NTNU – Trondheim
Norwegian University of
Science and Technology

Control-Switching Strategies for Reservoir Water-Flooding Management

Jon Anders Krogstad

Master of Science in Cybernetics and Robotics
Submission date: June 2015
Supervisor: Bjarne Anton Foss, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

NTNU Faculty of Information Technology,
Norwegian University of Science and Technology **Mathematics and Electrical**

Engineering

Department of Engineering Cybernetics

Master project

Name of candidate: Jon Anders Krogstad

Subject: Engineering Cybernetics

Title: Control-Switching strategies for reservoir water-flooding management.
Title (in Norwegian):

Reservoir water-flooding planning can be accomplished by a Nonlinear Model Predictive Control Algorithm (NMPC). To this end, the partial differential equations of the reservoir are discretized in space and in time and the optimal control problem is solved as a non-linear programming problem. Within this scheme, the independent variables to be decided upon are the control parameters describing how the reservoir should be operated.

Output constraints handling in reservoir management can be expensive due to the sensitivity calculations required to include these constraints. To limit overhead, reservoir simulators implement control-switching strategies. The strategies consist in switching control inputs during simulation in order to avoid constraint violations. In this way, the variable being violated will be treated as an independent control, and kept feasible. Therefore the inclusion of the constraints on the optimizer-side is no longer necessary.

This strategy introduces inconsistencies between the simulator and the optimizer. The simulator does not follow the control strategy dictated by the optimizer but an heuristic to prevent constraint violations. This research seeks to formalize the control-switching procedure to maintain consistency between the optimization procedures.

Task description:

1. Perform a literature on reservoir control optimization techniques including:
    - Reactive control technique with control-switching.
    - Constrained and unconstrained control optimization for reservoir management.
    - Optimization techniques based on embedded constraints on the simulators.
2. Propose suitable simple test cases to evaluate control-switching strategies.
3. Assess the merits of dealing with constraints directly within the simulator, and by including them in a NLP optimization algorithm.
4. Propose an algorithm to switch control strategies and assess its performance compared to traditional approaches.

Suggested literature:

[1] J. D. Jansen, “Adjoint-based optimization of multi-phase flow through porous media – A review,” Comput. Fluids, vol. 46, no. 1, pp. 40–51, Jul. 2011.
[2] A. Codas, B. Foss, and E. Camponogara, “Output constraint handling & parallelization for oil reservoir control optimization via multiple shooting,” submitted to the SPE Journal.

[3] K.-A. Lie, S. Krogstad, I. S. Ligaarden, J. R. Natvig, H. M. Nilsen, and B. Skaflestad, “Open-source MATLAB implementation of consistent discretisations on complex grids,” Computational Geosciences, vol. 16, no. 2, pp. 297–322, Aug. 2011.
[4] A. Codas,“Reservoir multiple shooting optimization code and cases,” 2014. [Online]. Available: https://github.com/iocenter/remso
[5] D. Kourounis, L. J. Durlofsky, J. D. Jansen, and K. Aziz, “Adjoint formulation and constraint handling for gradient-based optimization of compositional reservoir flow,” Computational Geosciences, vol. 18, no. 2, pp. 117–137, Jan. 2014.

Starting date:
End date:
Co-supervisor: Andres Codas, NTNU

Bjarne Foss
Professor/supervisor

Abstract

Waterflooding is a commonly used operation in the oil industry. It is used to increase oil recovery from the reservoir, and is considered as a secondary recovery technique. In short, the reservoir is flooded with water to increase pressure, and to displace oil from pore spaces. This thesis has explored three general methods for optimizing this operation, with special attention paid to treatment of constraints. As means of background theory, a brief introduction to petroleum engineering is provided, the concept of dynamic optimization is stated, and reservoir simulation is presented.

The three methods of interest are gradient-based control optimization, reactive control, and techniques based on simulator-embedded constraints. The popularity of gradient-based control optimization has grown during the last decade, and there has been conducted a vast amount of research on the topic during this period. Such optimization is usually referred to as adjont-based optimization, as efficient computation of gradients serves as a prerequisite for the method to remain tractable for reservoir optimization. Next, reactive control is widely used in the industry, and its popularity is among others due to simplicity, robustness, and model-independence. The last method considered is based on simulator-embedded constraints, which of today is a relatively unexplored area. A new heuristic has however been developed during this work, which combines output unconstrained control optimization and simulator-embedded constraints.

The three general approaches are implemented using the Matlab Reservoir Simulation Toolbox. A series of four case studies are employed to assess the merits of the methods. In short, if geological uncertainty is put aside, gradient-based optimization appeared as the preferred method. Furthermore, the popularity of the reactive control approach can be understood, as the method performed adequately - keeping its simplicity in mind. The heuristic that was developed showed some potential, but there are three issues at the current stage that must be addressed and resolved.

Sammendrag

Vanninjeksjon er en hyppig brukt metode i oljeindustrien for å øke utvinningsgraden av reservoaret, og er regnet som en sekundær utvinningsmetode. Metoden går ut på å flømme reservoaret med vann for å opprettholde trykk, og fortrenge oljen gjennom porene i reservoarsteinen. Denne masteroppgaven har utforsket tre metoder for å optimere denne prosessen, med fokus på å overholde begrensninger i systemet. Fundamental reservoarteori, dynamisk optimering, og reservoar simulering er beskrevet som bakgrunnsteori.

De tre metodene er gradient-basert kontroll optimering, reaktiv kontroll, og teknikker som bygger på begrensninger innebygget i simulatoren. Gradient-basert optimering har i løpet av de siste tiårene opplevd økede popularitet, og det forskes mye på dette per i dag. Slik optimering er ofte referert til som adjoint-basert optimering, fordi gradienter må kunne oppdrives effektivt for at denne fremgangsmåten skal være praktisk gjennomførbar. Videre er reaktiv kontroll mye brukt i industrien, som følge dens enkelthet, robusthet, og uavhengighet av modeller. Den siste metoden som bygger på begrensinger innebygget i simulatoren er per i dag et lite utforsket område. En ny heuristikk innenfor dette område har blitt utviklet i løpet av arbeidet med masteroppgaven, og den bygger på en kombinasjon av ubegrenset kontroll-optimering og begrensninger innebygget i simulatoren.

De tre metodene er implementert med Matlab Reservoir Simulation Toolbox. En serie av fire case-studier er brukt for å evaluere egenskapene til metodene. Kort oppsummert, hvis reservoar usikkerhet er satt til side, så fremstår gradient-basert kontroll-optimering som den foretrukne metoden. Det er også lett å skjønne popularitetene til reaktiv kontroll, ettersom metoden presenterer jevnt over godt, tatt dens enkelhet i betraktning. Heuristikken som ble utviklet klarte til en viss grad å løse problemene, men det er på dette stadiet tre uløste problemer – hvilke må løses for at metoden kan anses som nyttig.

Preface

This master thesis is written during the last semester of the five year Master of Science program in Engineering Cybernetics at the Norwegian University of Science and Technology. I have enjoyed working with this thesis, acquiring new knowledge of both reservoir engineering and dynamic optimization.

I would like to thank my supervisor Professor Bjarne A. Foss for guidance during the semester. A special thanks goes to PhD candidate Andres Duarte Codas for the time he has devoted to countless discussions, in addition to always helping me implementation-wise. I would also like to thank the fellow students at GG-44 for providing a positive, rewarding and supporting working environment. Finally, I would like to thank my girlfriend Madel for her support, and for coming with me to Trondheim.

Trondheim, 2015-06-06

Jon Anders Krogstad

Table of Contents

Abbreviations

AD	=	Automatic Differentiation
BHP	=	Bottom Hole Pressure
CLRM	=	Closed-Loop Reservoir Management
CP	=	Control Period
DAE	=	Differential-Algebraic Equations
EOR	=	Enhanced Oil Recovery
ICV	=	Inlet Control Valve
IVP	=	Initial Value Problem
MHE	=	Moving Horizon Estimator
MINLP	=	Mixed Integer Nonlinear Programming
MPC	=	Model Predictive Control
MRST	=	Matlab Reservoir Simulation Toolbox
MS	=	Multiple Shooting
NLP	=	Non Linear Programming
NMPC	=	Nonlinear Model Predictive Control
NPV	=	Net Present Value
OCP	=	Optimal Control Problem
ORAT	=	Oil Rate
OSSO	=	Output unconstrained Single Shooting Optimizer
QP	=	Quadratic Problem
RC1	=	See Table 5.1
RC2	=	See Table 5.1
REF1	=	See Table 5.1
REF2	=	See Table 5.1
REMSO	=	Reservoir Multiple Shooting Optimization
SQP	=	Sequential Quadratic Programming
SS	=	Single Shooting
WRAT	=	Water Rate

Chapter 1

Introduction

This thesis deal with methods for handling constraints in reservoir management, focusing on optimization of the water-flooding process. To this end, three particular approaches is of interest, namely reactive control, constrained and unconstrained control optimization, and control optimization with simulator-embedded constraints. It is assumed that the reader of this thesis has basic knowledge of mathematical optimization. Vectors and matrices are not emphasized with bold notation, but their dimensions are clarified when appropriate.

The thesis is outlined in the following manner:

- Chapter 1 gives a brief introduction to the oil industry and fundamental reservoir theory. The waterflooding operation is outlined, together with the concept of reservoir management.
- Chapter 2 is intended to give an introduction to dynamic optimization. This is relevant for understanding possible optimization-strategies for reservoir management.
- Chapter 3 includes general theory for reservoir simulation. The reservoir simulator Matlab Reservoir Simulation Toolbox (MRST) is treated separately, because the numerical examples presented in Chapter 5 are implemented on this simulator.
- Chapter 4 reviews different methods for optimizing the waterflooding operation, with special attention paid to treatment of constraints. Moreover, a new heuristic alongside with an optimizer is developed, in an attempt to combine existing methods.
- Chapter 5 presents the numerical examples, together with the results.
- Chapter 6 discusses the performance of the methods presented in Chapter 4, based on the results obtained in Chapter 5.
- Chapter 7 contains the conclusion, and recommendations for future work.

1.1 Motivation

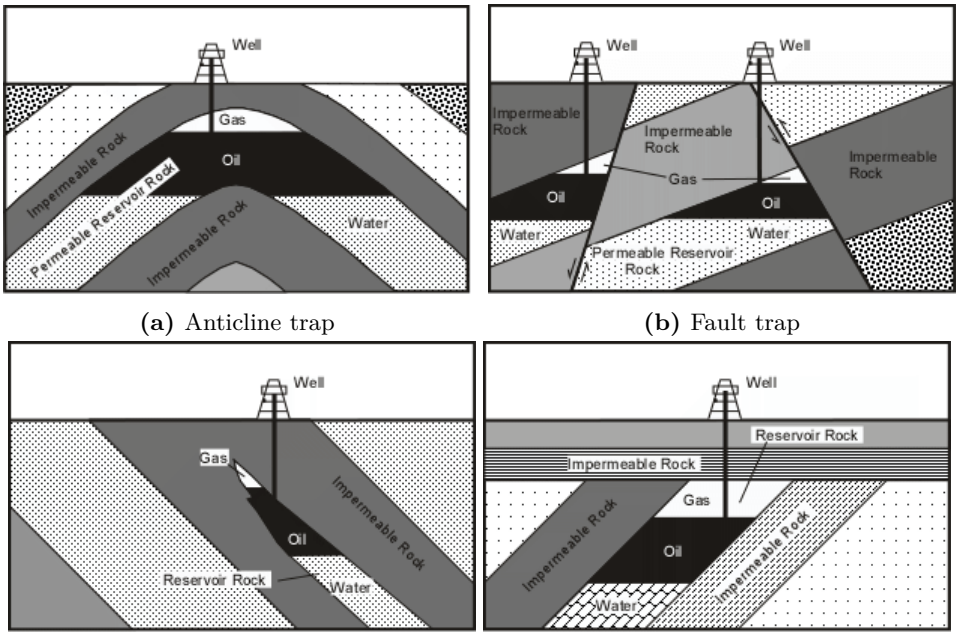
The U.S. Energy Information Agency (2013) projects that the worldwide demand of energy will grow with 56 % between 2010 and 2040. They also estimates that the energy supply will be divided almost equally between oil, gas, coal and other low carbon sources. Moreover, Conti (2014) projects that the worldwide consumption of petroleum and other liquid fuels will rise from 87 MMbbl/d in 2010[1], to 98 MMbbl/d in 2020, and 119 MMbbl/d in 2040, which indicates that oil and gas will remain an important source of energy in the years to come. One way of producing more oil is by increasing the recovery factor at existing fields, and to ensure that new fields are planned with high recovery. This is especially relevant for the Norwegian shelf which is considered mature, and many fields are in their tail-production. To put things into perspective in terms of value, the famous Ekofisk-field is estimated to contain 553.9 million Sm^3 oil (Oljedirektoratet, 2015) including oil already produced. It might seem that increasing the recovery factor of oil by 1% is small and negligible, but at an oil price of 80 dollar per barrel, this amount of oil represents a value of almost three billion dollars.

1.2 Reservoir Fundamentals

The oil and gas industry is usually divided into three major segments, namely *upstream*, *midstream* and *downstream*. The upstream segment involves exploration and production (E&P), which include searching for potential new fields, drilling of exploratory and production wells, and the production phase itself. The midstream segment involves transportation, storage and wholesale marketing of crude and refined products. The downstream sector commonly refers to refining of crude oil, and processing and purifying of raw natural gas. The scope of this thesis lies within the upstream segment.

The hydrocarbons, commonly referred to as oil and gas, are extracted from pools below the surface, which is known as the reservoir. However, several conditions must be met in order for the hydrocarbons to accumulate in the reservoir. The first condition is that source rocks rich on organic matter are deposited over time. Eventually, as this sedimentary layer is buried deep enough, a process that partially may be caused by tectonic plate movement and newer deposits, the source rock is exposed to high temperature and pressure. The source rock will at some point reach its maturation, at which the hydrocarbons are expelled from the rock. Because hydrocarbons usually are less dense than water, they naturally mitigate towards the surface. During this travel they may pass through porous rock, namely the reservoir rock. If a layer of impermeable rock surrounds the permeable reservoir rock sufficiently, the hydrocarbons are trapped, and may accumulate. This process can last for millions of years (Jahn et al., 2008). Common types of traps and reservoir formations are shown in Figure 1.1.

[1] MMbbl/day denotes one million barrel per day



(a) Anticline trap **(b)** Fault trap

(c) Stratigraphic lense **(d)** Stratigraphic unconformities

Figure 1.1: Different reservoir traps (Nelson, 2012).

Permeability

The first key property for the reservoir rock is the permeability. It characterizes the ability of a porous medium to transmit a single fluid through the interconnected pores, which is completely filled with this fluid (Lie, 2014). More precise, the permeability is the part of the proportionality constant in Darcy's law which relates discharge and fluid physical properties, to a pressure gradient applied to the porous media. This means that permeability can only be defined together with fluid flow

$$\vec{u} = -\frac{K}{\mu}\nabla\Phi$$

where K is the permeability, $\vec{u}$ is the superficial fluid flow velocity through the porous medium, μ is the dynamic viscosity of the fluid, and $\nabla\Phi$ is the applied pressure or potential gradient. Even though the SI-unit for the permeability is area [m^2], it is commonly measured in Darcy [D] which is approximately

$$1\ [\mathrm{D}] \approx 0.987 \times 10^{-12}\ [\mathrm{m}^2]$$

Because 1 Darcy is relatively high in a reservoir context, it is often specified in milli-Darcy [mD] instead. The permeability for a conventional reservoir is typically in the range 0.1 mD - 20 D for liquid flow (Lie, 2014).

Denote that the permeability in certain directions tends to depend on the permeability in other directions. It follows that that the permeability is a tensor, which can be represented by a matrix. For a Cartesian grid, this matrix yields

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

where the diagonal elements in the matrix represents direct flow in one direction, caused by pressure drop in the same direction. The cross-terms represents cross-flow, which is flow caused by pressure-drop in perpendicular directions to the flow. Moreover, the medium is said to be *isotropic* if the permeability equals in horizontal and vertical direction.

