absolute difference between them (lower) in the top layer of the reservoir

top row in Figure 3.5 represents the results from the conventional linearization after 15, 30 and 45 years of simulation. The middle row shows the results obtained with OBL using a resolution of 8 points at the same times. The lower row displays the absolute difference between the reference and OBL solutions. The injection-well position is marked with the blue circle; the production-well position is shown with the red one.

Analysis of Linearization Operators

In Figure 3.6, we present the most-nonlinear operators used in the proposed linearization approach. All of them are built based on the 64-point interpolation tables in parameter space. These operators correspond to the linearization of mass-accumulation α_w and flux β_w terms in the water-component mass equation and energy accumulation α_e and conduction γ_e in the energy equation (see Equation 2.10 and Equation 3.5). They are represented by isosurfaces in pressure, temperature

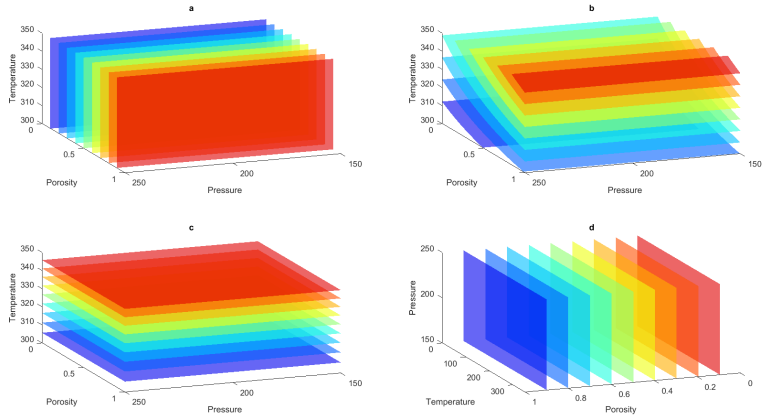


Figure 3.6: Physics-based operators of water mass accumulation α_w (a), energy accumulation α_e (b), water mass flux β_w (c) and energy conduction γ_e (d) terms

and porosity parameter space. As expected, all of the operators behave almost linearly in parameter space that explains why the results of simulation with just 8 points are so accurate.

3.1.6. Two-Component Low-Enthalpy Geothermal Model

Here, we demonstrate geothermal simulation with gas co-production using the same three-dimensional reservoir. The injection well injects cold water at T_{inj}=308 K and controlled by a water rate q = 1200 m^3/day. The initial reservoir composition of the gas component (methane) z_g = 0.1, with initial pressure p=100 bar, and initial temperature T=348 K. The production well is controlled by the bottom-hole pressure p_{prod}=70 bar. Since the injection rate is now 2 times lower, we increased simulation time to 100 years.

We used phase behaviour and densities based on the Peng-Robinson Equation of State [56] with critical parameters described in Table 3.2. For the enthalpy of the mixture, we used a correlation described in [65] with parameters from the same table. The Lohrenz-Bray-Clark (LBC) correlations were used for the viscosities of each phase [57].

Table 3.2: Parameters for properties

Comp.	T_c (K)	P_c (bar)	V_c	ACF	M_w	CPG_1	CPG_2	CPG_3	CPG_4
C_1	190.6	46.04	0.098	0.013	16.04	19.251	0.0521	1.197e-5	1.132e-8
H_2O	646.8	220.60	0.056	0.344	18.015	32.243	0.0019	1.055e-5	-3.596e-9

Convergence of Operator-Based Linearization

We performed a set of simulations with 6 different interpolation-table resolutions and compared solutions with the reference solution based on the conventional

approach. For parameterization, we used uniformly distributed points between p_{min}=60 bar and p_{max}=120 bar for pressure, T_{min}=300 K and T_{max}=360 K for temperature, and finally between 0 and 1 for both composition and porosity. The results can be seen in Table 3.3. Columns 1-4 are same as in Table 3.1, the fifth column shows the error in gas composition, and the last column shows a relative cost of Operator-Based Linearization per Newton iteration in comparison with AD-based linearization.

Table 3.3: Results of 3D, two-phase low-enthalpy simulation

Resolution	Newton iters.	E_p, %	E_T, %	E_{z_g}, %	Linearization cost per Newt.
Std.	1247	-	-	-	1
64	974	0.155	0.809	1.734	0.029
32	968	0.557	1.529	3.692	0.028
16	977	1.305	2.567	6.414	0.028
8	890	1.977	4.288	11.215	0.028
4	890	1.628	5.308	11.397	0.027
2	867	2.191	5.618	13.207	0.027

The two-component two-phase geothermal model is more challenging for the operator-based linearization approach in comparison to the previous case. However, the error of the OBL method drops significantly with the increasing resolution of interpolation tables. Here, the cost of the Operator-based Linearization is more than 30 times lower in comparison with the AD-based linearization. This happened because, in AD-GPRS, an iterative solution of EoS is required in the two-phase region, while in OBL, it only required for a limited number of parameterization points. For a higher OBL resolution, the linearization cost insignificantly increases.

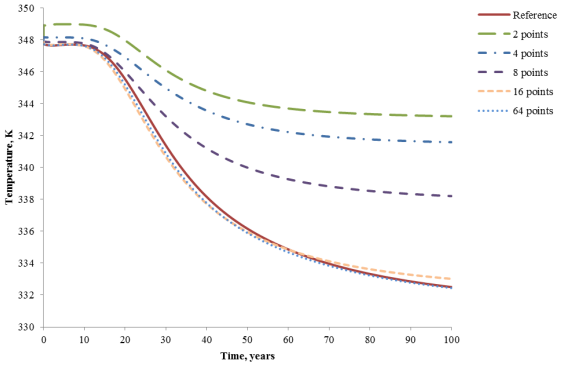


Figure 3.7: Comparison of temperatures at production well based on different linearization approaches in low-enthalpy model with co-production

Production temperatures for the reference solution and solutions based on linearization operators are shown in Figure 3.7. Here, the non-physical behaviour for

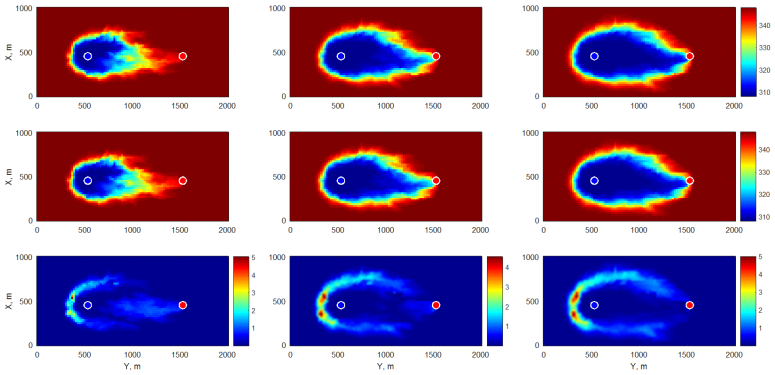


Figure 3.8: Temperature front after 20 (a), 40 (b) and 60 (c) years for the conventional linearization (upper), OBL with 64-point resolution (middle), and absolute difference between them (lower) in the top layer of the reservoir in low-enthalpy model with co-production

2-point resolution is similar to the previous case. This behaviour is quickly stabilized for the cases with a higher resolution.

The comparison of the thermal fronts is shown in Figure 3.8. Here, one can see that the reference solution (upper row) and the solution based on the OBL with 64 points (middle row) mostly match, and the largest errors of 5 degrees are primarily observed around the thermal front. Importantly, the maximum error does not grow along with the simulation. Compared to the previous case, the flow is now more influenced by the production well because of the changed pressure boundary conditions. Therefore, injected cold water primarily flows towards the production well causing faster breakthrough despite a lower injection rate.

Analysis of Linearization Operators

In Figure 3.9, the 3D iso-surfaces are shown to characterize the most nonlinear operators for the case of the two-phase geothermal model. These operators correspond to the linearization of mass accumulation α_g and flux β_g terms for the gas component in the mass equation and energy accumulation α_e and flux β_e in the energy equation. All of the operators are built based on the 64-point interpolation table. They are shown as functions of pressure, temperature and composition at a constant value$\phi = 0.2$. Unlike for the pure geothermal case, all operators are more nonlinear as functions of all state variables.

3.1.7. Two-Component High-Enthalpy Geothermal Model

Here, we demonstrate geothermal simulation with gas co-production for a high-enthalpy reservoir. The initial temperature, pressure and composition of the gas component (methane) was adjusted to T=500 K, p=100 bar, and z_g = 0.1, which makes the original mixture close to a critical fluid at reservoir conditions. The injection temperature stays the same T_{inj}=308 K, and the injection well operates under a constant water rate control q = 120 m^3/day. Only the top layer of the reservoir was modelled because of the significant reduction in simulation speed.

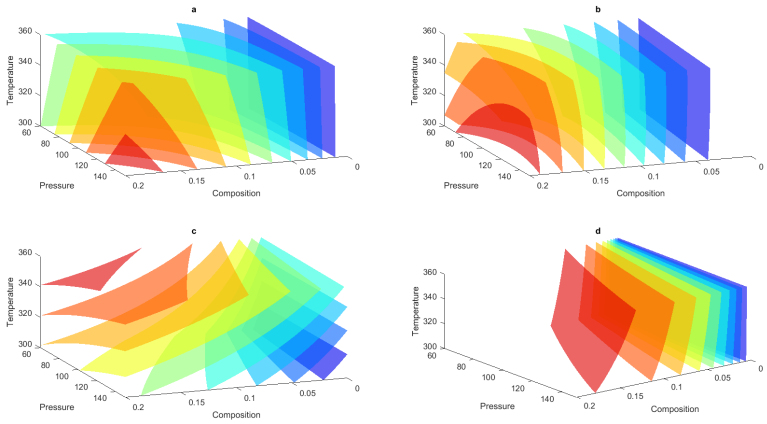


Figure 3.9: Physics-based operators of mass accumulation for gas component α_g (a), mass flux for gas component β_g (b), energy accumulation α_e (c) and energy flux β_e (d) terms in low-enthalpy model with co-production

This drop is related to more-expensive phase behaviour evaluations (in the near-critical region) for the reference solution and a lower limit of simulation timestep to suppress instabilities associated with high-enthalpy systems [66].

Convergence of Operator-Based Linearization

Similarly to the previous runs, we performed simulations with 6 resolutions of the interpolation table and compared them with the reference model. For parameterization, we used uniformly distributed points between p_{min}=60 bar and p_{max}=290 bar for pressure, T_{min}=300 K and T_{max}=510 K for temperature, and between 0 and 1 for both composition and porosity. The results can be seen in Table 3.4 with columns similar to Table 3.3.

Table 3.4: Results of 2D two-phase high-enthalpy simulation

Resolution	Newton iters.	E_p, %	E_T, %	E_{z_g}, %	Linearization cost per Newt.
Std.	7617	-	-	-	1
64	2715	0.023	0.092	0.711	0.027
32	2629	0.071	0.356	2.257	0.027
16	2489	0.096	0.496	3.443	0.026
8	2118	0.07	0.615	3.748	0.026
4	2113	0.088	1.104	3.99	0.027
2	1901	0.108	4.279	3.672	0.026

In comparison to the low-enthalpy case, the high-enthalpy simulation requires much more Newton iterations to converge. That is related to the fact that the high-enthalpy system corresponds to more nonlinear pressure-temperature dependencies. However, the OBL method introduces smaller errors, still requiring 64 points to keep the errors below 1%. We believe that overall accuracy increased

because the model has become 2-dimensional, therefore some factors affecting the solution, such as vertical conduction, are no longer present. Still, the error for OBL decreases as the resolution of interpolation tables increases. The cost of linearization per Newton iteration behaves similarly to the low-enthalpy case.

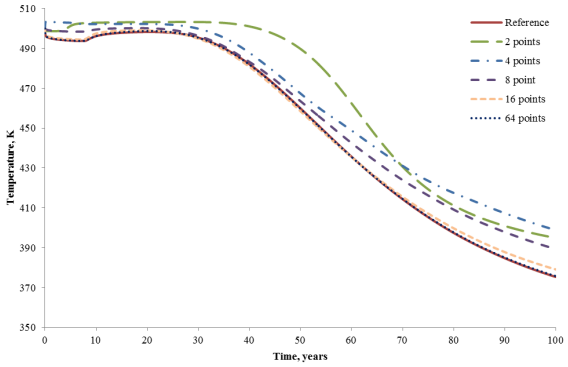


Figure 3.10: Comparison of temperatures at production well based on different linearization approaches in the high-enthalpy model with co-production

Production temperatures for the reference solution and solutions based on the linearization operator are shown in Figure 3.10. The lowest resolution in the physical tables introduces a large error in the temperature breakthrough time, as was expected from Table 3.4. With higher resolutions, the behaviour becomes closer to the reference solution, even though some non-physical results can be observed at intermediate resolutions. For example, the 4-point resolution in Figure 3.10 demonstrates production temperature higher than initial temperature right at the beginning of simulation.

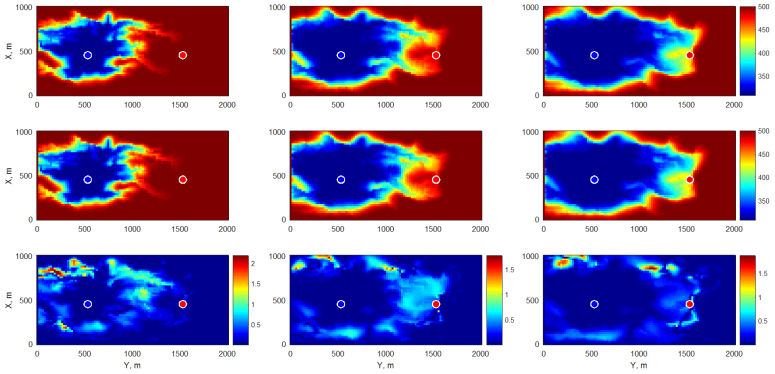


Figure 3.11: Temperature (condensation) front after 20 (a), 40 (b), and 60 (c) years for the conventional linearization (upper), OBL with 64-point resolution (middle), and absolute difference between them (lower) in high-enthalpy model with co-production

The spatial distribution of temperature demonstrates a certain discrepancy be-

tween two approaches — up to 2 degrees, as can be seen in Figure 3.11. As in previous cases, the maximum error does not increase along with the simulation and is concentrated around the thermal front.

Analysis of Linearization Operators

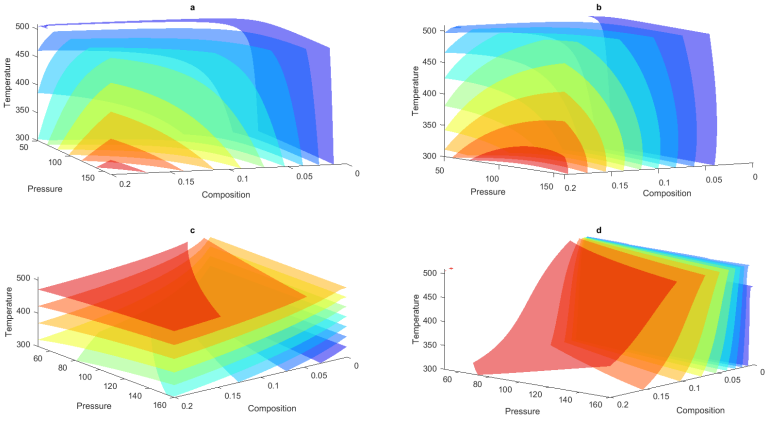


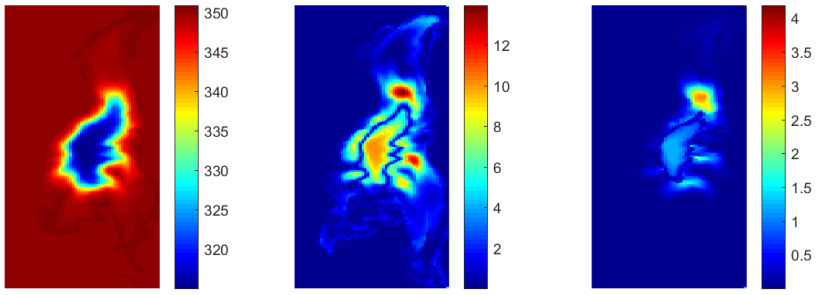
Figure 3.12: Physics-based operators of mass accumulation for gas component α_g (a), mass flux for gas component β_g (b), energy accumulation α_e (c) and energy flux β_e (d) terms in high-enthalpy model with co-production

In Figure 3.12, we plot again 3D isosurfaces to describe operators in the case of a high-enthalpy geothermal model with co-production. Here, we show the same operators as in Figure 3.9. For the high-enthalpy case, all operators demonstrate more nonlinear behaviour. That is partially due to the closeness to superheated conditions of the gas-water mixture and partially due to the larger interval of changes in temperature covered in simulations. We hope that in the future work, the detailed analysis of parametrized operators will help to improve the nonlinear solver in geothermal simulations.

3.1.8. Thermal Compositional Model With 4 Components

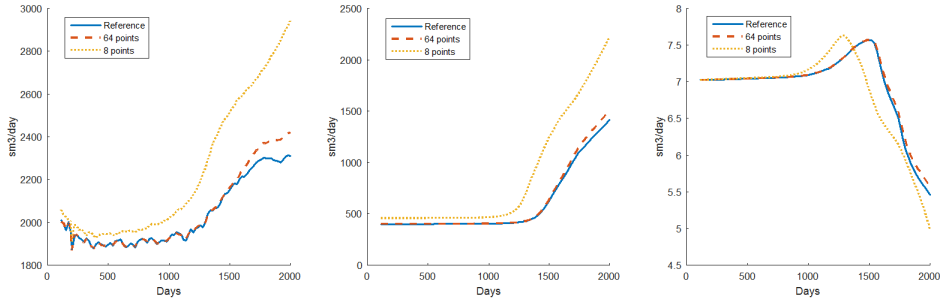
The next simulation model was built on a thermal-compositional physical kernel, extending the isothermal model described in subsection Subsection 2.7.2. The initial and injection conditions stayed the same as in the previous example except that the injection mixture had a lower temperature of T=315 K. The simulation period was 2000 days with a maximum time step of Δt=20 days. The temperature distribution at the last time step and the corresponding errors are depicted in Figure 3.13. The errors are concentrated near the cooling front (similar to composition errors located near the displacement front) and the injection well. The latter can be explained by a larger nonlinearity in the energy conservation equation, introduced by a correlation for enthalpy.

The convergence results of the thermal-compositional simulation, presented in Table 3.5, are similar to those for the isothermal model, provided by Table 2.2



(a) Reference (b) Error for 8 points, % (c) Error for 64 points, %

Figure 3.13: Temperature solution at t=2000 days



(a) Total gas injection rate (b) Total gas production rate (c) Total oil production rate

Figure 3.14: Total well rates for reference solution, OBL with 8 points, and OBL with 64 points

Table 3.5: Results of non-isothermal compositional simulation

Resolution	Iters.	E_p, %	E_T, %	E_{CO_2}, %	E_{C_1}, %	E_{C_4}, %	$E_{C_{10}}$, %	Space,%	CPU, sec
Std.	647	-	-	-	-	-	-	-	628
64	587	0.12	0.12	0.19	0.37	0.20	0.21	0.005	317
32	553	0.12	0.27	0.33	0.67	0.35	0.37	0.037	234
16	536	0.14	0.48	1.15	2.02	1.21	1.35	0.241	209
8	555	0.41	1.21	2.79	5.16	3.64	3.86	2.371	217

(the description of columns also matches, except that errors are provided for each component individually, but still according to Equation 2.28). In this simulation, the region of adaptive parameterization of physical space drops down to 0.001% which reflects the importance of the adaptive approach for higher dimensional systems (i.e., systems with more nonlinear unknowns per control volume).

3.2. Buoyancy Extension

In this section, another important extension of OBL to buoyancy dominated systems is described.

3.2.1. Phase Potential Upwinding (PPU)

In the conventional modelling approach, the introduction of buoyancy in multiphase flux calculations assumes that each phase has its own phase potential difference at a given interface ij, allowing counter-current flow:

$$\Phi_{p,ij} = \left(p_j - p_i - \delta_{p,ij}(D_j - D_i)\right) \tag{3.13}$$

$$\delta_{p,ij} = \begin{cases} \dfrac{\delta_{p,i} + \delta_{p,j}}{2}, & \text{if phase } p \text{ appears in both cell } i \text{ and } j; \\ \delta_{p,i}, & \text{if phase } p \text{ appears only in cell } i; \\ \delta_{p,j}, & \text{if phase } p \text{ appears only in cell } j; \\ 0, & \text{if phase } p \text{ doesn't exist in neither cell } i \text{ nor } j, \end{cases} \tag{3.14}$$

where $\delta_{p,i} = \delta_p(\boldsymbol{\omega}_i) = g\rho^m_{p,i}$, g - the acceleration of gravity, $\rho^m_{p,i}$ - mass density of a phase p in a control volume i. Numerical fluxes are usually computed using a phase potential upwinding (PPU) strategy, in which phase mobilities are selected depending on a sign of the corresponding phase potential difference separately for each phase.

Straightforward implementation of PPU within the OBL approach implies an increase in the number of flux operators from n_c to $n_c n_p$, since phases should be treated separately. In addition, a mass density operator δ_p has to be introduced for each phase. Striving to reduce the amount of required interpolations, we evaluate a single mass density value per phase for the united control volume of adjacent blocks, instead of averaging the two values obtained for each of the blocks. Hence, Equation 3.14 becomes:

$$\delta_{p,ij} = \delta_p(\boldsymbol{\omega}_{ij}) \tag{3.15}$$

$$\boldsymbol{\omega}_{ij} = \frac{\boldsymbol{\omega}_i + \boldsymbol{\omega}_j}{2} \tag{3.16}$$

Taking into account buoyancy, Equation 2.10 is transformed into:

$$a(\boldsymbol{\xi})\left(\alpha_c(\boldsymbol{\omega}) - \alpha_c(\boldsymbol{\omega}_n)\right) + \sum_{j\in L(i)}\sum_p b_p(\boldsymbol{\xi},\boldsymbol{\omega})\beta_{cp}(\boldsymbol{\omega}) + \theta_c(\boldsymbol{\xi},\boldsymbol{\omega},\boldsymbol{u}) = 0, \quad c = 1,\dots,n_c, \tag{3.17}$$

where

$$b_p(\boldsymbol{\xi},\boldsymbol{\omega}) = \Delta t \Gamma_{ij} \Phi_{p,ij}, \tag{3.18}$$

$$\beta_{cp}(\boldsymbol{\omega}) = x_{cp,ij}\rho_{p,ij}\lambda_{p,ij} = \begin{cases} x_{cp,i}\rho_{p,i}\lambda_{p,i} & \text{if } \Phi_{p,ij} < 0 \\ x_{cp,j}\rho_{p,j}\lambda_{p,j} & \text{otherwise.} \end{cases} \tag{3.19}$$

Similarly, the energy balance equation (Equation 3.5) becomes:

$$a_e(\boldsymbol{\xi})(\alpha_e(\boldsymbol{\omega}) - \alpha_e(\boldsymbol{\omega}_n)) + \sum_{j\in L(i)} \sum_p b_{ep}(\boldsymbol{\xi},\boldsymbol{\omega})\beta_{ep}(\boldsymbol{\omega}) + \sum_l c_e(\boldsymbol{\xi},\boldsymbol{\omega})\gamma_e(\boldsymbol{\omega}) + \theta_e(\boldsymbol{\xi},\boldsymbol{\omega},\boldsymbol{u}) = 0, \quad (3.20)$$

where

$$b_{ep}(\boldsymbol{\xi},\boldsymbol{\omega}) = b_p(\boldsymbol{\xi},\boldsymbol{\omega}), \quad (3.21)$$

$$\beta_{ep}(\boldsymbol{\omega}) = h_{p,ij}\rho_{p,ij}\lambda_{p,ij} = \begin{cases} h_{p,i}\rho_{p,i}\lambda_{p,i} & \text{if } \Phi_{p,ij} < 0 \\ h_{p,j}\rho_{p,j}\lambda_{p,j} & \text{otherwise.} \end{cases} \quad (3.22)$$

Therefore, OBL with PPU requires $n_c + n_c n_p + n_p$ operators for the isothermal problem with n_c components and n_p phases, and $n_c + n_c n_p + 2n_p + 2$ for the non-isothermal one.

3.2.2. Component-Potential Upwinding (CPU)

Striving to reduce the number of operators, we introduced a following component density paradigm:

$$\rho_c(\boldsymbol{\omega}) = \frac{\sum_{p=1}^{n_p} \rho_p^m x_{cp}\rho_p\lambda_p}{\sum_{p=1}^{n_p} x_{cp}\rho_p\lambda_p}, \quad c = 1,\dots,n_c. \quad (3.23)$$

Using this mobility-averaged density of a component allows to obtain a single component-potential at an interface ij for upwinding. Counter-current flow is still possible, but each component now moves only in one direction, apart from PPU:

$$\Phi_{c,ij} = \left(p_j - p_i - \delta_{c,ij}(D_j - D_i)\right), \quad (3.24)$$

$$\delta_{c,ij} = \delta_c(\boldsymbol{\omega}_{ij}) \quad (3.25)$$

$$\boldsymbol{\omega}_{ij} = \frac{\boldsymbol{\omega}_i + \boldsymbol{\omega}_j}{2} \quad (3.26)$$

$$\delta_c(\boldsymbol{\omega}_i) = g\rho_{c,i}. \quad (3.27)$$

The physical interpretation of this approximation is based on the fact that in cross-current flow, independently of phase directions, the total mass of the component is moving to a single direction. In addition, this scheme avoids summation across phases for each component and, thus, reduces the number of operators involved. It implies minimal changes in the original mass and energy balance equations (Equation 2.10, Equation 3.5) by making the space operator $b(\boldsymbol{\xi},\boldsymbol{\omega})$ component-dependent:

$$b(\boldsymbol{\xi},\boldsymbol{\omega}) = b_c(\boldsymbol{\xi},\boldsymbol{\omega}) = \Delta t\Gamma_{ij}\Phi_{c,ij}. \quad (3.28)$$

In this scheme, OBL requires $3n_c$ operators for the isothermal problem with n_c components and n_p phases, and $3n_c + 3$ for the non-isothermal one.

3.2.3. Independent Upwinding (IU)

Following the idea of hybrid upwinding (HU) [67] to the full extent, we completely separated computations of viscous- and buoyancy-induced flow. For each of the two parts, the upstream direction is obtained independently, based on the pressure difference for the first and based on the depth difference for the second. This implies

$$a(\boldsymbol{\xi})\left(\alpha_c(\boldsymbol{\omega}) - \alpha_c(\boldsymbol{\omega}_n)\right) + \sum_{j \in adj(i)} \left(b(\boldsymbol{\xi},\boldsymbol{\omega})\beta_c(\boldsymbol{\omega}) + b_g(\boldsymbol{\xi},\boldsymbol{\omega})\beta_{cg}(\boldsymbol{\omega})\right) + \theta_c(\boldsymbol{\xi},\boldsymbol{\omega},\boldsymbol{u}) = 0, \quad c = 1,\dots,n_c. \tag{3.29}$$

Here

$$b_g(\boldsymbol{\xi},\boldsymbol{\omega}) = \Delta t \Gamma_{ij}(-g(D_j - D_i)), \tag{3.30}$$

$$\beta_{cg}(\boldsymbol{\omega}) = \sum_{p=1}^{n_p} x_{cp,ij}\rho_{p,ij}\lambda\rho^m_{p,ij},$$

$$\text{where} \quad x_{cp,ij}\rho_{p,ij}\lambda_{p,ij}\rho^m_{p,ij} = \begin{cases} x_{cp,i}\rho_{p,i}\lambda_{p,i}\rho^m_{p,i} & \text{if } D_i < D_j \\ x_{cp,j}\rho_{p,j}\lambda_{p,j}\rho^m_{p,j} & \text{otherwise.} \end{cases} \tag{3.31}$$

This approximation can be interpreted as a compositional version of HU, where independently of flow direction, the gravity term of the lighter phase is always pointing up while for the heavier phase it is pointing down.

Similarly, the energy balance equation (Equation 3.5) becomes:

$$a_e(\boldsymbol{\xi})(\alpha_e(\boldsymbol{\omega}) - \alpha_e(\boldsymbol{\omega}_n)) + \sum_{j \in adj(i)} \left(b_e(\boldsymbol{\xi},\boldsymbol{\omega})\beta_e(\boldsymbol{\omega}) + b_g(\boldsymbol{\xi},\boldsymbol{\omega})\beta_{eg}(\boldsymbol{\omega})\right) + \sum_{j \in adj(i)} c_e(\boldsymbol{\xi},\boldsymbol{\omega})\gamma_e(\boldsymbol{\omega}) + \theta_e(\boldsymbol{\xi},\boldsymbol{\omega},\boldsymbol{u}) = 0, \tag{3.32}$$

where

$$\beta_{eg}(\boldsymbol{\omega}) = \sum_{p=1}^{n_p} h_{p,ij}\rho_{p,ij}\lambda\rho^m_{p,ij},$$

$$\text{where} \quad h_{p,ij}\rho_{p,ij}\lambda_{p,ij}\rho^m_{p,ij} = \begin{cases} h_{p,i}\rho_{p,i}\lambda_{p,i}\rho^m_{p,i} & \text{if } D_i < D_j \\ h_{p,j}\rho_{p,j}\lambda_{p,j}\rho^m_{p,j} & \text{otherwise.} \end{cases} \tag{3.33}$$

This approach matches the previous one in terms of the number of required state operators.