Permeability is a function of porosity, and if the flow is assumed laminar in a set of capillary tubes, the Carman-Kozeny equation links the two properties by the following relation

$$K = \frac{1}{8\tau A_v^2} \frac{\phi^3}{(1-\phi)^2}$$

where τ is the rock tortuosity and A_v is a specific surface area. The porous medium is usually saturated with two or three phases (oil, water, gas) which alter its ability to conduct fluids, thus reducing the effective permeability. The relative permeability links the effective permeability to the absolute permeability as a dimensionless ratio given by

$$k_{r,\alpha} = \frac{k_{e,\alpha}}{K}$$

where subscript r denotes relative permeability, e denotes effective permeability and α denotes the phase.

Porosity

The second key property for the reservoir rock is the already mentioned porosity, which is defined as a measure of the void space in a material

$$\phi = \frac{V_v}{V_b} = \frac{V_v}{V_v + V_r}, \quad \phi \in [0, 1)$$

where V_v is the void volume, V_r is the volume of the rock, and $V_b = V_r + V_v$ is the bulk volume of the material. The porosity is for most naturally occurring rocks in the range 0.1-0.4 (Lie, 2014). The void spaces in the rock can either be interconnected pores, which allows for flow of fluids, or it can be disconnected pores, where fluids are unable to flow. Disconnected pores are of no interest for flow simulation, and for this reason one rather consider the effective porosity. This only accounts for the void space of the interconnected pores. Furthermore, if the reservoir rock is

compressible, the porosity depends on pressure. The rock compressibility is defined

$$c_r = \frac{1}{\phi}\frac{\mathrm{d}\phi}{\mathrm{d}p} = \frac{\mathrm{d}\ln\phi}{\mathrm{d}p}$$

and the pressure-dependent porosity can be expressed

$$\phi(p) = \phi_0 e^{c_r(p-p_0)}$$

which within reservoir simulation is common to linearize

$$\phi = \phi_0\left(1 + C_r(p - p_0)\right)$$

To summarize, porosity and permeability are key parameters for the reservoir, but their spatial distribution throughout the field are often poorly known. In addition, position of seals, faults and other structural elements are also important information that is difficult to determine. Uncertainties are further discussed in Chapter 3.

1.3 Production

Put simply, the oil and gas is recovered by drilling wells into the reservoir, making it able to flow out the reservoir through the wells. The production phase starts once the *first-oil* flows through the well-bore for commercial purpose. The production phase can roughly be broken down into three periods, namely the *build-up* period, the *plateau* period, and the *decline*. During the build-up, the production from newly drilled production wells (producers) are brought on stream, slowly ramping up towards their designed capacity. The period when the facility runs at full capacity is commonly known as the plateau phase, which maintains constant production rate. New wells can also be brought on stream in this period, as other wells starts to decline in their production. The plateau usually lasts for 2-5 years (Jahn et al., 2008), but can also last longer, especially for gas-fields. Moreover, the plateu-phase can be extended by applying secondary recovery techniques. The final decline period, usually the longest in time, is the period where all production wells will exhibit declining production. The production phases are visualized in Figure 1.2.

Another common categorization of the production phase is based on the recovery techniques involved. The *primary* recovery phase denotes the period where no additional recovery techniques besides the original driving mechanism of the reservoir is required. Typical driving mechanisms are

- Natural water displacing oil downward into the well
- Expansion of the natural gas in the cap at the top of the reservoir
- Expansion of gas initially dissolved in the crude oil

- Gravity drainage resulting from the movement of oil within the reservoir from the upper to the lower parts where the wells are located

However, the pressure in the reservoir will at some point decline to a level that is not sufficient to lift the oil to the surface. As of today, the primary recovery is usually around 5-15% (Jahn et al., 2008).

The *secondary* phase attempts to maintain reservoir pressure, by supplying external energy into the reservoir. Energy is supplied by injecting water *(waterflooding)*, or gas *(natural gas re-injection)* into the reservoir. Injected water aims to maintain pressure and push the oil towards production wells. Injected gas supports the pressure in the reservoir, thus acts as an artificial driving force. Water is usually injected in the production zone, while gas is injected into the gascap. Even though external energy is supplied, most of the oil still remain trapped in the pore spaces after secondary recovery techniques has been applied. The total recovery usually stays somewhere in the range 10-50% (Jansen, 2013).

Some field also takes further measures with a *tertiary* phase, which is also known as *enhanced oil recovery* (EOR). At this stage, the goal is to increase the mobility of the oil. The mobility can be increased by injecting steam, or possibly even fire into the reservoir. An evident drawback from injecting fire is that some oil is burned during the process. Injecting surfactants is also a possible method, as surface tension is decreased. EOR techniques are usually applied to reservoirs where oil is heavier than normal crude oil.

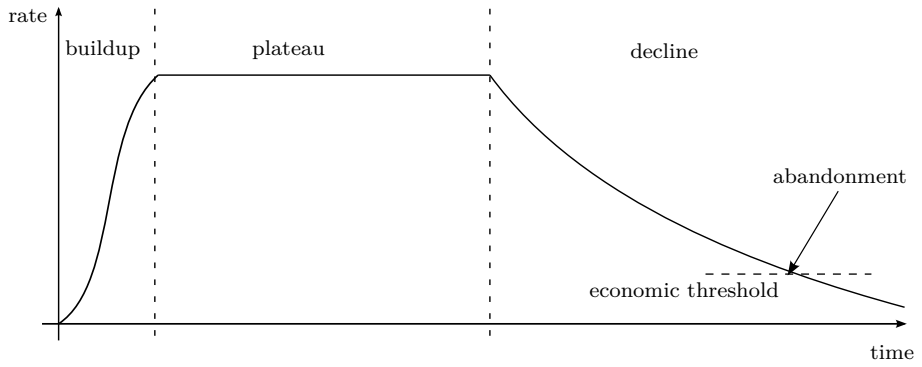


Figure 1.2: Production phases

Waterflooding

Special attention is paid to the waterflooding operation due to its importance for this thesis. The main goal for the operation is to increase oil-production rate, and ultimately the total recovery factor. This is achieved by means of voidage replacement - namely to inject water into the reservoir to replace produced fluids. The water is injected to increase or maintain reservoir pressure, and to displace oil from the pore spaces. The concept of injecting water into the reservoir appeared

in the 1920s (SPE, 2014) to dispose saline water and brine that were produced alongside with oil. However, it was quickly recognized that injecting water often resulted in increased recovery, in reservoirs were the initial natural energy had started to deplete. The fact that water is inexpensive, and often readily available also drove the development of the technology. The reason for its efficiency to displace oil were also discovered, which is due to the relative properties of water compared to oil; such as viscosity, density and wetting. The geology of the reservoir also affects the process. Waterflooding is an operation that usually takes decades to complete, and the technology is used with success on almost all reservoir types, all over the world. This include several fields on the Norwegian shelf, exemplified by the Ekofisk field.

There exist many patterns for configuring the location of wells in a waterflooding operation. One of the most common is the five-spot pattern, where four production wells (producers) are located around an injecting well (injector). The inverted five-spot pattern also appears frequently, where four injectors surround a producer. It is important for the operation that water-breakthrough is delayed as long as possible. The breakthrough occurs when injected water "breaks through" the rock, and reach a producer. This event can severely affect the oil-displacement process. However, smart-wells are often equipped with inlet control valves (ICV), to among others, handle such events. The ICV is an active component, installed as part of a well completion to partially or completely choke flow into a well. It is controlled to maintain flow conformance, and as the reservoir depletes, to stop unwanted fluids from entering the wellbore. The latter is exemplified by closing the ICV when the water-cut exceeds a certain threshold. Peters et al. (2010) showed that for a synthetic case, the optimized net present value for a flooding operation increased by 20%, using three ICV for each well instead of one. A schematic of the waterflooding operation is given in Figure 1.3. In figure 1.4, the distribution of oil-saturation over time is showed for a flooding operation simulated in MRST.

1.4 Reservoir Management

Uncertainties in the reservoir pose a great challenge, and it results from our inability to fully characterize the reservoir and its flow processes. Reservoir management is a dynamic process that aim to mitigate the effects of these uncertainties, by optimizing reservoir performance through a systematic application of integrated and multidisciplinary technologies. Reservoir management has been defined by numerous authors over the years, and the following definition is proposed by Thakur (1996).

Definition 1.4.1 (Reservoir Management)**.** *Sound reservoir-management practice relies on use of financial, technological, and human resources, while minimizing capital investments and operating expenses to maximize economic recovery of oil and gas from a reservoir. The purpose of reservoir management is to control operations to obtain the maximum possible economic recovery from a reservoir on the basis of facts, information, and knowledge.*

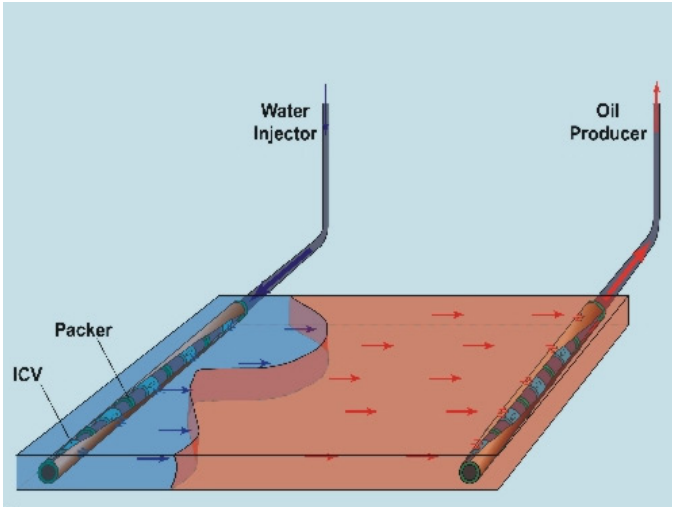


Figure 1.3: The water-flooding principle for horizontal wells (Brouwer, 2004)

Furthermore, the concept of *closed-loop reservoir management* (CLRM) has been around for years. This is often centered around attempts to improve reservoir characterizations from a geological point of view, typically achieved by history matching (Jansen, 2013). Here, measurements and previous inputs to the reservoir are used to update numerical models. Moreover, the concept is also often seen in connection with optimization of depletion strategies, such as waterflooding. Foss and Jensen (2011) argues that, on a conceptual level, CLRM can be interpreted as using real time data from multiple sources and mathematical models, to aid long-term strategic decision making, and medium term operational decision making. In this context, long term decisions include drainage strategies, technology and infrastructure development, and is considered as life-cycle optimization of the asset. Medium-term decisions involve well location, well design and targets for production rates, typically on horizons between months and years.

In addition, they consider short term decisions with horizons from days to weeks as (operational) production optimization. This is not reviewed further in this thesis, see for instance Gunnerud (2011). A decision pyramid that illustrate these layers is shown in Figure 1.5. It should be remarked that the layers are greatly interwoven, as decisions in each layer may influence others. For instance, long term optimization may impose constraints on lower-level decisions to avoid a short term strategy that harms long term recovery. Consequently, there exist no clear distinction between layers.

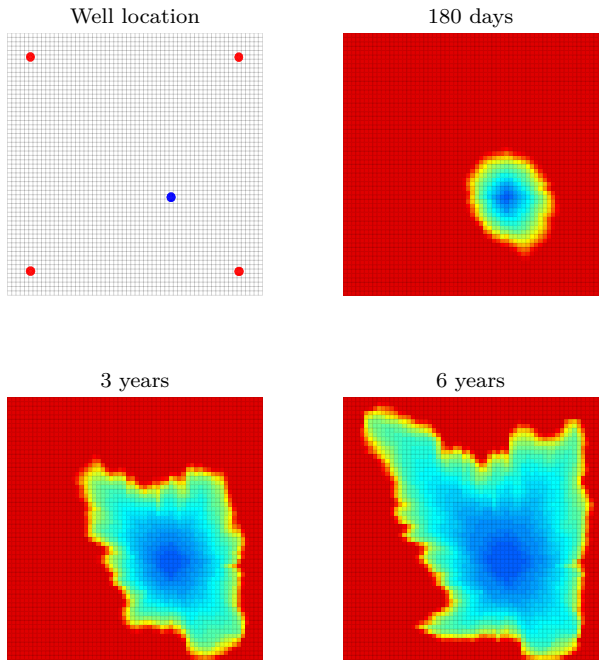


Figure 1.4: Oil-saturation distribution sampled in time, for five-spot waterflooding simulated in MRST. The model is proposed by Bellout et al. (2012). Producers are marked with a red dot, and the injector with a blue dot.

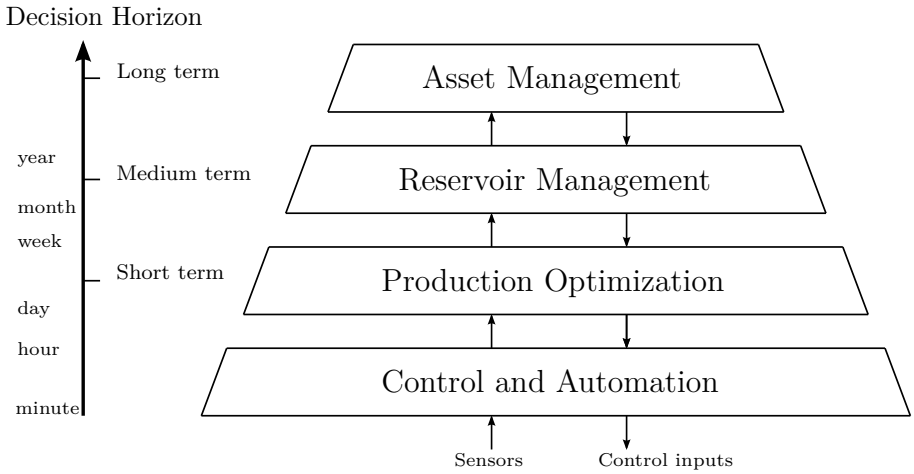


Figure 1.5: A multilevel control hierarchy illustrating the decision layers. Adapted from (Foss and Jensen, 2011)

Chapter 2

Dynamic Optimization

This chapter is concerned with theory that relates to dynamic optimization. It has been has been necessary to understand this theory in order to work with the problem description, and the tools required for doing so. In particular, it has been decisive in order to develop a single shooting based optimizer for output-unconstrained reservoir optimization, tightly interfaced to MRST. The optimizer is presented in Section 4.3.1.

The chapter is organized in the following manner; First, the concept of dynamic optimization is presented, and second, solution strategies are classified, whereas some are outlined.

2.1 Formulation

Dynamic optimization problems are typically divided into two groups, namely optimal control and parameter estimation. In both cases, the goal is to find a sequence of free variables that minimizes the objective. The objective is a functional that state the optimality of the trajectory of states, inputs and parameters. The optimal control problem deals with determining future input control signals to the system. The parameter estimation problem aims to estimate uncertain or even unknown states and parameters, based on a sequence of previous measurements and control inputs.

In either case, systems given by a fully implicit *Differential-Algebraic Set of Equations* (DAE) are considered. To keep it simple without loss of generality, the systems are specified with initial conditions given at zero ($t_0 = 0$). The *initial value problem* (IVP) is given by

$$F(z(t), \dot{z}(t), u(t), p, t) = 0 \tag{2.1a}$$

$$h(z(0)) = 0 \tag{2.1b}$$

where $z(t) \in \mathbb{R}^{n_z}$ are the states, $u(t) \in \mathbb{R}^{n_u}$ the controls, and $p \in \mathbb{R}^{n_p}$ represents model-parameters independent of time. The IVP is defined on the time-interval $t \in [t_0 \ t_f]$. The systems are also assumed to be time-invariant, such that the time variable t does not appear explicitly in the model. In general, it is quite difficult to analyze fully implicit DAEs, so instead only semi-explicit systems are considered. The reason is that they possess a nice structure, in which the states $z(t)$ are partitioned into differential variables $x(t)$ and algebraic (output) variables $y(t)$. Semi-explicit systems are still applicable for broad range of real physical processes (Binder et al., 2001). The semi-explicit IVP reads

$$\dot{x}(t) = f(x(t), y(t), u(t), p) \tag{2.2a}$$
$$x(0) = x_0 \tag{2.2b}$$
$$0 = g(x(t), y(t), u(t), p) \tag{2.2c}$$

where (2.2a) denotes the system dynamics, (2.2b) the initial condition, and (2.2c) denotes the algebraic equations.