3.2.4. One-Dimensional Dead-Oil Model with Gravity Segregation

We started with a simple one-dimensional domain for a vertical segregation model with buoyancy-driven flow. It consist of 5 grid cells $10 \times 10 \times 10$ m each, extending over a depth of 50 m overall. The rock permeability is equal to 100 mD and the porosity is equal to 0.2. We started with a dead-oil kernel, filling the top three grid cells with water (with a constant density $\rho_w^m = 1000$ kg/m^3) while the bottom two cells were filled with oil (with a constant density $\rho_o^m = 800$ kg/m^3). The reservoir was initialized with P_0=100 bar. We ran the model for 10,000 days until the system reached an equilibrium. The dynamic distribution of fluids is shown in Figure 3.15. It can be seen that the heavier water phase, placed on top, has exchanged position with the oil phase by the end of the simulation time.

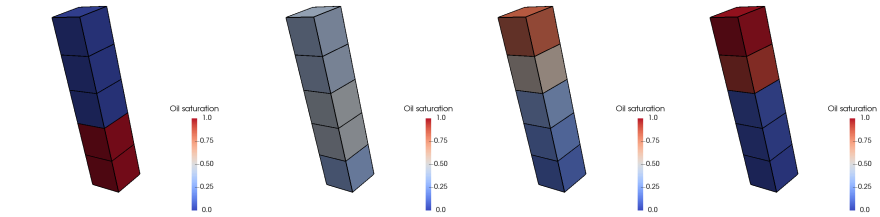
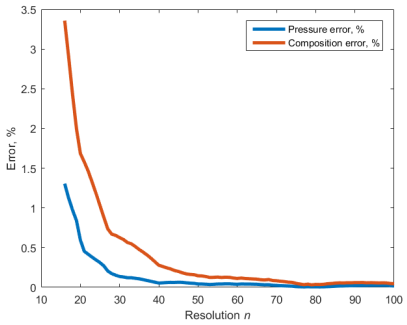


Figure 3.15: Dead-oil gravity segregation

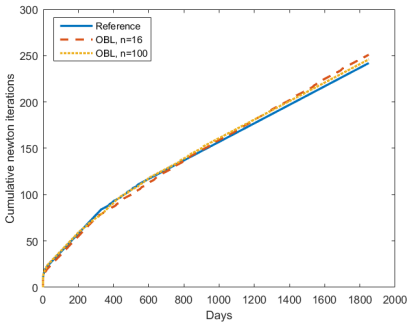
In Figure 3.16(a), the error between the reference and OBL simulations is shown depending on parameterization resolution. It is clear that the error is converging to zero at high OBL resolution. Next, Figure 3.16(b) demonstrates the dynamic cumulative number of nonlinear iterations versus time for two OBL resolutions of 16 and 100 points. The plot covers only the first 1800 days since later the system is close to equilibrium and requires a single Newton iteration to converge for all simulations. The lower resolution demonstrates better convergence at the beginning of the simulation requiring more iterations near its end. The finer resolution model behaves similarly to the model with the reference physics.

Table 3.6: Results of dead-oil gravity segregation with PPU

Resolution	Iters.	E_p, %	E_z, %	Space,%
Std.	1054	-	-	-
64	1076	0.05	0.11	4.350
32	1276	1.02	6.40	7.599
16	1119	0.92	2.03	13.281
8	-	-		-



(a) Final solution error of OBL compared to reference model



(b) Comparison of nonlinear solver behaviour between reference and OBL simulations

Figure 3.16: OBL behaviour for dead-oil kernel

Table 3.7: Results of dead-oil gravity segregation with CPU

Resolution	Iters.	E_p, %	E_z, %	Space,%
Std.	1054	-	-	-
64	1076	0.05	0.11	4.091
32	1276	1.02	6.40	7.248
16	1119	0.92	2.03	12.283
8	-	-		-

3.2.5. One-Dimensional Compositional Model with Gravity Segregation

Next, we ran the gravity segregation test with 4-component isothermal compositional model used before. The initial oil with 1% of CO_2, 11% of C_1, 38% of NC_4, and 50% of C_{10} was placed in the top three grid cells, while mixture of gas with 80% of CO_2 and 20% of C_1 was placed in the two bottom cells. The initial reservoir pressure was set to P_0=120 bar and temperature T_0=350 K forming a pure liquid phase in the top and a pure gas phase in the bottom. The dynamic distribution of phases is shown in Figure 3.17. Unlike in the dead-oil kernel, the gravity segregation here is combined with extensive mass exchange between liquid and vapor phases which drastically changes the composition of fluids.

In Figure 3.18(a), one can see the corresponding error in pressure and composition (maximum over all components) for the final solutions of OBL model, compared to reference physics, depending on the resolution of the OBL approach. It is clear that the error is converging slower than in the dead-oil kernel due to a more complicated dynamics of the process. At the same time, the nonlinear performance (Figure 3.18(b)) behaves more predictably, being better for the lower resolution and slightly worse for the high resolution in the OBL simulation. Again, the plot covers only the most intense first 140 days, as later the nonlinear behaviour of all simulations is equal.

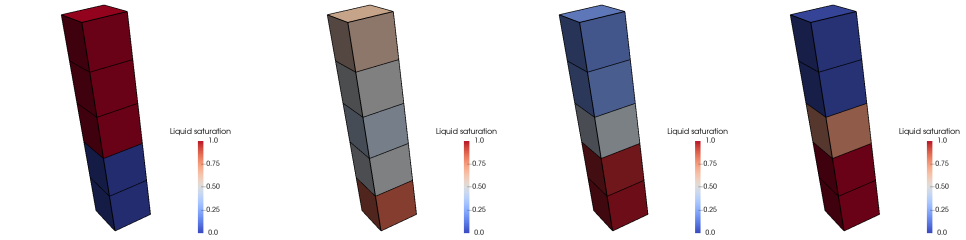
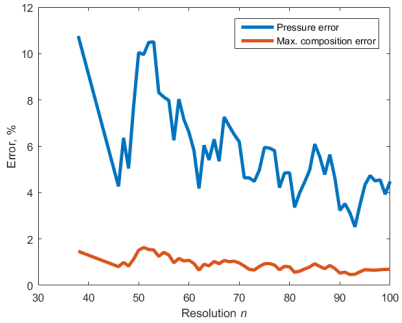
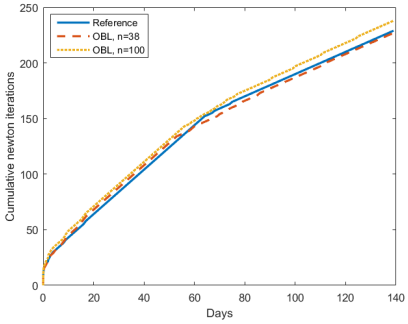


Figure 3.17: Compositional gravity segregation



(a) Final solution error of OBL compared to reference model



(b) Comparison of nonlinear solver behaviour between reference and OBL simulations

Figure 3.18: OBL behaviour for compositional kernel

Table 3.8: Results of compositional gravity segregation with PPU

Resolution	Iters.	E_p, %	E_{z1}, %	E_{z2}, %	Space,%
Std.	298	-	-	-	-
64	424	45.95	1.50	0.83	0.399
32	445	208.96	2.92	1.68	1.782
16	303	441.45	6.66	1.99	5.382
8	-	-	-		-

3.2.6. Brugge Field Model

To demonstrate the applicability of the OBL approach for a full field three-dimensional model, we employ the Brugge field, which is often used as an optimization benchmark for reservoir simulation study [68]. This model is based on realistic reservoir structures and properties shown in Figure 3.19. The simulation time spans 10 years with BHP controls changing every 3 months for both injection and production wells. For reservoir parameters and well controls, we used the base case realization described in [69].

Table 3.9: Results of compositional gravity segregation with IU

Resolution	Iters.	E_p, %	E_{z1}, %	E_{z2}, %	Space,%
Std.	298	-	-	-	-
64	130	385.23	20.22	28.92	0.202
32	129	473.79	20.79	29.17	0.998
16	124	379.98	19.40	29.70	3.601
8	117	502.72	17.22	36.50	11.963

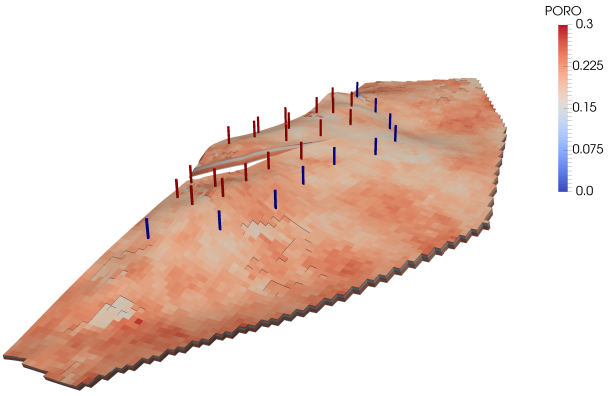


Figure 3.19: Porosity distribution of the Brugge field

In Figure 3.20, we compare total well rates for both production and injection wells for the reference dead-oil kernel and the OBL implementation (with the resolution $n = 64$) with and without gravity. It can be seen, that the buoyant forces play an important role in this model, and only the simulation using OBL with buoyancy successfully recovers reference well rates.

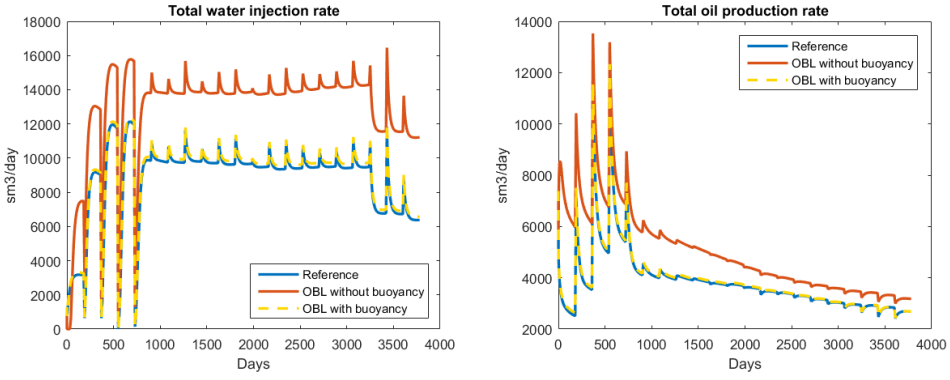
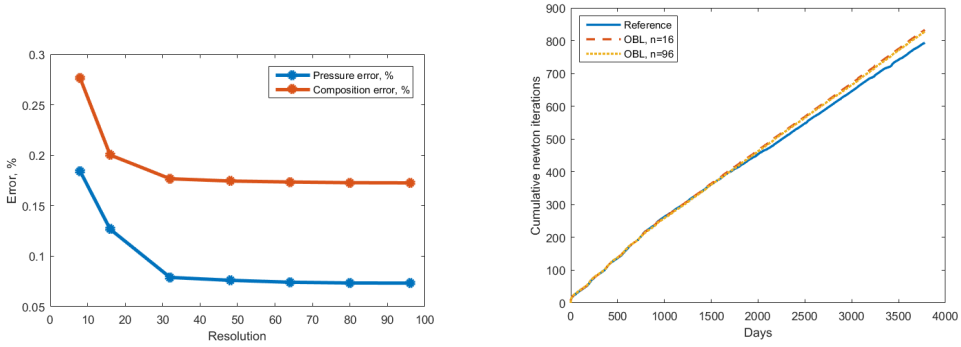


Figure 3.20: Well rates comparison for Brugge field

The corresponding errors and nonlinear behaviour are shown in Figure 3.21. It is clear that the overall error is quite insignificant even for the coarsest resolution ($n = 8$). The error stabilizes at $n = 32$ and remains so up to the finest resolution $n = 96$. The nonlinear behaviour is almost equivalent between the reference solution and OBL at different resolutions in the first half of the simulation. After that OBL-based simulations require slightly more nonlinear iterations to converge. The CPU cost of simulation with the reference physics is comparable with the OBL approach without buoyancy (580 vs. 582 seconds respectively, while linearization takes around 260 seconds in both). The corresponding number of operators in the OBL approach with gravity grows significantly (from 2 to 6 for dead-oil kernel) increasing the overhead due to ADETL. That can explain why the simulation time for the OBL approach with buoyancy is larger. Table 3.10 and Table 3.11 demonstrate the results for OBL approach with PPU and IU accordingly. The amount of newton iterations in case of PPU is slightly higher than for reference physics, while in case of IU is even higher that for PPU. At the same time, the error remains very low for both approaches. The simulation time for OBL in this case is 734-790 seconds with linearization cost of around 470 seconds. As was already mentioned above, the proper implementation of the OBL approach can speed up Jacobian assembly by a factor of 14x. Moreover, the migration and optimization of algorithms for emerging architectures (e.g., GPU) improves the linearization performance by another order of magnitude [28].



Final solution error Nonlinear solver behaviour

Figure 3.21: OBL error behaviour for Brugge field model

Table 3.10: Results of Brugge model with PPU

Resolution	Iters.	E_p, %	E_z, %	Space,%	CPU, sec
Std.	817	-	-	-	579.7
64	829	0.13	0.01	20.521	786.6
32	856	0.13	0.01	23.757	799.9
16	824	0.13	0.01	28.693	766.2
8	826	0.16	0.02	44.318	769.1

Table 3.11: Results of Brugge model with IU

Resolution	Iters.	E_p, %	E_z, %	Space,%	CPU, sec
Std.	817	-	-	-	579.7
64	960	0.37	0.13	20.513	782.1
32	956	0.37	0.13	23.806	773.2
16	980	0.37	0.13	29.601	791.2
8	893	0.40	0.14	43.403	734.6

4

Prototype Implementations of Operator-Based Linearization

4.1. Extension of Existing Simulation Framework

Automatic Differentiation General Purpose Research Simulator (AD-GPRS) is a flexible and efficient reservoir simulation research laboratory with extensible modelling and solution capabilities. AD-GPRS is one of a few simulators based on automatic differentiation (AD) framework (see also [26]). It has a modular object-oriented design, while all of the code is written in standard C++. This design is convenient for researchers to extend the simulator by incorporating new physics, introducing complex processes, or adding new formulations and solution algorithms. It was a natural choice to implement OBL as an alternative nonlinear formulation in AD-GPRS. To better understand the specifics, the structure of the AD-GPRS framework will be briefly observed first.

4.1.1. General Structure of AD-GPRS

The system model shows the basic classes and their relations, and it is very helpful for understanding the structure of AD-GPRS. Due to the complexity of AD-GPRS, the system model can be addressed level by level using multiple figures. The description will be concentrated on the most necessary aspects for the understanding of OBL implementation.

Figure 4.1 shows the overall structure of the entire simulator. *SimMaster* is a manager of all simulation-related objects. When *SimMaster* is created, in turn it instantiates a specific *NonlinearFormulation* object. After that, common objects including *Reservoir*, *Facilities*, *NonlinearSolver* are constructed. The data members of *SimMaster* also include the global variable set (*adX*) that stores all the simulation variables with both values and gradients, as well as the global backup set (*adX_n*)

Parts of this chapter have been published in the proceedings of SPE Reservoir Simulation Conference (2017)[28]

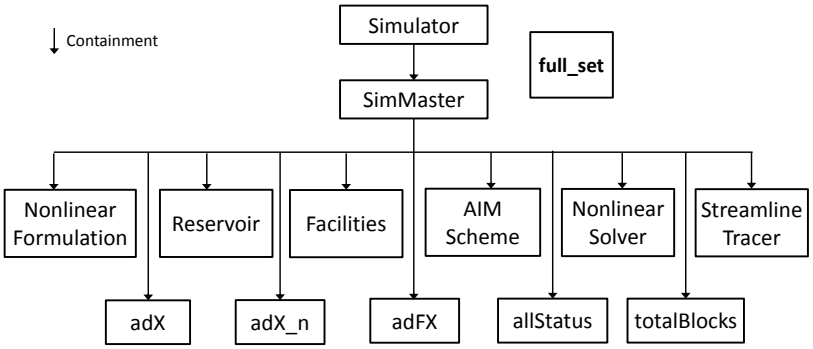


Figure 4.1: Overall structure of the entire simulator

that saves a copy of values of all variables. Those can include, in particular, pressure, temperature, phase saturations, component molar fractions, phase mobilities, and so on. Special status in a variable formulation activates a part of the corresponding variables to be independent and leaves the rest of the variables to be dependent, without changing their values.

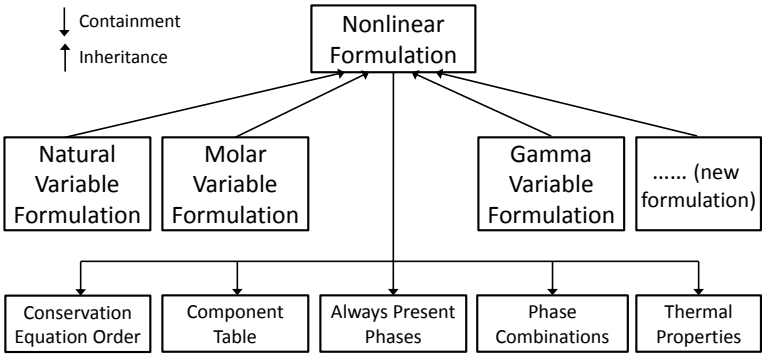


Figure 4.2: Structure of the NonlinearFormulation

Now, the *NonlinearFormulation* class will be discussed in detail. Figure 4.2 shows its structure. This is the abstract base class of various formulations, such as *NaturalVariableFormulation*, *MolarVariableFormulation*, and so on. By constructing new inherited classes of *NonlinearFormulation* (or one of its derived classes) with the same interfaces but possibly different realizations, we are able to introduce new formulations.

NonlinearFormulation contains a variety of formulation-related virtual member functions. Some functions are called in the initialization stage to specify the phase, component and variable structure of a formulation. This includes, among many others, *fluidPropertiesCalculation*, which computes all fluid properties for a given control volume, and *computeMassFluxTerm*, which performs computations related

to a specific interface between two control volumes.

It is essential to understand that nearly all computations at the nonlinear level in AD-GPRS are performed using data structures from Automatic Differentiation Expression Template Library (ADETL). The most important ones are aforementioned *adX* and *adX_n*, but there are many other locally used data storages. Any operation with ADETL types additionally involves augmented gradient computations, which are happening behind the scenes according to the selected primary variable set. Each nonlinear formulation defines such a specific set. Then, in order to construct a Jacobian matrix, one should only care of residual equations, and all gradients are computed automatically.

4.1.2. Additional Nonlinear Formulation with OBL

It was a natural choice to implement OBL in the AD-GPRS framework as an additional nonlinear formulation. *MolarVariableFormulation* was chosen to be modified, since it provided a primary variable set which does not require variable switching (as, for instance, in *NaturalVariableFormulation* where the number of phases is changing). In order to apply OBL to computations of accumulation and flux terms of a residual equation, *fluidPropertiesCalculation* and *computeMassFluxTerm* functions of the new nonlinear formulation were modified accordingly.

All ADETL-based computations related to the evaluation of state operators were replaced by corresponding multilinear interpolation procedures. Nevertheless, all gradients obtained through interpolation had to be injected back into ADETL structure *adX* for further processing in Jacobian assembly. This was implemented in additional function *assembleADScalar*. Of course, these manipulations of gradient data can be seen as unnecessary overhead computations, which can be avoided in a stand-alone simulator designed on top of the OBL from the very beginning.

All state operators were interpolated in an adaptive manner, which is thoroughly described in Section 2.6. It allowed to run OBL-based simulations with up to 6 degrees of freedom (see Subsection 2.7.2). Each operator was stored and interpolated separately. On one hand, that allowed applying a different parameterization resolution for each of them. On the other hand, it was not found really beneficial, and, for consistency, in the majority of simulations an identical parameterization resolution was applied to all operators. In this case, the search of values of their supporting points and interpolation was also performed independently, despite the fact that a large degree of those computations was redundant.

Even though the implementation of OBL within AD-GPRS had the aforementioned issues, the performance benefits of the approach were confirmed (see Chapter 2, Chapter 3). To estimate the full performance advantage provided by OBL, a stand-alone simulation capability with combined operator storage was needed. All implementation stages are described in the sections below.

4.2. One-Dimensional Simulator in MATLAB

As a first step, a stand-alone simulator was created in Matlab. Its goal was to model conservation equations in operator form, while operator values were generated

by AD-GPRS implementation and provided to Matlab externally via text files. To simplify the analysis, the model was limited to a 1D reservoir with Cauchy boundary conditions on the left and right sides. This made the spatial discretization simpler, yielding to the following equation in vector form (the length of vector corresponds to the number of components n_c) for the block i:

$$\begin{aligned} \boldsymbol{r}_i(\boldsymbol{\omega}_{i-1}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_{i+1}, \boldsymbol{\omega}_i^n) &= \left(\boldsymbol{\alpha}(\boldsymbol{\omega}_i) - \boldsymbol{\alpha}(\boldsymbol{\omega}_i^n)\right) a_i \\ &- \boldsymbol{\beta}(\boldsymbol{\omega}_i) b_{i+}(\boldsymbol{\omega}_i, \boldsymbol{\omega}_{i+1}) \\ &+ \boldsymbol{\beta}(\boldsymbol{\omega}_{i-1}) b_{i-}(\boldsymbol{\omega}_i, \boldsymbol{\omega}_{i-1}) = 0, \end{aligned} \tag{4.1}$$

where

$$a_i = \phi_{0_i} V_i, \tag{4.2}$$
$$b_{i+}(\boldsymbol{\omega}_i, \boldsymbol{\omega}_{i+1}) = \Delta t T_{i,i+1}(p_{i+1} - p_i), \tag{4.3}$$
$$b_{i-}(\boldsymbol{\omega}_i, \boldsymbol{\omega}_{i-1}) = \Delta t T_{i-1,i}(p_i - p_{i-1}). \tag{4.4}$$

In the case of total velocity formulation

$$b_{i+}(\boldsymbol{\omega}_i, \boldsymbol{\omega}_{i+1}) = \Delta t T_{i,i+1}(p_{i+1} - p_i)\Lambda(\boldsymbol{\omega}_i) \tag{4.5}$$
$$b_{i-}(\boldsymbol{\omega}_i, \boldsymbol{\omega}_{i-1}) = \Delta t T_{i-1,i}(p_i - p_{i-1})\Lambda(\boldsymbol{\omega}_{i-1}). \tag{4.6}$$

Next, we assume a homogeneous reservoir with V, ϕ_0 and T constants. Equation 4.1 written for internal reservoir block can be simplified:

$$\boldsymbol{r_i} = \left(\boldsymbol{\alpha}_i - \boldsymbol{\alpha}_i^n\right) + \gamma\left(\boldsymbol{\beta}_i b_{i+} + \boldsymbol{\beta}_{i-1} b_{i-}\right), \tag{4.7}$$
$$\boldsymbol{\alpha}_i = \boldsymbol{\alpha}(\boldsymbol{\omega}_i), \tag{4.8}$$
$$\boldsymbol{\beta}_i = \boldsymbol{\beta}(\boldsymbol{\omega}_i), \tag{4.9}$$
$$\gamma = \Delta t \frac{T^{ab}}{\phi_0 V}, \tag{4.10}$$
$$b_{i+} = p_i - p_{i+1}, \tag{4.11}$$
$$b_{i-} = p_i - p_{i-1}. \tag{4.12}$$

In the case of total velocity formulation

$$b_{i+} = (p_i - p_{i+1})\Lambda(\boldsymbol{\omega}_i) \tag{4.13}$$
$$b_{i-} = (p_i - p_{i-1})\Lambda(\boldsymbol{\omega}_{i-1}). \tag{4.14}$$

Now, the internal i-th Jacobian block row can be written as:

$$\mathbf{J_i} = \begin{bmatrix} \gamma \boldsymbol{B}_{i-1} b_{i-} + \gamma \boldsymbol{\beta}_{i-1} \times \boldsymbol{b'_{i-,i-1}} \\ \boldsymbol{A}_i + \gamma\left(\boldsymbol{B}_i b_{i+} + \boldsymbol{\beta}_i \times \boldsymbol{b'_{i+,i}} + \boldsymbol{\beta}_{i-1} \times \boldsymbol{b'_{i-,i}}\right) \\ \gamma \boldsymbol{\beta}_i \times \boldsymbol{b'_{i+,i+1}} \end{bmatrix}^T \tag{4.15}$$

where

$$\boldsymbol{A}_i = \left[\frac{\partial \boldsymbol{\alpha}_i}{\partial \boldsymbol{\omega}_i}\right] = \left[\frac{\partial \alpha_c}{\partial p_i} \quad \frac{\partial \alpha_c}{\partial z_{i,1}} \quad \cdots \quad \frac{\partial \alpha_c}{\partial z_{i,n_c-1}}\right], c = 1, \ldots, n_c, \tag{4.16}$$

$$\boldsymbol{B}_i = \left[\frac{\partial \boldsymbol{\beta}_i}{\partial \boldsymbol{\omega}_i}\right] = \left[\frac{\partial \beta_c}{\partial p_i} \quad \frac{\partial \beta_c}{\partial z_{i,1}} \quad \cdots \quad \frac{\partial \beta_c}{\partial z_{i,n_c-1}}\right], c = 1, \ldots, n_c, \tag{4.17}$$

$$\boldsymbol{b}'_{i-,i-1} = \left[\frac{\partial b_{i-}}{\partial \boldsymbol{\omega}_{i-1}}\right]^T = \begin{cases} \left[-1 \underbrace{0 \ldots 0}_{n_c-1}\right]^T, & \text{conventional} \\ \left[(p_i - p_{i-1})\dfrac{\partial \lambda_{i-1}}{\partial \boldsymbol{\omega}_{i-1}} + \lambda_{i-1}\left[-1 \underbrace{0 \ldots 0}_{n_c-1}\right]\right]^T, & \text{total velocity} \end{cases} \tag{4.18}$$

$$\boldsymbol{b}'_{i-,i} = \left[\frac{\partial b_{i-}}{\partial \boldsymbol{\omega}_i}\right]^T = \begin{cases} \left[1 \underbrace{0 \ldots 0}_{n_c-1}\right]^T, & \text{conventional} \\ \lambda_{i-1}\left[1 \underbrace{0 \ldots 0}_{n_c-1}\right]\Big]^T, & \text{total velocity} \end{cases}, \tag{4.19}$$

$$\boldsymbol{b}'_{i+,i} = \left[\frac{\partial b_{i+}}{\partial \boldsymbol{\omega}_i}\right]^T = \left[\frac{\partial(-b_{(i+1)-})}{\partial \boldsymbol{\omega}_{(i+1)-1}}\right]^T = -\boldsymbol{b}'_{(i+1)-,(i+1)-1}, \tag{4.20}$$

$$\boldsymbol{b}'_{i+,i+1} = \left[\frac{\partial b_{i+}}{\partial \boldsymbol{\omega}_{(i+1)}}\right]^T = \left[\frac{\partial(-b_{(i+1)-})}{\partial \boldsymbol{\omega}_{i+1}}\right]^T = -\boldsymbol{b}'_{(i+1)-,(i+1)}. \tag{4.21}$$

To maximize the simulation performance, functions performing operator interpolation and Jacobian assembly were vectorized. In addition, the Jacobian was composed as a tridiagonal block sparse matrix directly of three diagonals (i.e., bands). That helped to speed up the assembly itself and, more importantly, to ensure the usage of efficient sparse direct solver.

4.3. Stand-Alone Simulator with OBL for CPU and GPU Architectures

In order to evaluate the genuine performance of the OBL approach, we decided to develop a new C++ prototype of the compositional simulator, without using an Automatic Differentiation (AD) library. Due to the limited time allotted for the development of both CPU and GPU versions of the simulation code, we restricted the number of components to $n_c = 2$. The target implementation was designed to closely follow the implementation in AD-GPRS to allow a proper performance comparison between them. To ensure that, we aligned the initialization and simulation loop in both simulators.

The initialization stage consisted of several parts. First, the mesh was initialized in both AD-GPRS and prototype simulators from a connection list augmented

by transmissibility, volume, and porosity values. This allowed to gain simplicity and unstructured grids support [70]. Second, the initial state of the reservoir was defined directly from the values of state variables, dumped from AD-GPRS after initialization. Finally, the look-up tables required by the OBL approach were processed similarly: pre-calculated in AD-GPRS, stored, and then loaded in the simulator prototype. Thereby, both AD-GPRS and C++ prototypes were initiated identically.

The linearization stage with the linear and nonlinear solvers comprise the main simulation loop. The OBL was implemented equivalently in all simulation codes with the only difference in the Jacobian storage selection. While the AD-GPRS framework used AD-based specific storage, the new prototype employed a standard Block Compressed Sparse Row (BCSR) matrix format. We selected the GMRES linear solver preconditioned by ILU(0) [34], because this combination is well known for its good convergence behaviour [71], and its implementation is widely available. AD-GPRS already had this setup implemented [72].

In the new prototype, we used our own implementation of the CPU linear solver, while for the GPU version, the linear solver library from [73] was used. The basic Newton-Raphson nonlinear solver was used in all modelling approaches. The nonlinear convergence criterion was implemented based on the L2-norm of a residual for all simulations. Finally, we achieved an identical behaviour of the iteration process and the time stepping across all OBL-based modelling approaches, delivering simulation results that closely match the conventional AD-GPRS results as long as the parameterization resolution n is high enough.