Optimal Control Problem

The *Optimal Control Problem* (OCP) deals with finding an optimal sequence of inputs to the system over a future horizon. The length of the horizon determines how far the behavior of the system is predicted, subject to the input sequence. The prediction is based on mathematical models. As mentioned initially, the optimality of the control sequence is measured with a performance index, namely the objective function. A possible formulation for the OCP reads

$$\min_{x,y,u,p} \quad J(x(t), y(t), u(t), p) \tag{2.3a}$$

$$\begin{aligned}
\text{s.t} \quad & \dot{x}(t) = f(x(t), v(t), u(t), p, t) && \text{(2.3b)}\\
& x(0) = x^0 && \text{(2.3c)}\\
& g(x(t), y(t), u(t), p) = 0 && \text{(2.3d)}\\
& c(x(t), y(t), u(t), p) \leq 0 && \text{(2.3e)}\\
& u_l \leq u(t) \leq u_h && \text{(2.3f)}\\
& p_l \leq p \leq p_h && \text{(2.3g)}\\
& y_l \leq y(t) \leq y_h && \text{(2.3h)}\\
& x_l \leq x(t) \leq x_h && \text{(2.3i)}\\
& t \in [t_s \ t_f]
\end{aligned}$$

where (2.3a) denotes the performance-index, (2.3b)-(2.3d) the system dynamics, and (2.3e) specifies constraints on the trajectory. Furthermore, (2.3f)-(2.3i) bounds the different variables to stay within specified regions. Typically, the objective (2.3a) is given by a Bolza-functional

$$J(x(t), y(t), u(t), p) = \underbrace{E(x(t_f), y(t_f), u(t_f))}_{\text{Mayer-term}} + \underbrace{\int_{t_0}^{t_f} L((x(t), y(t), u(t)))}_{\text{Lagrange-term}} \tag{2.4}$$

which consist of two terms, namely a *Mayer*-term and a *Langrange*-term. The Lagrange term contributes to the cost over the entire horizon, while the Mayer-Term is treated as terminal cost. A control strategy that is based on the OCP is the *Model Predictive Control* (MPC). At each timestep, the MPC solves a discretized OCP, where the current state of the system is used as the initial condition (2.3c) for the OCP[1]. Once the solution of the OCP has been obtained, the first entry in control-sequence is applied to the system. At the next timestep, the initial conditions for the OCP is updated, based on current measurements from the physical system, and the problem is solved all over again. This way of continuously shifting the horizon is known as the *receding horizon* principle. The MPC is among others discussed by Foss and Heirung (2013), in which instructively merge optimization and control theory, and by Rawlings and Mayne (2009) which provide an extensive framework.

In context of reservoir management, the nonlinear model predictive control (NMPC) is a state of the art tool to plan and support waterflooding operations. This is further discussed in Section 4.1.

Moving Horizon Estimator

The second branch of dynamic optimization is the parameter estimation problem, commonly known as the *moving horizon estimator* (MHE). Whereas the OCP seeks to find a future control sequence, the MHE uses measurements from the past, together with previous control signals, to estimate unknown states and/or parameters. An important feature of the MHE is the ability to estimate both states and parameters simultaneously, on-line, especially in the presence of noise.

Kühl et al. (2011) proposes the following approach; consider the semi-explicit IVP

$$\dot{x}(t) = f(x(t), y(t), u(t), p, w(t)) \quad , \quad x(t_0) = x_0 \tag{2.5a}$$
$$0 = g(x(t), y(t), u(t), p) \tag{2.5b}$$
$$\gamma(t) = h(x(t), y(t), p) \tag{2.5c}$$

denote that this formulation include the output function γ, and also state-noise $w(t)$. The continuous model in Equation (2.5) is first transformed into a discrete model, i.e

$$x_{k+1} = F(x_k, y_k, u_k, p) + w_k \tag{2.6a}$$
$$0 = g(x_k, y_k, u_k, p) \tag{2.6b}$$
$$\gamma_k = h(x_k, y_k, p) \tag{2.6c}$$

At time t_i a horizon containing of N measurements $\{\gamma_{i-N+1}, \gamma_{i-N+2}, \ldots, \gamma_i\}$ is considered. These measurements are taken at times $t_{i-N+1} < \ldots < t_i$. Here, the length of the estimation horizon is $t_e = t_i - t_{(i-N+1)}$, where $i - N + 1 \triangleq L$ is defined

[1] Strategies for solving the OCP are outlined in Section 2.1

for ease of notation. The optimization problem of the MHE is then formulated in a least-squares sense

$$\min_{x_k, y_k, p} \quad \left(\left\| \begin{matrix} x_L - \tilde{x}_L \\ p - \tilde{p}_L \end{matrix} \right\|^2 + \sum_{k=L}^{i} ||\gamma_k - h(x_k, y_k, p)||^2 \right) \tag{2.7a}$$

$$\text{s.t} \quad x_{k+1} = F(x_k, y_k, u_k, p) \tag{2.7b}$$

$$0 = g(x_k, y_k, u_k, p) \tag{2.7c}$$

$$p^l \leq p_k \leq p^h \tag{2.7d}$$

$$y^l \leq y_k \leq y^h \tag{2.7e}$$

$$x^l \leq x_k \leq x^h \tag{2.7f}$$

where the first part in 2.7a is known as the *arrival cost*. At each time t_i, measurements are updated, and the optimization is solved all over again. The MHE is especially relevant for history matching in reservoir management. Ongoing measurements and control inputs are used together with a mathematical model of the reservoir, to estimate uncertain model parameters and reservoir states.

2.2 Solving Dynamic Optimization Problems

There exist several methods for solving DAE optimization problems. As an extension of calculus of variations, optimal control theory has become a branch of mathematical optimization for deriving optimal control laws. Optimal control laws can either be obtained by *Pontryagin's minimum principle* (Pontryagin, 1987), which is a necessary condition for an optimum, or by solving the *Hamilton-Jacobi-Bellman* (HJB) equation (Navasca and Krener, 2000). The latter is both a necessary and sufficient condition for optimality when it is solved over the entire state space. Both methods can be used to derive optimal control laws continuous in time, specified by differential equations that describe the optimal path of the controlled variable. As a results, these methods does not require a discretized mathematical model of the system. However, these methods are not suitable for reservoir optimization. The remainder of this chapter will instead focus on direct methods. To this end, the concept of non-linear programming is needed, and is therefore presented first.

2.2.1 Non-Linear Programming

In mathematics, *nonlinear programming* (NLP) is the process of solving an optimization problem. The problem is defined by a system of equalities and inequalities, over a set of unknown real variables, along with an objective function to be maximized or minimized. The variable is here given by $x \in \mathbb{R}^n$. Some of the constraints,

or the objective function, are nonlinear. A common, general formulation reads

$$\min_{x} \quad J(x) \tag{2.8a}$$

$$\text{s.t} \quad c_i(x) = 0, \quad i \in \mathcal{E} \tag{2.8b}$$

$$c_i(x) \leq 0, \quad i \in \mathcal{I} \tag{2.8c}$$

where Equation (2.8a) is the objective, Equation (2.8b) represents equality constraints, and (2.8c) inequality constraints. The reader is referred to Nocedal and Wright (1999) for further fundamental theory on mathematical optimization.

Several algorithms exists for solving NLP problems. If gradients are available, sequential quadratic programming (SQP) (Gill et al., 2002; Schittkowski, 1986), interior point (Kawajir et al., 2010) and generalized augmented Lagrangian (Nocedal and Wright, 1999, chapter 17) are different approaches used to solve NLPs. It goes without saying that gradients are crucial for these algorithms, both for runtime performance and convergence properties. Methods for obtaining gradients are evaluated in Appendix A.

NLP problems can also be solved by *derivative-free optimization* (DFO) algorithms. These algorithms are practical for problems where gradients are not available. However, DFO algorithms suffer from many drawbacks, and are usually outperformed by derivative-based algorithms[2]. As of today, there has yet to be discovered how DFO algorithms can robustly handle general constraints. Moreover, according to (Nocedal and Wright, 1999, pg. 221), derivitive-free algorithms are only effective for small-scaled problem. An example of DFO is the well-known Nelder-Mead algorithm (Nelder and Mead, 1965).

2.2.2 Direct Methods

A popular idea is to solve the dynamic optimization problem as a NLP problem. Strategies built on this idea are known as direct methods. Because problems that are continuous in time consists of infinitively many decision variables, they must be discretized in time in order for a NLP algorithm to solve the problem. The concept for all direct methods is to first discretize the problem, and then optimize in the next step using NLP algorithms. This is why direct methods are refereed to as *discretize, then optimize* methods.

Direct methods are divided into *sequential* and *simultaneous* approaches. In short, sequential methods execute optimization and simulation sequentially, while simultaneous methods execute optimization and simulation simultaneously. For applications within the process industry, Binder et al. (2001) provides an extensive description of problem formulations, and solution strategies. Indirect methods are also included.

A key distinction between sequential and simultaneous approaches is that sequential methods strictly uses manipulated variables as optimization variables, whereas

[2]If gradients exits, and can be obtained sufficiently efficient.

simultaneous methods also embeds discretized state variables as optimization variables. There also exists in-between solutions like the Multiple Shooting method, which utilizes a trade-off between the sequential and simultaneous approach. Solution strategies for DAE Optimization problems are categorized in figure 2.1. A neat summery of the approaches is provided by Binder et al. (2001):

- **Sequential simulation and optimization**
  A numerical integrations method solves the model equations exactly in every iteration of the optimization algorithm, given initial conditions and a set of controls.

- **Simultaneous simulation and optimization**
  State equations and control profiles are fully discretized. The resulting set of equations and constraints then enter the transcribed optimization problem as nonlinear constraints. These constraints are allowed to be violated during the optimization procedure. At the solution however, they need to be satisfied.

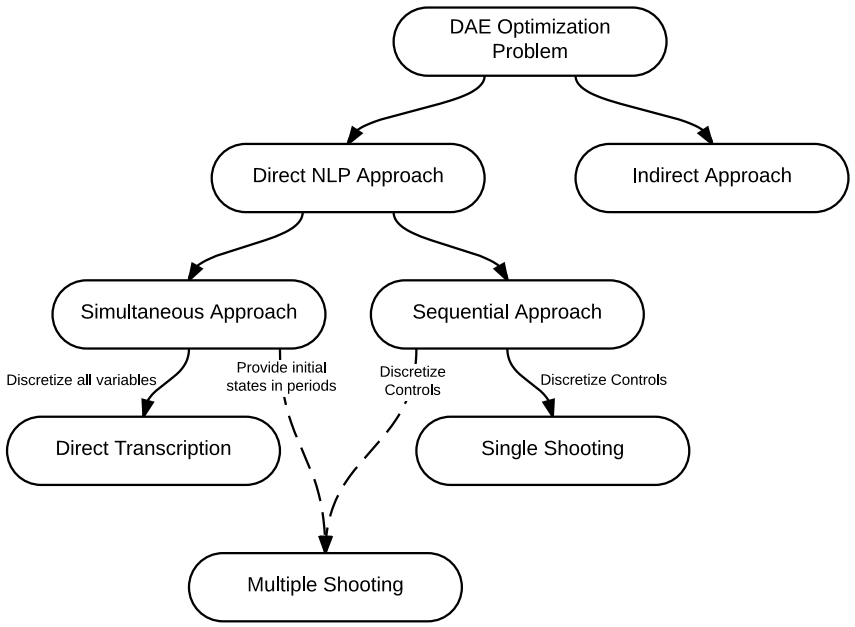


Figure 2.1: Classification of DAE Optimization strategies. Adapted from (Biegler, 2010).

An advantage for all direct methods is that they directly find good approximated solutions, feasible to the state equations. On the other hand, the quality of the methods are subject to the level of accuracy when discretizing the controls, and

possibly the states. The *single shooting* (SS), and the *multiple shooting* (MS) method are presented in the upcoming sections.

2.2.3 Single Shooting

The direct single shooting method is a sequential strategy that transforms an infinite dimensional DAE optimization problem into a finite dimensional problem. This is achieved by discretization of the control signal $u(\cdot)$. This is why the method is also referred to as *control vector parameterization* (Sargent and Sullivan, 1978). The single shooting method approximate the state-profiles by solving the DAE with a suitable numerical scheme, given a set of parameterized controls. Consequently, the states can be viewed as a function of the controls only, such that state-variables are eliminated as decisions-variables in the OCP. The SS-algorithm sequentially optimizes the parameterized controls, and solves the DAE. This procedure is illustrated in figure 2.2. Denote that in practice, the DAE is often embedded into a dedicated simulator, such that it is necessary to interface the NLP algorithm with the simulator.

The first step when deriving the method is to divide the entire time-horizon $[t_s \;\; t_f]$

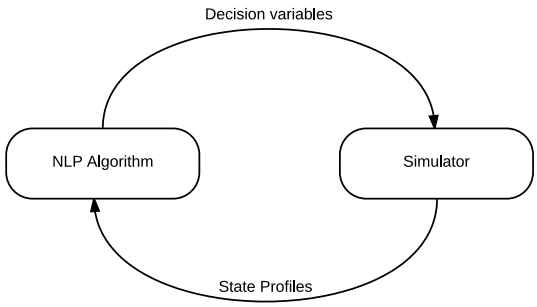


Figure 2.2: The sequential strategy for the single shooting method. At each iteration, a set of intermediate decision variables are generated by the NLP. These are given to a suitable numerical DAE solver (the simulator), which compute the state-profiles. The state profiles are then again fed back to the NLP, which computes a new set of decision variables. In addition, the simulator must provide gradients if it is required by the NLP algorithm.

into smaller segments

$$t_s = t_0 < t_1 < ... < t_{N-1} < t_N = t_f \tag{2.9}$$

For ease of notation, let the interval $I = [t_0 \;\; t_N]$ denote the entire horizon, and let the interval $I_i = [t_i \;\; t_{i+1}]$ denote a local interval. Next, the controls must be discretized. To this end, a parameterization on the form $\tilde{u}(t, q)$ is used, which depends on the control-parameter vector $q \in \mathbb{R}^{n_q}$. The simplest parameterization

is piecewise constant controls, which yields

$$\tilde{u}(t, q^0, q^1, \ldots, q^{n-1}) \triangleq q^i, \quad t \in I_i \tag{2.10}$$

This type of parameterization also allow for more complex control trajectories. However, only piecewise constant controls are considered in this thesis. In the OCP defined in Equation (2.3), the objective function is comprised by state variables, algebraic variables, and control inputs. As mentioned, state profiles are obtained by solving the DAE based on the control-vector, such that the objective may be viewed as a function of the controls only. Therefore, both the constraints and objective may be seen as implicit functions of $u(\cdot)$. As such, the single shooting approach allows for the compact NLP formulation

$$\min_{\tilde{u}} \quad J(\tilde{u}) \tag{2.11a}$$

$$\text{s.t} \quad c_i(\tilde{u}) = 0, \quad i \in \mathcal{E} \tag{2.11b}$$

$$c_i(\tilde{u}) \leq 0, \quad i \in \mathcal{I} \tag{2.11c}$$

where $u \in \mathbb{R}^{Nn_u}$. An appropriate NLP algorithm is used to solve the problem. The single shooting approach leads to a small, unstructured problem, as the problem is reduced in space, by eliminating state and algebraic variables as decision variables.

A drawback for the SS method is that it might not be suitable for unstable systems, as state-profiles may be unbounded. In addition, as simulation is performed coherently from t_0 until t_f, it is not possible to utilize parallelism to speed up simulation. Parallelism is desirable for large-scale systems where simulation is computationally expensive. On the other hand, the single shooting approach is often simple to implement, and easy to understand. An illustration of the method is shown in Figure 2.3.

2.2.4 Multiple Shooting

A major drawback for the SS approach is that it may struggle when encountering unstable, or even poorly conditioned dynamic systems (Biegler, 2010). In particular, this may lead to one of the three following issues; state-profiles that explodes, failure of the DAE solver, or that the NLP solver experience troubles with convergence. An elegant solution to this problem is to incorporate more state variables into the NLP formulation - which is the idea behind the multiple shooting method. The method is discussed in several papers, among others (Biegler, 2007), (Biegler et al., 2002) and (Leineweber et al., 2003).