The new simulator prototype is conceptually close to the approach described by [74], having the same initialization driver for both GPU and CPU versions, which are developed as interchangeable parts. The GPU version first loads the required initial data to GPU memory and then performs all major computations on the device. The CPU is only used to control the main execution logic and to launch the GPU kernels. All kernels were implemented on a thread-per-cell basis, avoiding any communication between threads. Restricted by only 2 components, we achieved a streaming multiprocessor occupancy of 100% for the bi-linear interpolation kernel and 52% for the Jacobian assembly kernel. Gaining the performance by using the BCSR storage and native data types, we did not specifically tune neither our CPU nor our GPU kernels. Carefully applied vectorization, memory padding and alignment, and mixed precision will be able to improve the computational performance further.

4.4. Numerical Resuls and Performance Comparison

We compared the computational performance of four modelling approaches:

1. the default overall molar formulation in AD-GPRS [10],
2. the OBL-based molar formulation in AD-GPRS [75],
3. the prototype of the OBL-based compositional simulator on CPU, and
4. the prototype of the OBL-based compositional simulator on GPU.

The first three approaches, which use only the CPU, were executed in a serial mode on a single server with two Intel Xeon E5-2620 v2 processors clocked at 2.1 GHz. We performed all executions on CPU in serial mode to escape considerations about the efficiency of shared-memory implementation. The fourth approach was run on NVIDIA Tesla K40m at 875 MHz. All OBL-based approaches used a constant resolution of n =64.

4.4.1. Benchmark Model

The SPE10 test case [76], initially created to compare upscaling techniques, now is probably the most commonly used model to benchmark the performance of reservoir simulators. Due to its highly heterogeneous permeability distribution, achieving 10 orders of magnitude, and considerable size of 1.1 million cells, this model is quite challenging for both linear and nonlinear solvers. Here, we selected the SPE10 test case as a benchmark to compare Jacobian construction times. We chose the simplest linear and nonlinear solution strategies, described above, to perform the consistent comparison of all four implementations. Wells were modelled as simple source/sink terms controlled by pressure. We introduced two wells, an injector and a producer, at the opposite corners of the model, with perforations in all vertical layers. The original porosity of the SPE10 model was adjusted to a minimum limit of $\phi_{min} = 0.001$ to avoid the presence of nonactive cells in the model.

In addition to the standard SPE10 model, we used a homogeneous model with the same geometrical characteristics, but constant porosity $\phi = 0.2$, horizontal $K_h = 10$ mDarcy, and vertical $K_v = 0.4$ mDarcy permeabilities. For each of the models, we performed a waterflooding simulation described by the original SPE10 dead-oil properties and gas injection simulation based on EoS properties.

Conducting the benchmarks, we were limited by both small simulation timesteps, caused by the choice of simple linear and nonlinear solvers, and a short time-frame of exclusive access to the server. Due to these restrictions, we ran all simulations for a limited run-time. For a more involved simulation and a detailed analysis of the impact of OBL on solution see [75].

4.4.2. Waterflooding in Heterogeneous Reservoir

We employed a standard dead-oil model, where oil and water components exist only in their corresponding phases and do not mix. Most of the properties are described as table-based correlations. The water injection well operated with a pressure control at $p_i = 400$ bar, and the production well operated at $p_p = 100$ bar. The initial pressure distribution was set at $p_0 = 200$ bar and the initial water saturation was set at $S_w = 0$. All simulations were run for 0.1 days with a limited timestep to avoid convergence issues in both the linear and the nonlinear solvers.

Table 4.1 demonstrates the performance results for the simulators, listed in the first column. Due to identical initial conditions, tolerances and convergence conditions for both the linear and the nonlinear solvers, the number of timesteps, nonlinear iterations, and linear iterations, presented in the second, the third, and the fourth columns respectively, match for all OBL-based simulators and are close to those for the standard AD-GPRS.

Table 4.1: Performance results for the dead-oil simulation in the heterogeneous reservoir

Simulator	Ts	Newt. it.	Lin. it.	Jac., s	Single Jac., s	Lin. solv., s	Total, s
AD-GPRS	16	51	1891	592.892	8.849	976.459	1569.351
AD-GPRS + OBL	16	52	1887	505.644	7.436	964.846	1470.490
Prototype (CPU)	16	52	1887	35.400	0.521	715.000	751.000
Prototype (GPU)	16	52	1887	2.330	0.034	118.540	122.900

The difference in Jacobian construction time, which is shown in the fifth column, between the standard linearization and OBL within the AD-GPRS framework shows the advantage of the approach. Furthermore, the same approach, built in the prototype simulator, speeds up the Jacobian construction by an order of magnitude on the CPU platform, and by another order of magnitude on the GPU platform, resulting in a 257x speedup over the Jacobian assembly performed in the default AD-GPRS implementation. There are several reasons explaining these results. First, all simulation runs on the CPU platform were performed on a single processor core, as we did not want to rely on the efficiency of a multithreaded parallel implementation. Second, as was mentioned before, AD-GPRS is based on the extensive use of the AD technique [24] which introduces a certain overhead caused by augmented algebra computations, storage selection, and compiler optimization.

The next column in the table represents an average time spent on a single Jacobian assembly. It was estimated by dividing the Jacobian construction time by the sum of nonlinear iterations and timesteps, which represents the number of Jacobian evaluations. We calculated this value to compare the Jacobian construction performance regardless of the number of nonlinear iterations or timesteps made. According to [74], their GPU implementation of Jacobian assembly takes 8 s in SPE10 simulation with 68 timesteps and 418 Newton iterations on the similar NVIDIA Tesla K40, which gives an average of 0.016 s per single Jacobian assembly, assuming there were no time step cuts. In this case, our first prototype implementation of a general purpose simulator is only 2x slower.

Linear solver execution time is shown in the 7-th column. The linear solver, used in the CPU version of the prototype, performs better than that in the default implementation in AD-GPRS. The GPU-based linear solver has a convincing advantage over the fastest CPU solver, performing 6x times faster. The total simulation times, excluding initialization, are shown in the last column of the table. AD-GPRS-based simulations have a little difference due to the different linearization approaches. The CPU version of the prototype is 2x faster than the standard AD-GPRS owing to a 16x faster Jacobian assembly and a more efficient linear solver implementation. Finally, the GPU version of the prototype is only 12x times faster than the reference simulation, even though the Jacobian assembly now takes only 2.3 s. The reason is in a lower scalability of the linear solver, probably caused by employing substantially sequential ILU(0) as a preconditioner in this simulation. However, an additional speedup can be obtained by use of multi-GPU systems.

4.4.3. Dead-oil Waterflooding in Homogeneous Reservoir

Here, we simplify the initial model by setting porosity and transmissibility values to constant. This allowed us to set a larger timestep and run the simulation for 10 days. As Table 4.2 shows, the numbers of timesteps and nonlinear iterations still match across all simulations, while the number of linear solver iterations is slightly higher for the reference approach than that for OBL-based approaches. The cost of a single Jacobian construction remains practically unchanged for all simulators.

Table 4.2: Performance results for the dead-oil simulation in the homogeneous reservoir

Simulator	Ts	Newt. it.	Lin. it.	Jac., s	Single Jac., s	Lin. solv., s	Total, s
AD-GPRS	10	37	1420	411.776	8.761	677.327	1089.103
AD-GPRS + OBL	10	37	1395	354.758	7.548	670.299	1025.057
Prototype (CPU)	10	37	1395	21.380	0.455	474.150	496.000
Prototype (GPU)	10	37	1395	1.630	0.035	84.100	87.200

4.4.4. Gas Injection in Heterogeneous Reservoir

Here, we present the results of gas injection into oil composed of $\{CO_2, C_{10}\}$ to demonstrate the applicability of the developed simulators to compositional problems. The gas was injected at a pressure of $P_i = 100$ bar, while the oil was produced at $P_p = 60$ bar. The initial oil contained 31% carbon dioxide and 69% decane, while the injected mixture was composed of 79% of CO_2 and 21% of C_{10}. The reservoir was initialized uniformly at a pressure of $P_0 = 80$ bar and a temperature of $T_0 = 372$ K. As before, the simulations were performed with a limited timestep and ran for 0.1 days.

Table 4.3: Performance results for the compositional simulation in the heterogeneous reservoir

Simulator	Ts	Newt. it.	Lin. it.	Jac., s	Single Jac., s	Lin. solv., s	Total, s
AD-GPRS	16	19	1403	304.540	8.701	872.068	1176.608
AD-GPRS + OBL	16	19	1403	228.267	6.522	870.073	1098.340
Prototype (CPU)	16	19	1403	17.310	0.495	697.580	715.200
Prototype (GPU)	16	19	1403	1.110	0.032	97.720	100.200

Table 4.3 shows that the cost of a single Jacobian evaluation remained close to the dead-oil case for all approaches in compositional simulation. It can be explained by a small run-time period used in our simulations. The real impact of phase behaviour computations to the linearization stage and advantages provided by the OBL approach can be found in [75].

4.4.5. Gas Injection in Homogeneous Reservoir

The results of $CO_2 + C_{10}$ injection in the homogeneous reservoir are presented in Table 4.4. The observations and conclusions in this case are similar to the previous results. Notice that the time of a single Jacobian assembly for the OBL approach

implemented in the prototype simulator is almost independent of the complexity of the physics across all test cases.

Table 4.4: Performance results for the compositional simulation in the homogeneous reservoir

Simulator	Ts	Newt. it.	Lin. it.	Jac., s	Single Jac., s	Lin. solv., s	Total, s
AD-GPRS	10	20	1497	238.613	7.954	885.941	1124.554
AD-GPRS + OBL	10	20	1497	178.292	5.943	886.649	1064.941
Prototype (CPU)	10	20	1497	13.500	0.450	705.890	719.700
Prototype (GPU)	10	20	1497	0.980	0.033	100.030	102.200

4.4.6. Gas Injection in One-Dimensional Homogeneous Reservoir

In order to add the MATLAB prototype into the comparison, we also modeled gas injection into oil composed of $\{CO_2, C_4, C_{10}\}$ in a one-dimensional reservoir with 1000 grid blocks. The gas was injected at a pressure of $P_i = 100$ bar. The initial oil contained 1% carbon dioxide, 65% buthane and 34% decane, while the injected mixture was composed of 79% CO_2, 20% C_4, and 1% C_{10}. The reservoir was initialized uniformly at a pressure of $P_0 = 60$ bar and a temperature of $T_0 = 353$ K. The simulations were performed with a limited timestep of 1 day and ran for 200 days. For OBL simulations, the resolution of 32 was used.

Table 4.5: Performance results for the compositional simulation in the one-dimensional heterogeneous reservoir

Simulator	Ts	Newt. it.	Lin. it.	Jac., s	Single Jac., s	Lin. solv., s	Total, s
AD-GPRS	203	635	635	2.6	0.0031	0.6	4.5
AD-GPRS + OBL	203	580	580	2.1	0.0026	0.5	3.4
Prototype (CPU)	203	580	580	0.2	0.0003	0.17	0.6
Prototype (MATLAB)	203	580	-	7.8	0.009	1	9.3

Table 4.5 demonstrates that the performance difference between the C++ and MATLAB-based simulators is around an order of magnitude. The main contribution is made by linearization, which is almost 40x slower in MATLAB despite the vectorized implementation. The linear solver is only 5x behind the C++ version (it should be noted that the MATLAB version employed a direct linear solver, whereas the CPU prototype uses an iterative GMRES + BILU(0) scheme). The difference in linearization performance between the AD-GPRS and CPU prototypes did not change for the models of reduced size and remained around an order of magnitude.

5

Delft Advanced Research Terra Simulator (DARTS)

After several prototype implementations of OBL were developed, tested, and validated, a certain level of maturity was reached. Existing code has been significantly refactored and extended, exploiting the advantages of the approach to its limits. The main goal for the new Delft Advanced Research Terra Simulator (DARTS, [80]) was to preserve its simplicity and computational efficiency, but make it modular and extendable as much as possible. At the same time, general purpose reservoir simulation capabilities were required, with the potential to extend them further.

5.1. Combined Implementation in Python/C++

In order to reach these goals, it was decided to complement two technology stacks in the simulator: C++ and Python. The former is the most popular compiled programming language, inherently providing the required computational efficiency and additionally allowing the usage of coarse-grained and fine-grained parallelism through OpenMP and CUDA language extensions. C++ was used in DARTS for implementation of critical for performance parts, such as linearization, interpolation, and solution of a linear system. Python is one of the most popular interpreted languages, providing flexibility and simplicity of development. Moreover, it integrates with C++ with minimal overhead, allowing even to inherit existing C++ interface classes and therefore extend original, already compiled functionality with a custom script.

Python was used in DARTS mainly for data pre- and post-processing, where performance fades into the background, yielding functionality in importance. For a constantly developing research simulator, it is imperative to adapt existing and

Parts of this chapter have been published in the proceedings of 16th European Conference on the Mathematics of Oil Recovery (2018)[77], in the proceedings of SPE Reservoir Simulation Conference (2019)[78], and in the proceedings of 44th Workshop on Geothermal Reservoir Engineering (2019) [79]

introduce new input data arrays. Traditionally used input keywords, rigidly implemented in C++ core code, become an obstacle here. It is much more natural to prepare all input data in Python by reading text or binary files or generating it according to any desired algorithm, and then feed it into DARTS.

It is also true for the output data: having all results as Python variables, it is up to a user whether to save it in the desired format, analyse using various scientific libraries, (e.g., NumPy, SciPy or Pandas [81–83]), or export to a powerful visualization tool such as Paraview [84]. Finally, the entire simulator can be wrapped by a single Python function call, immediately opening opportunities for inverse modelling without redundant Input/Output overhead.

5.2. Decoupling of Physical Properties

Following the main idea of OBL, the DARTS framework distinguishes operators from governing equations and treats them in a special way. Operators are functions of the state in a single control volume. Typically, they represent a combination of fluid and rock properties and correspond to the most complex and nonlinear part of the governing equations. Sometimes, the dependency of operators on the state is determined through indirect procedures like phase-split, and therefore it is hard to linearize them in a general way. Suggesting an alternative to an automatic or direct hand-differentiation solution to this problem, OBL replaces the operators with their piece-wise multilinear approximations. For those, it is possible to express their derivatives with respect to state variables in a general way.

In DARTS, approximated operators values (along with partial derivatives) are computed via multilinear interpolation, where the amount of dimensions matches the number of nonlinear variables (i.e. the length of the state, or the number of degrees of freedom) in a single control volume. The true operator values, which are used in interpolations, are called supporting points or base points. They are computed in an adaptive manner during simulation [59] only once for a given state. Supporting points then are saved in a special two-level sparse storage designed for efficient lookup and re-use.

This approach has proven to be especially effective when property calculations, involved in the evaluation of operator values, are computationally expensive (e.g., involve complex phase behaviour). Since supporting points are values of functions of the state, they do not depend on spatial location and can be applied either across the whole reservoir or at least within the sub-regions where fluid and rock properties remain constant (e.g., PVT regions). Thereby in DARTS, property calculations occur relatively rare and their amount depends not on the spatial discretization, but rather on a discretization of the parameter space used for operator approximation and development of the simulation in that space.

From the perspective of the simulation nonlinear loop, the operator interpolation replaces properties calculations in the Jacobian assembly step. In addition, it also 'shadows' physical phenomena behind the operators, leaving out only the values of supporting points, which are rarely computed but utilized all the time during interpolation. This allows to detach fluid and rock properties calculations (now only performed during operator evaluation at supporting points) from the main nonlinear

loop, as well as to relax the performance requirements for such calculations. The Jacobian assembly now depends on the choice of the nonlinear variables and the governing physical mechanisms which are taken into account. The former determine the dimensionality of parameter space, while the latter define the operators required for the assembly. Once the choice is made, the Jacobian assembly is simply the right combination of approximated operator values and partial derivatives with spatial properties and states, encapsulated in a simulation engine.

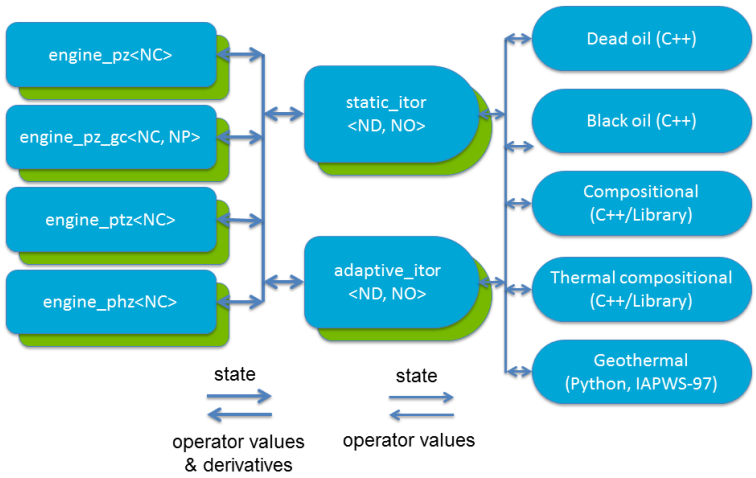


Figure 5.1: DARTS modular structure

The modularity of DARTS is demonstrated in Figure 5.1. On the left, four simulation engines are shown:

- engine_pz – multiphase multi-component mass transport, $\boldsymbol{\omega} = \{p, z_1, \dots, z_{n_c-1}\}$;
- engine_pz_gc – multiphase multi-component mass transport with gravity and capillarity, $\boldsymbol{\omega} = \{p, z_1, \dots, z_{n_c-1}\}$;
- engine_ptz – multiphase multi-component mass and energy transport, $\boldsymbol{\omega} = \{p, T, z_1, \dots, z_{n_c-1}\}$;
- engine_phz – multiphase multi-component mass and energy transport, $\boldsymbol{\omega} = \{p, h, z_1, \dots, z_{n_c-1}\}$.

All engines are written in a general manner for NC components (and NP phases for engine_pz_gc). Notion $< NC >$ here indicates that the variable represents integer

template parameter of corresponding class, known at compile time. This approach allows to maximize various compiler optimizations (e.g., loop unrolling).

Next, two interpolators are available (Figure 5.1, middle):

- static_itor – pre-computes all supporting points in advance, and can be useful for coarse physical representation and low-dimensional parameter space;
- adaptive_itor – adaptively computes supporting points along with the simulation, as described in Section 2.6.

Both interpolators are written in a general way for ND degrees of freedom and NO operators. All operator values are stored together to benefit from faster search and interpolation. Moreover, operator values computed during simulation can be stored and loaded, which can be extremely beneficial in case of running multiple models with the same physical properties (inverse modelling, optimization).

Finally, several operator sets are present (Figure 5.1, right):

- Dead oil – water and oil components, water and oil phases, $\boldsymbol{\omega} = \{p, z_w\}$;
- Blackoil – water, oil, and gas components, water, oil, and gas phases, $\boldsymbol{\omega} = \{p, z_g, z_o\}$;
- Compositional – n_c components, liquid and vapor phases, $\boldsymbol{\omega} = \{p, z_1, \dots, z_{n_c-1}\}$;
- Thermal compositional – n_c components, liquid and vapor phases, $\boldsymbol{\omega} = \{p, T, z_1, \dots, z_{n_c-1}\}$;
- Geothermal – water component, liquid and vapor phases, $\boldsymbol{\omega} = \{p, h\}$.

All the items above are implemented in C++, except the geothermal operator set, which is implemented purely in Python, as a wrapper over the IAPWS library [85]. Nevertheless, as is shown in Section 5.7, it does not diminish simulation performance unless excessive parameterization accuracy is used. With sufficient parameterization resolution, DARTS is significantly faster than other simulators (see Subsection 5.7.4).

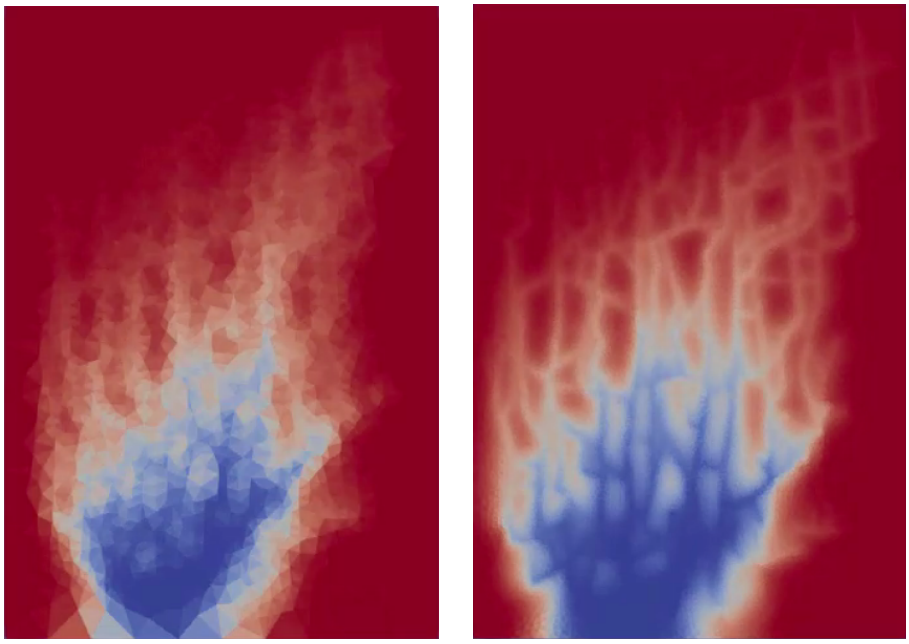
Note, that it is relatively easy to refactor existing C++-based operator sets with an AD library. In this case, it will be possible to skip the interpolation link and directly supply the AD-based gradient from the operator set to the simulation engine. This mode momentarily converts DARTS to a conventional simulator with the exact representation of physical properties. It can be used to obtain a reference solution or simulation time for accurate estimation of OBL accuracy and performance.

In addition, DARTS opens the opportunity to perform the entire simulation on GPU architecture by offloading only engines, interpolators, and linear solvers. All operator-related computations may be still performed on CPU without significant impact on simulation performance thanks to adaptive parameterization.

Detaching operators and engines create unique opportunities in terms of both flexibility and performance. From the perspective of the engine, the exact implementation of evaluation of operator supporting points is not relevant, because operators values along with their derivatives are computed by interpolation. Receiving those, the Jacobian assembly is then done using straightforward computation of derivatives, which is feasible because of the simplicity of the governing equations written in operator form, even for complex physical applications.

5.3. Unstructured Grids

In order to keep the framework general and flexible, the space discretization procedure is left out of the simulation engines. They are initialized by a connection list, which represents peer-to-peer connectivity between control volumes in the reservoir and can be built in the same format for both structured and unstructured grids. The connection list for TPFA is defined by the total amount of grid blocks and a list of connections. Each connection is defined by the set (i, j, Γ, Γ_d), where i and j are indices of neighbouring control volumes, Γ is transmissibility of fluxes and Γ_d is diffusion transmissibility. The sparsity pattern of the Jacobian matrix is computed directly based on the connection list and remains fixed during the simulation.

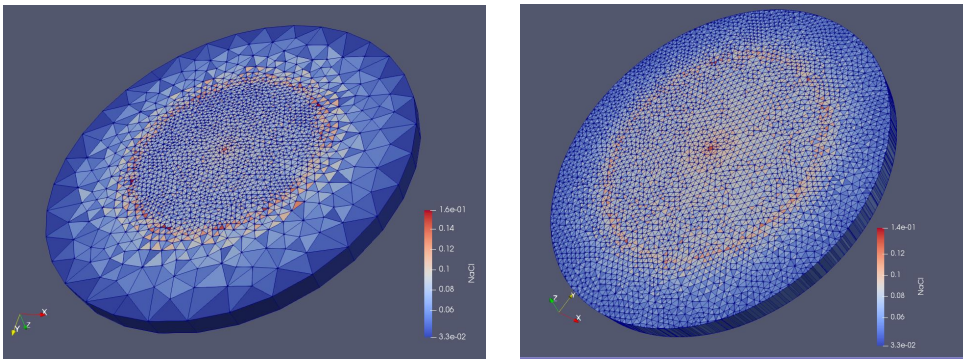


(a) Coarse grid with 3722 control volumes and 1376 fracture elements

(b) Fine grid with 31746 control volumes and 3610 fracture elements

Figure 5.2: Unstructured grids for Descrete Fracture Model (DFM)

This approach allows to run the same simulation code on structured grids, grids build within Discrete Fracture Model (DFM) concept (described among many others



(a) Coarse grid (b) Fine grid

Figure 5.3: Unstructured radial grids

in [86]), or radial unstructured grids (see examples in Figure 5.2, Figure 5.3).

5.4. Multisegment Wells

Following the general unstructured grid framework, a well is discretized by a set of control volumes of well segments, chained together by connections. The current implementation only includes the homogeneous flow model in segments. Any well segment can be connected with an arbitrary number of reservoir control volumes, representing well perforations. For the well discretization, we use a connection-based approach, suggested by [87].

Each perforation is characterized by geometrical transmissibility representing the connectivity of corresponding well segments to the reservoir, also referred to as a well index. Similar to the connections between reservoir grid blocks, well indices are computed outside simulation engine taking into account geometry and orientation of wellbore and perforated grid block (along with associated rock properties) in a general unstructured grid. In addition, the top well segment is also connected to a ghost control volume, which has exactly one connection and is used as a placeholder for well control equations (see details in [88]).

Two examples are shown in Figure 5.4: a one-segment well configuration (similar to a regular well) is on the left and a multi-segment well configuration is on the right. Reservoir control volumes are shown in gray; well control volumes including the top segment w_1 - in blue; the well ghost control volume w_0 - in red. The interface between w_0 and w_1 is denoted as w. Black arrows represent connections between reservoir control volumes; blue arrows - perforation connections; red arrows - intra-well connections. Even though the examples show a structured grid case with a vertically oriented wellbore, the well configuration in DARTS can be arbitrary owing to the connection-based approach to describe well perforations.

All well control volumes are considered as extensions of a reservoir and treated exactly the same way during Jacobian assembly, except for w_0. Naturally, due to the absence of the porous media inside a wellbore, the phase relative permeabilities

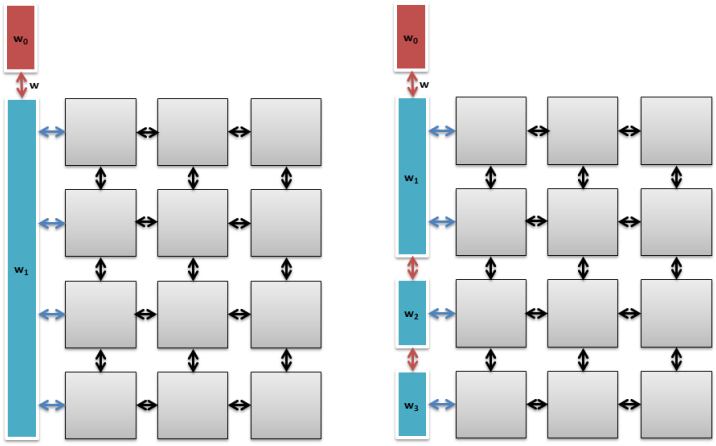


Figure 5.4: Example of multi-segment well discretization for structured reservoir

will be different from those in the reservoir. This can be modelled accordingly via operator regions (Section 5.5). Each well segment is defined by a volume-dependent on a wellbore diameter and a segment length, while other properties are neglected. The flow in the multi-segment well is following the homogeneous multiphase flow in an idealized tube without roughness or slip [88].

5.4.1. BHP Well Control

One of the two most common controls for wells in reservoir simulation is fixed bottom hole pressure. The following system of equations is applied to the w_0 control volume instead of Eq. Equation 2.10 in order to maintain target pressure p^{target}:

$$p - p^{target} = 0, \tag{5.1}$$

$$z_c - z_c^{up} = 0, \quad c = 1, \dots, n_c - 1, \quad z_c^{up} = \begin{cases} z_c^{inj} & \text{for injector} \\ z_c^{w_1} & \text{for producer} \end{cases} \tag{5.2}$$

5.4.2. Rate Well Control

Another common way to define the well regime is to specify volumetric phase rate at surface conditions. In order to parametrize this rate, we first define the state at the separator (or surface) conditions using the overall composition of the flux β_c^w over interface w, which is evaluated according to Equation 2.13:

$$\boldsymbol{\omega}^{SC} = [p^{SC}, T^{SC}, \frac{\beta_1^w}{\sum_c \beta_c^w}, \dots, \frac{\beta_{n_c-1}^w}{\sum_c \beta_c^w}] \tag{5.3}$$

Next, we can obtain a target rate introducing rate operator $\zeta_p(\boldsymbol{\omega})$:

$$Q_p = \frac{b^w \sum_c \beta_c^w}{dt} \frac{S_p(\boldsymbol{\omega}^{SC})}{\rho_t(\boldsymbol{\omega}^{SC})} = \frac{b^w}{dt} \zeta^w(\boldsymbol{\omega}), \quad \zeta_p^w(\boldsymbol{\omega}) = \begin{cases} \zeta_p(\boldsymbol{\omega}) & \text{for injector} \\ \zeta_p(\boldsymbol{\omega}^w) & \text{for producer} \end{cases} \tag{5.4}$$

Finally, we write down equations for the control volume w_0 to maintain target rate Q_p^{target}:

$$\frac{b^w}{dt} \zeta_p^w(\boldsymbol{\omega}) - Q_p = 0, \tag{5.5}$$

$$z_c - z_c^{up} = 0, \quad c = 1, \dots, n_c - 1, \quad z_c^{up} = \begin{cases} z_c^{inj} & \text{for injector} \\ z_c^{w_1} & \text{for producer} \end{cases} \tag{5.6}$$

Due to more nonlinear relations involved in operator evaluation for well controls, parameterization tables with resolution higher than in reservoir simulation are often needed to control the error.

Note, that physical state $\boldsymbol{\omega}$ for well control volumes, including w_0, is defined exactly the same way as for the reservoir control volume. Hence, the choice of nonlinear variables and their order is also identical. This approach simplifies effective preconditioning of the linear system (see Section 5.6).

5.4.3. Validation

Another source of error comes from operators involved in the approximation of properties for well controls. Multi-segment wells provide the most accurate solution when cross-flow effects coupled with complex phase behaviour occur in the reservoir model. In order to mimic these conditions, we took a synthetic model comprised of three layers with lateral permeabilities of K_{xy} =100, and 500 mD, while the vertical permeability was set at $K_z = \frac{K_{xy}}{100}$. Each layer consisted of 10x10 grid blocks of 100x100x10m with a porosity of 25%. The initial oil is composed of C_1, C_4, and C_{10} at corresponding compositions: 1% methane, 35% n-butane, and 64% n-decane. The description of the phase behaviour and properties is based on the Peng-Robinson Equation of State [56].

Two vertical multi-segment wells with three segments each are placed at the opposite corners of the model. Each segment is connected to the corresponding layer with different well indices of 10, 20 and 30. We inject a mixture of 99% of C_1 and 1% of C_4 at a constant gas rate $Q_g = 1.5 \times 10^5$ sm^3/day. The production well operates at a constant oil rate $Q_o = 800$ sm^3/day with a minimum BHP constraint of 10 bar. In order to model single-phase gas injection into a single-phase liquid, we set the initial pressure at $P_0 = 60$ bar and temperature $T_0 = 77$ °C. The simulation period is 4000 days.

The comparison between DARTS and AD-GPRS with multi-segment well model is shown in Figure 5.5, Figure 5.6. The two results match very well. The injector BHP climbs up due to gas compressibility till breakthrough happens after roughly

1600 days of simulation, as it can be seen from Figure 5.5. After that the injection pressure rapidly drops, while the producer cannot satisfy the oil rate control anymore after roughly 2300 days of simulation and therefore switches to the BHP constraint of 10 bar.

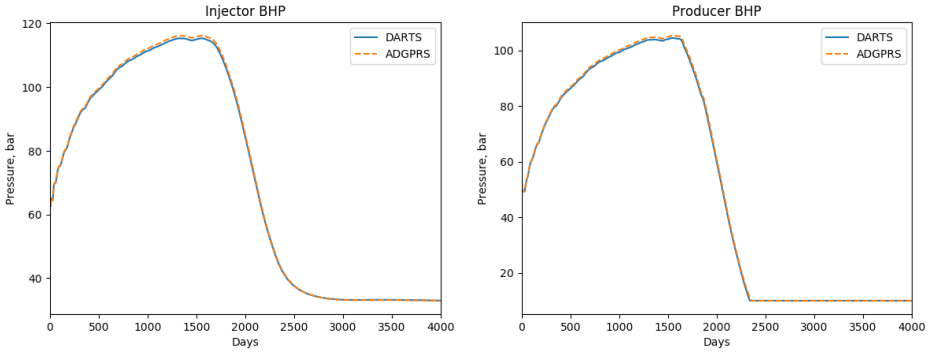


Figure 5.5: BHP for injector and producer provided by DARTS and AD-GPRS

Producer gas and oil rate comparisons are shown in Figure 5.6. There is a very good match between the DARTS and AD-GPRS results. The producer gas rate starts at 0, then rapidly increases after breakthrough and falls back after the producer switches to the BHP control. The production oil rates computed by the DARTS and AD-GPRS multi-segment well models (denoted as AD-GPRS_ms) match well. The results provided by the AD-GPRS standard well model, denoted as AD-GPRS_std, underestimate oil production after breakthrough and illustrate the substantial difference between the two well models in this case.

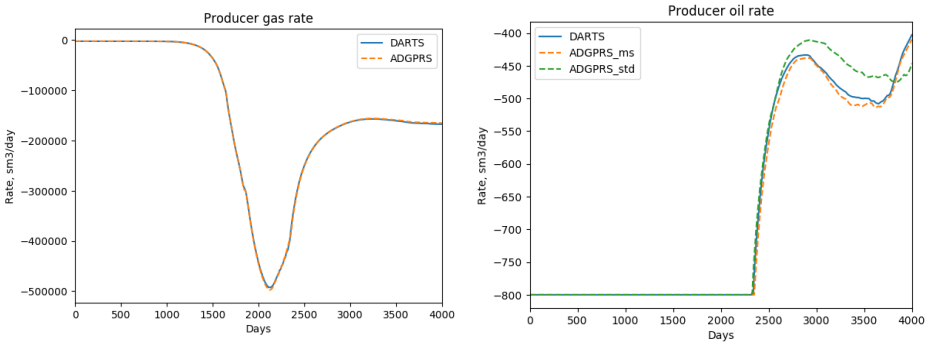


Figure 5.6: Producer oil and gas rates provided by DARTS and AD-GPRS

In order to test the same physical set up in a highly heterogeneous reservoir, we took the top layer of the SPE10 test case [76] and applied an inverted five- spot well pattern. The reservoir's initial physical state and injection mixture are the same as in the previous case. The injection well starts with the BHP control at 180 bar, and

after 500 days switches to gas rate control $Q_g = 2300\,\mathrm{sm}^3/\mathrm{day}$ with a maximum BHP constraint of 199 bar. The production wells, placed at the corners, operate with gas rate control $Q_g = 200\,\mathrm{sm}^3/\mathrm{day}$ with a minimum BHP constraint of 30 bar. The simulation period is 2500 days.

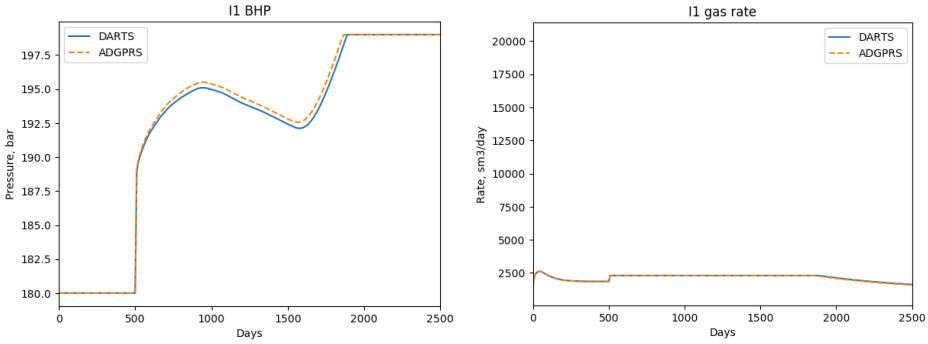


Figure 5.7: Injector BHP and gas rate provided by DARTS and AD-GPRS

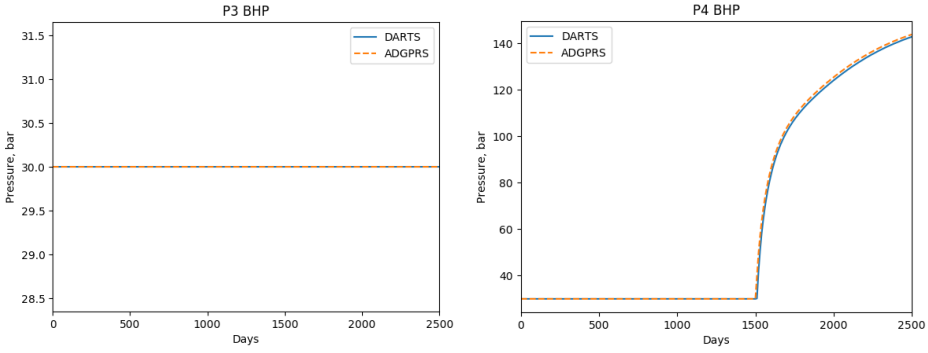


Figure 5.8: P3 and P4 BHP provided by DARTS and AD-GPRS

Figure 5.7, Figure 5.8, Figure 5.9 , Figure 5.10 demonstrate a close match between the DARTS and AD-GPRS results. The injection well switched to gas rate control after 500 days of simulation and reached the BHP limit by roughly the 1900th day, as can be seen in Figure 5.7. The P1 and P2 production wells happened to be perforated in low permeable reservoir area producing negligible amounts of fluids, so we omit their results. The BHPs for P3 and P4 are shown in Figure 5.8. For P3 it remains constant throughout the simulation, while for P4 it starts raising after 1500 days of simulation - the well switches to gas rate constraint after the breakthrough. This is confirmed by Figure 5.9 and Figure 5.10. The oil production rate immediately decreases while the gas production rate increases until its limit of Q_g=200 $\mathrm{sm}^3/\mathrm{day}$ once the breakthrough is reached for P4. A small increase in the gas production rate by the end of the simulation indicates the breakthrough for the P3 well.

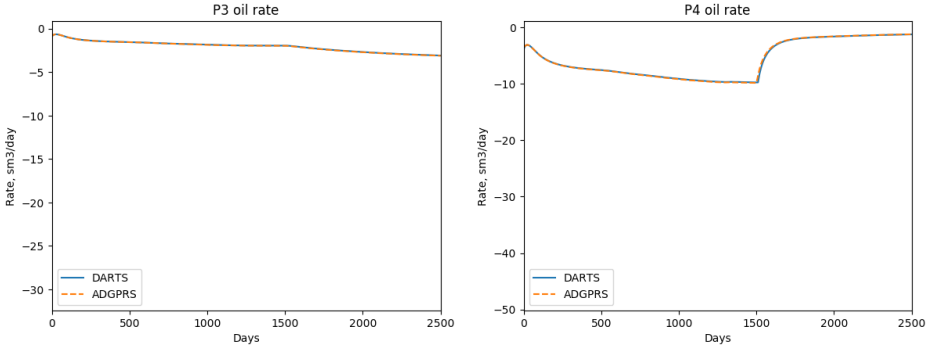


Figure 5.9: P3 and P4 oil rates provided by DARTS and AD-GPRS

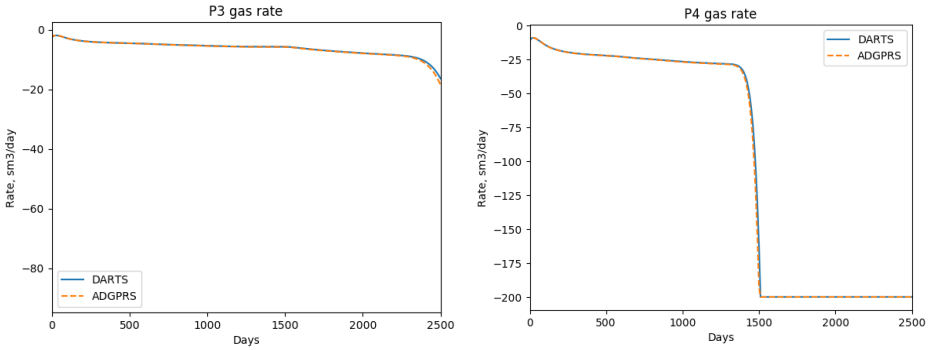


Figure 5.10: P3 and P4 gas rates provided by DARTS and AD-GPRS

5.5. Operator Regions

One of the important features, usually implemented in industry-level simulators, is the ability to model fluids and rock with spatially different properties in a single reservoir. It is essential for modelling of complex multilayered reservoirs with significantly different characteristics, especially when wells are perforated in several layers. Traditionally, each option of fluid PVT, SCAL, or rock compaction properties is associated with a single corresponding region. Then, every control volume in the computational mesh is explicitly assigned to a specific region via corresponding keywords for each of spatially variable property (e.g., PVTNUM, SATNUM, ROCKNUM in [22]). Usually, regions represent large reservoir partitions, their amount is limited, and partitioning remains fixed during a simulation.

DARTS supports this feature introducing operator regions. Since all physical properties of fluid and rock are represented by corresponding state operators, there is no need to define a separate set of regions for every property. Therefore, it is sufficient to associate every control volume of the computational mesh to a single operator region (via OPNUM, similarly to above-mentioned keywords), whereas each operator region represents spatial variability in any of fluid or rock properties.

Table 5.1: Validation model parameters

Parameter	Value
Reservoir dimensions, m	1000x10x1
Reservoir grid	100x1x1
Permeability, mD	100
Porosity	0.2
Initial pressure, bar	100
Initial water saturation	0.2
Injector BHP, bar	150
Producer BHP, bar	50
Maximum timestep, days	10
Simulation period, days	3000

(a) SWOF 1

S_w	K_{rw}	K_{ro}
0.2	0.0	1.0
0.5	0.2	0.6
1.0	1.0	0.0

(b) SWOF 2

S_w	K_{rw}	K_{ro}
0.2	0.0	1.0
0.5	0.5	0.5
1.0	1.0	0.0

Validation of the numerical solution obtained by DARTS in case of spatial variability of fluid properties was done by comparison against the solution obtained from the Eclipse 100 simulator [22]. Conventional waterflooding based on Dead-Oil PVT description was modelled for an homogeneous one-dimensional reservoir with high resolution in space, time, and physical properties discretizations (only for DARTS). In this model, gravity and capillarity effects were not taken into account.

The model parameters are represented in Table 5.1. The injector was placed at the first grid block, the producer at the last one. The first half of the model was considered as region 1, the second half - region 2. Each of the regions was assigned to a specific oil-water relative permeability curve introducing spatial variability in fluid properties. In the Eclipse 100 model, two SWOF tables (Table 5.1a, Table 5.1b) were used along with the SATNUM keyword. In the DARTS model, two independent sets of operators were initialized with corresponding tables. Since PVT properties

were not different between the regions, values for accumulation operators matched between sets, however flux operators values (for physical states corresponding to water saturation above connate water saturation) were different.

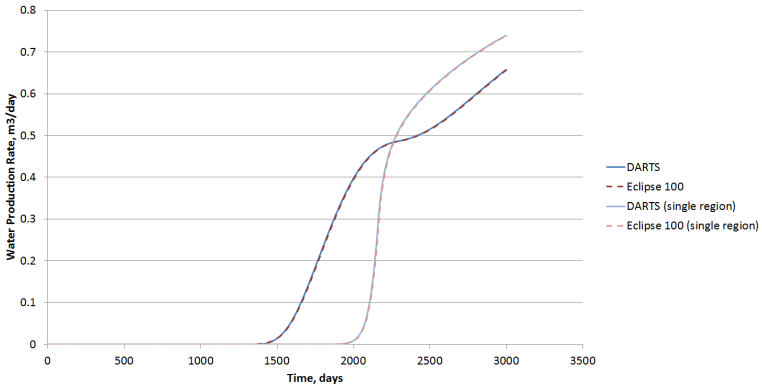


Figure 5.11: Comparison of water production rates between DARTS and Eclipse 100 models for 2 regions and a single region

Figure 5.11 shows the water production rate at the producer well for DARTS and Eclipse 100 models. It is easy to see that the match between the two solutions is almost ideal. In order to confirm the significance of spatial variability in fluid properties in this case, also a single region model was simulated and compared. Here, only the table corresponding to the first region was used for the entire reservoir. Since Table 5.1a corresponds to the less mobile water phase, the breakthrough happens later compared to the 2 regions model, and the water production rate curve is substantially different. Still, the solutions obtained by DARTS and Eclipse 100 match very well.

5.6. Linear Solvers

For the vast majority of practical reservoir simulations, linear solution occupies the most of simulation time. Efficient implementation of linear solvers is determined not only by the actual algorithms and quality of their implementation but starts from the choice of the underlying data storage and further depends on the number of implicit transformations it undergoes at various levels of the linear solver.

One of the most common storage formats for large sparse matrices is Compressed Sparse Row (CSR) [34]. Its goal is to minimize data transfers between memory and CPU by storing contiguously nonzero entries belonging to a single row. Linear systems originating from problems with n degrees of freedom per control volume (element) solved with fully implicit schemes, usually exhibit $n \times n$ dense blocks of nonzero entries in matrix portraits.

The Block Compressed Sparse Row (BCSR) matrix format, introduced by Pinar and Heath [89], exploits this feature to reduce the memory access even further. The amount of row and column indices is then reduced by a factor of n. More

importantly, all linear combinations involving the matrix (e.g., sparse matrix-vector multiplications) are performed block-wise, operating on dense $n \times n$ sub-matrices, which is more efficient on modern CPU architectures due to more intense usage of register and cache storage.

DARTS engines perform Jacobian assembly directly into a BCSR storage in a single pass, filling in both diagonal and off-diagonal values. This choice is conditioned by the goal to develop a general and efficient framework for simulation on unstructured grids. Depending on the upwind direction, certain derivative values result in zeroes. Nevertheless, they are still stored explicitly inside matrix blocks to maintain uniform data storage format.

Since the computational mesh is assumed to be fixed for the entire simulation, and well opening/closing does not entail changes in the number of variables, the BCSR structure is computed once at the beginning of a simulation and does not change in the process. Only matrix values are recomputed on every iteration, while its portrait stays constant. This assumption allows to apply the same approach at a linear level: internal structures are initialized only once, further increasing the performance of the linear solution.

The standard choice of linear solvers for large reservoir simulation are Krylov subspace iterative solvers with a sophisticated preconditioning strategy [34]. Preconditioning is an essential technique to reduce the condition number of the linear system increasing the convergence rate of the iterative solver substantially. Typically, different preconditioners work with various efficiency when applied to systems with different nature. For example, the Algebraic Multigrid Method (AMG) is efficient for near-elliptic problems [32], while Incomplete LU factorization with 0 level of fill-in (ILU(0)) is successfully applied for near-hyperbolic equations [90].

Reservoir simulation equations with a Fully Implicit approximation scheme lead to the linear system where both types of unknowns are present. It is comprised of a near-elliptic pressure equation, a near-hyperbolic composition (saturation) equation, while the temperature equation can be either type depending on whether the process is conduction- (thermal diffusion) or convection-dominated. For the efficient treatment of such systems, a Constrained Pressure Residual (CPR) approach was designed in [29, 30].

The CPR method is a two-stage preconditioner, where at the first stage, the pressure system is decoupled from the full system and solved separately with AMG-based scheme. Often, a single V-cycle is enough for efficient preconditioning. At the second stage, the full system is processed by an ILU(0) preconditioner using the pressure solution from the first stage. This strategy has proved to be very robust and efficient even for highly heterogeneous reservoirs with strong coupling between elliptic and hyperbolic parts of the linear system. This results in stable convergence within a limited number of linear solver iterations even when simulation time steps are very large.

The linear system in DARTS is solved using the Flexible Generalized Minimum Residual (FGMRES) iterative method [91]. All matrix operations are performed in native BCSR format. The two-stage CPR preconditioning strategy is employed. The pressure system is decoupled from the full FIM system using a True-IMPES reduction

approach directly from the BCSR storage. Then, a single V-cycle of the AMG solver is used to obtain an approximation of the pressure solution. Finally, it is substituted in the full system and the block ILU(0) preconditioner is applied.

5.7. Geothermal Benchmark

In this section, we perform a set of benchmark tests and compare the simulation results of DARTS with the state-of-the-art research simulators TOUGH2 and AD-GPRS. First, we describe the geothermal model used in the benchmark. Next, we compare simulation results of pressure and temperature solution under both low- and high-enthalpy conditions. Finally, we display the performance of different simulators in the geothermal formulation. In the benchmark study, because of the complexity of data pre-processing in TOUGH2 and some convergence issues in AD-GPRS for the high-enthalpy condition, a single layer of the model is used to run and compare under low- and high-enthalpy condition within 3 simulators. The full model is only compared with AD-GPRS under the low-enthalpy condition.

5.7.1. Three-dimensional Geothermal Model

A synthetic geological model from [92] is used in this section for benchmark tests. All properties in the model are populated with a dataset from fluvial Nieuwerkerk formation of the West Netherlands Basin similar to Subsection 3.1.3. The reservoir dimensions are 1.8 km $\times$ 1.2 km $\times$ 0.1 km as shown in Figure 5.12. The discretized model contains 60 $\times$ 40 $\times$ 42 grid blocks. One doublet is placed on the middle-line along the X-axis with 1.2 km spacing as shown in Figure 5.13. The fluvial sandstone is distributed along the X-axis of the reservoir. The open flow boundary condition is set along the Y-axis of the reservoir, and a no-flow boundary condition is defined along the X-axis of the reservoir. Two energy-transfer mechanisms are considered in this process: convective and conductive heat flow.

Table 5.2: Parameters used in benchmark tests

Parameter	Value
Depth, m	2300
Pressure, bar	200
Temperature, K	348.15
Porosity	0.16 ~ 0.36
Permeability, mD	6~ 3360
Sand heat capacity, $\mathrm{kJ/m^3/K}$	2200
Sand thermal conductivity, $\mathrm{kJ/m/day/K}$	180

5.7.2. Comparisons of DARTS and TOUGH2

Table 5.3 shows the initial conditions and well controls used in the validation with TOUGH2. The results are shown in Figure 5.14 and Figure 5.15 for low- and high-enthalpy conditions respectively. The TOUGH2 solution is taken as a reference. A good match can be observed between DARTS and TOUGH2 results under both

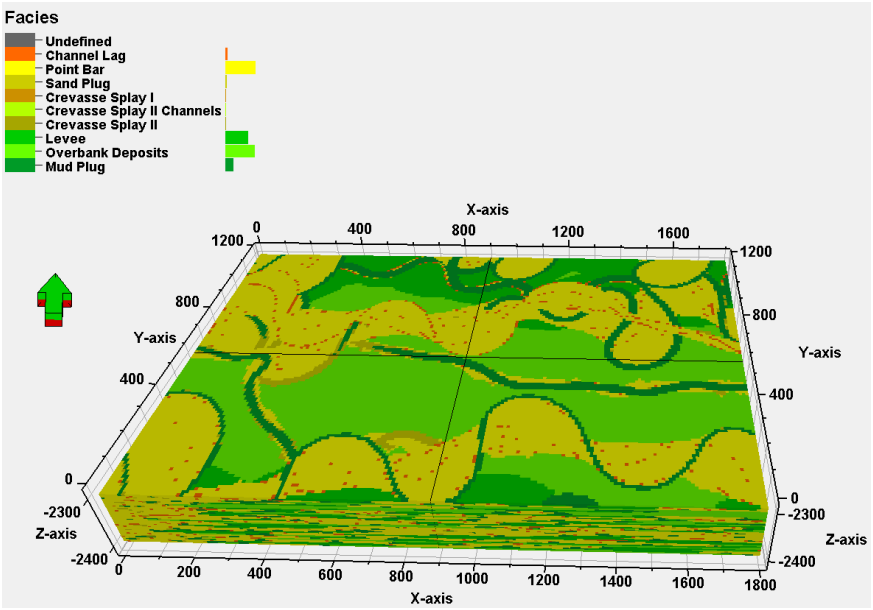


Figure 5.12: Schematic of the facies distribution for synthetic geothermal model

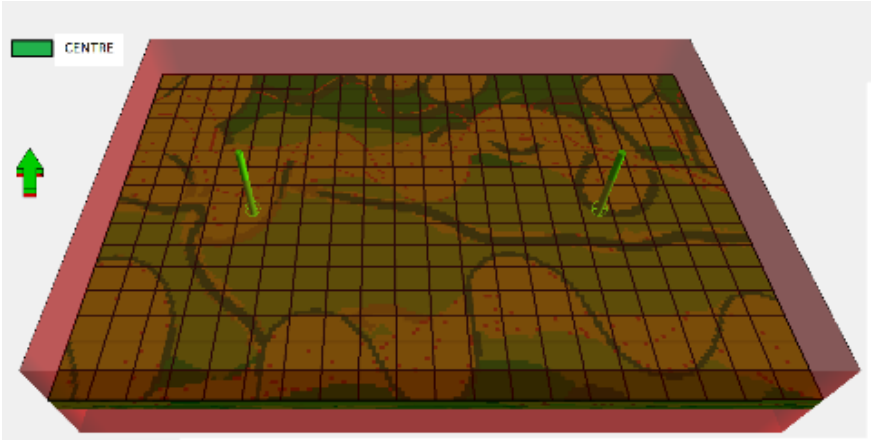


Figure 5.13: Geothermal doublet location for synthetic geothermal model

conditions. The maximum relative temperature difference is just around 1.6% for the low-enthalpy case rising to 4% and concentrating along the cold front for the high-enthalpy case. The higher difference in the high-enthalpy case, apart from being attributed to higher nonlinearity of the process, can also be explained by a higher amount of timesteps for TOUGH2 (see Table 5.5), leading to a certain difference in time truncation errors between the simulators. The saturation solution exhibits differences up to 10% only along the front, where both liquid and vapor

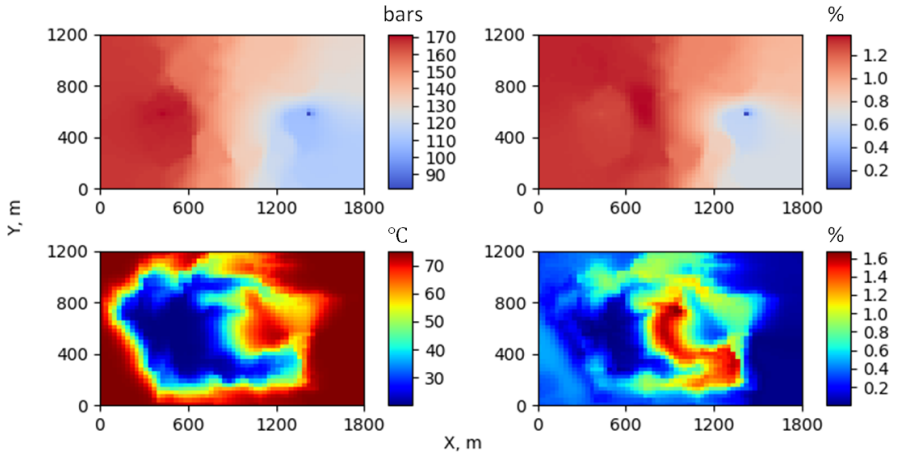


Figure 5.14: Comparison of final solutions for low-enthalpy conditions obtained from DARTS and TOUGH2 for pressure (top row) and temperature (bottom row). In each row, the left column corresponds to the final solution from TOUGH2, while the right column represents the relative difference between solutions from TOUGH2 and DARTS

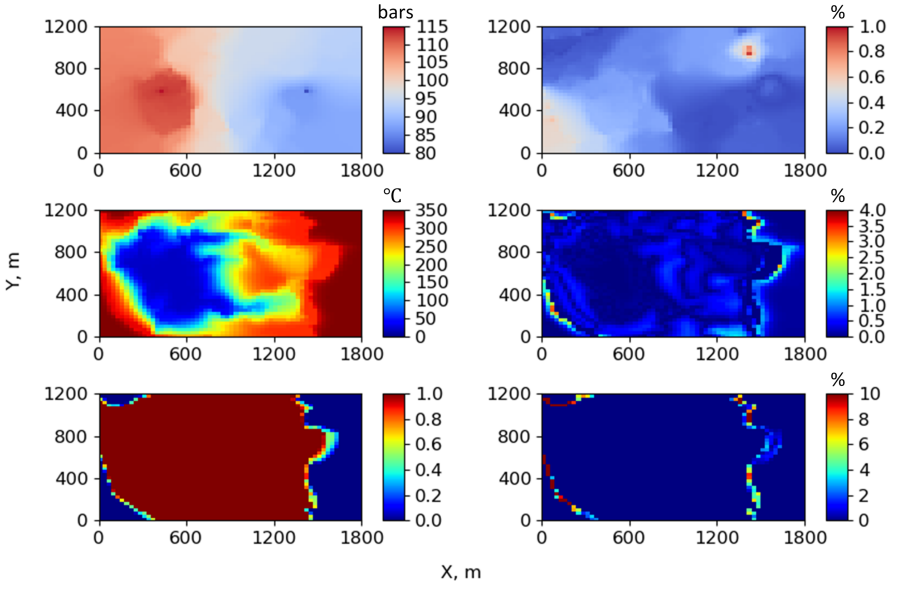


Figure 5.15: Comparison of final solutions for high-enthalpy conditions obtained from DARTS and TOUGH2 for pressure (top row), temperature (middle row), and water saturation (bottom row). In each row, the left column corresponds to the final solution from TOUGH2, while the right column represents the relative difference between solutions from TOUGH2 and DARTS

Parameter	Low-enthalpy	High-enthalpy
Initial temperature, K	348.15	623.15
Initial pressure, bar	100	100
Injection enthalpy, kJ/kg	100	100
Injection rate, $\mathrm{m}^3/\mathrm{day}$	36	36
Production pressure, bar	80	80
Simulation time, years	100	100

Table 5.3: Simulation parameters used for comparison between DARTS and TOUGH2 in low-enthalpy and high-enthalpy cases

phases are present.

5.7.3. Comparison of DARTS and AD-GPRS

Here, we take the AD-GPRS solution as a reference. Figure 5.16 and Figure 5.17 show the solution and difference of a single-layer model. Figure 5.18 shows the solution of the 20th layer of the full three-dimensional model. That layer corresponds to the highest differences in the solution since its average permeability is also the highest.

Parameter	Low-enthalpy	High-enthalpy
Initial temperature, K	348.15	623.15
Initial pressure, bar	100	100
Injection temperature, K	298.15	298.15
Injection rate, $\mathrm{m}^3/\mathrm{day}$	40	40
Production pressure, bars	80	80
Simulation time, years	100	100

Table 5.4: Simulation parameters used for comparison between DARTS and AD-GPRS in low-enthalpy and high-enthalpy conditions

As shown in Figure 5.16, for the model with the low-enthalpy condition, the maximum temperature difference between DARTS and AD-GPRS is around 3% of the overall temperature variation range. For the high-enthalpy case (Figure 5.17), the temperature variation range is from 25 to 225°C. The maximum temperature difference, in this case, goes up to 3.5% similarly to the comparison with TOUGH2. Again, partly the difference can be explained by the different amounts of timesteps required for simulators to converge. However, the highest differences are not only distributed along the front but are also present inside the liquid phase region. The difference in saturation solution remains at 10 % and is predictably distributed over the two-phase region along the front, exactly as in the case with TOUGH2.