A relatively simple description of the method is proposed by Biegler (2010), which starts out by dividing the time-interval into smaller segments. Unlike pure sequential approaches, MS solves an independent IVP for each shooting interval, based on the DAE together with guessed initial conditions. Control-signals are treated in the same manner as with SS, thus a control-vector parameterization is still applied. In addition, new equality constraints must be included to enforce that the initial condition at each interval is equal to the state at the end of the trajectory at

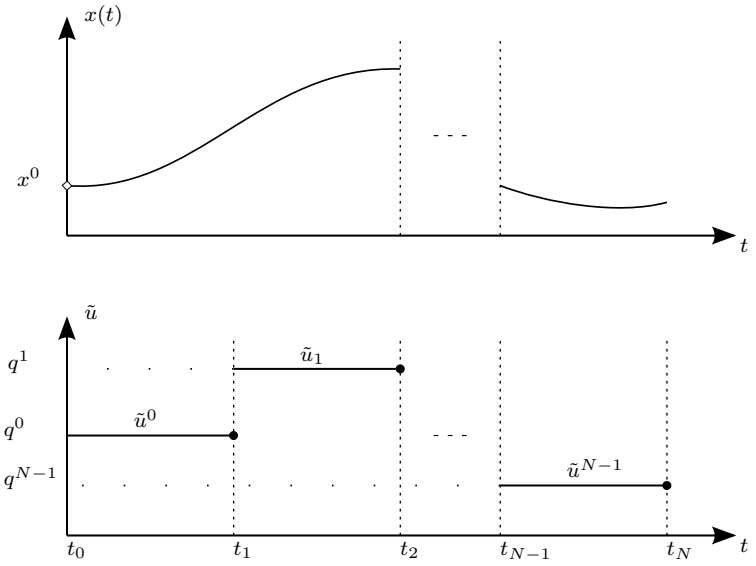


Figure 2.3: An illustration of a single shooting simulation. Here, both the controls and the states are scalars, thus $x, q_i \in \mathbb{R}$. The states profile (illustrated in the upper graph) is found by solving the DAE, given a set of parameterized controls (lower graph).

the previous interval. However, infeasible path algorithms may utilize that these constraints can be violated during intermediate iterations. A possible multiple shooting formulation reads

$$\min_{\tilde{u}} \quad J(\tilde{u}) \tag{2.12a}$$

$$\begin{aligned}
\text{s.t} \quad & x^{i-1}(t_{i-1}) = x_0^i, && i = 2, \dots, N && \text{(2.12b)}\\
& x^i(0) = x_0^i && && \text{(2.12c)}\\
& u_L^i \le u^i \le u_U^i, && i = 1, \dots, N && \text{(2.12d)}\\
& x_L^i \le x^i(t_i) \le x_L^i, && i = 1, \dots, N && \text{(2.12e)}\\
& \tilde{u}_L^i \le \tilde{u}^i(t_i) \le \tilde{u}_L^i, && i = 1, \dots, N && \text{(2.12f)}
\end{aligned}$$

Together with the DAE system

$$\begin{aligned}
\dot{x}^i(t) &= f^i\left(x^i(t), y^i(t), \tilde{u}\right) && \text{(2.12g)}\\
x^i(t_{i-1}) &= x_0^i && \text{(2.12h)}\\
g^i\left(x^i(t), y^i(t), \tilde{u}\right) &= 0, && \text{(2.12i)}\\
t &\in (t_{i-1}, t_i], \quad i = 1, \dots, N
\end{aligned}$$

where $\tilde{u} \in \mathbb{R}^{Nn_u}$, Equation (2.12a) is the objective, (2.12b) the new equality constraints that enforces continuity across intervals, (2.12c) initial conditions, and (2.12d) - (2.12f) bounds the variables $\tilde{u}, y, x$. Furthermore, Equation (2.12g) denotes the systems differential equations partitioned into intervals, (2.12h) gives

initial conditions for each interval, and (2.12i) are algebraic equations partitioned into intervals. A relative simple illustration of the Multiple Shooting method is shown in Figure 2.4. The multiple shooting approach leads to a large, but struc-

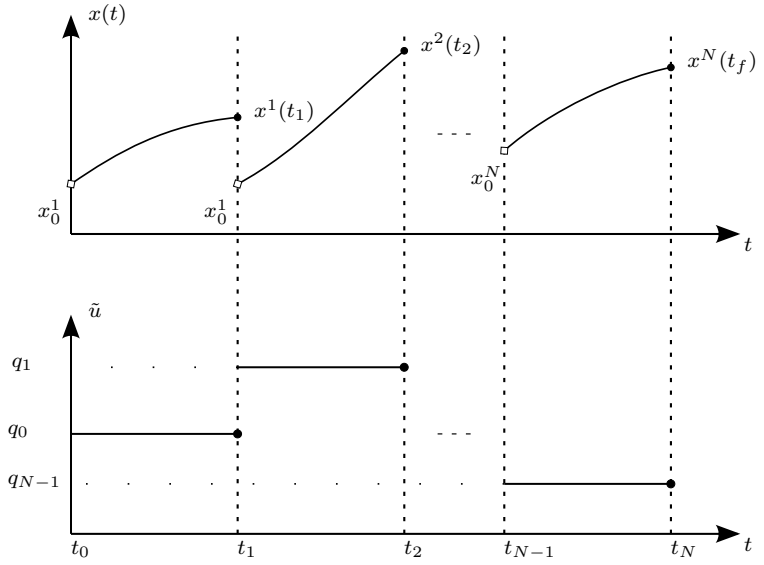


Figure 2.4: Multiple Shooting applied to an unstable system. Adapted from (Biegler, 2010)

tured problem.

Chapter 3

Reservoir Simulation

Reservoir simulation is a field within engineering aiming to simulate and predict how fluids flows through porous media. A reservoir simulator is here a tool for evaluating large-scale numerical models describing this subsurface flow. Simulation of reservoirs started in mid 1950s (Lie, 2014), and the technique was to a large degree developed during the 1970s and 1980s (Carlson, 2003). At the time, the technique was the newest reservoir tool, and consequently it encountered a great deal of skepticism. The technique has however matured over the years, and the scepticism has slowly diminished. Reservoir simulation is today an important tool for oil companies, providing qualitative and quantitative prediction of fluid flow.

3.1 Background

A major obstacle for reservoir engineering is the inability to directly reach the reservoir, due to its location beneath the surface. The reservoir can be anything between a small pocket of oil located right beneath the surface, to a huge reservoir that stretches across several square kilometres, often under remote seas (Lie, 2014). In contrast to topside facility equipment, the reservoir cannot be directly observed nor manipulated. This poses a great challenge, because the reservoir properties can only be determined to a certain degree. The information that can be assembled from the reservoir with today's technology, often restricts to seismic surveys, measurements obtained from drilled wells, and rock samples. Unfortunately, this information is highly uncertain. For instance, information obtained from a well is only valid in a small region around the well, and the uncertainty increases with the distance from the well. Seismic surveys can have more than one solution to a particular set of data. Furthermore, as mentioned in Chapter 1, the spatial distribution of important parameters such as porosity and permeability are also often poorly known. The true model of the reservoir is consequently impossible to construct, and the goal should rather be to keep the models as accurate as possible.

Lie (2014) argues that the reservoir models used in the early days of simulation usually were a Cartesian two-dimensional grid, with somewhere around $10^2 - 10^3$

gridcells building the whole reservoir. As of today, the technology is able to handle models with gridcells down to the meter scale, which can lead to models consisting of millions of cells. He further argues that stratigraphic grids building the volumetric descriptions is the industry standard, but other more complex models based on unstructured grid experience increasing popularity. See his publication for an explanation of various types of grids.

Reservoir simulation has numerous areas of application, relevant for all phases of the reservoir life cycle. In the preproduction phase, simulators can be used as a supporting tool for determining the number of wells, and their location within the field. They are also employed to help maximizing economic recovery, by generating long-term depletion strategies for the reservoir. Bellout et al. (2012) even proposes a joint optimization method for well placement and optimizing controls. Other typical usages during preproduction is to help determining the *original oil in place* (OOIP) and *original gas in place* (OGIP), figures which are of great significance for long-term field development decisions. This can be exemplified by selection and scaling of top-side facility equipment.

During the production phase, the operator typically seeks to maximize *net present value* (NPV), delaying water-break through, maintaining reservoir pressure and monitoring *gas to oil ratio* (GOR). Moreover, one is often interested in estimating uncertain parameters through history matching. Reservoir simulators can be helpful tools for achieving these goals.

Reservoir simulation is dependent upon reliable reservoir models in order to provide meaningful results. The conceptual model of the reservoir should include every relevant aspect of the reservoir, such as its shape and location, fluids contained in the field, properties of the reservoir rock, the natural driving mechanisms, and visualizations of flow patterns. However, reservoir properties changes with time, which should be accounted for in the model. Good simulation results requires continuous updates based on newly gathered information. From time to time this may even require that the entire model must be discarded, and a new model must be built up from scratch.

3.2 Mathematical Modeling

This section is based on the two-phased isothermal flow model proposed by Lie and Mallison (2013), and intends to briefly introduce mathematical modeling of flow through porous medium. The two-phase model accounts for the fact that the reservoir is filled with both hydrocarbons and water. The model could also be extended to a multiphase model, where additional phases like a gaseous phase, could be included. The fluids are referred to as phases if they are immiscible, and separated by a sharp interface. A two-phase system is commonly divided into a wetting[1] and a non-wetting phase. This is on a micro-scale given by the contact angle between the solid surface and the fluid-fluid interface. On a macro-scale

[1] Wetting denotes the ability of a fluid to maintain contact with a solid surface.

however, both phases are assumed to be present at the same location. To this end, the term *saturation* is used to denote the volumetric fraction that is occupied by each phase. The saturations must always sum to unity, thus $\sum_i S_i = 1$. For the two-phased system composed of the wetting and non-wetting phase, this yields $S_w + S_n = 1$. Now, a multiphase extension of Darcy's laws founds the basis as the first basic equation for reservoir modeling

$$\vec{v}_\alpha = -\frac{K k_{r\alpha}}{\mu_\alpha} \left(\nabla p_\alpha - \rho_\alpha \vec{g} s\right) \tag{3.1}$$

where $\alpha \in \{w, n\}$ denotes the wetting and non-wetting phases, $\vec{v}_\alpha$ is the superficial fluid velocity, K is the permeability, $k_{r\alpha}$ the relative permeability, μ the fluid viscosity, $\vec{g}$ the gravity vector, ρ density and p pressure. The second basic equation is the mass conservation of each phase

$$\frac{\partial(\rho_\alpha S_\alpha \phi)}{\partial t} + \nabla \cdot (\rho_\alpha \vec{v}_\alpha) = q_\alpha \tag{3.2}$$

where q_α denotes the fluid source/sink term that represents the wells which either extract or inject fluids. Furthermore, interfacial tension may cause pressure difference between the phases, known as the capillary pressure

$$p_{cnw} = p_n - p_w$$

The capillary pressure is on a macro-scale often assumed to be a function of saturation. It is common to reformulate the basic equations to flow equations for fluid pressure, and transport equations for saturations, to help reveal their nature. A manipulation of the equations leads to a system for one phase pressure and one saturation. The capillary pressure appears explicitly in the latter, and the resulting equations are nonlinear and strongly coupled. The coupling can be reduced by introducing a global pressure $p = p_n - p_c$. The complementary pressure p_c contains saturation dependent terms, which is defined

$$\nabla p_c = f_w \nabla p_{cnw}, \qquad f_w = \frac{\lambda_w}{(\lambda_w + \lambda_n)}$$

where f_w is a dimensionless fractional-flow function that measures the fraction of the total flow that contains the wetting phase. This is defined from the phase mobilities

$$\lambda_\alpha = \frac{k_{r\alpha}}{\mu_\alpha}$$

For the incompressible and immicisble case, the basic equations can now be expressed in a so-called fractional form, which consists of an elliptic pressure equation

$$\nabla \cdot \vec{v} = q, \qquad \vec{v} = -K\left(\lambda_n + \lambda_w\right) + \nabla p + K(\lambda_W \rho_w + \lambda_n \rho_n)\vec{g} \tag{3.3}$$

where the total velocity reads $\vec{v} = \vec{v}_n + \vec{v}_w$. The second basic equation becomes the following parabolic saturation equation

$$\phi \frac{\partial S_w}{\partial t} + \nabla \cdot f_w(S_w)\left[\vec{v} + K\lambda_n\left(\rho_w - \rho_n\right)\vec{g} + K\lambda_n \nabla p_{cnw}\right] = \frac{q_w}{\rho_w} \tag{3.4}$$

3.2.1 Multiphase, Multicomponent Flow

The two-phase equations describing immiscible flow may easily be extended to include more phases. However, this leads to issues where parameters like relative permeability can be more challenging to define. Each phase may consist of more than one single chemical specie, which are grouped into fluid components. The basic conservation laws are expressed for each component ℓ because components may transfer between phases and change the composition

$$\frac{\partial}{\partial t}\left(\phi \sum_{\alpha} c_{\alpha}^{\ell} \rho_{\alpha} S_{\alpha}\right) + \nabla \cdot \left(\sum_{\alpha} c_{\alpha}^{\ell} \rho_{\alpha} \vec{v}_{\alpha}\right) = \sum_{\alpha} c_{\alpha}^{\ell} q_{\alpha} \tag{3.5}$$

c_{α}^{ℓ} denotes the mass fraction of component ℓ in phase α, ρ_{α} is the density of phase α, $\vec{v}_{\alpha}$ is phase velocity, and q_{α} is phase source. However, additional equations are needed to the model, and closure relations such as PVT models and phase equilibrium conditions are needed for specific fluid systems. Different choices for closure relationships are appropriate for different reservoirs and different recovery mechanisms.

3.2.2 Black-Oil Model

The black-oil model is the most common model within reservoir simulation. This model uses a simple PVT description, where chemical species are lumped together to form two components at surface conditions. The first component contains the heavy hydrocarbon component, the oil, while the second contains the light component, the gas. For these lumped components the chemical composition remain constant for all times. Denote that the lightweight component may partially, or completely, be dissolved in the oil phase at reservoir conditions. However, the liquid and vapour phases of hydrocarbons do not dissolve in the water phase. For many reasons, the black oil models are often formulated as conservation of volumes at standard conditions, rather than conservation of component masses. This is achieved by introducing the formation volume factors

$$B_{\alpha} = \frac{V_{\alpha}}{V_{\alpha s}}$$

where V_{α} is the volume occupied by component α at reservoir condition, and $V_{\alpha s}$ denotes the same volume at surface conditions. Similarly, the gas solubility factor is given by

$$R_{so} = \frac{V_{gs}}{V_{os}}$$

and represents the the volume of the gas measured at standard conditions, dissolved at reservoir conditions in a unit of stock-tank oil (at surface conditions). The resulting conservation laws then reads

$$\frac{\partial}{\partial t}\left(\frac{\phi \rho_s^{\alpha}}{B_{\alpha}} S_{\ell}\right) + \nabla \left(\frac{\rho_s^{\alpha}}{B_{\alpha}} \vec{v}_{\ell}\right) = q^{\alpha} \qquad \alpha \in \{o, w\} \tag{3.6a}$$

$$\frac{\partial}{\partial t}\left(\frac{\phi\rho_s^g}{B_g}S_g + \frac{\phi R_{so}\rho_s^g}{B_o}S_\ell\right) + \nabla\cdot\left(\frac{\rho_s^g}{B_g}\vec{v}_g + \frac{R_{so}\rho_s^g}{B_o}\vec{v}_\ell\right) = q^g \tag{3.6b}$$

3.2.3 Well Model

The last important piece of the puzzle is the well model, which couples the reservoir to the rest of the facility, by representing flow of fluids in and out of the reservoir. Today, wells are built in any thinkable vertical and horizontal layout, but is in the simplest form a vertical hole that penetrates the reservoir, in which fluids can flow in and out. A well that penetrates several sedimentary layers in the reservoir can also introduce cross-flow between layers. The well model intends to accurately represent the flow through the wellbore, and should provide equations that relate the pressure in the wellbore to the flow. In this way, the equations should compute the flow if the pressures are known, and oppositely, compute the pressure if the flow is known.