For the full three-dimensional model with low-enthalpy conditions, the observed maximum temperature difference is observed for the 20th layer. As shown in Figure 5.18, it reaches around 2% for both temperature and pressure.

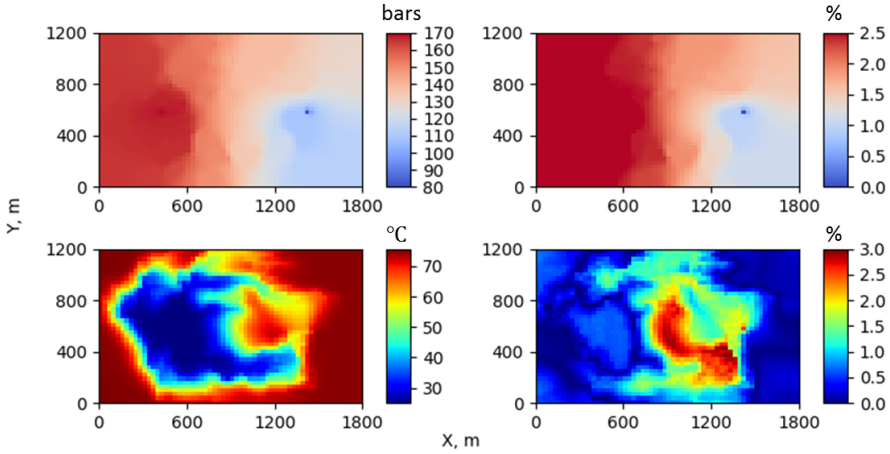


Figure 5.16: Comparison of final solutions for low-enthalpy conditions obtained from DARTS and AD-GPRS for pressure (top row) and temperature (bottom row). In each row, the left column corresponds to the final solution from AD-GPRS, while the right column represents the relative difference between solutions from AD-GPRS and DARTS

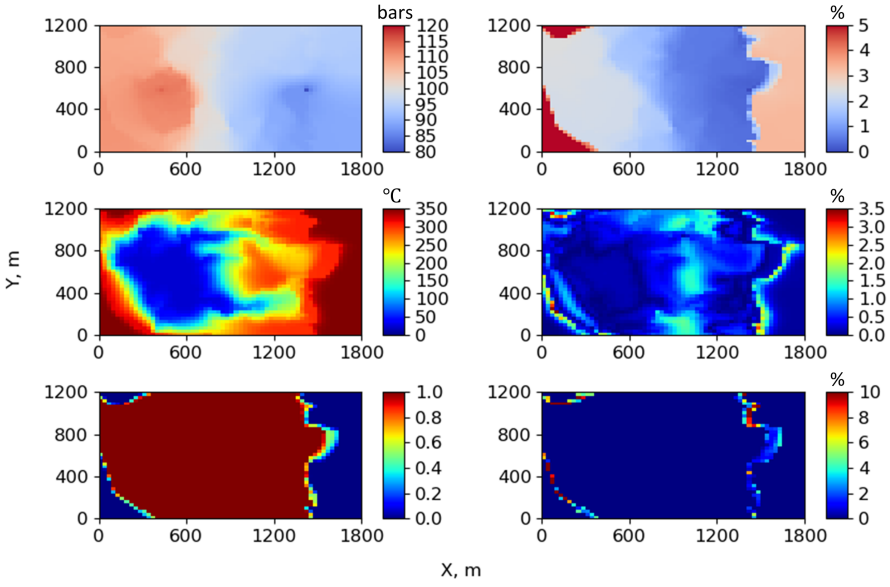


Figure 5.17: Comparison of final solutions for high-enthalpy conditions obtained from DARTS and AD-GPRS for pressure (top row), temperature (middle row), and water saturation (bottom row). In each row, the left column corresponds to the final solution from AD-GPRS, while the right column represents the relative difference between solutions from AD-GPRS and DARTS

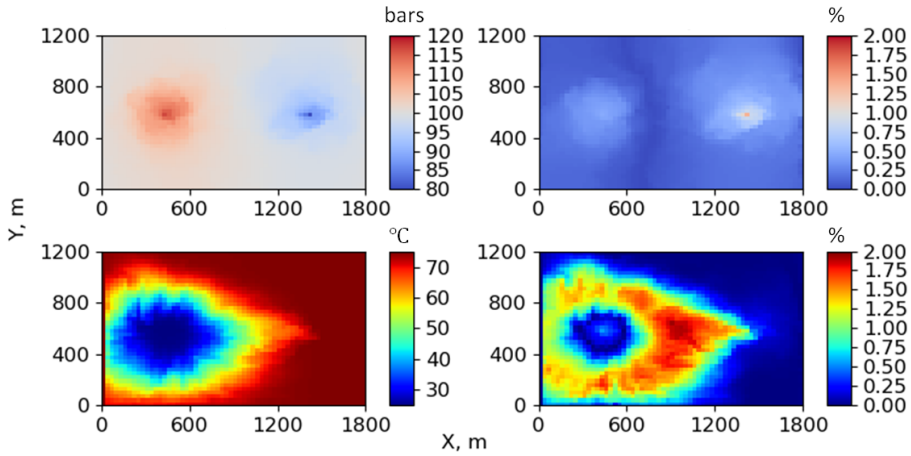


Figure 5.18: Comparison of final solutions for 20th layer of full three-dimensional model with low-enthalpy conditions obtained from DARTS and AD-GPRS for pressure (top row) and temperature (bottom row). In each row, the left column corresponds to the final solution from AD-GPRS, while the right column represents the relative difference between solutions from AD-GPRS and DARTS

5.7.4. Performance comparison

Table 5.5 shows the performance of different simulators as run on an Intel(R) Xeon(R) CPU 3.50GHz desktop. All runs have been performed in a single thread regime. Despite that in some cases, there are significant differences in the number of timesteps, nonlinear, and linear iterations, it is clear that DARTS achieves much better performance than TOUGH2 and AD-GPRS among these runs. A smaller timestep of 20 days is employed for high-enthalpy conditions. Nevertheless, the amount of timesteps for TOUGH2, in this case, is significantly higher than for DARTS, possibly due to certain limitations in the nonlinear convergence for the former (see [93]). Fast simulation in DARTS can be attributed to the OBL approach, which significantly simplifies the calculation of state-dependent properties and Jacobian assembly. A slightly higher number of nonlinear iterations in DARTS runs in comparison to AD-GPRS in the low enthalpy cases is related to different convergence criteria.

5.8. Performance on Realistic Full-Field Models

For general purpose simulation, it is important to deal with realistic full-field models at different levels of complexity. In this section, we demonstrate the applicability of DARTS to reservoir models with different physics, reservoir structure, and model size.

Test case	Simulator	Max ts (days)	Ts	Newt. it.	Lin. it.	CPU time (s)
Low-enthalpy	DARTS	365	115	259	1950	2.9
one layer model	TOUGH2	365	115	–	–	24.1
High-enthalpy	DARTS	20	2020	6834	95032	97.9
one layer model	TOUGH2	20	2997	–	–	942.0
Low-enthalpy	DARTS	365	115	259	1950	2.9
one layer model	AD-GPRS	365	115	253	1616	5.5
High-enthalpy	DARTS	20	2173	10855	125160	126.6
one layer model	AD-GPRS	20	2075	9742	159929	475.6
Low-enthalpy	DARTS	365	115	261	2841	159.3
full model	AD-GPRS	365	115	264	2437	446

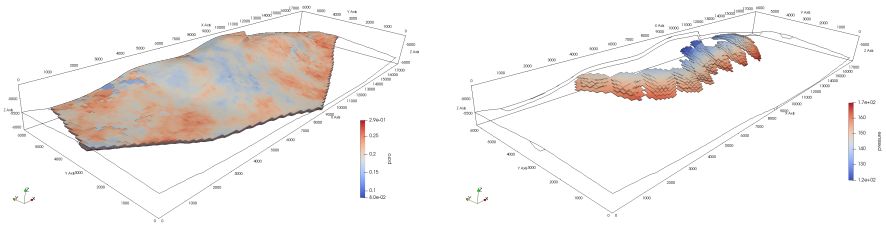
Table 5.5: Comparison of simulation performance of different simulators

5.8.1. Numerical Models

First, we describe different test cases utilized for performance comparisons. Models introduced by ascending order in the number of control volumes, the number of unknowns per control volume and the complexity of physics.

Brugge Field Model

The Brugge test case is often used as an optimization benchmark problem in reservoir simulation [68]. In our study, we used a particular permeability realization and production scenario described in [69]. The simulation time spans 10 years with BHP controls changing every 3 months for both injection and production wells. In this study, we only use this test case for performance comparisons. The detailed convergence analysis and the comparison with the reference physics can be found in [59]. The number of control volumes in this model is equal to 43,846 with two unknowns per each. There are in total 124,370 connections and dead-oil reference physics is used.



(a) Porosity scaled 3 times along Z axis

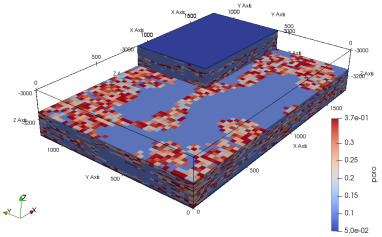
(b) Pressure along the water distribution

Figure 5.19: Brugge field

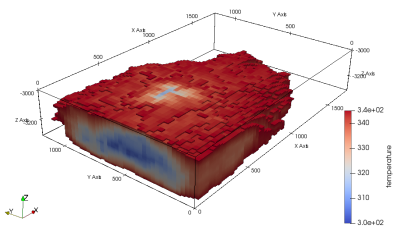
Geothermal Model

To test thermal-compositional formulation in the DARTS framework, we use a geothermal model described in Subsection 5.7.1. Here, we use a modification of the original

model where brine and dissolved methane are present at reservoir conditions. Details on validation of the geothermal model and the convergence analysis for OBL resolution accuracy can be found in [60]. This model has 100,800 control volumes with 3 independent unknowns (pressure, enthalpy and composition), 295,882 connections and mixed EoS-based thermal-compositional reference physics [94].



(a) Porosity distribution scaled 3 times along Z axis



(b) Temperature solution with threshold below 345 K

Figure 5.20: Geothermal model

SPE10 Model

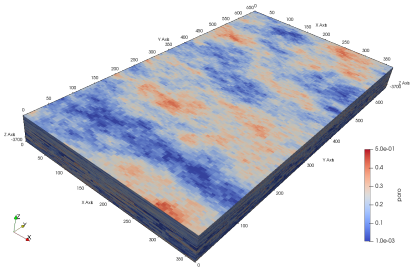
The results of the isothermal compositional simulation for gas injection processes is demonstrated using a four-component model described in [59]. In this model, the original distribution of permeability and porosity was taken from SPE10 problem [76]. The compositional properties were processed using the Peng-Robinson Equation of State [56] with original oil composition from [55]. The details of the model, comparison with the reference physics and convergence analysis for numerical results can be found in [59]. This model has 1,122,000 control volumes with 4 nonlinear unknowns per control volume, 3,329,020 connections and compositional reference physics [40].

5.8.2. Sensitivity to OBL Resolution

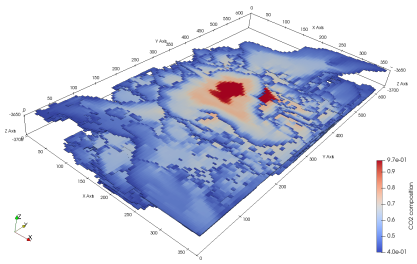
Here, we present the results of numerical simulation for the models described above. The models are described in ascending order of complexity where the first model has in total 87,728 degrees of freedom and simplest physics, the second model has 302,400 degrees of freedom and more complicated physics and the last model has 4,488,000 degrees of freedom with the most nonlinear physics.

Table 5.6 presents the inclusive simulation time for each model where the first subcolumn 'Sim' corresponds to the total simulation time, the second subcolumn 'Jac' represents the linearization time (Jacobian assembly) and the last subcolumn 'Gen' corresponds to the time spent on the generation of supporting points in the OBL parameterization.

It is clear that for the simplest (Dead-Oil) physics in the first model, the generation time is almost negligible since the property evaluation is extremely cheap

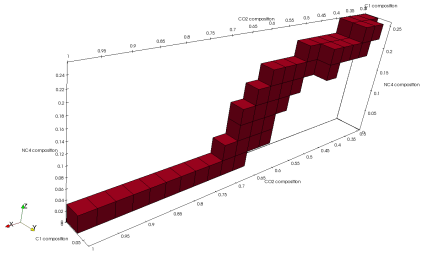


Porosity

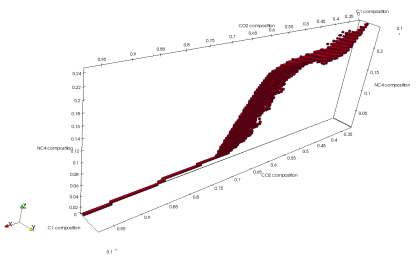


CO_2 composition

Figure 5.21: SPE 10 model



32 points resolution



256 points resolution

Figure 5.22: SPE 10 model compositional parameter space for pressure range 53-85 bar

Table 5.6: Performance results

Resolution	Brugge			Geothermal			SPE10		
	Sim, s	Jac, s	Gen, s	Sim, s	Jac, s	Gen, s	Sim, s	Jac, s	Gen, s
16384	84.6	17.2	5.23	1441.1	1316.1	1300.56	-	-	-
8192	83.5	15.5	4.02	1214.6	1082.7	1066.76	-	-	-
4096	81.5	13.4	2.19	1089.6	959.7	944.00	-	-	-
2048	80.0	11.7	0.88	809.1	683.4	668.40	14731.3	2639.2	2029.9
1024	79.6	11.0	0.30	491.5	368.5	354.22	13359.9	1020.1	419.6
512	79.1	10.4	0.09	266.3	139.3	125.19	10947.5	583.3	71.8
256	80.3	10.3	0.03	170.3	45.1	32.10	9627.5	476.8	12.3
128	78.0	9.8	0.01	137.0	17.6	6.35	7360.8	366.1	2.3
64	82.7	10.2	0.00	133.4	11.3	1.14	6323.8	327.3	0.5
32	84.8	10.3	0.00	130.2	9.7	0.21	5425.8	290.7	0.1
16	81.5	10.0	0.00	129.0	9.4	0.03	5432.4	307.2	0.1

(table-based). Even at the most expensive OBL resolution, the total cost of linearization is below 15% of total simulation time. For the larger model with more complex binary thermal-compositional physics, the cost of generation is growing much faster and soon enough (with the resolution above 256 points) becomes dominant in the simulation. For the bigger and more involved four-component compositional model, the linearization cost only becomes noticeable at extremely high OBL resolutions. This is happening since the compositional model, even in the most realistic setting, has a strong hyperbolic behaviour with a limited spread of compositions in the parameter space [11].

Notice that according to our previous investigations [see 46, 59], a resolution above 64 points already guarantees an error in simulation results below 1% in the most cases. It is also worth to mention that our multiphase flash solver is not optimized for performance and only tuned for accuracy of the phase behaviour prediction especially in the near-miscible gas injection regime (close to the critical point). In addition, the parameterized points in OBL can be effectively reused for repeated simulations since the solution in compositional space is mostly controlled by the thermodynamics of the problem [11]. Therefore, for subsequent launches of models, the effective simulation time remains nearly constant for any resolution of parameter space discretization.

6

DARTS Perspectives and Applications

6.1. Parameterization

In the previous chapters, all provided examples with OBL used a uniform parameterization of physical space. Despite that choice proves to be simple and efficient, a resolution increase leads to refinement in the entire (used) parameter space. Consequently, the generation of supporting points starts to dominate in the Jacobian construction (see Table 5.6) with highly accurate OBL parameterizations. At the same time, the refinement is useful only for those regions of the parameter space where the behaviour of state operators is the most nonlinear. Besides, "blind" uniform parameterization can be inaccurate when handling functions with a strong discontinuity of partial derivatives. One important example is the parameterization of phase boundaries in compositional simulation. While most of the phase properties are continuous when composition crosses these boundaries, the derivative of these properties can be highly discontinuous [40]. Intermediate solutions like local grid refinement (LGR) can be applied to preserve a rectilinear parameterization mesh, and therefore still applicable for piecewise multilinear interpolation. However, the introduction of unstructured non-uniform parameterization is more attractive, since it allows to minimize the number of supporting points and prescribe them freely and accurately to any position in parameter space (e.g., at the phase boundaries). Also, this choice forces to replace multilinear interpolation based on N-dimensional rectangles (hyperrectangles) with, for example, piecewise linear interpolation based on N-dimensional triangles (simplexes). Despite that searching for simplex locations is more difficult, the interpolation itself is much less expensive especially for high-dimensional problems (see [50]). Therefore, we investigate two strategies of non-uniform parameterization of physical space for compositional simulation.

Parts of this chapter have been published in the proceedings of 16th European Conference on the Mathematics of Oil Recovery (2018)[77, 95]

6.1.1. Physics-Based Non-Uniform Parameterization

The OBL approach simplifies the description of fluid and rock properties by building approximation interpolants for the operators α_c, β_c and θ_c within the parameter space of a simulation problem [see 59, for details]. Those interpolants are then used in the course of simulation to obtain the values and partial derivatives of the operators with respect to nonlinear unknowns for the Jacobian assembly. Uniform discretization of parameter space, which was used previously, proved to be a simple, efficient, and robust approach. However in this approach, approximation quality and consequently the accuracy of a simulation depend only on the number of supporting points, while their locations are prescribed blindly.

Next, we describe a tie-line-based non-uniform parameterization approach. The key idea is to use the knowledge of the thermodynamic behaviour of a system to discretize the parameter space more efficiently. Using this parameterization, we can reduce the interpolation error with the same parameterization accuracy (number of supporting points).

Representation of Phase Behaviour in Compositional Space

We show the application of tie-line parameterization for a three-component isothermal system with two hydrocarbon phases for the sake of simplicity. However, similar parameterization techniques can be extended to a multi-dimensional parameter space with multiple phases [11, 96]. For each pressure within the interval of interest, we construct a ternary diagram for phase behaviour representation in compositional space.

An example of such representation is shown in Figure 6.1. Here, the one-phase region is shown in green and the two-phase region in red. A tie-line is a key concept in the thermodynamical description of a multiphase multi-component mixture at equilibrium assumptions. It is a line within the two-phase region between a bubble point B_i and a dew point D_i with equal compositions of liquid and vapour phases. Along with this line, overall compositions $\mathbf{z}$ and phase saturations $\mathbf{S}$ keep changing, but molar fractions of components within phases $\mathbf{x}$ remain constant.

All tie-lines can be extended through the one-phase region to the sides of the diagram covering the sub-critical region $C_1 L_{cr} R_{cr} C_2$ (see Figure 6.1). If a critical point z_{cr} exists for the system under given p, T, then the tie-line which passes through that point has zero length and is called a critical tie-line. The part of the one-phase region which is above the extended critical tie-line $L_{cr}R_{cr}$ is called super-critical region. The approach provides a separate parameterization treatment for these regions. If the critical point does not exist for given p, T then we assume that the sub-critical region covers the entire compositional space $[C_1, C_2, C_3]$.

Parameterization of Sub-Critical Region

We use an extension of tie-lines to parametrize the entire sub-critical region. First, we obtain the number of intermediate tie-lines between the critical $L_{cr}R_{cr}$ and the base (longest) C_1C_2 tie-lines based on the distance between their midpoints M_{cr}, M_0 and discretization parameter Δx:

$$n_{it} = \left\lceil \frac{|M_0 M_{cr}|}{\Delta x} \right\rceil. \tag{6.1}$$

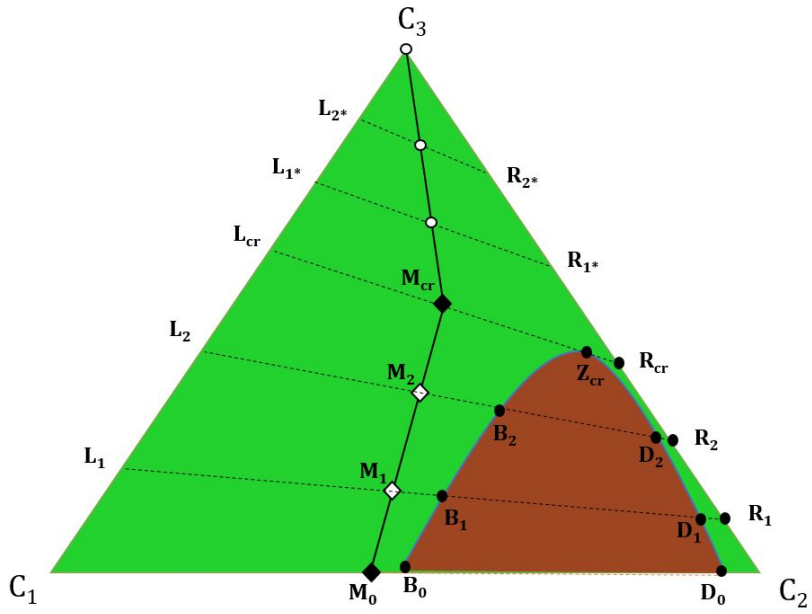


Figure 6.1: Tie-line based compositional space parameterization

Next, we split M_0M_{cr} into $n_{it}+1$ equal segments and get the midpoints M_i of intermediate extended tie-lines L_iR_i, where $i = 1 \ldots n_{it}$. Now, for every intermediate tie-line, we place supporting points at L_i, B_i, D_i, and R_i. Naturally, we also place them at $C_1, B_0, D_0, C_2, L_{cr}, Z_{cr}$, and R_{cr}. The segments L_iB_i, B_iD_i, and D_iR_i are evenly divided by the fixed number of supporting points into sub-segments, similarly to M_0M_{cr}. As the result, each sub-segment becomes shorter than Δx. Segments $C_1B_0, B_0D_0, D_0C_2, L_{cr}Z_{cr}$, and $Z_{cr}R_{cr}$ are treated in the same way.

Parameterization of Super-Critical Region

The super-critical region can not be parameterized using tie-lines, because they neither exist nor extend there. Instead, we apply a uniform parameterization with gridlines parallel to the critical tie-line. First, we determine the number of such lines:

$$n_{il} = \left\lceil \frac{|M_{cr}C_3|}{\Delta x} \right\rceil. \tag{6.2}$$

Then, we split each of the segments $L_{cr}C_3, R_{cr}C3$ into $n_{il}+1$ equal subsegments. Finally, we place supporting points at C_3, L_i^*, R_i^* and along segments $L_i^*R_i^*$, so that the segments become sliced into the equal subsegments shorter than Δx. Later, we test the tie-line-based parameterization against a uniform parameterization proposed in [43, 46].

Numerical Results

The quality of a parameterization approach can be assessed using the accuracy of the interpolator built on the parameterization. Previously, the piece-wise multi-linear interpolation was used, since the supporting points were evenly distributed over parameter space. This condition is no longer valid with the tie-line-based parameterization approach. Therefore, to compare two types of parameterization, we apply Delaunay triangulation to the set of supporting points and then use a simplex-based interpolation based on triangular simplices in both cases [11, 50].

The accuracy of an interpolator I can be estimated directly at any point of parameter space by computing the absolute difference between the interpolated value and the true value. A generalized error can be obtained by multiple application of this procedure at every point in set $\boldsymbol{\Omega}$, covering densely the entire parameter space. The error is computed at every point $i \in \boldsymbol{\Omega}$ of compositional space using the L_2 norm of the operator for all components:

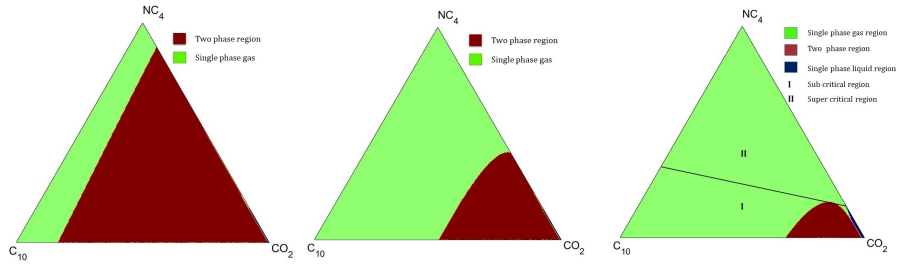
$$||E_\alpha||_i = \frac{\sqrt{\sum_{c=1}^{n_c} (I_{\alpha_c^D}(\boldsymbol{\omega}_i) - \alpha_c(\boldsymbol{\omega}_i))^2}}{\max_{j,c} |\alpha_c(\boldsymbol{\omega}_j)|}. \tag{6.3}$$

Here, $I_{\alpha_c^D}$ is the interpolant of operator α_c in discrete parameter-space $\boldsymbol{\Omega^D}$, c corresponds to the component and $\boldsymbol{\omega}_i$ corresponds to the state at i.

We demonstrate the results of the approach by modelling a fluid mixture of CO_2, NC_4, and C_{10} at three particular pressures of 20, 60, and 100 bar, while the temperature is fixed at 345 K.

Phase diagrams for the mixtures at all 3 pressures are shown in Figure 6.2. In Figure 6.2(a), the two-phase region occupies almost the entire parameter space. Figure 6.2(b) shows the phase behaviour at $p = 60$ bar. Here, the size of the two-phase region has reduced, but the extension of the two-phase region still parameterizes the entire compositional space.

A similar diagram was generated for $p = 100$ bar in Figure 6.2(c). Here, the two-phase and entire sub-critical regions occupy a smaller portion of the compositional space and a large portion of space is present in the critical region.



(a) p = 20 bar (b) p = 70 bar (c) p = 100 bar

Figure 6.2: Ternary diagram for the system CO_2, NC_4, C_{10} at T = 345 K

Figure 6.3 and Figure 6.4 show the norm of combined operators α and β at

different pressures respectively. These figures confirm that the thermodynamic properties of the system dictate the behaviour of accumulative and flux operators. The border between the one-phase and two-phase regions accounts for the most abrupt changes in operator values. With growing pressure, the two-phase region shrinks and the nonlinearity increases, especially for the convection operator β.

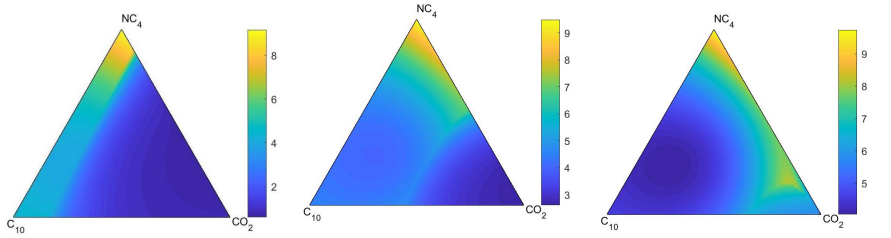


Figure 6.3: Euclidian norms of accumulation operator α at p = 20, 60 and 100 bar

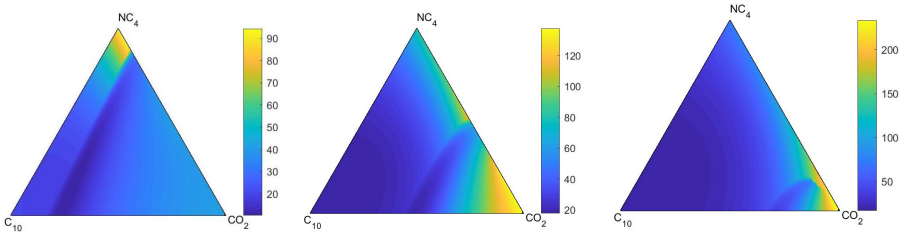


Figure 6.4: Euclidian norms of convection operator β at p = 20, 60 and 100 bar

For a meaningful comparison, the number of supporting points in uniform and adaptive parameterizations should be equivalent. This is easy to control in the uniform parameterization but more difficult in the adaptive version, since the number and density of supporting points depend on the form of the two-phase region. In the following comparison, we generate the adaptive parameterization first with fixed parameters. Next, we select the resolution of the uniform parameterization to match the number of points in the adaptive parameterization as close as possible. This approach lets us compare the errors consistently.

In Figure 6.5 and Figure 6.6, we demonstrate the normalized interpolation error $E_\alpha(\boldsymbol{\omega})$ for the operator α in case of adaptive and uniform parameterization respectively. The corresponding mesh is also shown for both types of parameterization. It can be clearly seen how the tie-line-based mesh adapts to the shape of the two-phase region. The number of points for adaptive parameterization at pressures p = 20, 60, and 100 bar is 46, 36, and 51 points respectively. To compare the errors, the resolution of uniform parameterization was tuned to generate 45, 36, and 55 points for these pressures respectively. The error maps show that the main error is concentrated at the boundary of the two-phase region and is more pronounced for the uniform parameterization. For the convection operator β, the error

in the adaptive parameterization is closer to the error in the uniform parameterization, see Figure 6.7 and Figure 6.8 for the adaptive and uniform parameterization respectively.