A volumetric discretization of the flow equations leads to a significant difference between the average pressure in the perforated grid-block and the pressure in the wellbore. As the size of the wellbore is relatively small compared to the grid-block, a large pressure gradient appears in a small region inside the perforated block. The common approach is to use either an analytical or a semi-analytical solution on the form

$$-q = \mathrm{WI}(p_b - p_{wb}) \tag{3.7}$$

that relates wellbore pressure p_{wb} to the numerically computed average perforated grid-block pressure p_b and the flow q. The term WI is the well index, representing geometric characterizations of the well, and properties of the surrounding rock. According to (Lie and Mallison, 2013) the most used well model assumes steady state radial flow, and a 7-point finite difference characterization. For an isotropic medium, on a Cartesian grid with cells $\Delta x \times \Delta y \times \Delta z$, this reads

$$\mathrm{WI} = \frac{2\pi K\Delta z}{\ln r_0/r_w}, \qquad r_0 = 0.14(\Delta x^2 + \Delta y^2)^{\frac{1}{2}}$$

where K is the permeability, r_w the radius of the well and r_0 the effective block radius, at which the steady state pressure equals the computed block pressure.

3.2.4 Discretization

In order to numerically evaluate the reservoir equations on a computer they must be discretized in both time and space. Commercial simulators often use fully implicit discretization to solve the nonlinear system (3.6), but there also exists sequential methods. These methods can vary in choice of primary unknowns, linearizion, temporal and spatial discretization, and also the order of which the different steps are applied to derive the set of discrete equations. An example of the latter is the

implicit pressure, explicit saturation (IMPES) method which is an experimental solver for compressible, miscible black-oil type flow and transport (Lie and Mallison, 2013). For further information regarding the IMPES method, and additional information on typical explicit and implicit Euler discretization for reservoir simulation, see (Jansen, 2013, pg. 90-99). See (Gravdahl and Egeland, 2002, part 5) for general numerical simulation.

3.3 MRST

The *MATLAB Reservoir Simulation Toolbox* (MRST) is an open-source reservoir simulator, developed and distributed by SINTEF Applied Mathematics, under the terms of GNU General Public License (GPL). Though it initially was intended as a toolbox for rapid prototyping, demonstration of new simulation methods, and modeling concepts on unstructured grids, it has been applied to large and complicated grids with success. However, it is not intended to be as realistic and comprehensive as commercial simulators like Eclipse (Schlumberger), Nexus (Haliburton) and CMG Suite (CMG). An important drawback for most commercial simulators is that the source code is not available for the user, implying that the simulator must be treated as a black-box, containing possibly unknown logic. This is not ideal from a research point-of-view. On the other hand, it is of great significance for reproducibility of research, and for interpreting results, that MRST is open-source.

MRST consists of two main parts. The first part is the core, which offers the following basic functionality (Lie et al., 2012)

- **Grids**: *a common data structure and interface for all types of grids.*
- **Parameters**: *a data structure for petrophysical parameters, common interface for fluid models, routines for setting and manipulating boundary conditions, source/sinks, well models etc.*
- **Units**: *MRST works in strict SI-units, but supports conversion to and from other unit system such as field units. Unless reading from en Eclipse input format, the user is responsible for explicit conversion and consistency of units.*
- **Reservoir state**: *data structure for pressure, fluxes and saturations.*
- **Postprocessing**: *visualization routines for scalar cell and face data, and also well information.*
- **Solvers**: *the toolbox contains several flow and transport solvers.*
- **Linear algebra**: *MRST relies on MATLAB's built-in linear solvers, but these can also be replaced by specialized solvers.*

The second part consist of add-on modules, such as (Lie et al., 2012)

- **Deck reader**: *contains support for input of complete simulation decks in the ECLIPSE format, including input reading, conversion to SI units, and construction of MRST objects for grids, fluids, rock properties, and wells.*

- **Adjoint formulations**: *implements strategies for production optimization based on adjoint formulations.*
- **Fully implicit solver**: *This module contains a set of fully implicit solvers for a variety of flow problems, and uses automatic differentiation to calculate Jacobians.*

In order to describe the subsurface flow-process, MRST use two models (Lie et al., 2012). The first is a mathematical flow model that describes how fluid flows through a porous medium, consisting of partial differential equations based on mass conservation of fluid phases, alongside with necessary constitutive equations. The second is a geological model that describes the reservoir, which is built up as a grid, where each grid-cell is populated with geophysical properties. The properties in the geological model are used as inputs to the flow model. The two models, in combination with models of the wells, builds up the entire simulation model.

MRST also support automatic differentiation, such that derivatives can be computed efficiently by either forward or adjoint sensitivity, outlined in Appendix A. Efficient computation of gradients is especially relevant for gradient-based optimization and history matching, applications that are important for most simulators.

Control switching is implemented as a default option, in order to calculate feasible simulations with respect to well constraints, such as limitations on wellbore pressure and maximum-flow. The concept of control switching is further discussed in Section 4.2 as an instance of reactive control. However, there are only certain constraints that can be specified in MRST, which is

- Maximum pressure for injectors
- Maximum rate for injectors, phase specific or for total liquid rate.
- Minimum pressure for producers
- Maximum rate for producers, phase specific or for total liquid rate.

For the work in this thesis, it is unfortunate that minimum rate constraints on producers and injectors cannot be specified, because it negatively impacts the heuristic presented in Section 4.3.2. The heuristic uses control switching in MRST to handle output constraints.

3.3.1 Two-Phase Black-Oil Flow Model

The case studies in Chapter 5 are only concerned with two-phased black oil models, consisting of a liquid oil-phase and an aqueous water phase. For a comprehensive derivation of different flow-models used in MRST, the reader is referred to the original publications (Lie, 2014), (Krogstad et al., 2015) and (Lie et al., 2012). However, Codas et al. (2015) provides a neat summary of the two-phase black-oil

flow model that is used for the cases in Chapter 5, which the remainder of this chapter is based on. The two-phase model can be derived by considering the mass conservation principle, and Darcy's law together with the capillary pressure. The continuous-time model reads

$$\frac{\partial}{\partial t}\left(\frac{\phi S_\alpha}{B_\alpha}\right) = \nabla \cdot (T_\alpha \nabla \Phi_\alpha) + \frac{q_\alpha}{B_\alpha}, \qquad \alpha \in \mathcal{P} \tag{3.8a}$$

$$T_\alpha = \frac{k_{r,\alpha}}{\mu_\alpha B_\alpha} k, \qquad \alpha \in \mathcal{P} \tag{3.8b}$$

$$\Phi_\alpha = p_\alpha - \frac{\rho_\alpha}{B_\alpha} \|g\| z, \qquad \alpha \in \mathcal{P} \tag{3.8c}$$

$$S_w + S_o = 1 \tag{3.8d}$$

$$p_{c,w} = p_o - p_w \tag{3.8e}$$

The nomenclature for the symbols in Equation (3.8) is listed in Table 3.1.

Table 3.1: Nomenclature for two-phase flow model

Symbol	Definition
$\mathcal{P} \in \{o, w\}$	The set of phases denoted as subscript, oil and water respectively
ϕ	Rock porosity
S_α	Fluid saturation
B_α	Fluid formation volume factor
q_α	Volumetric flow rate
$k_{r,\alpha}$	Fluids relative permeability
μ_α	Fluid viscosity
p_α	Absolute pressure of fluid
ρ_α	Density at standard conditions
$\|\mathbf{g}\|$	Gravity absolute value
z	Height, increases in the same direction as gravity
$p_{c,w}$	Capillary pressure between phases
$\mathbf{T}_\alpha$	Fluid transmissibility
λ_α	Fluid mobility ($= \frac{k_{r,\alpha}}{\mu_\alpha B_\alpha}$)

Furthermore, the reservoir porous medium is denoted by $\Omega \in \mathbb{R}^3$, in which the model (3.8) is valid. It is assumed that ϕ, B_p and μ_p depend on p_o, while $k_{r,p}$ and $p_{c,w}$ depend on S_w. The flow q_p through the wells is a boundary condition, given as a rate itself, or as well bottom hole pressures along with well equations, relating both variables. The solution of a reservoir simulation are the functions p_o and S_w, on the reservoir domain Ω, and on the time range $(t_o, t_f]$, given initial conditions $p_o(t_0)$ and $S_w(t_0)$.

3.3.2 Discretization

As discussed in Section 3.2.4, the model defined in Equation (3.8) must be discretized in order to be numerically evaluated on a computer. To this end, it is discretized in space by the *Control Volume Finite Element method* (CVFE), while the implicit integration scheme *Backward-Euler* (Gravdahl and Egeland, 2002) ensures discretization in time. Furthermore, wells are linked to the reservoir by including discrete well equations. The discrete model reads (Codas et al., 2015)

$$0 = \frac{1}{t_k - t_{k-1}} \left(\left[\frac{\phi^i S^i_\alpha}{B^i_\alpha} \right]_k - \left[\frac{\phi^i S^i_\alpha}{B^i_\alpha} \right]_{k-1} \right) - \left[\sum_{j \in N(\Omega_i)} \lambda^{i,j}_\alpha T^{i,j} (\Phi^j_\alpha - \Phi^i_\alpha) + \frac{q^i_\alpha}{B^i_\alpha} \right]_k \tag{3.9a}$$

$$0 = \frac{q^i_{\alpha,k}}{B^i_{\alpha,k}} - \left[\sum_{w \in \mathcal{W}^i} W^I_{w,i} \lambda^{w,i}_\alpha \left(p^w_{\text{bh}} - p^i_\alpha - \rho^i_\alpha \left\| g \right\| \left(z^w_{\text{bh}} - z^i \right) \right) \right]_k \tag{3.9b}$$

$$0 = 1 - \left(S^i_w + S^i_o \right)_k \tag{3.9c}$$

$$0 = \left(p^i_{c,w} - p^i_o + p^i_w \right)_k \tag{3.9d}$$

The nomenclature for the symbols used in (3.9) is listed in Table 3.2. The discretized system has four variables describing the reservoir conditions, namely the saturations $S^i_{w,k}$, $S^i_{o,k}$ and pressures $p^i_{o,k}$, $p^i_{w,k}$, for each step k and each block i. In addition, one variable describes the well BHP. Using Equation (3.9b)-(3.9d), the variables $p^i_{w,k}$, $S^i_{o,k}$ and $q^i_{p,k}$ are placed in (3.9a). This is possible due to the approximation of the density in the well tubing in (3.9b) by the density of the perforation grid-block. Thus, at each time instant, and in each grid block, one are left with two equations and two variables ($p^i_{o,k}$ and $S^i_{w,k}$) describing the grid block conditions, and $|\mathcal{W}|$ variables (p^w_{bh}) describing well conditions.

Table 3.2: Nomenclature for the discretized two-phase flow model (3.9)

Symbol	Definition
k	Subscript, the time step in which the expression is evaluated
$i \in \mathcal{G}$	Superscript, expresses averaging on the corresponding variable in the grid block i.e $\phi^i = \int_{\Omega_i} \phi \mathrm{d}w / \int_{\Omega_i} 1 \mathrm{d}w$
$\mathcal{G}$	The set containing all grid blocks that constitute the reservoir domain Ω
$N(\Omega_i)$	The set of neighboring grid block to Ω_i
$\lambda_\alpha^{i,j}$	Upstream mobility, where $j \in N(\Omega_i)$. $\lambda_\alpha^{i,j} = \lambda_\alpha^j$ if $\Phi_\alpha^j > \Phi_\alpha^i$, otherwise $\lambda_\alpha^{i,j} = \lambda_\alpha^i$
$T^{i,j}$	Constant that depends on block geometry
$\mathcal{W}^i$	A set which can have cardinality 0 or 1, containing the well with a perforation in the grid block i
$W_{w,i}^I$	Productivity index, dependent on the geometry of the well w in the grid block i
$\lambda_\alpha^{i,j}$	Well perforation upstream mobility. It is the mobility of the associated perforation grid block if the well is a producer, or the mobility if the injected fluid at the perforation grid block condition if the well is an injector
p_{bh}^w	Well bottom hole pressure measured at height z_{bh}^w

3.3.3 Implicit Simulation Step

First, let $x_k = (p_o^i, S_w^i)_k$ define the discrete reservoir state, where $i \in \mathcal{G}$, and let $y_k = (q_\alpha^w, p_{\mathrm{bh}}^w)_k$ define the algebraic output variables, where $\alpha \in \mathcal{P}$ and $w \in \mathcal{W}$. Given initial conditions x_{k-1}, the solution of the simulation step must satisfy the following DAE

$$0 = R_c(x_{k-1}, x_k, y_k) \tag{3.10a}$$

$$0 = q_{\alpha,k}^w - \sum_{i \in \mathcal{G}^w} q_{\alpha,k}^i \qquad \alpha \in \mathcal{P},\, w \in \mathcal{W} \tag{3.10b}$$

$$0 = B(y_k, u_k) \tag{3.10c}$$

where

- R_c in Equation (3.10a) equals (3.9a) after substituting in (3.9b)-(3.9d)
- Equation (3.10b) links the flow variables in y_k to their corresponding perforation flow variables. The well perforation flow variables, obtained by (3.9b), are related by their common well BHP. This is represented in an aggregated way as $0 = Q_w(x_k, y_k)$

- Equation (3.10c) implements one equality constraint for each well. It links well variables y_k to the well control target u_k. A well control target is usually a fixed BHP or flow during a time-step, therefor, these equations are linear equalities.
- There are $|\mathcal{P}||\mathcal{G}| + |\mathcal{P}||\mathcal{W}| + |\mathcal{W}|$ variables, and the number of equations are equal.

In each timestep, MRST solves (3.10) with Newton steps that yields

$$\begin{aligned} -R_c^l &= \frac{\partial R_c^l}{\partial x^l}\Delta^l x + \frac{\partial R_c^l}{\partial y^l}\Delta^l y \\ -q_{\alpha,k}^{l,w} + \sum_{i\in\mathcal{G}^w} q_{\alpha,k}^{l,i} &= \frac{\partial Q_w^l}{\partial x^l}\Delta^l x + \frac{\partial Q_w^l}{\partial y^l}\Delta^l y \\ -B^l(y_k, u_k) &= \frac{\partial B^l}{\partial y^l}\Delta^l y \end{aligned}$$

For ease of notation, the set of equations described in Equations (3.10) are compressed into

$$0 = R(x_{k-1}, x_k, v_k, u_k) \tag{3.11}$$

The Newton-system can then be put as the following matrices

$$-R^l = \begin{bmatrix} \frac{\partial R_c^l}{\partial x^l} & \frac{\partial R_c^l}{\partial y^l} \\ \frac{\partial Q_w^l}{\partial x^l} & \frac{\partial Q_w^l}{\partial y^l} \\ 0 & \frac{\partial B^l}{\partial y^l} \end{bmatrix} \begin{bmatrix} \Delta^l x \\ \Delta^l y \end{bmatrix} \tag{3.12}$$

the superscript variable l introduced in the latter equations represent the Newton iteration number. For ease of notation, subscript k is removed. The variable Δ represents the size of the Newton step. Furthermore, it is assumed that the Jacobian $\left[\frac{\partial R_k}{\partial x_k} \ , \ \frac{\partial R_k}{\partial v_k}\right]$ is non-signular, a property which can be ensured by proper parameterization of the fluid model and reservoir discretization.

As most of the time for a reservoir optimization procedure is used to solve (3.12), it is according to (Codas et al., 2015) important to tune the linear solvers correctly, and to configure automatic differentiation procedures correctly, to exploit the sparsity of the Jacobian $\left[\frac{\partial R_c^l}{\partial x^l}\right]$.

Chapter 4

Handling Constraints in Reservoir Optimization

The overall goal for this thesis is to evaluate methods that deal with constraints in optimization for reservoir management. The particular optimization problem of interest is to manage well-trajectories in a medium term waterflooding operation, by determining targets for pressures and rates. Three approaches are outlined and reviewed in this chapter, which are constrained and unconstrained control optimization, the reactive control strategy, and methods with simulator-embedded constraints. In addition, an in-between strategy of unconstrained optimization and simulator-embedded constrained is developed.

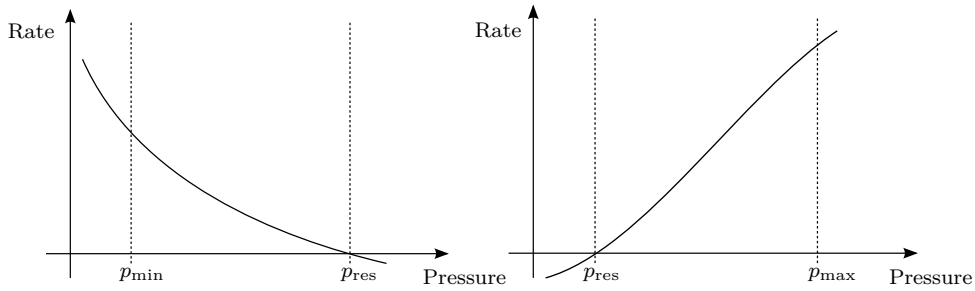
An important property that is exploited in this chapter is the time-varying mapping between flow and pressure at the wellbore. This relation is given by the well model, and a possible instant was given in Equation (3.7), which for convenience is restated

$$-q = \mathrm{WI}(p_b - p_{wb})$$

stating the relation between the average pressure p_b in the perforated gridblock, the bottom hole pressure p_{wb} and the flow q. This is sketched for a producer and an injector in Figure 4.1a and 4.1b, respectively.

There exist many possible constraints for wells in an oil-gathering production network. Examples of constraints on a producer are:

- The pressure in the wellbore should not drop below a certain threshold, in order to lift the fluid to the surface.
- The pressure in the wellbore should not be greater than the surrounding reservoir pressure, as this reverses the flow-direction.
- There could also be constraints on the production facility, that requires that either the total flowrate, or the rate of a particular phase, must be limited. This constraint can appear on particular wells, on groups of wells, or on all wells combined.



(a) Producer **(b)** Injector

Figure 4.1: Sketched relation between pressure and rate. Note that the flow is defined positive out of the injector, and into the producer.

And likewise for an injector

- The pressure in the wellbore should not exceed a certain threshold, as this may harm the reservoir by damaging the rock. In a long perspective, this can affect long-term recovery issues.
- The pressure in the wellbore should not be lower than the surrounding reservoir pressure, as this reverses the flow-direction.
- The amount of available fluid to inject can be limited, such that the flowrate must stay below a limit. This constraint can appear for particular wells, on groups of wells, or all wells combined.

Combined rate constraints are however not considered further in this thesis.

Remark 4.1. The physical actuator that manipulates the well is usually a choke-valve, and possibly an additional valve to control gas-lift. At a medium term perspective the fast dynamics of the valve-control can be disregarded. Instead, it is rather pressure or rate that is considered as the controlled variable.

4.1 Constrained and Unconstrained Control Optimization

During the last decades there has been a tremendous evolution in terms of available computational power. This has made it possible to use complex optimization algorithms in a broad range of applications. Typical large scale examples ranges from crew scheduling issues in the airline business, to planning production levels in the industry for several years ahead. Optimizing the waterflooding operation is an example of the latter.

According to Jansen (2011), the first publication on gradient-based optimization of waterflooding was proposed by Asheim (1988). They achieved NPV improvements of 2-11% for a hypothetical case. Later followed (Virnovsky, 1991), (Zakirov et al., 1996), (Sudaryanto and Yortsos, 2000), (Sudaryanto and Yortsos, 2001) and (Dolle et al., 2002). Jansen further argues that the industry uptake of gradient-based methods was almost not present until the terms *smart field* and *smart wells* appeared in the industry, and the interest started to flourish. DFO has also been considered for reservoir management, and Echeverría Ciaurri et al. (2010) demonstrated the applicability of derivative-free methods for challenging problems.

Furthermore, Brouwer and Jansen (2004) shows how dynamic optimization can be used to increase the NPV for a flooding-operation with smart-wells. They also categorize how pressure and rate constraints, in addition to the discount factor, impact the results of the optimization. Another interesting study was conducted by Peters et al. (2010), which invited a group of different companies to perform history matching and optimization of controls. A synthetic field was considered, and the study is known as the Brugge Benchmark. The companies selected different approaches, ranging from adjoint-based optimization to strategies based on neural networks. The outcome of the study showed big variations in the results obtained, but most of the companies that went for adjoint-based optimization showed promising results.

4.1.1 Optimization Problem

The optimality of the waterflooding operation is usually measured with a *net present value* (NPV) function, and the objective is to maximize this function, subject to the operational constraints. To this end, the optimization variable u is typically bottomhole pressures or flow set-points for wells. Jansen (2011) proposes the following formulation for optimizing the waterflooding operation. First, let the reservoir equations be given by

$$R_k^x(u_k, x_{k-1}, x_k) = 0, \qquad k = 1, 2, \ldots, K$$

where x_k are the reservoir states, usually given by the pressure and saturation in each gridcell. This equation can be seen as the part of Equation (3.11) that relates the reservoir states between time-steps k. The initial conditions of the reservoir is specified by $x_0 = \breve{x}_0$, a vector of initial state variables. Furthermore, output

equations in terms of well models are included to link the wells to the reservoir

$$R_k^y(u_k, x_k, y_k) = 0, \qquad k = 1, 2, \ldots, K$$

where $R_k^y(\cdot)$ is a vector valued function, and y is a vector of output variables. Equation (3.9b) is an example of such equation. Both input and output variables are, as mentioned, rates and pressures down hole. Each phase (oil, gas, water) may be considered as outputs for producers. Moreover, because most simulators solve the reservoir equations implicitly, it follows that the Jacobians of partial derivatives $\frac{\partial R_k^x}{\partial u_k}$, $\frac{\partial R_k^x}{\partial x_{k-1}}$ and $\frac{\partial R_k^x}{\partial x_k}$ tends to be available from the simulator. This is of great significance for gradient calculation, which is further discussed in Appendix A. Now, let the following general function represent the objective function

$$J(u_{1:K}, y_{1:K}(u_{1:K})) = \sum_{k=1}^{K} J_k(u_k, y_k)$$

where J_k is the contribution of J in each time-step k. The objective is typically given as a NPV function. A common used function for a two-phased oil-water system reads

$$J = \sum_{k=1}^{K} \frac{r_o q_{o,k} - r_w q_{o,w} - r_i q_{i,k}}{(1+b)^{\frac{t_k}{\tau_t}}} \Delta t_k \tag{4.1}$$

where r_0 denotes the fixed oil price, r_w and r_i are the water production and the water injection cost, respectively. These parameters are assumed to remain constant over the horizon. The production-rates of oil and water are denoted by $q_{o,k}$ and $q_{w,k}$, and the injection-rate of water by $q_{i,k}$. However, these variables are usually a subset of either the input variable u or the output variable y. To account for depreciation, the discount rate b is added for a certain reference time τ_t. Δt_k specifies the step-size for the k^{th} step. Note that this size may vary between steps.

The beginning of this chapter introduced possible constraints on wells. To generalize further, it can sometimes be desirable to also constrain reservoir states. Let all such constraints be represented by the following equalities and inequalities

$$c_k(u_k, y_k, x_k) = 0, \qquad k = 1, 2, \ldots, K$$

$$d_k(u_k, y_k, x_k) \geq 0, \qquad k = 1, 2, \ldots, K$$

The optimization problem is then summarized by

$$\max_{u_{1:K}} \quad \sum_{k=1}^{K} J_k(u_k, y_k) \tag{4.2a}$$

$$\text{s.t} \quad R_k^x(u_k, x_{k-1}, x_k) = 0, \quad k = 1, 2, \ldots, K \tag{4.2b}$$

$$x_0 = \breve{x}_0 \tag{4.2c}$$

$$R_k^y(u_k, x_k, y_k) = 0, \qquad k = 1, 2, \ldots, K \tag{4.2d}$$

$$c_k(u_k, y_k, x_k) = 0, \qquad k = 1, 2, \ldots, K \tag{4.2e}$$

$$d_k(u_k, y_k, x_k) \geq 0, \qquad k = 1, 2, \ldots, K \tag{4.2f}$$

where (4.2b) represents the reservoir equations, (4.2c) the initial condition of the reservoir, (4.2d) output equations, (4.2e) equality constraints and (4.2f) inequality constraints. The problem has the following dimensions

- n inputs, i.e. $u \in \mathbb{R}^n$
- m states, i.e. $x \in \mathbb{R}^m$
- p outputs, i.e. $y \in \mathbb{R}^p$
- q equality constraints, i.e. $c \in \mathbb{R}^q$
- r inequality constraints, i.e. $d \in \mathbb{R}^d$

The gradient of Equation (4.2a) with respect to the controls reads

$$\frac{\mathrm{d}J}{\mathrm{d}u_k} = \frac{\partial J_k}{\partial u_k} + \sum_{i=k}^{K} \frac{\partial J_i}{\partial y_i} \left(\frac{\partial y_i}{\partial u_k} + \frac{\partial y_i}{\partial x_i} \frac{\partial x_i}{\partial u_k} \right) \tag{4.3}$$

Assuming that the output equation (4.2d) is an explicit linear algebraic equation, the terms $\frac{\partial y_i}{\partial u_k}$ and $\frac{\partial y_i}{\partial x_i}$ can be computed directly. The terms $\frac{\partial J_k}{\partial u_k}$ $\frac{\partial J_i}{\partial u_i}$ are also easily computed. On the other hand, the term $\frac{\partial x_i}{\partial u_k}$ is not trivial to evaluate, because it implies that the recursive system (4.2b) of discrete time differential equations must be solved, to connect the state vectors x_i at times $i = k, k+1, \ldots, K$ to the input u_k at time $i = k$. Therefore, gradients are usually computed by forward or adjoint sensitivity equations, for reservoir optimization problems. Forward and adjoint sensitivity, in addition to finite difference approximation of gradients, are included in Appendix A. The Lagrangian for Problem (4.2) can be written as

$$\mathcal{L}(x_{1:K}, y_{1:K}, u_{1:K}, \lambda_{g,1:K}, \lambda_{h,1:K}, \lambda_{c,1:K}, \mu_{1:K}) = \sum_{k=1}^{K} J_k(u_k, y_k) \tag{4.4}$$

$$- \sum_{k=1}^{K} \lambda_{x,k} R_k^x(u_k, x_{k-1}, x_k) - \sum_{k=1}^{K} \lambda_{y,k} R_k^y(u_k, x_k, y_k) \tag{4.5}$$

$$- \sum_{k=1}^{K} \lambda_{c,k} c_k(u_k, y_k, x_k) - \sum_{k=1}^{K} \mu_k d_k(u_k, y_k, x_k)$$

where $\lambda_{x,k}$ is the multiplier associated with the reservoir states, $\lambda_{y,k}$ relates to the output equations, $\lambda_{c,k}$ relates to the equality constraints, and μ_k relates to the inequality constraints. The KKT conditions for the problem can then be formulated

(the notation $1:k$ is left out for ease of notation)

$$\nabla_u \mathcal{L}(x^*, y^*, u^*, \lambda_g^*, \lambda_h^*, \lambda_c^*, \mu^*) = 0 \tag{4.6a}$$
$$R_k^x(u_k^*, x_{k-1}^*, x_k{}^*) = 0 \qquad k = 1, 2, \ldots, K \tag{4.6b}$$
$$R_k^y(u_k^*, x_k^*, y_k^*) = 0 \qquad k = 1, 2, \ldots, K \tag{4.6c}$$
$$c_k(u_k^*, y_k^*) = 0 \qquad k = 1, 2, \ldots, K \tag{4.6d}$$
$$d_k(u_k^*, y_k^*, x_k^*) \geq 0 \qquad k = 1, 2, \ldots, K \tag{4.6e}$$
$$\mu_k^* \geq 0 \qquad k = 1, 2, \ldots, K \tag{4.6f}$$
$$\lambda_{x,k}^* R_k^x(u_k^*, x_{k-1}^*, x_k{}^*) = 0 \qquad k = 1, 2, \ldots, K \tag{4.6g}$$
$$\lambda_{y,k}^* R_k^y(u_k^*, x_k^*, y_k^*) = 0 \qquad k = 1, 2, \ldots, K \tag{4.6h}$$
$$\lambda_{c,k}^* c_k(u_k^*, y_k^*, x_k^*) = 0 \qquad k = 1, 2, \ldots, K \tag{4.6i}$$

A suitable NLP algorithm is used to find a control sequence u^* satisfying (4.6). If the inequality constraints (4.2f) were not present, a control-sequence could instead be found by solving the first-order necessary Euler-Lagrange equations.

Solving Problem (4.6) yields in an open-loop solution given by u^*. However, by repeatedly re-solving the problem during the reservoirs life-cycle, and continuously updating the initial state x_0 based on newly gathered information from the real field, the loop is closed by means of a NMPC, as outlined in Section 2.1.

4.1.2 Robust Optimization

Handling uncertainty was introduced as a great challenge for reservoir simulation in Chapter 1 and 3. The parameters that are used to describe the reservoir, such as permeability and porosity, are only estimates of the true values. According to Jansen (2013), this is a large obstacle on the road to industry uptake of gradient-based optimization methods. This problem is addressed by van Essen et al. (2009) with a synthetic reservoir, commonly known as *The Egg Model*. They do not consider a single deterministic reservoir, but instead an ensemble that consists of 100 different realizations of the permeability field. As such, multiple subsurface models are present. They propose a method to maximize the expected value of the production strategy, subject to all realizations. The traditional approach is rather to optimize the NPV for a single deterministic realization, which they refer to as *Nominal Optimization* (NO). The robust optimization problem reads

$$\max_{u_{1:K}} \mathrm{E}_\theta \left[J(u_{1:k}, y_{1:k}^{1:N_r}(u_{1:k}), \theta^{1:N_r}) \right] \approx \max_{u_{1:K}} \frac{1}{N_r} \sum_{i=1}^{N_r} J\left(u_{1:k}, y_{1:k}^i(u_{1:k}), \theta^i\right)$$

where θ^i are uncertain model parameters, and y^i are output vectors of realization $i = 1, 2, \ldots, N_r$. They considered three strategies for optimizing the production strategy, namely reactive control, nominal optimization and robust optimization. See their publication for further information. However, it is worth mentioning that the results of the study showed that it is beneficial to use robots optimization, rather

than nominal optimization, in the presence of multiple geological realizations. According to Jansen (2013), not only is the mean recovery of the 100 simulations highest with the robust strategy, the standard deviation is also the lowest. However, the robust approach comes at a price, as it needs to perform many additional, costly simulations. Indeed, both forward and adjoint simulation are needed for all geological realizations at each iteration.

To keep the scope of this thesis tractable, robust optimization has not been further considered. However, a simplified version of the egg model has been used to conduct numerical experiments, presented in Chapter 5.

4.1.3 REMSO

A particular algorithm made to solve problems like (4.2) is proposed by Codas et al. (2015), with the *reservoir multiple shooting optimization* (REMSO) algorithm. It is tailored to solve output constrained oil-reservoir control optimization problems, by means of multiple shooting. REMSO is tightly interfaced to MRST, but can also be programmed to interface other simulators. The following features are implemented:

- Reduced sequential quadratic programming (rSQP)
- The lift-opt trick (Albersmeyer and Diehl, 2010), alongside with a reduction procedure.
- A multiplier-free penalty strategy to tune an ℓ_1 merit function for line-search
- Non-monotone linesearch strategy, commonly known as the watchdog strategy (Nocedal and Wright, 1999, pg. 446). Used to prevent the Maratos-effect.
- Adjoint and forward sensitivity calculations as suitable.
- Damped BFGS (Nocedal and Wright, 1999, pg. 537) for updating the Hessian approximation.
- Initial guesses of simulation profiles for each simulated point. The profiles are based on a linear prediction of the simulation profiles x and y. See their publication for further details, in particular Appendix A.