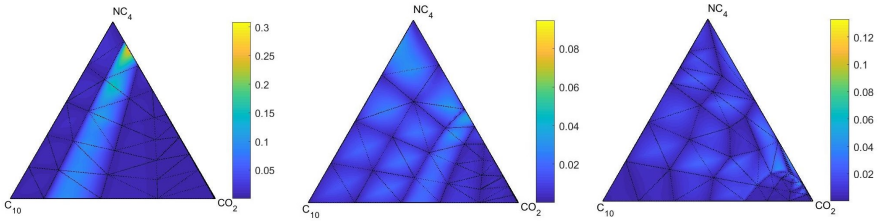


Figure 6.5: Errors for adaptive parameterization of accumulation operator α at p = 20, 60, and 100 bar

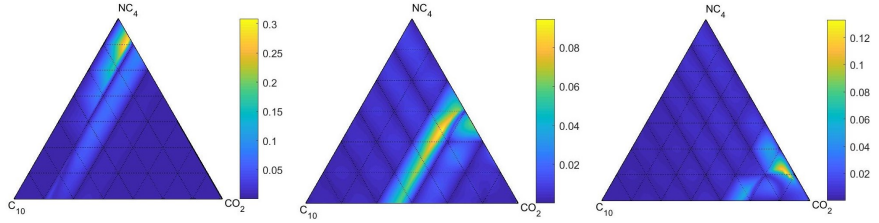


Figure 6.6: Errors for uniform parameterization of accumulation operator α at p = 20, 60, and 100 bar

The maximum errors of a combined operator are considered for sensitivity analysis as

$$\|E_\alpha\| = \max_i \|E_\alpha\|_i. \tag{6.4}$$

The variation of error with the increase in the number of supporting points is shown on the semi-log plot on the X-axis in Figure 6.9 and Figure 6.10. Five intervals are chosen between parameterizing distance $\Delta x = 0.1$ and $\Delta x = 0.01$. Since the nonlinearity is strongly correlated with the two-phase shape, the error behaves non-monotonically at the highest pressure for both parameterizations. However, in most cases, the error at lower resolutions for the adaptive mesh is smaller than the error for the uniform mesh. At lower Δx, the difference in errors between uniform and adaptive mesh reduces since the proximity between the supporting points increases.

6.1.2. Automatic Non-Uniform Parameterization

Compared to the physics-based parameterization approach developed for compositional simulation, described in Subsection 6.1.1, automatic non-uniform parameterization is a more general technique and can be applied to a physical model of any kind.

The key difference here is that the placement of supporting points does not require prior knowledge of underlying physics. The technique requires only the ability to compute operator values at any point in parameter space and aims to detect parameter space locations where the operators are changing the most. Once

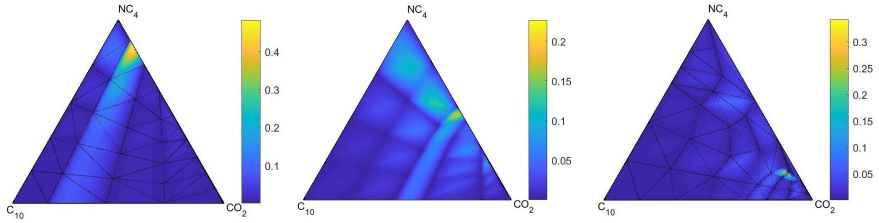


Figure 6.7: Errors for adaptive parameterization of convection operator β at p = 20, 60, and 100 bar

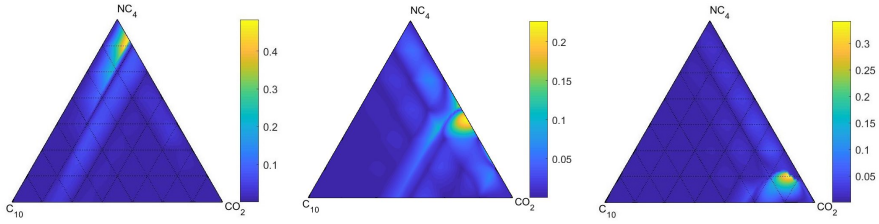


Figure 6.8: Errors for uniform parameterization of convection operator β at p = 20, 60, and 100 bar

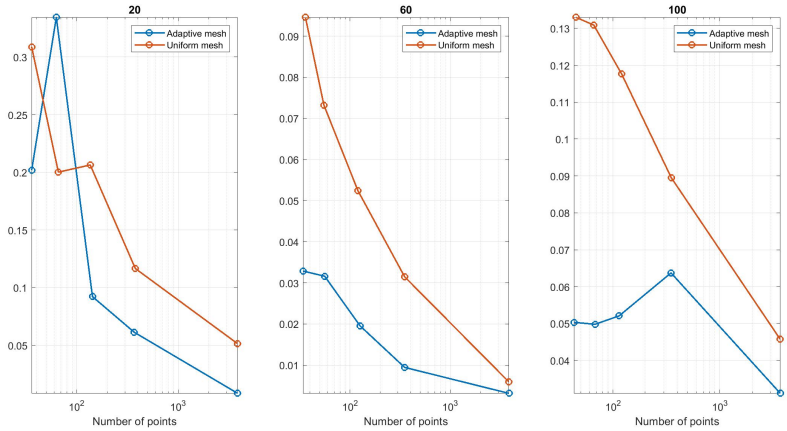


Figure 6.9: Mean error comparison for accumulation operator with various parameterization resolutions at p = 20, 60, and 100 bar

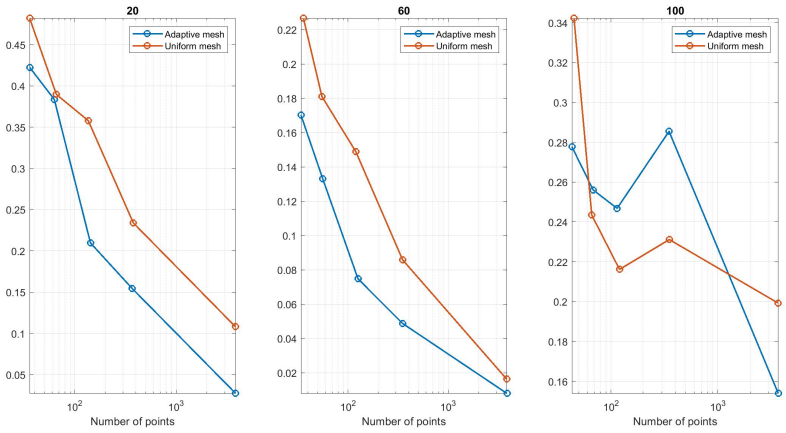


Figure 6.10: Mean error comparison for flux operator with various parameterization resolutions at p = 20, 60, and 100 bar

scattered supporting points are placed, similar to Subsection 6.1.1, triangulation is introduced and linear barycentric interpolator is built to approximate a continuous form of nonlinear operator. As a result, the error can be significantly reduced within the same number of parameterization points compared to uniform parameterization. Besides, simplex-based interpolation complexity is $O(D+1)$, which makes it highly attractive for highly-dimensional problems.

The suggested automatic parameterization approach requires three stages:

1. Initialization
2. Structuring
3. Enrichment

The first step accounts for the definition of boundaries in parameter space. The corner locations of the parameterized region in parameter space constitute the initial set of supporting points. The second step adds the locations corresponding to operator extrema, forming a coarse set of supporting points. The third step adds a limited amount of supporting points targeting to reduce approximation error as much as possible. It can be performed multiple times until the parameterization is enriched enough to meet the desired accuracy.

Several implementations of the suggested algorithm were developed and compared. The flux operator for decane in a binary compositional mixture of CO_2 and N_{10} was employed as a function to be parametrized and approximated. Setting constant temperature $T = 350$ K allowed to have only two degrees of freedom – pressure and overall CO_2 composition. This choice was made for simplicity and better visualization purposes. Pressure was bounded in a range of [30; 150] bar to include both single-phase and two-phase regions. Composition is naturally bounded

by the [0; 1] interval. Therefore, the supporting point set is initialized with {(0, 30), (0, 150), (1, 30), (1, 150)}.

Structuring Stage

A local constrained optimization algorithm can be used to detect extrema of a general nonlinear function. In particular, a sequential least squares programming (SLSQP) algorithm, based on the Han–Powell quasi-Newton method, was used because of its general robustness [97]. Local optimization algorithms converge to a local extremum in a reasonable time, starting from a certain initial guess and following the steepest descent (ascent) based on the local gradient. On the other hand, there is no guarantee that the local extremum coincides with the global one. Therefore, to make the algorithm more robust, several optimization processes can be launched with different initial guesses. All local extrema which are found in such process, even though they may not coincide with the global extrema, are still important for accurate approximation.

Figure 6.11 illustrates the results of the described process with a variable amount of initial guesses. The flux operator for decane in mixture of CO_2 and N_{10} shapes a surface with a quite nonlinear behaviour (Figure 6.11 **A-E**). The orange points correspond to locations where supporting points are placed. All subfigures contain such points at the corner locations, coming from the initialization stage. Subfigure **A** has one additional point, resulting from the local optimization process with a single initial guess at the centre of parameter subspace under consideration. Subfigure **B** has more points coming from 4 (2x2) initial guesses, uniformly scattered over the parameter subspace. Similarly, **C-E**, illustrate the results of the structuring stage for 3x3, 4x4 and 5x5 uniformly scattered initial guesses.

Note that some optimization processes may fail or converge to very close positions, therefore the total number of supporting points can be smaller than the number of initial guesses. It is easy to see that 4x4 and 5x5 schemes result in too many local extrema (caused by the saw-tooth shape of the surface on the edge of the cliff), therefore the 3x3 scheme was chosen to proceed with. Sometimes, when the underlying function is monotonous, the entire structuring stage can be wasteful, which is illustrated by subfigure **F**.

Enrichment Stage

Upon completion of the structuring stage, the coarsest parameterization is obtained. To increase its accuracy, additional supporting points have to be added. This is done in an iterative way, in which one or more points are added at a time. One of the ways to process this enrichment is to continue using the optimizer, but apply it to the absolute approximation error. Since it computes the absolute difference between the accurate and approximated operator values, its local maxima represent the locations where additional points are needed the most. This process is illustrated in Figure 6.12.

Addressing Boundaries

As experience has shown, the chosen optimizer was not performing well enough along the boundaries. This resulted in the relatively inaccurate approximation of

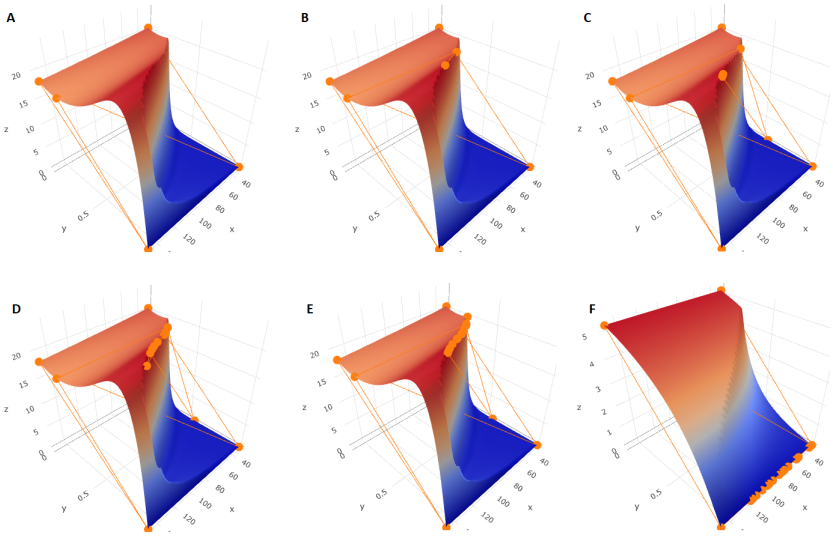


Figure 6.11: **A-E:** Flux operator for decane in mixture of CO_2 and N_{10} and detected extrema points with 1x1 (**A**), 2x2 (**B**), 3x3 (**C**), 4x4 (**D**) and 5x5 (**E**) initial guesses, **F:** accumulation operator for decane in mixture of CO_2 and N_{10} and detected extrema points with 5x5 initial guesses

an operator on the edges of the parameter subspace under consideration. Figure 6.13a, Figure 6.13c demonstrate the issue: only a few supporting points are placed along the border leading to a large error.

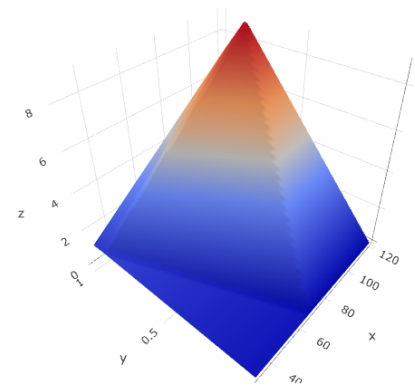
To address the issue, a separate optimization process was launched at every boundary. Since every boundary has one of the degrees of freedom fixed, such an optimization problem will be less expensive than the main optimization procedure. On the other hand, the number of boundary searches will grow for problems with more degrees of freedom.

Therefore, every iteration of the enrichment stage specifically addresses the edges of the parameter subspace under consideration. It includes several optimization problems with various dimensions and boundaries. As a result, the number of supporting points and consequently the approximation quality significantly improves along the borders, as Figure 6.13b, Figure 6.13d confirm.

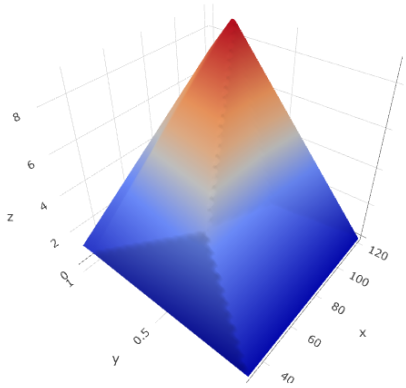
Comparison Against Uniform Parameterization

Here, the quality of the automatic non-uniform parameterization is compared to a uniform one. The mean and maximum errors are considered. Each of these is found using the difference between approximated and actual operator values at every point of a very dense uniform grid over the parameter subspace of interest. Two different operators with highly nonlinear behaviour, displayed in Figure 6.14, were used for comparison.

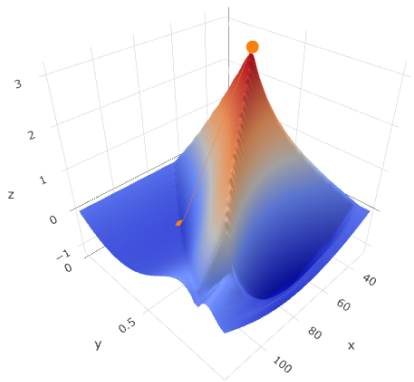
Figure 6.15 shows the mean error comparison for 16, 64, 256, 1024, and 4096 supporting points between the non-uniform (denoted as ”Adaptive grid”) and uni-



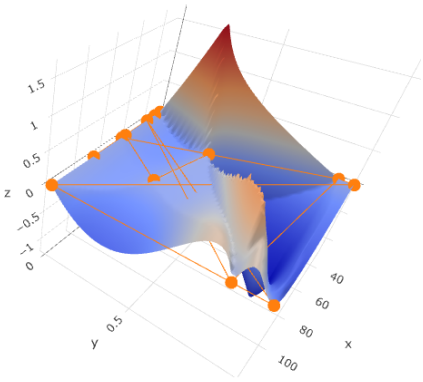
(a) Interpolated function before enrichment stage



(b) Interpolated function after enrichment stage



(c) Approximation error before enrichment stage. Detected minimum and maximum locations are shown



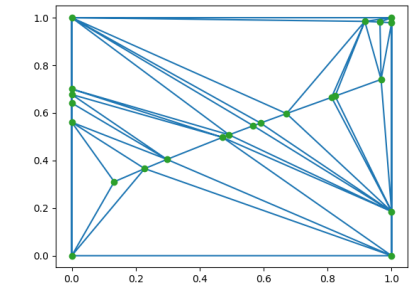
(d) Approximation error after single iteration of enrichment stage. Locations of all supporting points are shown

Figure 6.12: Approximated function and corresponding error before and after single iteration of enrichment stage for accumulation operator for carbon dioxide in mixture of CO_2 and N_{10}

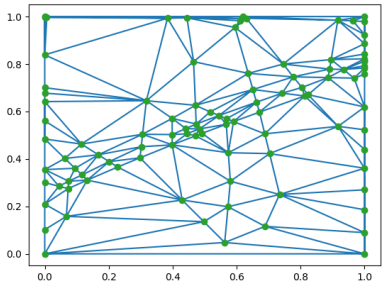
form (denoted as "Regular grid"). Note, that the mean error axis has a log scale. It is easy to see that automatic parameterization provides significantly better parameterization accuracy on average for both operators. Furthermore, the difference between the approaches increases as the parameterization resolution grows.

The standard deviation of the error, shown in Figure 6.16, characterizes its spread. It is clear, that the errors for non-uniform parameterization are mostly closer to the mean error than those for the uniform one.

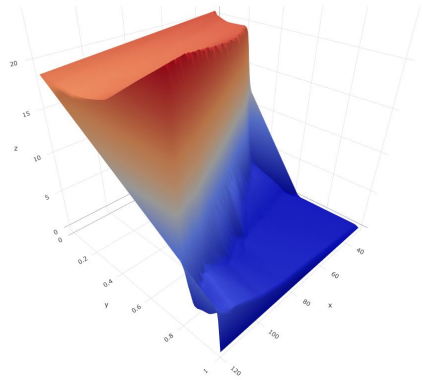
However, the comparison of the maximum errors for the approaches, provided



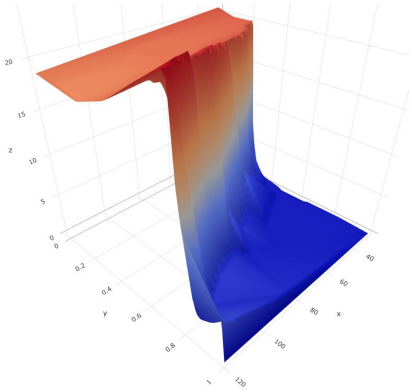
(a) Delaunay triangulation after complementation stage



(b) Delaunay triangulation after enrichment stage including boundaries treatment



(c) Approximated operator after complementation stage



(d) Approximated operator after enrichment stage including boundaries treatment

Figure 6.13: Delaunay triangulation and corresponding approximated operator after different stages of parameterization

by Figure 6.17, shows that the non-uniform parameterization can induce a slightly higher approximation error. The approximation errors for both parameterizations for one of the cases are shown in Figure 6.18. Not only it confirms the conclusions made from Figure 6.15 - Figure 6.17, but also indicates that high values of maximum errors for non-uniform parameterization are caused by the nature of local optimization algorithms employed. Given that the area around high peaks is mostly flat, as Figure 6.18 shows, it is complicated for a gradient-based SLSQP optimizer to spot them.

6.2. Proxy Models in Physics

Proxy modelling is widely used in practice to obtain the best possible prediction when time or/and computational resources are limited. For example, simplified

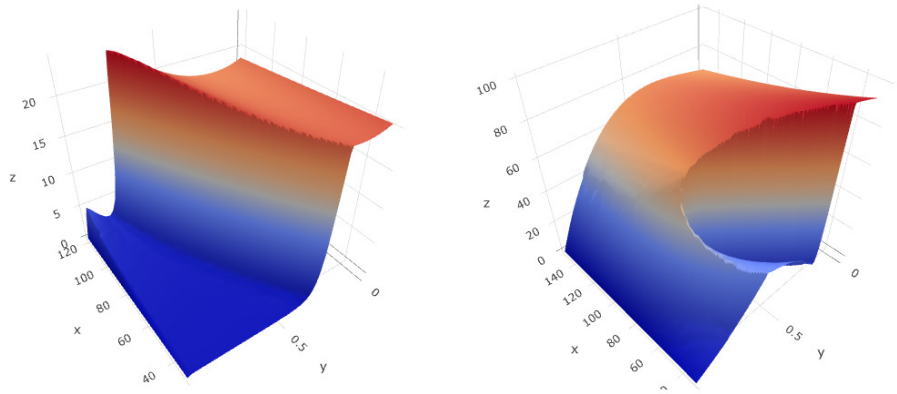


Figure 6.14: Two operator functions used for comparison

Figure 6.15: The mean error comparison between automatic non-uniform and uniform parameterization at different parameterization resolutions for both operators

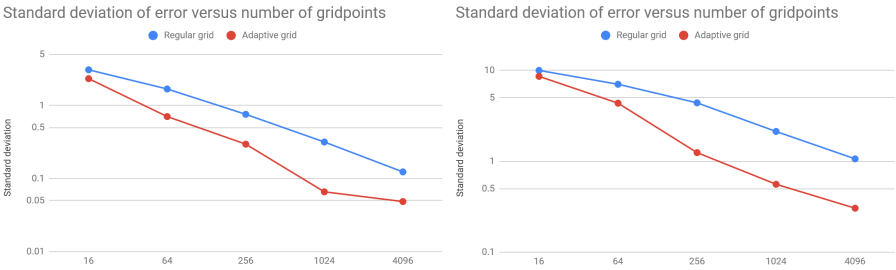


Figure 6.16: Comparison of standard deviation of approximation errors between automatic non-uniform and uniform parameterization at different parameterization resolutions for both operators

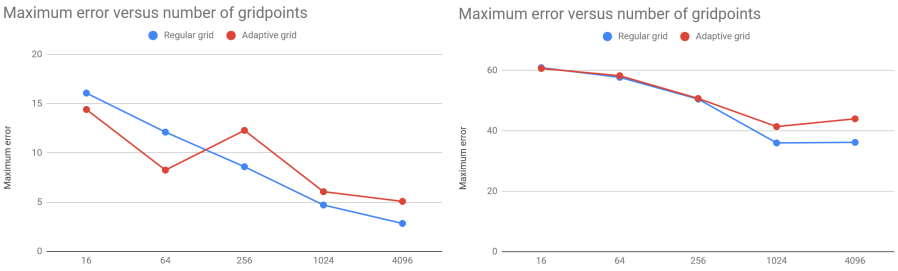
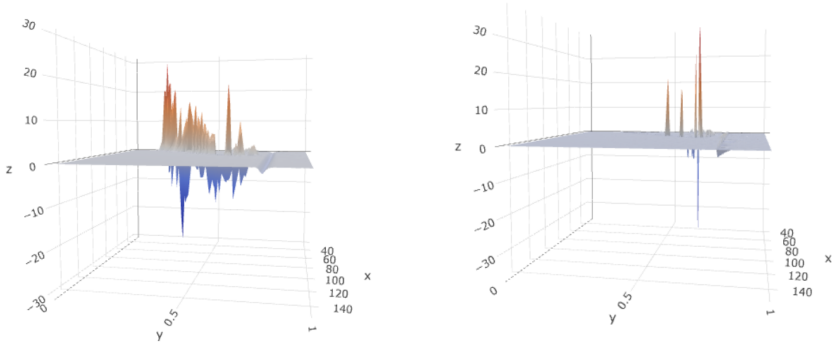


Figure 6.17: Comparison of maximum approximation error between automatic non-uniform and uniform parameterization at different parameterization resolutions for both operators



(a) Approximation error the uniform parameterization

(b) Approximation error the non-uniform parameterization

Figure 6.18: Approximation errors for uniform and non-uniform parameterizations of the second operator for 1024 points

models are useful for global optimization in various scenarios of reservoir production/development, or decision support in real-time management of field production. Here, we describe how the DARTS framework can be straighforwardly utilized for the development of proxy models in physics.

Several efforts have been made to improve the performance of compositional reservoir simulators by enhancing phase-behaviour computations [98–100], spatial coarsening of compositional models [101, 102] or reformulation of the compositional nonlinear problem [103]. While the improvement of phase behaviour computations usually does not introduce errors in results, its influence on the total computational time is limited. The biggest improvement is usually achieved by the coarsening in spatial representation (upscaling) since it can significantly reduce the number of unknowns in the linear system of equations. However, upscaling always introduces an error in computational results.

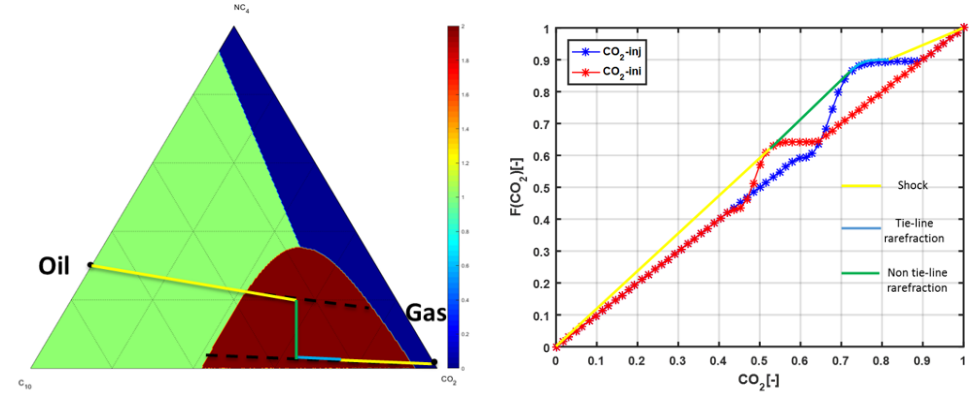
As an alternative to the upscaling, the Algebraic Multi-Scale (AMS) approach

was initially proposed to solve an elliptic flow problem by [104]. Several extensions of this method have been successfully developed [105–107]. However, most of the AMS methods were focused exclusively on the flow solver and did not address the transport problem, except [108], where an adaptive Multiscale Finite Volume Method was proposed to accelerate the transport solver. Based on these ideas, a Multi-Scale Compositional Transport (MSCT) method for reconstruction of the compositional transport problem with an arbitrary number of components was developed in [109].

This approach suggests a two-stage reconstruction, where at the first stage, the boundary of a two-phase region is recovered, while the detailed solution in the two-phase region is reconstructed in the second stage. MSCT utilizes the OBL technique proposed by [43] and is implemented within the DARTS framework.

6.2.1. Multi-Scale Compositional Transport

The solution of a compositional transport problem can be shown in a phase diagram by the solution path in compositional space. Such a path defines the compositional changes between the initial and injection mixtures. Conservation principles and fractional-flow theory form the foundation for the general solution method [110]. The compositional path of a conventional gas injection problem where single phase gas is injected into single-phase oil always results in two shocks (leading and trailing shocks) between the single- and two-phase regions. In a ternary diagram (Figure 6.19a), it is presented as yellow lines connecting the initial oil and injected gas composition.



(a) (b)

Figure 6.19: Gas-injection solution in ternary system: (a) ternary diagram with displacement path and two key tie-lines and (b) fractional-flow curves for component CO_2 with solution path

The shocks between single- and two-phase regions are always aligned along two key tie-lines (black dashed lines) defined by liquid x_i and vapor y_i fractions of each component. For a fixed pressure, x_i and y_i remain constant and it is possible to construct the fractional-flow curve corresponding with compositional transport,

see Equation 6.5. Figure 6.19b gives the injection and initial fractional-flow curves for CO_2 in a ternary system corresponding to the injection and initial tie lines in Figure 6.19a.

$$F_i = x_i\left(1 - f_g\right) + y_i f_g, \quad i = 1, ..., n_c - 1 \tag{6.5}$$

The proposed Multi-Scale Compositional Transport approach consists of two stages [111]. The first stage utilizes a set of restriction-prolongation operators for reconstructing two-phase boundaries (the trailing and leading shocks). The restriction here reduces the $n_c - 1$ transport equations to a single equation with a special flux operator based on the pseudo-fractional-flow curve. Once the restriction solution is obtained, a simple interpolation-based prolongation operator is applied to reconstruct the solution in the single-phase regions.

In the second stage, the set of restriction-prolongation operators is applied in the two-phase region to reconstruct the solution structure of the two-phase displacement. This stage is based on the invariance of two-phase solutions in tie-line space reported in [112] and adapted for practice in [113].

The proxy model for compositional simulation, utilized in this work, uses the first-stage multi-scale reconstruction from [111]. A restriction operator combines two fractional-flow curves for injection and initial tie-lines, defined as:

$$F_l^{ini} = x_l^{ini}(1 - f_g) + y_l^{ini} f_g, \quad F_l^{inj} = x_l^{inj}(1 - f_g) + y_l^{inj} f_g. \tag{6.6}$$

The equivalent fractional-flow curve, serving as the restriction operator, is constructed by taking a convex hull on the union of both curves:

$$F_R = \text{conv}(F_l^{inj} \cup F_l^{ini}) \tag{6.7}$$

In Figure 6.20, this curve is shown in green. Next, the equivalent values of F_i and z_i from the green curve are tabulated into the restriction operator and the reduced system is solved. The reduced system of equations includes the conventional pressure equation and the restricted transport equation based on the constructed pseudo-fractional-flow curve. In structure, this system is very close to the conventional binary compositional problem.

Figure 6.21 gives an example of the operators which are tabulated from the analytical fractional flow curve. Those operators are utilized in the OBL framework [46] to solve the first-stage restricted system.