Codas et al. (2015) shows that the linear prediction drastically reduces the number of required Newton-Iterations to solve the implicit simulation steps in MRST, see section 3.3.3. REMSO formally solves the following problem

$$\min_{\Theta_c} \quad J = \sum_{k \in \mathcal{K}} J_k \left(x_k^f, y_k, u_{\kappa(k)} \right) \tag{4.7a}$$

$$\text{s.t} \quad x_k^f - x_{k+1} = 0, \qquad k \in \mathcal{K}, \tag{4.7b}$$

$$R \left(x_k, x_k^f, y_k, u_{\kappa(k)} \right) = 0, \quad k \in \mathcal{K} \tag{4.7c}$$

$$b_l^x \leq x_k \leq b_u^x, \qquad k \in \mathcal{K} \tag{4.7d}$$

$$b_l^y \leq y_k \leq b_u^y, \qquad k \in \mathcal{K} \tag{4.7e}$$

$$b_l^u \leq u_j \leq b_u^u, \qquad j \in \mathcal{U} \tag{4.7f}$$

for the simulation steps $k \in \mathcal{K} = \{1, 2, \ldots, K\}$. The optimization variable Θ_c consist of the control variables u, the initial states x, the final states x^f, and the algebraic output variables y.

The control is parameterized, and divided into n_u steps, such that $\mathcal{U} = \{1, 2, \ldots, n_u\}$ is the set of control periods. There is a surjective function that maps the simulation-step indices to the control-step indices, $\kappa : \mathcal{K} \to \mathcal{U}$. It follows that $\kappa(k_1) \leq \kappa(k_2)$ if $k_1 < k_2$, thus $n_u \leq K$. Furthermore, Equation (4.7a) is the negative separable NPV function[1]. Equation (4.7b) represents the additional MS constraints that enforces continuity over shooting intervals. Equation (4.7c) denotes the reservoir dynamics defined in Section 3.3.3 with Equation (3.11). Equation (4.7d) specifies bounds on reservoir states, Equation (4.7e) is an output constraint that bounds the algebraic output well-variables, and Equation (4.7f) bounds the controlled variable u. Thus, (4.7d)- (4.7f) can be viewed as part of (4.2f).

However, as the Jacobian $[\frac{\partial R_k}{\partial x_k^f}, \frac{\partial R_k}{\partial y_k}]$ is assumed not-singular by construction, see Section 3.3.3, y_k and u_k determine an unique solution due to the implicit function theorem. Consequently, a reformulated optimization problem reads

$$\min_{\Theta} \quad J = \sum_{k \in \mathcal{K}} J_k \left(x_{k+1}, y_k, u_{\kappa(k)} \right) \tag{4.8a}$$

$$\text{s.t} \quad x_{k+1} = R_k^x \left(x_k, u_{\kappa(k)} \right), \quad k \in \mathcal{K} \tag{4.8b}$$

$$y_k = R_k^y \left(x_k, u_{\kappa(k)} \right), \quad k \in \mathcal{K} \tag{4.8c}$$

$$b_l^x \leq x_k \leq b_u^x, \quad k \in \mathcal{K} \tag{4.8d}$$

$$b_l^y \leq y_k \leq b_u^y, \quad k \in \mathcal{K} \tag{4.8e}$$

$$b_l^u \leq u_j \leq b_u^u, \quad j \in \mathcal{U} \tag{4.8f}$$

Equation (4.8b) represents state-transitions, as $x_k^f = x_{k+1}$, ensuring continuity across shooting intervals. The algebraic output variables y are obtained by (4.8c). The optimization variable is consequently reduced to Θ, which consists of x, v and u

Codas et al. (2015) show that the multiple shooting approach requires the computationally expensive state sensitivity matrix, which often is why the single shooting approach instead is favoured for reservoir optimization. The reader is referred to their publication for techniques used to overcome these issues, but some advantages of their approach are worth mentioning:

- The multiple shooting approach allows for parallelism opportunities, which for large-scale reservoirs can be of great significance.
- It is trivial to implement state constraints on the shooting interval boundaries, as these variables are independent decision variables with the MS approach.
- Numerical accuracy can be differentiated between the shooting intervals. As most geological models are inaccurate in the first place, there is usually no

[1] $\max J(u)$ equals $\min -J(u)$, see Nocedal and Wright (1999).

need of calculating very accurate solutions towards the end of the horizon, due to unavoidable error propagation throughout simulation.

- MS converges differently than SS, and can be faster.

REMSO is one of the methods that are used to solve the numerical examples in Chapter 5. The REMSO implementation is distributed by (Codas, 2015). The version that is used in this work simulates the reservoir in parallel.

4.1.4 IPOPT

The optimizer IPOPT (Kawajir et al., 2010) is also used to solve the numerical examples in Chapter 5. It is included as a single shooting alternative to REMSO, using output constrained adjoint-based optimization. Keeping the notation from the previous section, IPOPT solves the following problem

$$\min_{u} \quad J = \sum_{k \in \mathcal{K}} J_k \left(x_{k+1}, y_k, u_{\kappa(k)} \right) \tag{4.9a}$$

$$\text{s.t} \quad x_{k+1} = R_k^x \left(x_k, u_{\kappa(k)} \right), \quad k \in \mathcal{K} \tag{4.9b}$$

$$y_k = R_k^y \left(x_k, u_{\kappa(k)} \right), \quad k \in \mathcal{K} \tag{4.9c}$$

$$b_l^y \leq y_k \leq b_u^y, \quad k \in \mathcal{K} \tag{4.9d}$$

$$b_l^u \leq u_j \leq b_u^u, \quad j \in \mathcal{U} \tag{4.9e}$$

In addition, state-constraints like (4.8d) could easily be included also in IPOPT. They are however left out, because there is of no interest to bound the reservoir states in the numerical examples in Chapter 5, and they have no further purpose for the single-shooting approach. Unlike REMSO, a linear prediction of each simulated step is not implemented, such that Newton-Iterations are not warm-started. As a result, combined with the inability to parallelize simulation, it is expected that IPOPT will be a relatively slow algorithm compared to REMSO.

The single shooting approach reduces the problem in space, as the decision variables only consist of the parameterized controls, as outlined in Section 2.2.3. All other variables and functions that depend on simulations, are calculated from a forward simulation. Gradients are obtained through adjoint backward simulations.

IPOPT is downloaded as binaries, and easily linked to Matlab. A framework that interfaces IPOPT to MRST, together with efficient calculation of gradients, is distributed by Codas (2015).

4.2 Reactive Control

Reactive control is a simple and intuitive heuristic for operating both producers and injectors. Unlike other methods, this approach does not require complex optimization procedures. Its simplicity and fairly proven efficiency is among others why this method is often a preferred practice in the industry. van Essen et al. (2009) argues that the approach does not suffer from an inaccurate, or even completely wrong, geological model of the reservoir when it is applied to a real field. This is because the strategy is model-free. Model-errors, such as deviations from true permeability and porosity field, can possibly severely affect the performance of controls optimized by model-based procedures.

It is typical to fix production wells at either maximum allowed rate, or at the minimum allowed BHP with the reactive strategy. Injectors can similarly be fixed at maximum rate, or maximum pressure. Furthermore, it is common to shut down the producer entirely when a certain condition is met, often when the water-cut exceeds a specified threshold. This implies that the well is not longer profitable, and only contributes negatively to the NPV. In reservoir simulation, the well can also be shut exactly once it starts to contribute negatively to the NPV, accounting for the discount factor. If shut-in is not implemented, the method allows for injecting water directly to a producer. This will, of course, negatively affect the NPV, because both injecting water and handling produced water comes at a price. The total injection rate should be scaled accordingly if the injectors are controlled by rate, and producers are shut.

Reactive control can also be extended to comply with output constraints. For instance, if a producer is operated by rate control, the minimum pressure limit can be reached as the reservoir depletes and the surrounding pressure declines. A reactive control strategy to handle this event can be to switch the well from rate control to pressure control, in order to maintain the minimum required pressure. This is illustrated in Figure 4.2. Similarly, if the well is operated by pressure control, it can lead to a flowrate through the wellbore that exceeds a maximum rate limit. At this point, the well can be switched from BHP control to rate control. Injectors can be operated in the same manner, switching between rate and pressure control as appropriate. This kind of reactive control is actually how MRST copes with output constraints that are embedded directly into the simulator. Moreover, MRST can switch to rate-control of specific phases (oil/gas/water) to handle phase-specific rate constraints. Even though phase-specific rate control can be difficult to apply at real fields[2], it can easily be used for simulation based optimization.

Reactive control is a greedy approach, aiming to maximize current throughput of fluids. This can contradict with long-term depletion strategies, as long-term recovery can be damaged. A negative consequence that can follow from reactive control is early water-breakthrough.

[2]It needs sufficiently accurate phase-specific flow measurements

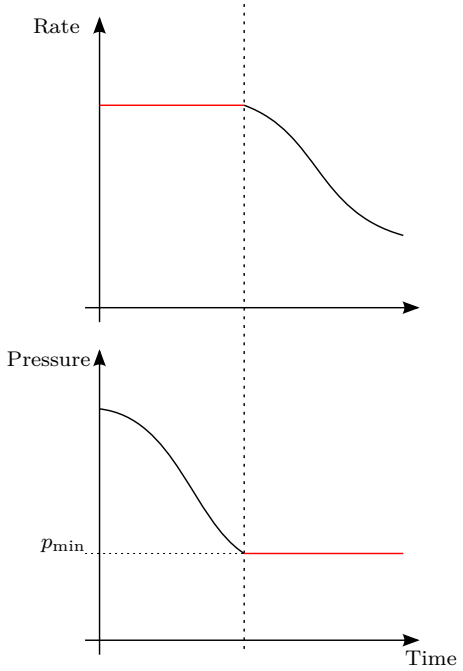


Figure 4.2: Control-switch based reactive control illustrated for a producer. The controlled variable is highlighted in red.

4.2.1 Implementation

Reactive control is implemented and used for the numerical experiments in Chapter 5. It combines control-switching to handle constraints and shut-in of producers. Moreover, wells are initially set to operate at their maximum/minimum allowed limit in the following manner

- Producers controlled by BHP operate at the minimum allowed BHP.
- Producers controlled by rate (water/oil/both) operate at the maximum allowed rate.
- Injectors controlled by BHP operate at the maximum allowed BHP.
- Injectors controlled by rate (water) operate at the maximum allowed rate.

Each well switches control-mode within MRST when necessary, in order to comply with output constraints. The producers are permanently shut for all remaining simulation steps, if they at some point start to contribute negatively to the NPV.

4.3 Optimization Techniques with Simulator Embedded Constraints

To the authors knowledge, there seems to exists little formal research on optimization techniques with simulator-embedded constraints. However, reactive control based on control-switching, as outlined previously in this chapter, are though implemented on simulators like MRST and Eclipse. This leaves the simulator responsible for handling the output constraints.

In the following sections, two other heuristics are presented, whereas the first is developed during this work. This heuristic needs a modified NLP algorithm, which is presented first. The major workload of this thesis was to implement the heuristic and the NLP algorithm, as modifications deep within the software project (Codas, 2015), and MRST were needed. This required that large, multidisciplinary software had to be explored and understood.

4.3.1 OSSO: an Output-unconstrained Single Shooting Optimizer

In order to implement the reformulation-based heuristic presented in Section 4.3.2, it is necessary to interface MRST with an open-source optimization algorithm. This is because modifications are needed within the optimizer to detect, and more importantly, to handle control switching in forward simulations. For these reasons, a new output unconstrained gradient-based optimizer is developed. The optimizer REMSO is used as template. However, the complexity that comes with the multiple shooting approach is undesirable in this context, especially because continuity of state-profile cannot be ensured. This may cause that constraints are being violated in an intermediate multiple shooting simulation, that would not be violated in a consistent forward simulation. In particular, this can make the simulator switch the controlled-target for a well, based on violated constraints that in reality would be feasible in a single shooting simulation. Also, the large problem size that comes with the multiple shooting approach sets high requirement on the hardware used for optimization, due to the vast amount of memory needed.

With the above challenges in mind, the optimizer has been been modified to interface MRST in a single shooting fashion. The new optimizer is from here on denoted as *Output unconstrained Single Shooting Optimizer* (OSSO). Important features that are reused from REMSO is a damped BFGS update of the Hessian (Nocedal and Wright, 1999, pg. 537), and the third-order polynomial approximation of the linefunction. Gradients are still computed by forward or adjoint sensitivity as appropriate, using the same framework developed by Codas et al. (2015). In addition, the linear predictor is kept, providing each simulated point an initial guess of the simulation profile. The watchdog strategy used in REMSO is however removed. This is because it introduces unnecessary complexity, when one of the goals is to keep the optimizer simple, in order to highlight it's important features related to control-switching. Instead, OSSO is implemented with a tradi-

tional linesearch strategy, using a polynomial approximation of the line-function. OSSO solves the following problem

$$\min_{u} \quad J = \sum_{k \in \mathcal{K}} J_k \left(x_k, y_k, u_{\kappa(k)} \right) \tag{4.10a}$$

$$\text{s.t} \quad x_{k+1} = R_k^x \left(x_k, u_k \right), \quad k \in \mathcal{K} \tag{4.10b}$$

$$y_k = R_k^y (x_k, u_k), \quad k \in \mathcal{K} \tag{4.10c}$$

$$b_l^u \leq u_j \leq b_u^u, \quad j \in \mathcal{U} \tag{4.10d}$$

where Equation (4.10b)-(4.10c) is solved with single shooting, based on the current control vector $u_{\kappa(k)}$. Similarly to REMSO, (4.10a) denotes the negative NPV function. However, MRST is allowed to switch the controlled variable at each step k. Due to the single approach, the problem is reduced in space, as all variables can be obtained through a forward simulation, given the controls. In each iterate, the search direction Δu is computed by solving a QP which is common for reduced SQP algorithms (Schmid and Biegler, 1994; Nocedal and Wright, 1999). This reads

$$\min_{\Delta u} \quad \frac{1}{2} \Delta u^T M \Delta u + \nabla_u J \Delta u \tag{4.11a}$$

$$\text{s.t} \quad b_l^u \leq u + \Delta u \leq b_u^u \tag{4.11b}$$

$$\tag{4.11c}$$

in which the step Δu honor the input constraint (4.10d). M is the approximated hessian of the objective J. The QP problem given in (4.11) approximates the NLP (4.10) in a neighborhood of the current iteration point, and consequently convergence cannot be ensured from any starting point. Shorter steps may however improve the objective value at the iteration, and the length of the step Δu must be adjusted to ensure improvement with respect to the previous iteration. Equation (4.11b) ensures that the algorithm never visit infeasible areas w.r.t the input constraints.

As the problem is unconstrained in terms of violation of states and output variables, the merit-function simply equals the objective function. The line-search strategy on the merit is reused from the REMSO implementation (Codas, 2015), and Codas et al. (2015) proposes the following approach to approximate the line function by a third-order polynomial: The line-function is evaluated at the bounds, and a third-order polynomial is adjusted to the function values and directional gradients. The minimum of the polynomial is evaluated, and a new function evaluation is calculated at this point. If this point satisfies the Wolfe-conditions (Nocedal and Wright, 1999, p. 33), the line-search procedure finished successfully. Otherwise new polynomials are fit to the two new segments. A new function evaluation is performed at the minimum of the polynomial for these segments. If neither point still do not satisfies the the Wolfe-conditions, the process is repeated until a predefined maximum of number of iterations is reached. The polynomial approximation of the line-search is shown in Figure 4.3.