Once the solution of the restricted system is found, the full system is reconstructed based on the prolongation operator. This operator applies interpolation between initial and injection compositions using the solution of the restricted system $\boldsymbol{\kappa}(z_R)$ as an indicator:

$$\boldsymbol{\kappa}(z_R)\left[\mathbb{R}^1 \Longrightarrow \mathbb{R}^{n_c - 1}\right] : \mathbf{z} = \mathbf{I}_{\{z^{ini}, z^{inj}\}}(z_R). \tag{6.8}$$

Here, $\boldsymbol{\kappa}$ is the interpolation-prolongation operator, z_R is the restricted solution and $\mathbf{I}$ is the piecewise linear interpolation function. Referring to this linear interpolation, the transport solution of other components in the multi-component system is reconstructed and used as a proxy model in place of the full compositional model. Notice

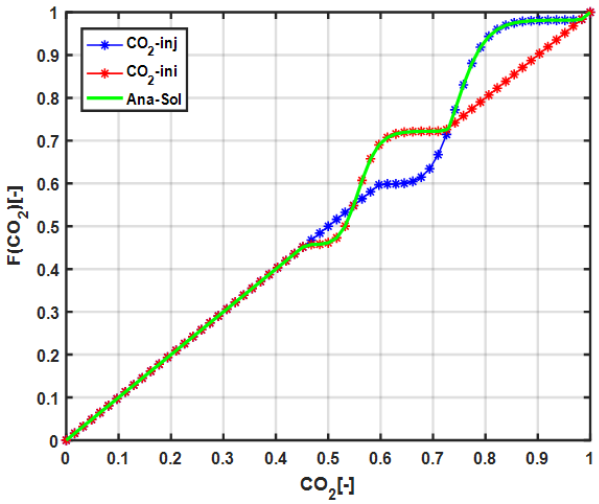
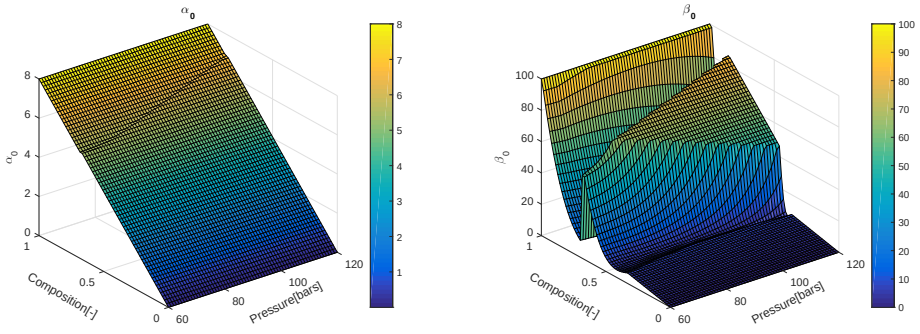


Figure 6.20: Analytical fractional flow for CO_2



(a) (b)

Figure 6.21: Operators for a restricted compositional system parameterized at N=64

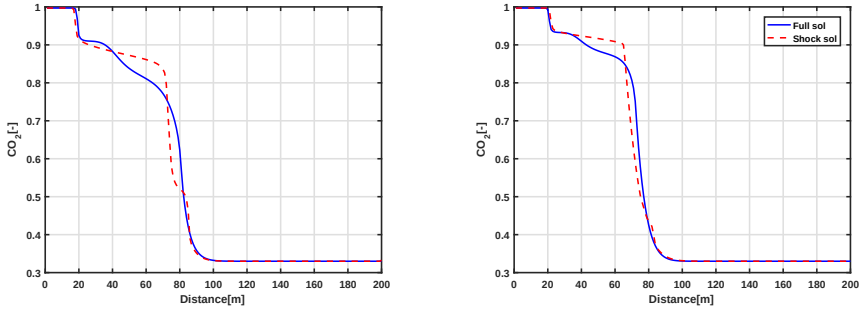
that this system can accurately predict only the boundaries of the two-phase region and their dynamic propagation in space; for an accurate solution, the second-stage multi-scale reconstruction should be applied [111].

6.2.2. Restricted Solution

Next, the comparison between solutions of a full compositional model and a corresponding proxy model is demonstrated. Here, we limit our investigation to a conceptual 1D reservoir model for simplicity. In this model, the injection well on the left operates at a constant gas rate when the production well on the right is controlled by Bottom-Hole Pressure (BHP).

Figure 6.22 shows the restricted solution z_R, which yields the shock reconstruction curves for simulation results for the growing BHP at the production well. All simulation results are shown for the model with parameters after 1000 days of simulation. The K-value table in this work is obtained from the embedded Constant Composition Expansion (CCE) experiments in [114] based on the PR EoS. The K-value system does not develop miscibility even when BHP provides the pressure at the displacement front close to the First-Contact Minimal Miscibility Pressure (FC MMP) for this system (around 126 bar at T = 373 K). This happens due to the inability of the K-value model to predict miscibility accurately since compositional dependency is not captured in this model.

It can be overcome by either extension of the K-value parameterization with additional degrees of freedom [e.g. 115] or incorporation of EoS-based phase behaviour [113]. However, the two-phase boundaries can be accurately represented by the restricted model for K-value based physics. Besides, the complexity and structure of the restricted solution are invariant to the number of components present and only depends on initial and injection tie-lines in the multi-component system [see 111, for details].



(a) Restricted solution for BHP = 85 bar

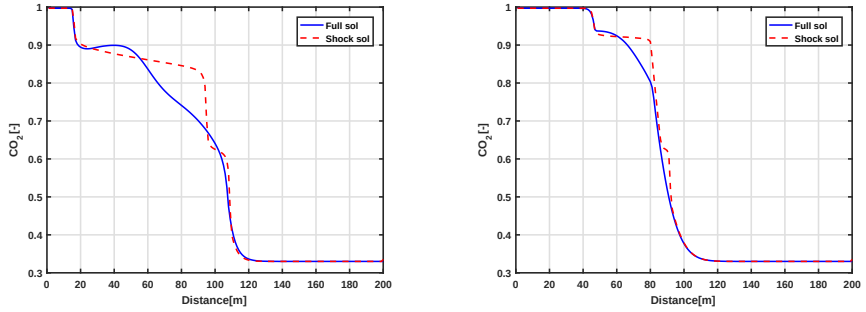
(b) Restricted solution for BHP= 120 bar

Figure 6.22: Shock reconstruction of the four-component system for two different BHP controls at production well (K-values)

Next, the results of the restricted solution for the compositional problem based on the EOS is shown. The structure of the compositional transport solution depends on key tie-lines [110]. For the restricted solution, we follow the same strategy as before and construct the restriction operator based on combined fractional flow (Equation 6.6) according to the first stage of the MSCT approach [111]. The solution of the restricted transport equation reconstructs the boundaries of the two-phase region using one transport equation instead of $n_c - 1$ equations in the conventional compositional model.

The results of quaternary system reconstruction are shown in Figure 6.23. Here, one can see that for a high BHP value, the structure of the solution is much closer to miscibility (leading and trailing shocks stay closer to each other) than in the K-value approximation. This happens because the EOS-based phase behaviour correctly represents the compositional dependence of the solution. Similar to the K-value

model, the leading and trailing shocks are accurately reconstructed by the proxy model.



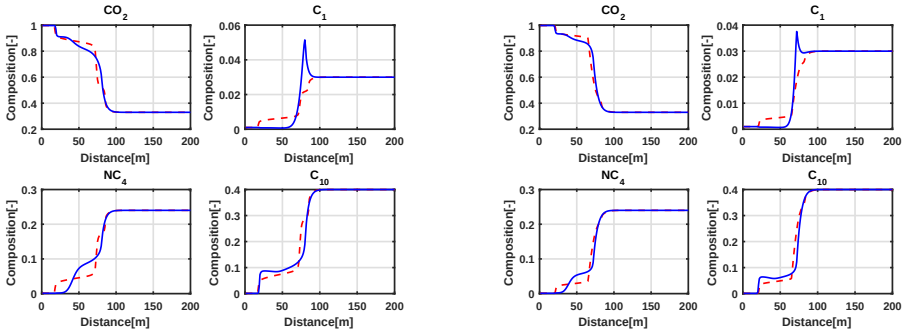
(a) Restricted solution for BHP = 85 bar

(b) Restricted solution for BHP = 120 bar

Figure 6.23: Shock reconstruction of the four-component system for two different BHP controls at production well (EoS model)

Prolongation of Proxy Model

Now, we illustrate the construction of the proxy model using an interpolation-based prolongation operator (Equation 6.8) for both cases. It can be seen in Figure 6.24 and Figure 6.25 that the restriction stage does not reconstruct the full structure of the solution, but only one indicator component. For the full solution, the prolongation stage should be applied (see [111] for details).

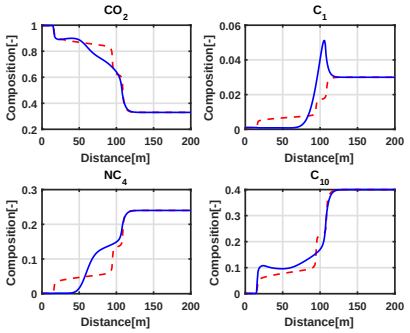


(a) Full solution for BHP = 85 bar

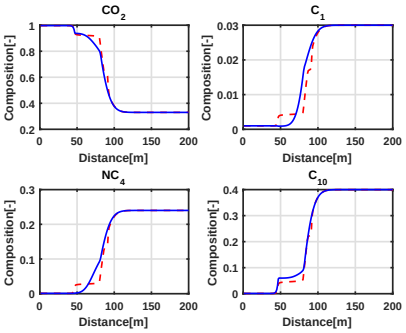
(b) Full solution for BHP = 120 bar

Figure 6.24: Proxy model for a four-component system (K-value based)

The prolongation stage yields a full compositional solution in every control volume, which then can be used in a multiphase flash procedure to predict phase behaviour. This phase behaviour provides the boundary of the two-phase region in space. This prediction can be used to compute phase rates at wells and evaluate the net present value (NPV) for a proxy model. The comparison of NPV for one and



(a) Full solution for BHP = 85 bar



(b) Full solution for BHP = 120 bar

Figure 6.25: Proxy model for a four-component system (EoS based)

two controls are shown in Figure 6.26 and Figure 6.27. The details on economic parameters can be found in [95].

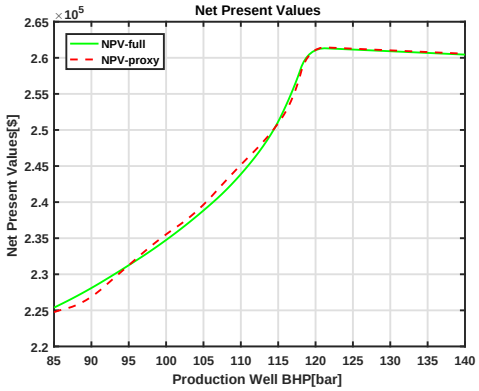
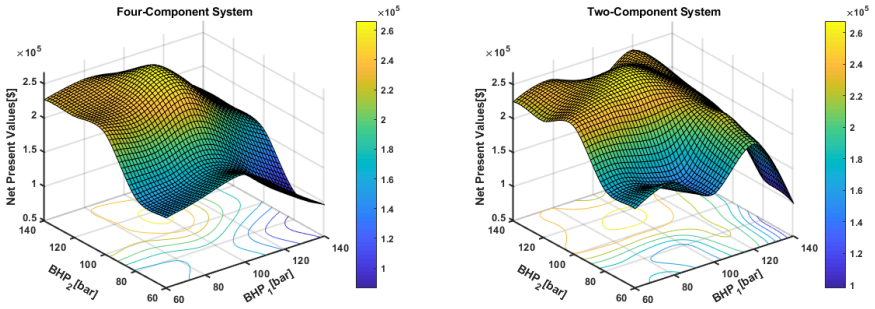


Figure 6.26: Comparison of NPV with one control parameter obtained from full and proxy models



(a) Full model (b) Proxy model

Figure 6.27: Comparison of NPV with two control parameters obtained from full and proxy models

7

Recapitulation and Conclusions

7.1. Operator-Based Linearization

A new linearization method for multiphase thermal-compositional fluid flow and transport in the subsurface with two-point flux approximation (TPFA) and Fully Implicit (FI) time approximation is established in Chapter 2. In this approach, the governing equations of a general purpose reservoir simulation problem are represented in the operator form where each term is a product of two types of operators. The first type of operators is fully defined by the physical state of the problem, while the second is characterized by spatial and temporal discretization. Mass conservation equations in operator form are characterized by component mass accumulation and flux state-dependent operators.

To perform the linearization of the governing equations, we introduced a uniform parameterization in the space of physical unknowns. Each state-dependent operator is evaluated at supporting points of the parameter space. This defines discretization in the physical description of fluid and rock. Piecewise multilinear interpolation is applied to compute both values and partial derivatives of state-dependent operators based on the created parameterization. Once that is done, the linearization of the governing equations in simplified operator form is completed in a conventional manner using analytical derivatives by chain rule between derivatives of two operator types.

Adaptive parameterization in the discretized thermodynamic space is developed to address the performance limitations of accurate parameterization for high-dimensional problems. Because of the near-hyperbolic behaviour of several unknowns in the nonlinear solution (i.e., saturation or overall molar composition), only a limited amount of supporting points in parameter space is usually used in a simulation. Therefore, operator values at every new supporting point, required by the simulation process, can be computed adaptively on-the-fly and then stored for

reuse during the same or a subsequent simulation.

We demonstrated the applicability of the OBL approach to general purpose reservoir simulation problems. In particular, we applied different physical kernels that include black-oil, isothermal compositional kernels with 4 and 6 components. We showed that the OBL approach reproduces the results of the reference solution at any reasonable resolution with insignificant errors, localized at the displacement front. As a rule of thumb, a resolution of 64 uniformly distributed points along each of the parameter space axes within the required range is sufficient for an accurate representation of fluid and rock properties. On the other hand, the limited coarsening of parameter space improves the nonlinear convergence in most cases.

The performance of the OBL approach benefits from the simplified assembly of the Jacobian of the simulation problem and an almost complete bypass of phase behaviour calculations (except for supporting points). Compared to linearization based on Automatic Differentiation (AD), the new approach is relieved from the computational overhead related to augmented algebra computations, while providing almost the same level of flexibility for the extension of the physical model in a simulation framework.

Proving the last point, Chapter 3 introduces an extension of the OBL approach to account for thermal effects. It demonstrates the applicability of the approach to the simulation of thermal-compositional multiphase flow in porous media. In addition to mass conservation, the energy conservation equation was also transformed to the operator form, forming energy accumulation, convection, and conduction state-dependent operators. Initial porosity was chosen to enrich the vector of state variables to reduce the number of state-dependent operators. However, the dimensionality of the parameter space for energy accumulation and conduction operators is consequently increased by one. A similar approach can be used to handle the changes in the mass of solid phase(s) due to the chemical precipitation and dissolution.

To test the geothermal application, we used a realistic model of a channelized system of the Delft Sandstone member (DSSM), situated at the West Netherlands Basin. Simulation results showed that the proposed approach reproduces the reference solution results quite accurately with a reasonable parameterization resolution. For a single-component low-enthalpy geothermal model, a relatively coarse resolution of interpolation tables can handle all governing nonlinearities and matches the reference solution based on full physics precisely. For a two-component geothermal model with natural gas co-production, the required resolution of interpolation tables is higher. This happens due to the highly nonlinear nature of linearization operators in case of two-phase systems. However, the simulation with a coarser resolution still can be used as fast proxy models in the inversion and uncertainty quantification process.

It is important to notice that the proposed linearization approach significantly improves the performance of the AD-based linearization. The relative cost of OBL does not grow significantly with the increased resolution while the number of nonlinear iteration is decreasing with coarser representation. For all observer cases, the parameterization resolution of 64 points provided a sufficient level of accuracy,

keeping the average error below 1%.

In addition, Chapter 3 describes the buoyancy extension of OBL. We introduced three types of upwinding based on phase potential, component potential, and independent treatment. The first type can be seen as the conventional approach, where its OBL representation implies a substantial increase in the number of flux operators compared to the case without the buoyancy effect. The two other types attempt to simplify the conventional description and decrease the number of flux operators while maintaining the accuracy of the solution. The developed approaches were validated on dead-oil and compositional simplified gravity segregation models. Besides, we investigated the sensitivity of the OBL solution accuracy and nonlinear solver convergence to the parameterization resolution for a realistic model of the Brugge field with buoyancy.

In overall, Chapter 2 and Chapter 3 confirm the applicability of the OBL approach to general purpose reservoir simulation based on thermal-compositional description. It is shown that coarsening of the physical description of rock and fluid introduces an additional trade-off between the accuracy of numerical simulation and the performance of the nonlinear solver. Similarly to the discretization in space and time, the OBL description can be coarsened, loosing certain accuracy in favor of simulation performance to speed up optimization, uncertainty quantification, or inverse modelling.

7.2. Delft Advanced Research Terra Simulator

While the entire concept was matured and improved, OBL has had a few prototype implementations described in Chapter 4. The first implementation was performed in the AD-GPRS framework. It was a natural choice: in the beginning, it was vital to validate the applicability of the new linearization method to a wide range of reservoir simulation problems. Once that was done, it became clear that OBL provides a significant performance advantage (by 15-30%) over the conventional linearization despite non-ideal implementation from the performance point. In the original implementation, AD-based storages were abused by artificial injection of externally computed gradients. Therefore, to estimate the true performance capabilities of the method, a stand-alone implementation was needed.

As a first attempt, a MATLAB-based prototype was coded. The parameterization information was generated in AD-GPRS, exported to a set of text files and then imported to the simulator. The entire code was under 1000 lines, but capable to solve one-dimensional isothermal compositional flow and transport problem in a fully implicit manner. This development confirmed the efficiency of the concept: decoupling physical properties from the main simulator core allows to simplify and generalize Jacobian assembly and make it portable to alternative computational architectures.

Then, a standalone high-performance prototype of a compositional simulator based on C++\CUDA was developed for both CPU and GPU computing architectures. Note, that the GPU version executes the entire simulation loop on GPU. The prototype supports general unstructured grids via connection lists, hence one-, two-, and three-dimensional reservoirs are supported. To perform a fair comparison,

initialization, nonlinear and linear solvers were aligned with AD-GPRS. Benchmarks showed that the single-threaded CPU version performs the Jacobian construction up to 19x times faster. The GPU version of the prototype boosts the linearization by a factor of 260x. Careful tuning of GPU kernels and the use of multi-GPU systems can improve the performance even further.

Finally, the Delft Advanced Research Terra Simulator (DARTS) was introduced and described in Chapter 5. It combined the experience and knowledge obtained during previous iterations of OBL implementation. Having kept all performance-critical parts of the simulator core in C++, DARTS exploits physical description decoupling to the full extent, providing a Python-based plugin interface to customize fluid and rock properties. Since property computations are performed for a limited amount of discrete locations (e.g., supporting points), the computational performance requirements for such plugins are relaxed. Moreover, the results of physics parameterization can be simply saved and reused in subsequent simulations when uncertainty quantification, inversion modelling or production optimization are performed. This allows introducing advanced thermodynamic and chemical equilibrium computations based on complex software libraries, and couple them with flow and transport in a fully implicit manner without sacrificing performance.

Parameterization, evaluation of operator values and their gradients are controlled by DARTS interpolators. Uniform parameterization with piecewise multilinear interpolation can be either performed statically (i.e., all supporting points are pre-computed) or adaptively (on-the-fly along the course of a simulation). Adaptive parameterization allows to achieve extreme resolutions in parameter space discretization (e.g., 4.3×10^{12} supporting points for 3-dimensional space) and still provide reasonable simulation time. Alternatively, the entire interpolator can also be seamlessly replaced by AD-based operator evaluator. Then, DARTS would support the conventional treatment of the rock and fluid properties description and provide an exact reference solution.

The Jacobian assembly of the simplified operator form of the governing equations is taken care of by DARTS engines. Depending on the desired amount of components, primary variables, and physical effects to be taken into account, one or another engine is chosen. Multisegment wells are introduced in DARTS via the OBL approach and contribute to the Jacobian assembly in a way which keeps the blocked structure of the matrix. Once the Jacobian is assembled, it is passed over to the linear solver, which includes a CPR-based preconditioner, de-facto standard for a fully implicit scheme. The major computational load of the simulation process is delegated to a relatively simple engine object and linear solver. Since various linear solvers become available on different computing architectures such as a GPU (e.g., see [116]), the whole simulation can be executed there, while the calculations of operator values can still be performed on a CPU.

Petroleum and geothermal industries constantly exert pressure on reservoir simulation for more rigorous models to improve accuracy and concurrently demands faster turnaround times to speed up history-matching and uncertainty quantification. While the traditional CPU architectures are currently limited in providing consistent acceleration to reservoir simulation codes, the GPU architectures evolve

rapidly and promise to take the initiative. Hence, aside from the purely algorithmic aspect of simulation performance, it is essentially important to take into account the efficient implementation of the chosen methods in terms of both software and hardware. DARTS demonstrates how the architecture of a reservoir simulator can reveal the performance potential of OBL in three independent dimensions: improved nonlinear performance (algorithmic level); actual performance of linearization stage (software level); portability to alternative computing architectures including a GPU (hardware level).

7.3. DARTS Applications

Chapter 6 discusses current DARTS applications and future developments. It starts with the investigation of two different approaches of non-uniform parameterization of physical space. The first is designed specifically for compositional formulation. It uses prior knowledge about the shape of state operators - phase envelope, and therefore relatively cheap. The second is more expensive since it directly detects the most problematic regions requiring refinement, using local optimization algorithms. Both methods confirm a significant increase in parameterization accuracy compared to uniform parameterization with a similar amount of supporting points in most cases. Proper implementation of non-uniform parameterization approaches coupled with the simplex-based interpolation will allow evaluating the resulting performance of this alternative.

Proxy models in physics built within DARTS and their applications are also discussed in Chapter 6. Usually, proxy models provide significant improvement in simulation performance by introducing various simplifications into full-physics models. Multi-Scale Compositional Transport (MSCT) simplifies the compositional description of a multi-component system with a specially built binary system. This allows reducing the size of the corresponding linear system by $\frac{n_c}{2}$ times. The resulting proxy model is straightforwardly constructed within DARTS simply by substituting restricted fractional flow curves into operators. It can accurately predict leading and trailing shocks, which is enough for judgement of miscibility development. Consequently, cheap yet accurate NPV estimation can be constructed and used for production optimization based on a proxy model.

Employment of DARTS as a workhorse in production optimization and inverse modelling is especially attractive because of the minimal input/output overhead provided by the Python integration. It is especially valuable for proxy models, when the runtime of a forward simulation is measured in seconds. At the moment, only numerical derivatives can be used for gradient-based optimization methods. The implementation of adjoint-based gradient calculations within DARTS would be simplified owing to operator form of the governing equations, and would further improve gradient-based optimization performance ([117]).

In overall, additional accuracy-performance tradeoff provided by OBL, simplified manipulation of simulation model via the Python interface, and exceptional computational performance make DARTS an efficient platform for research for both forward and inverse modelling. Its architecture allows to change existing formu-

lations and even introduce new physics with minimal efforts. Furthermore, the complete transition of the main simulation loop to GPU, along with the implementation of adjoint gradients will allow taking the inverse modelling performance to a new level.

References

[1] K. Aziz and T. Settari, *Petroleum Reservoir Simulation* (Applied Science Publishers, 1979).

[2] L. P. Dake, *Fundamentals of reservoir engineering*, Vol. 8 (Elsevier, 1983).

[3] D. W. Peaceman, *Fundamentals of numerical reservoir simulation*, Vol. 6 (Elsevier, 2000).

[4] K. Aziz and T. Wong, *Considerations in the development of multipurpose reservoir simulation models,* proceedings of the 1st and 2nd International Forum on Reservoir Simulation, Alpbach, Austria (1989).

[5] K. H. Coats, *An equation of state compostional model,* SPE Journal **20**, 363 (1980).

[6] G. Acs, S. Doleshall, and E. Farkas, *General purpose compositional model,* SPE Journal **25**, 543 (1985).

[7] D. Collins, L. Nghiem, Y.-K. Li, and J. Grabenstetter, *Efficient approach to adaptive-implicit compositional simulation with an equation of state,* SPEJ **7**, 259 (1992).

[8] C. Rasmussen, K. Krejbjerg, M. Michelsen, and K. Bjurstrom, *Increasing computational speed of flash calculations with applications for compositional, transient simulations,* SPE Reservoir Evaluation and Engineering **9**, 32 (2006).

[9] D. Voskov and H. Tchelepi, *Compositional space parameterization: Theory and application for immiscible displacements,* SPE Journal **14**, 431 (2009).

[10] D. Voskov and H. Tchelepi, *Comparison of nonlinear formulations for two-phase multi-component eos based simulation,* Journal of Petroleum Science and Engineering **82-83**, 101 (2012).

[11] R. Zaydullin, D. Voskov, and H. Tchelepi, *Nonlinear formulation based on an equation-of-state free method for compositional flow simulation,* SPE Journal **18**, 264 (2013).

[12] H. Cao, R. Zaydullin, and E. Obi, *Nonlinear convergence for near-miscible problem: A mystery unveiled for natural variable simulator,* (2017).

[13] J. R. Appleyard, I. M. Cheshire, *et al.*, *The cascade method for accelerated convergence in implicit simulators,* in *European Petroleum Conference* (Society of Petroleum Engineers, 1982).

[14] F. Kwok and H. Tchelepi, *Potential-based reduced newton algorithm for nonlinear multiphase flow in porous media,* Journal of Computational Physics **227**, 706 (2007).

[15] F. P. Hamon and H. A. Tchelepi, *Ordering-based nonlinear solver for fully-implicit simulation of three-phase flow,* Computational geosciences **20**, 909 (2016).

[16] Ø. Klemetsdal, A. F. Rasmussen, O. Møyner, and K.-A. Lie, *Nonlinear gauss-seidel solvers with higher order for black-oil models,* in *ECMOR XVI-16th European Conference on the Mathematics of Oil Recovery* (2018).

[17] Ø. S. Klemetsdal, A. F. Rasmussen, O. Møyner, and K.-A. Lie, *Efficient reordered nonlinear gauss–seidel solvers with higher order for black-oil models,* Computational Geosciences , 1 (2019).

[18] R. Younis, H. A. Tchelepi, K. Aziz, *et al.*, *Adaptively localized continuation-newton method–nonlinear solvers that converge all the time,* SPE Journal **15**, 526 (2010).

[19] S. M. Sheth, R. M. Younis, *et al.*, *Localized solvers for general full-resolution implicit reservoir simulation,* in *SPE Reservoir Simulation Conference* (Society of Petroleum Engineers, 2017).

[20] T. Xu, N. Spycher, E. Sonnenthal, G. Zhang, L. Zheng, and K. Pruess, *Toughreact version 2.0: A simulator for subsurface reactive transport under non-isothermal multiphase flow conditions,* Computers and Geosciences **37**, 763 (2011).

[21] K. Vanden and P. Orkwis, *Comparison of numerical and analytical jacobians,* AIAA Journal **34**, 1125 (1996).

[22] Schlumberger, *ECLIPSE Reference Manual 2011.2*, Tech. Rep. (Schlumberger, Houston, 2011).

[23] H. Cao, P. Crumpton, and M. Schrader, *Efficient general formulation approach for modeling complex physics,* (2009) pp. 1075–1086.

[24] R. Younis, *Modern Advances in Software and Solution Algorithms for Reservoir Simulation*, Ph.D. thesis, Stanford University (2011).

[25] Y. Zhou, H. Tchelepi, and B. Mallison, *Automatic differentiation framework for compositional simulation on unstructured grids with multi-point discretization schemes,* SPE Reservoir Simulation Symposium (2011).

[26] S. Krogstad, K.-A. Lie, O. Møyner, H. M. Nilsen, X. Raynaud, B. Skaflestad, *et al.*, *Mrst-ad–an open-source framework for rapid prototyping and evaluation of reservoir simulation problems,* in *SPE reservoir simulation symposium* (Society of Petroleum Engineers, 2015).

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

[28] M. Khait, D. Voskov, *et al.*, *Gpu-offloaded general purpose simulator for multiphase flow in porous media,* in *SPE Reservoir Simulation Conference* (Society of Petroleum Engineers, 2017).

[29] J. Wallis, *Incomplete gaussian elimination as a preconditioning for generalized conjugate gradient acceleration,* Proc. of 7th SPE Symposium on Reservoir Simulation (1983).

[30] J. Wallis, R. Kendall, T. Little, and J. Nolen, *Constrained residual acceleration of conjugate residual methods,* Proc. of 8th SPE Symposium on Reservoir Simulation (1985).

[31] H. Cao, H. A. Tchelepi, J. R. Wallis, H. E. Yardumian, *et al.*, *Parallel scalable unstructured cpr-type linear solver for reservoir simulation,* in *SPE Annual Technical Conference and Exhibition* (Society of Petroleum Engineers, 2005).

[32] K. Stüben, *Algebraic multigrid (amg): experiences and comparisons,* Applied mathematics and computation **13**, 419 (1983).

[33] K. Stüben, T. Clees, H. Klie, B. Lu, M. F. Wheeler, *et al.*, *Algebraic multigrid methods (amg) for the efficient solution of fully implicit formulations in reservoir simulation,* in *SPE Reservoir Simulation Symposium* (Society of Petroleum Engineers, 2007).

[34] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed. (Philadelphia, Pennsylvania, USA: Society for Industrial and Applied Mathematics, 2003).

[35] P. Jenny, H. A. Tchelepi, and S. H. Lee, *Unconditionally convergent nonlinear solver for hyperbolic conservation laws with s-shaped flux functions,* Journal of Computational Physics **228**, 7497 (2009).

[36] X. Wang and H. A. Tchelepi, *Trust-region based solver for nonlinear transport in heterogeneous porous media,* Journal of Computational Physics **253**, 114 (2013).

[37] B. Li and H. A. Tchelepi, *Nonlinear analysis of multiphase transport in porous media in the presence of viscous, buoyancy, and capillary forces,* Journal of Computational Physics **297**, 104 (2015).

[38] D. V. Voskov, H. A. Tchelepi, R. Younis, *et al.*, *General nonlinear solution strategies for multiphase multicomponent eos based simulation,* in *SPE Reservoir Simulation Symposium* (Society of Petroleum Engineers, 2009).

[39] A. Abadpour and M. Panfilov, *Method of negative saturations for modeling two-phase compositional flow with oversaturated zones,* Transport in porous media **79**, 197 (2009).

[40] D. Voskov, *An extended natural variable formulation for compositional simulation based on tie-line parameterization,* Transport in Porous Media **92**, 541 (2012).

[41] D. Voskov and H. Tchelepi, *Compositional nonlinear solver based on trust regions of the flux function along key tie-lines,* in *SPE Reservoir Simulation Symposium*, Vol. 2 (2011) pp. 799–809.

[42] K. Mansour Pour, *Advanced nonlinear solver for a discrete fracture modelling,* (2018).

[43] D. V. Voskov, *Operator-based linearization approach for modeling of multiphase multi-component flow in porous media,* Journal of Computational Physics **337**, 275 (2017).

[44] K. Haugen and B. Beckner, *A general flow equation framework,* SPE Reservoir Simulation Symposium 2015 **2**, 1310 (2015).

[45] M. Khait and D. Voskov, *Operator-based linearization for non-isothermal multiphase compositional flow in porous media,* in *ECMOR XIV-15th European Conference on the Mathematics of Oil Recovery* (2016).

[46] M. Khait and D. V. Voskov, *Operator-based linearization for general purpose reservoir simulation,* Journal of Petroleum Science and Engineering **157**, 990 (2017).

[47] M. L. Michelsen, *The isothermal flash problem. part ii. phase-split calculation,* Fluid Phase Equilibria **9**, 21 (1982).

[48] CMG, *STARS users's guide* (2009).

[49] *Ieee standard for floating-point arithmetic,* IEEE Std 754-2008 , 1 (2008).

[50] A. Weiser and S. Zarantonello, *A note on piecewise linear and multilinear table interpolation in many dimensions,* Mathematics of Computation **50**, 189 (1988).

[51] R. Zaydullin, D. Voskov, S. James, H. Henley, and A. Lucia, *Fully compositional and thermal reservoir simulation,* Computers and Chemical Engineering **63**, 51 (2014).

[52] J. E. Killough, *Ninth spe comparative solution project: A reexamination of black-oil simulation,* in *SPE Reservoir Simulation Symposium* (1995).

[53] P. Sammon, B. Rubin, *et al.*, *Practical control of timestep selection in thermal simulation,* SPE Reservoir Engineering **1**, 163 (1986).

[54] R. Lantz *et al.*, *Quantitative evaluation of numerical diffusion (truncation error),* Society of Petroleum Engineers Journal **11**, 315 (1971).

[55] F. Orr Jr., B. Dindoruk, and R. Johns, *Theory of multicomponent gas/oil displacements,* Industrial and Engineering Chemistry Research **34**, 2661 (1995).

[56] D.-Y. Peng and D. B. Robinson, *A new two-constant equation of state,* Ind Eng Chem Fundam **15**, 59 (1976).

[57] J. Lohrenz, B. Bray, and C. Clark, *Calculating viscosities of reservoir fluids from their compositions,* SPEJ (1964).

[58] K. Haugen and B. Beckner, *Highly optimized phase equilibrium calculations,* (2013) pp. 96–104.

[59] M. Khait and D. Voskov, *Adaptive parameterization for solving of thermal/compositional nonlinear flow and transport with buoyancy,* SPE Journal **33**, 522 (2018), sPE-182685-PA.

[60] M. Khait and D. Voskov, *Operator-based linearization for efficient modeling of geothermal processes,* Geothermics **74**, 7 (2018).

[61] Z. Y. Wong, R. Horne, and D. Voskov, *A geothermal reservoir simulator in ad-gprs,* in *World Geothermal Congress* (2015).

[62] *COMSOL Multiphysics Reference Manual, version 5.1*, COMSOL AB, Stockholm, Sweden (2015).

[63] S. Saeid, R. Al-Khoury, H. M. Nick, and M. A. Hicks, *A prototype design model for deep low-enthalpy hydrothermal systems,* Renewable energy **77**, 408 (2015).

[64] C. Willems, T. Goense, H. Nick, and D. Bruhn, *The relation between well spacing and net present value in fluvial hot sedimentary aquifer geothermal doublets; a west netherlands basin case study,* in *41st Workshop on geothermal aquifer engineering Stanford University* (2016).

[65] D. Voskov, R. Zaydullin, and A. Lucia, *Heavy oil recovery efficiency using sagd, sagd with propane co-injection and strip-sagd,* Computers & Chemical Engineering **88**, 115 (2016).

[66] Z. Y. Wong, R. Horne, and D. Voskov, *Comparison of nonlinear formulations for geothermal reservoir simulations,* (2016).

[67] S. Lee, Y. Efendiev, and H. Tchelepi, *Hybrid upwind discretization of nonlinear two-phase flow with gravity,* Advances in Water Resources **82**, 27 (2015).

[68] E. Peters, R. Arts, G. Brouwer, C. Geel, S. Cullick, R. Lorentzen, Y. Chen, K. Dunlop, F. Vossepoel, R. Xu, P. Sarma, A. Alhutali, and A. Reynolds, *Results of the brugge benchmark study for flooding optimization and history matching,* SPE Reservoir Evaluation and Engineering **13**, 391 (2010).

[69] V. Bukshtynov, O. Volkov, L. Durlofsky, and K. Aziz, *Comprehensive framework for gradient-based optimization in closed-loop reservoir management,* Computational Geosciences **19**, 877 (2015).

[70] K.-T. Lim, D. Schiozer, and K. Aziz, *New approach for residual and jacobian arrays construction in reservoir simulators,* (1994) pp. 265–270, cited By 0.

[71] H. A. Van der Vorst and C. Vuik, *The superlinear convergence behaviour of gmres,* Journal of computational and applied mathematics **48**, 327 (1993).

[72] Y. Zhou, Y. Jiang, and H. A. Tchelepi, *A scalable multistage linear solver for reservoir models with multisegment wells,* Computational Geosciences **17**, 197 (2013).

[73] A. Yuldashev, R. Gubaidullin, and N. Repin, *Development of parallel linear solver for reservoir simulation on hybrid computing systems with gpus,* in *CEUR Workshop Proceedings*, Vol. 1482 (2015) pp. 392–398, (in Russian).

[74] K. Esler *et al.*, *Realizing the potential of gpus for reservoir simulation,* in *ECMOR XIV-14th European Conference on the Mathematics of Oil Recovery* (2014).

[75] D. Voskov and M. Khait, *Adaptive coarsening in physical representation for the robust thermal-compositional simulation,* (2017) pp. 1610–1625, cited By 1.

[76] M. Christie and M. Blunt, *Tenth spe comparative solution project: A comparison of upscaling techniques,* SPE Reservoir Evaluation and Engineering **4**, 308 (2001).

[77] M. Khait, D. Voskov, and G. Konidala, *Tie-simplex parametrization for operator-based linearization for non-isothermal multiphase compositional flow in porous,* (2018) cited By 0.

[78] M. Khait and D. Voskov, *Integrated framework for modelling of thermal-compositional multiphase flow in porous media,* in *SPE Reservoir Simulation Conference* (Society of Petroleum Engineers, 2019).

[79] Y. Wang, M. Khait, D. Voskov, S. Saeid, and D. Bruhn, *Benchmark test and sensitivity analysis for geothermal applications in the netherlands,* in *44th Workshop on Geothermal Reservoir Engineering* (2019).

[80] DARTS, *Delft Advanced Research Terra Simulator.* (2019).

[81] T. E. Oliphant, *A guide to NumPy*, Vol. 1 (Trelgol Publishing USA, 2006).

[82] E. Jones, T. Oliphant, P. Peterson, *et al.*, *SciPy: Open source scientific tools for Python,* (2001–), [Online; accessed <today>].

[83] W. McKinney *et al.*, *Data structures for statistical computing in python,* in *Proceedings of the 9th Python in Science Conference*, Vol. 445 (Austin, TX, 2010) pp. 51–56.

[84] J. Ahrens, B. Geveci, and C. Law, *ParaView: An end-user tool for large-data visualization* (2005) pp. 717–731, cited By 286.

[85] H. J. Kretzschmar and W. Wagner, *International Steam Tables: Properties of Water and Steam based on the Industrial Formulation IAPWS-IF97* (Springer Science & Business Media, 2007).

[86] P. Dietrich, R. Helmig, M. Sauter, H. Hötzl, J. Köngeter, and G. Teutsch, *Flow and transport in fractured porous media* (Springer Science & Business Media, 2005).

[87] K.-T. Lim *et al.*, *A new approach for residual and jacobian arrays construction in reservoir simulators,* SPE Computer Applications **7**, 93 (1995).

[88] Y. Jiang, *Techniques for modeling complex reservoirs and advanced wells* (PhD Thesis, Stanford University, 2007).

[89] A. Pinar and M. T. Heath, *Improving performance of sparse matrix-vector multiplication,* in *SC'99: Proceedings of the 1999 ACM/IEEE Conference on Supercomputing* (IEEE, 1999) pp. 30–30.

[90] A. Behie and P. Forsyth Jr, *Comparison of fast iterative methods for symmetric systems,* IMA Journal of Numerical Analysis **3**, 41 (1983).

[91] Y. Saad, *A flexible inner-outer preconditioned gmres algorithm,* SIAM Journal on Scientific Computing **14**, 461 (1993).

[92] S. Shetty, D. Voskov, and D. F. Bruhn, *Numerical strategy for uncertainty quantification in low enthalpy geothermal projects,* in *Workshop on Geothermal Reservoir Engineering 2018* (2018).

[93] J. O'Sullivan, A. Croucher, A. Yeh, and M. O'Sullivan, *Further improvements in the convergence of tough2 simulations,* in *In 11th World Congress on Computational Mechanics, Barcelona, Spain* (2014).

[94] R. Zaydullin, D. Voskov, and H. Tchelepi, *Phase-state identification bypass method for three-phase thermal compositional simulation,* Computational Geosciences , 1 (2015).

[95] Y. Chen, D. Voskov, and M. Khait, *Optimization of co2 injection using multi-scale reconstruction of compositional transport,* (2018) cited By 0.

[96] A. Iranshahr, D. Voskov, and H. Tchelepi, *Tie-simplex based compositional space parameterization: Continuity and generalization to multiphase systems,* AIChE Journal **59**, 1684 (2013).

[97] D. Kraft, *A software package for sequential quadratic programming,* Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt (1988).

[98] H. A. Voskov, D. V.and Tchelepi, *Compositional space parameterization: Multicontact miscible displacements and extension to multiple phases,* SPE Journal **14**, 441 (2009), sPE-113492-PA.

[99] H. Pan and H. A. Tchelepi, *Compositional flow simulation using reduced-variables and stability-analysis bypassing,* in *SPE Reservoir Simulation Symposium* (2011) sPE-142189-MS.

[100] A. Iranshahr, D. Voskov, and H. Tchelepi, *Generalized negative flash method for multiphase muticomponent systems,* Fluid Phase Equilibria, **299**, 272 (2010).

[101] A. Iranshahr, Y. Chen, and D. V. Voskov, *A coarse-scale compositional model,* Computational Geosciences **18**, 797 (2014).

[102] A. Salehi, *Upscaling of Compositional Flow Simulation Based on a Non-Equilibrium Formulation*, Ph.D. thesis, Stanford University (2016).

[103] R. Zaydullin, D. Voskov, and H. A. Tchelepi, *Nonlinear formulation based on an equation-of-state free method for compositional flow simulation. society of petroleum,* SPE Journal **18**, 264 (2012), sPE-146989-PA.

[104] P. Jenny, S. H. Lee, and H. A. Tchelepi, *Multi-scale finite-volume method for elliptic problems in subsurface flow simulation,* J. Comput. Phys. **187**, 47 (2003).

[105] H. Hajibeygi, G. Bonfigli, M. A. Hesse, and P. Jenny, *Iterative multiscale finite-volume method,* Journal of Computational Physics **227**, 8604 (2008).

[106] J. E. Aarnes, S. Krogstad, and K.-A. Lie, *A hierarchical multiscale method for two-phase flow based upon mixed finite elements and nonuniform coarse grids,* Multiscale Modeling & Simulation **5**, 337 (2006).

[107] M. Wolff, B. Flemisch, and R. Helmig, *An adaptive multiscale approach for modeling two-phase flow in porous media including capillary pressure,* Water Resources Research **49**, 8139 (2013).

[108] H. Zhou, S. H. Lee, and H. A. Tchelepi, *Multiscale finite-volume formulation for saturation equations,* SPE Journal **17**, 198 (2012), sPE-119183-PA.

[109] C. Ganapathy, *Multiscale Reconstruction of Compositional Transport*, Master's thesis, Delft University of Technology (2017).

[110] F. M. Orr, *Theory of Gas Injection Process* (Tie-Line Publications, Denmark:Holte, 2007).

[111] C. Ganapathy, Y. Chen, and D. Voskov, *Multiscale reconstruction of compositional transport,* in *ECMOR 2018-16th European Conference on the Mathematics of Oil Recovery* (2018).

[112] D. Voskov and V. Entov, *Problem of oil displacement by gas mixtures,* Fluid Dynamics **36**, 269 (2001).

[113] D. Voskov and H. Tchelepi, *Compositional space parameterization: Theory and application for immiscible displacements,* SPE Journal **14**, 431 (2009).

[114] Geoquest, *PVTi Reference Manual* (Schlumberger, 2008).

[115] G. Rannou, D. Voskov, and H. Tchelepi, *Tie-line-based k-value method for compositional simulation,* SPE Journal **18**, 1112 (2013).

[116] H. Knibbe, C. W. Oosterlee, and C. Vuik, *Gpu implementation of a helmholtz krylov solver preconditioned by a shifted laplace multigrid method,* Journal of Computational and Applied Mathematics **236**, 281 (2011).

[117] J. Jansen, *Adjoint-based optimization of multi-phase flow through porous media – a review,* Computers & Fluids **46**, 40 (2011), 10th ICFD Conference Series on Numerical Methods for Fluid Dynamics (ICFD 2010).

Curriculum Vitæ

Mark KHAIT

19-10-1984 Born in Ufa, Russia

Education

1990–2001 Secondary School
Gymnasium № 82, Ufa, Russia

1999–2001 Distance Education School of Physics & Technology
Moscow Institute of Physics and Technology, Moscow, Russia

2001-2006 Master of Science in Information Security
Ufa State Aviation Technical University, Ufa, Russia
Thesis: Algorithms and Software for Cryptographic System
Promotor: Dr. V.E. Kladov

2015-2019 Doctor of Philosophy in Geoscience & Engineering
Delft University of Technology, Delft, The Netherlands
Thesis: Delft Advanced Reservoir Terra Simulator. General Purpose Reservoir Simulator with Operator-Based Linearization
Promotor: Dr. D.V. Voskov
Copromotor: Prof. dr. ir. J.D. Jansen

Professional Experience

2003-2007	Research Assistant Computational Research Center Ufa State Aviation Technical University, Ufa, Russia
2007-2008	Visiting Researcher Department of Energy Resources Engineering Stanford University, Stanford, USA
2007-2015	Senior Researcher IT Department Ufa R&D Institute, Rosneft Oil Company, Ufa, Russia
2017	Research & Development Intern Geology Technology Team Aramco Services Company, Houston, USA
2019	Computational Scientist Intern Stone Ridge Technology, Bel Air, USA

Volunteering

2011-2013	Chairman of Apartment Building Board
2013-2015	Member of Precinct Election Comission

List of Publications

Journal Articles

9. **A. Blinovs, M. Khait, D. Voskov, V.Elichev**, *Physics-Based Data-Driven Model for Short-Term Production Forecast*, in preparation.

8. **Y. Wang, D.V. Voskov, M. Khait, S. Saeid, D. Bruhn**, *Sensitivity analysis of a geothermal application in the Netherlands*, in preparation.

7. **Y. Wang, D.V. Voskov, M. Khait, D. Bruhn**, *An efficient numerical simulator for geothermal simulation: a benchmark study*, submitted to Applied Energy Journal.

6. **T.D. Jobe, E. Vital-Brazil, M. Khait**, *Geological Feature Prediction Using Image-Based Machine Learning*, Petrophysics **59(6)**, 750-760 (2018).

5. **M. Khait, D.V. Voskov**, *Adaptive Parameterization for Solving of Thermal/Compositional Nonlinear Flow and Transport With Buoyancy*, SPE Journal **23(2)**, 522-534 (2018).

4. **M. Khait, D.V. Voskov**, *Operator-Based Linearization for Efficient Modeling of Geothermal Processes*, Geothermics **74**, 7-18 (2018).

3. **M. Khait, D.V. Voskov**, *Operator-Based Linearization for General Purpose Reservoir Simulation*, Journal of Petroleum Science and Engineering **157**, 990-998 (2017).

2. **V.A. Baikov, I.K. Badykov, O.S. Borschuk, I.F. Saifullin, M.L. Khait, M.A. Linvinenko, A.V. Timonov**, *Digital Experimental Filtration Laboratory*, Scientific and Technical Bulletin "PJSC "Rosneft Oil Company" **3(28)**, 43-47 (2012). [in Russian]

1. **V.A. Baikov, O.S. Borschuk, V.I. Savichev, M.L. Khait**, *Developmental experience of high-performance computing systems for hydrodynamic modelling of oil and gas production processes*, Oil Industry **10**, 100-103 (2006). [in Russian]

Conference Proceedings

13. **S. Saeid, Y. Wang, A. Daniilidis, M. Khait, D.V. Voskov, D. Bruhn**, *Lifetime and Energy Prediction of Geothermal Systems: Uncertainty Analysis in Highly Heterogeneous Geothermal Reservoirs (Netherlands)*, Proceedings of World Geothermal Congress (2020).

12. **A. Daniilidis, M. Khait, S. Saeid, D. Bruhn, D.V. Voskov**, *A High Performance Framework For Optimization Of Energy Generation, Field Lifetime And Economic Output Of Geothermal Systems*, Proceedings of World Geothermal Congress (2020).

11. **Y. Wang, M. Khait, D.V. Voskov, S. Saeid, D. Bruhn**, *Benchmark test and sensitivity analysis for Geothermal Applications in the Netherland*, Proceedings of 44th Workshop on Geothermal Reservoir Engineering(2019).

10. **M. Khait, D.V. Voskov**, *Integrated Framework for Modelling of Thermal-Compositional Multiphase Flow in Porous Media*, Proceedings of SPE Reservoir Simulation Conference (2019).

9. **Y. Chen, M. Khait, D.V. Voskov**, *Optimization Of CO2 injection using multi-scale reconstruction of compositional transport*, Proceedings of 16th European Conference on the Mathematics of Oil Recovery(2018).

8. **M. Khait, D.V. Voskov, G.K. Konidala**, *Tie-Simplex Parametrization For Operator-Based Linearization For Non-Isothermal Multiphase Compositional Flow In Porous*, Proceedings of 16th European Conference on the Mathematics of Oil Recovery (2018).

7. **M. Khait, D.V. Voskov**, *GPU-Offloaded General Purpose Simulator for Multiphase Flow in Porous Media*, Proceedings of SPE Reservoir Simulation Conference (2017).

6. **D.V. Voskov, M. Khait**, *Adaptive Coarsening in Physical Representation for the Robust Thermal-Compositional Simulation*, Proceedings of SPE Reservoir Simulation Conference (2017).

5. **M. Khait, D.V. Voskov**, *Operator-based Linearization for Non-isothermal Multiphase Compositional Flow in Porous Media*, Proceedings of 15th European Conference on the Mathematics of Oil Recovery (2016).

4. **O.S. Borschuk, I.F. Saifullin, M.L. Khait**, *Development of parallel version of software suite for modelling of hydrocarbon flow in subsurface NGT BOS*, Proceedings of Parallel Computational Technologies Conference (PCT'2010) (2010). [in Russian]

3. **M.L. Khait**, *Efficient Parallel Reservoir Simulation Using Multicore Architectures*, Proceedings of SPE Reservoir Simulation Symposium (2009).

2. **V.A. Baikov, O.S. Borschuk, R.K.Gazizov, I.F. Saifullin, M.L. Khait, A.V.Yuldashev**, *Modelling of hydrocarbon filtration flows in porous medium using multiprocessor computing systems*, Proceedings of Scientific Internet Service: Parallel Programming Technologies Conference (2007). [in Russian]

1. **V.A. Baikov, R.K.Gazizov, S.Y. Lukaschuk, M.L. Khait, A.V.Yuldashev**, *Modelling of hydrocarbon filtration flows in porous medium using multiprocessor computing systems*, Proceedings of Scientific Internet Service: Parallel Programming Technologies Conference (2006). [in Russian]

Acknowledgements

This dissertation would never be even started, neither accomplished the way it has been, without the help and support of many people. I would like to express my highest gratitude for their involvement, in one way or another.

Dear Denis, there are not enough words to thank you for all you have done for me. We became friends yet in Stanford in 2006-2007, and making that call in August 2014 I had no doubt in your help. Thank you for believing in me and creating this wonderful opportunity! I feel proud and honoured to be your first PhD student at Delft University of Technology. Thank you for the endless energy, passion, and motivation you provided during these 4 years. I always knew where to go in the moments of doubt and frustration. Thank you for many evenings our families spent together in your house when after a delicious dinner with a glass of wine we inevitably converged to discussions about work and future plans. Our relationships grew over friendship and transformed into something bigger. You are my fine example in many respects, related to both professional and personal life. Thank you, Denis.

Dear Prof. Jan Dirk Jansen, Prof. William Rossen, I am so thankful to you for trusting in me and being favourable during the intake interview! Jan Dirk, I appreciate your unconditional support during my PhD journey, always honest and critical comments on my work. William, thank you for all the assistance and support that was greatly relieving the beurocratic pressure thus making more room for research. I also appreciate your and Janice's initiative of organizing a work session with wonderful live piano music in Aula hall for all of us - what an unforgettable experience that was!

Dear Prof. Kees Vuik and Dr. Hadi Hajibeygi, I thank you for favouring me during my Go/No-Go meeting though not all my answers were perfect! Kees, thank you for making your lecture on linear solvers and preconditioners available on Youtube, and especially the inspirational story about ICCG solver you told about when two bad methods combined together can result in a really good method. Hadi, I am grateful for all the DARSim - related activities you initiated and tirelessly maintained, making it possible for all students to listen not only to each other but also to leading experts in our field and to have lots of fruitful discussions afterwards. Thank you for your passionate lectures, they served me as a fine example of how deeply a lecturer should be involved in his topic.

Dear Prof. Hans Bruining, I am grateful for your interesting questions and curious comments during many department seminars. I am especially thankful for the words of wisdom you always have for everyone around you, including me - they help to stop the daily rush and look at things from a different perspective. Dear Dr. Phil Vardon, I thank you for organizing both informative and informal geothermal meetings, where I was able to learn a bit more about current geothermal activi-

ties in our department and to find collaboration opportunities. Special thanks for sharing the wonderful picture of the DAP project, it made the best thesis cover I could ever have! Dear Dr. Olwijn Leeuwenburgh, I am grateful to you for explaining to me many practical aspects of optimization and inverse modelling. Dear Ir. Kees Lemmens, I appreciate the long and detailed discussion about various techniques related to CUDA that we had. I remember your excitement about Continuity containers and look forward to making use of it!

During many conferences and workshops I took part during this 4 years, in I was lucky to discuss my research and get valuable comments from leading experts in our field, including Prof. Patrick Jenny, Prof. Hamdi Tchelepi, Prof. Rainer Helmig, Prof. Knut-Andreas Lie, Prof. Vasily Demyanov, Alberto Cominelli, Dr. Ahmad Abushaikha, Dr. Hui Cao and Dr Olav Moyner. Special gratitude to Hamdi for hosting me during my visits to Stanford University many years back. In fact, that was a mind-opening experience and a turning point for me. I am also grateful to Rainer for inviting me to the University of Stuttgart and giving the opportunity to present my research. I would also like to thank Dr. John O'Sullivan for his help in setting up the high-enthalpy geothermal simulations with TOUGH2.

Coming from a medium-sized city in Russia to work in Delft University of Technologies, how big are the chances to have a townsman sitting in the same room, two tables apart from yours? I was very lucky indeed to have Nikita always ready for help with anything - even like organizing a private Geoscience & Engineering lab tour for my parents! I am also grateful to Nik for his permanent efforts to bring all PhDs together for dinner, football training, or somewhere else. Along with Rahul-Mark, Durgesh, Siavash, Daniel Victor, Elisa and other 'previous wave' PhD candidates, you maintained a special spirit and warm atmosphere of support and involvement in our room. My special thanks to Rahul-Mark for his wise advice and insights on choosing further career step towards the end of my PhD contract.

Jiakun, thank you for steadily keeping those traditions, for your generous help with taking photos during our defence ceremonies, and for being always available with a good piece of advice on virtually any question. Matteo, Rafael, Eduardo, it was always a pleasure to work near you charging the positive energy coming all the way from sunny Italy and Brazil! Matteo and Rafael, you guys opened to me a whole new world of picanha steak and taught me how to cook it! Picanha has become my favourite cut, and I am spreading this taste further to my family and friends. Eduardo, thank you for inviting me to Tuesday football training sessions, they were fun and friendly thanks to your attitude. I remember that guided by your enthusiasm, our section once has won the MV football cup!

Matei, we worked at the same office for around 3 years, but my most vivid memories by far are those several weeks in Houston we happened to be at the same time - you were finishing your internship and I was just starting. We survived a Category 4 hurricane and subsequent flooding, celebrated the success of your appointment with a delicious piece of steak in a fancy restaurant, and same evening managed to get home just in time before the curfew hours (literally minutes, who knew they even had a curfew!).

Jakolien, thank you for inspiring to learn at least basics of Dutch language and

your warm smiles for everyone in the office. Bander, thank you for being my deskmate, always ready to provide an opinion, comment or suggestion. You were my fine example of an intelligent and polite petroleum engineer.

Swej, I appreciate our casual chatting just about everything, from movies to Formula 1, and thank you for sharing my passion for a glass of good whisky. The taste of Paneer Tikka Masala that you made for us using probably more spices in that single dish than I ever had in my life before - I will never forget, so delicious it was! Martijn, I am thankful for our discussions about Netherlands culture and traditions. You always share your opinion in the most direct and honest way! Brandon, you were always the one to talk about serious things - life choices, expectations, and consequences. Thank you for those discussions and may we both become a bit more easy-going! Ahmed, thank you for sharing your positive energy and the most curious travelling stories. I admire your chef skills: all deserts you shared with us were made with scientific precision. Needless to say, they tasted perfect, cheering everyone up for the rest of the day!

Kai, you were always able to help whenever I needed it, being tough but fair rival during football training sessions. Sian, I appreciate your care about the most personal and touching gifts for those of us who just defended their thesis. In between our graduations, every now and then you made and shared with us the most delicious cakes and cookies! I remember a couple of times when you and Ahmed both brought something special on the same day - and I felt very happy despite any current hurdles on my agenda. Mohsen, I am thankful for your sincere and informative stories about Iranian culture and traditions. Longlong, it was a pleasure to have you in our office some years back, and it is exciting that our paths crossed again for a collaboration project between HBKU and TU Delft.

Dear Xiaocong, Yang, Stephan, you have made solid and significant contributions to this thesis, I appreciate it deeply. Dear Kiarash, you have just started working on one of the most difficult problems - and I am sure the results will come. I am thankful and proud to work with all of you guys on the development of DARTS, and look forward to what we can achieve together!

I am grateful to all master students that I had a chance to work with - I believe to have learned something useful from each of you and I hope that worked both ways. I would especially like to thank Geetha and Arturs for their valuable contributions to this thesis and DARTS. Emil, I was so lucky to meet you and Irina! I appreciate the time we spent together, always missing your jokes and looking forward to seeing you again.

Dear Alex and Sanaz, I am so thankful for our collaboration and your trust placed in DARTS. The fact that you decided to use it in your own research provided me with extra inspiration and motivation!

Dear Rustem, Houston is my favourite travel destination because of your and Diana's hospitality - I feel like home in your house. I am also grateful for introducing me to Yan - that half an hour meeting resulted in a 3-month internship later same year, which in turn triggered serious changes in my research, career, and life itself. Thanks to Harvey and hospitality of you both, we happened to spend a lot of time together and become good friends. Energized by Yan's passion for Python language,

I figured that it was the perfect instrument to complement my rigid C++ code, thus setting the cornerstone for DARTS architecture.

I appreciate Susan Agar and the whole GTT team in Aramco Research Center in Houston for the wonderful experience I had during my internship. I am proud of our achievements.

Dear Vincent and Ken, it was a great pleasure and honour to be a part of Stone Ridge Technology, even for just one month. I wish it could be longer, but still more than satisfied with the outcome. Timur, it was so nice to catch up with you after so many years! Thank you and Vika for showing me around some nice places, we had a great time. I am happy for you and Yan complementing the team, making Maryland another attractive travel destination!

I would like to express my special gratitude to Marlijn Ammerlaan, Lydia Broekhuijsen-Bentvelzen, Margot Bosselaar-Perk, and Ralf Haak for their continuous help and support with any request I had. Joost van Meel, Guus Lohlefink, Robert Keulen, Rob van Laarhoven, thank you for the stable and faultless maintenance of cluster systems and Gitlab repositories. Jolanda van Haagen, I appreciate your help and patience during lab sessions for AES1320 course, and thank you for your involvement in the improvement of the Dietz Tongue experiment procedure!

Dear Oleg and Arthur, I appreciate your help with linear solver libraries. I am also thankful to all members of NGT BOS group for the team spirit and work atmosphere, which I keep warm memories of.

Over these four years, I was only able to get back to Ufa a couple of times, but every time Timofey, Artyom, Andrey, Vladislav and Alex made me feel like I have never left. My dear, true friends, I cherish our bond. Vis unita fortior!

I would not be able to stand where I am without all my family and their endless and unconditional support. My dear parents, Fanaziya and Leonid, I have no words to acknowledge the sacrifices you made and the dreams you had to let go, just to give me a shot at achieving mine. Marina, my sister, I am thankful for your love, care and support.

Alina, we started this journey together and you have been standing by my side all along, charging me with confidence, softening sharp corners, keeping the light of our family - I simply could not make it without you. Thank you, my love!