The linear predictor builds an estimate of the state difference Δx and the output difference Δy in each major iteration, based on the (full step) point $\hat{u} = u_k + \Delta u$, thus

$$\tilde{x} = x_k + \alpha \Delta x$$
$$\tilde{y} = y_k + \alpha \Delta y$$

where $\alpha \in (0, 1]$ denotes possible backtrack. See Codas et al. (2015) for computation of Δx and Δy. $\tilde{x}$ and $\tilde{y}$ are given to MRST as initial guesses of the simulation profiles, to warm-start the Newton-Iterations that solves the reservoir equations. Remark that, if constraints are not embedded into MRST, OSSO works just as a

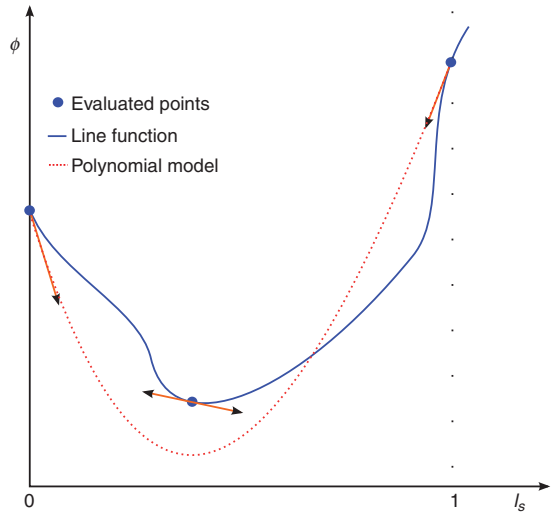


Figure 4.3: Third order polynomial approximation of the line function ϕ. The figure is originally published in (Codas et al., 2015).

regular (output and state unconstrained) SQP-based algorithm. OSSO is outlined in Algorithm 1. Remark that $D(a; b)$ denotes the derivative of a in the direction of b.

4.3.2 Heuristic Based on Problem Reformulation

A new heuristic for control optimization in reservoir management has been developed during this thesis. This is based on problem-reformulation, and simulator embedded constraints. The concept is to leave the simulator responsible for handling output constraints, while using an output-unconstrained optimizer to search for the optimal control sequence. To this end, some of the complexity is removed from the optimizer. If a control-sequence generated by the optimizer make the simulator switch control during simulation, in order to handle output constraints,

Algorithm 1: OSSO

Data: Simulator $\Upsilon(u, \tilde{x}, \tilde{y}, J(\cdot))$, initial controls u^0, objective function $J(u, x, y)$, optimality tolerance ε, Hessian initialization M^0, line-search parameter η

Result: u, x, y, J, M

Set $i \leftarrow 0$;

if *Initial guess of x and y not provided* **then**

  Compute $(x^0, y^0, J^0) \leftarrow \Upsilon\left(u^0, \sim, \sim, J(\cdot)\right)$;

  if *Simulator switches control* **then**

    Return $(u^0, x^0, y^0, J^0, M^0)$;

  end

end

repeat

  Compute gradient $\nabla_u J$ by adjoint simulation;

  Compute a step Δu by solving the QP given in (4.11);

  Update M^i with BFGS ;

  Set $\hat{u} = u^i + \Delta u$;

  Build linear prediction Δx and Δy based on $\hat{u}$;

  Set $\tilde{x} = x^i + \Delta x$;

  Set $\tilde{y} = u^i + \Delta y$;

  Compute $(x^i, y^i, J^i) \leftarrow \Upsilon(\hat{u}, \tilde{x}, \tilde{y})$;

  if *Simulator switches control* **then**

    Return $(\hat{u}, x^i, y^i, J^i, M^i)$;

  end

  if $J(\hat{u}) \leq J(u^i) + \eta D(J(u^i); \Delta u)$ **then**

    Set $u^{i+1} \leftarrow \hat{u}$;

  else

    Find α^i by third-order polynomial approximation of J, such that;

    $\hat{u} \leftarrow u^i + \alpha^i \Delta u$;

    $(x^i, y^i, J^i) \leftarrow \Upsilon(\hat{u}, \tilde{x}, \tilde{y})$;

    $J(\hat{u}) \leq J(u^i) + \eta \alpha^i D(J(u^i); \Delta u)$;

    Set $u^{i+1} \leftarrow \hat{u}$;

  end

  $i \leftarrow i + 1$;

until $\|\Delta u\| < \varepsilon$;

the gradient w.r.t the control-sequence can not be computed. This is because the simulator changed what was the controlled variable, to become an algebraic variable instead. This variable is rather calculated from the output well-equation. To continue optimization at this stage, two options are considered. The first option is to perform a backtrack operation along the current search direction[3] Δu by reducing the step-length parameter α, and check if the simulator still switches control with the backtracked control sequence. If this do not occur, then optimization may safely proceed in traditional manner. On the contrary, if control switching still occurs, the second option is considered; namely to terminate the entire optimization problem at the current iterate, and start an entirely new optimization with reformulated controls. The troublesome variable(s) are switched between rate control and pressure control as appropriate. Backtracking is however not used for the cases in Chapter 5.

The last trialpoint in the terminated optimization is used as the starting point (x_0, y_0, u_0) for the new problem. Constraints are implicitly handled, as the violated variable(s) becomes controlled variable(s) in the subsequent problem. As mentioned earlier, a single phase (oil, water) can be rate controlled in MRST. The heuristic is outlined in Algorithm 2.

The motivation for developing the heuristic is that handling output constraints with a formal treatment in the NLP algorithm can be computationally demanding. In addition, the heuristic may remove some of the complexity that are associated with traditional constrained optimization.

An additional augmented version of the heuristic has also been tested in Chapter 5, which uses optimized controls from a constrained optimzier, such as REMSO and IPOPT, as the initial control. The hope for this version is that the ability to refine control periods where it is appropriate may lead to higher NPV, without having to solve a more complex problem from scratch. The approximated Hessian from the initial constrained optimization is reused.

However, there are three major issues that must be addressed, which are discussed in the upcoming sections.

Absence of certain output constraints

As discussed in Section 3.3, only certain constraints can be specified and handled by MRST. The following constraints are troublesome:

- Minimum rate, producers and injectors
- Maximum pressure, producers
- Minimum pressure, injectors

This is clearly a challenge, because MRST is not able to handle them as output constraints. For instance, if a producer is controlled by rate, it is easy to ensure positive

[3]Each trial-point is found by the line-search procedure $u = u^i + \alpha \Delta u$

Algorithm 2: Heuristic 1: Problem reformulation

Data: Simulator $\Upsilon(u, \tilde{x}, \tilde{y}, J(\cdot))$, initial controls u^0, objective $J(u, x, y)$
Result: u, x, y, J
Choose initial controls u_0;
Set initial Hessian approximation $M \leftarrow I$;
Set x, y empty;
repeat
  $(J, u, x, y, M) \leftarrow \text{OSSO}(J(\cdot), \Upsilon(\cdot), u^0, x, y, M)$;
  if *OSSO reports control switch in the forward model* **then**
    $u^0 \leftarrow$ Rebuild controls ;
    $M \leftarrow$ Initialize Hessian;
  end
until *OSSO returns without control-switching*;

flow, because it is a free variable. However, if the same producer is controlled by BHP, positive flow cannot be ensured because it is an output variable, which again must be handled by output constraints. This becomes particularly cumbersome for the heuristic, because all wells are allowed to switch between pressure-control and rate-control. This is further discussed in Chapter 6.

Reformulation

It is not straight forward to reformulate the controls for the new optimization problem. Note that the number of control periods $j \in \mathcal{U}$ often are smaller then the number of simulation periods $k \in \mathcal{K}$. As each control period contains a target for each well, an optimization problem with n_w wells and n_u control periods has $n_w \times n_u$ decision variables.

If control switching occur in a simulation step where the corresponding control period only cover that particular step, the reformulation is simply to only switch the wells that switched control during simulation. This does not affect the organization of the control periods. However, if switching occur in a simulation step where the corresponding control period cover multiple simulation steps, then several actions are possible. The simplest is to keep the original organization of control periods also in the reformulated problem, and switch the control mode for the troublesome well. However, this strategy also influences the other simulation steps that are covered by the control period, as the control is also switched for other simulation steps. This can introduce undesirable effects. In particular, it can leave the heuristic in a state where it continuously alternate between two (or more) formulations of the optimization problem. This is a problem if it is not making progress towards convergence.

Another possibility is to expand the number of control periods, aiming to keep the reformulated controls feasible in the initial simulation of the restarted optimization. In this way, earlier simulation steps in the control period that were not switched in

the terminated optimization, are not affected by the reformulation. This strategy is illustrated in the following example.

Example 4.3.1 (Control reformulation). Consider an optimization problem with four simulation periods ($K = 4$), one control period ($n_u = 1$), and only one well controlled by rate ($n_w = 1$). During a simulation called by the optimizer, the simulator switches the well from rate control to pressure control in the third simulation step. The control periods are reorganized in the following manner

Simulation period (k)	1	2	3	4
(Old) control period (j)	1	1	1	1
(New) control period (j)	1	1	2	2

where an additional control period has been added. The simulation steps that followed after the third simulation step are also set to use the new control period.

Unfortunately, the number of the control periods can blow up by reformulating the problem in this manner, which ultimately can result in $n_c = K$. On the contrary, the goal is to achieve the following:

- Counteract that the heuristic alternate between possible formulations.
- Refine control periods in segments where maintaining original control periods may be difficult.

Hessian initialization

For both performance, and efficiency, it is important to reuse the Hessian approximation obtained in the terminated optimization[4]. Unfortunately, there has not been time during this work to perform a proper in-depth analysis of methods for doing so. Instead, only a simple method has been used. Put simply, rows and columns are interchanged as suitable, and the matrix is expanded when the number of control periods are increased. It is not argued that the approach taken is necessarily a good one. In fact, the numerical cases in Chapter 5 showed that this also may introduce problems, which is further discussed in Chapter 6.

First, the Hessian of the function $f(u)$ with $u \in \mathbb{R}^n$ reads

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial u_1^2} & \frac{\partial^2 f}{\partial u_1 u_2} & \cdots & \frac{\partial^2 f}{\partial u_1 u_n} \\ \frac{\partial^2 f}{\partial u_2 u_1} & \frac{\partial^2 f}{\partial u_2^2} & \cdots & \frac{\partial^2 f}{\partial u_2 u_n} \\ \frac{\partial^2 f}{\partial u_3 u_1} & \frac{\partial^2 f}{\partial u_2^2} & \cdots & \frac{\partial^2 f}{\partial u_3 u_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial u_n u_1} & \frac{\partial^2 f}{\partial u_n u_2} & \cdots & \frac{\partial^2 f}{\partial u_n^2} \end{bmatrix}$$

The rows and columns in the Hessian directly related to the affected variable u_i, are set to an identity vector of appropriate dimension. The next example illustrates

[4]Keeping in mind that SQP-based algorithms use the Hessian to compute the search direction Δu

how the Hessian is initialized if the number of control periods are kept in the reformulated problem

Example 4.3.2 (Hessian initialization, 1). Consider an optimization problem with one simulation period ($K = 1$), one control period ($n_u = 1$) and four wells ($n_w = 4$), all of which are controlled by BHP. During optimization, a simulation called by the optimizer switches the third well to rate control, and the optimization is terminated at the current iterate. M_1 denotes the approximated Hessian at the terminated point. The optimization is restarted with all wells controlled by BHP, except for the third, which is now controlled by rate. The Hessian initialization M_2 that is provided to the optimizer then reads

$$\underbrace{\begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} & m_{1,4} \\ m_{2,1} & m_{2,2} & m_{2,3} & m_{4,4} \\ m_{3,1} & m_{3,2} & m_{3,3} & m_{4,4} \\ m_{4,1} & m_{4,2} & m_{4,3} & m_{4,4} \end{bmatrix}}_{\text{old approximation } M_1} \longrightarrow \underbrace{\begin{bmatrix} m_{1,1} & m_{1,2} & 0 & m_{1,4} \\ m_{2,1} & m_{2,2} & 0 & m_{4,4} \\ 0 & 0 & 1 & 0 \\ m_{4,1} & m_{4,2} & 0 & m_{4,4} \end{bmatrix}}_{\text{new approximation } M_2}$$

The next example illustrates how the Hessian is initialized if the number of control periods are increased during problem reformulation.

Example 4.3.3 (Hessian initialization, 2). Consider an optimization problem with five simulation periods ($K = 5$), two control period ($n_u = 2$) and two wells ($n_w = 2$), all of which are controlled by BHP. The periods are organized in the following manner

Simulation period (k)	1	2	3	4	5
Control period (j)	1	1	1	2	2

During optimization, a simulation called by the optimizer switches the second well to rate control at the second simulation step, and the optimization is terminated with the current approximated Hessian M_1. The optimization is restarted with all wells controlled by BHP, except for the second well in the (newly appended) second control period, which is now controlled by rate. Note that what was the second control period in the previous formulation now has shifted to become the third control period. The control periods are reorganized in the following manner

Simulation period (k)	1	2	3	4	5
Control period (j)	1	2	2	3	3

And the Hessian initialization M_2 that is provided to the optimizer reads

$$\underbrace{\begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} & m_{1,4} \\ m_{2,1} & m_{2,2} & m_{2,3} & m_{4,4} \\ m_{3,1} & m_{3,2} & m_{3,3} & m_{4,4} \\ m_{4,1} & m_{4,2} & m_{4,3} & m_{4,4} \end{bmatrix}}_{\text{old Hessian } M_1} \longrightarrow \underbrace{\begin{bmatrix} m_{1,1} & m_{1,2} & 0 & 0 & m_{1,3} & m_{1,4} \\ m_{2,1} & m_{2,2} & 0 & 0 & m_{2,3} & m_{2,4} \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ m_{3,1} & m_{3,2} & 0 & 0 & m_{3,3} & m_{3,4} \\ m_{4,1} & m_{4,2} & 0 & 0 & m_{4,3} & m_{4,4} \end{bmatrix}}_{\text{new Hessian } M_2}$$

However, it can be argued the both the upper-left and lower-left block matrix that is reused in Example 4.3.3 is wrong. They are based on contributions from simulation periods controlled by different control-periods, in the restarted problem. An idea that has not been further explored in this work, is to scale these values accordingly. Another simple idea for improvement, is related to the identity block matrix in the center. If some wells are not switched in the control period, there should be possible to reuse information for these wells.

It was also considered to approximate affected rows in the Hessian by finite difference. Although, this may not be tractable when the number of controls $n_u \times n_w$ is large. The last problem to be mentioned, is that the BFGS in OSSO is not allowed to correct the Hessian, if a restarted problem is infeasible in the initial run. This can typically occur if a control period cover multiple simulation steps.

4.3.3 Combining Unconstrained Optimization and Reactive Control

There exists at least one additional heuristic based on simulator-embedded constraints. Kourounis et al. (2014) proposes an heuristic for output constrained optimization problems, where an output unconstrained optimization is performed first. When the initial optimization has converged, the forward model is run once more using the optimized controls from the output unconstrained optimization. However, during the final forward simulation, output constrained are handled by control-switching in the simulator. In this way, the computational effort for the heuristic is only a little more than that required for optimizing the first problem, as only one additional forward run of the model is required. For further information about implementation, and the method itself, the reader is referred to the original publication.

Even though this heuristic is clearly approximate, the authors argue that is has some potential advantages over formal methods. The heuristic treatment allows the simulator to switch controls at any time step during simulation. In some sense, the heuristic enables a more fine-grained response, which can be viewed as having many more control variables (although not formally optimized).

To assess the merits of the heuristic, they compare it against a formal treatment of the constraint within the optimizer SNOPT (Gill et al., 2002). To this end, a series of four case studies are considered, which are simulated on the automatic differentiation-based compositional flow simulator AD-GPRS[5]. Two of the cases are small and simple grids in two dimensions, while the remaining two are more complicated grids. In the simple cases, they found that the formal treatment outperformed the heuristic. In the other cases, the heuristic generated better results in terms of run-time and NPV.

Their heuristic is implemented, and used to solve the numerical experiments in the next chapter. The version that is implemented is outlined in Algorithm 3.

[5]Stanford's Automatic Differentiation-based General Purpose Research Simulator

Algorithm 3: Heuristic 2: Output constrained optimization and reactive control. (Kourounis et al., 2014)

Data: Simulator $\Upsilon(\cdot)$, initial controls u_0, objective $J(\cdot)$
Result: u, x, y, J
Solve an output unconstrained optimization problem;
$\hat{u} \leftarrow \text{OSSO}(J(\cdot), \Upsilon(\cdot), u_0)$;
Perform a forward run with simulator embedded output constraints, using the unconstrained optimized controls;
$(u, x, y, J)) \leftarrow \Upsilon(\hat{u})$

Chapter 5

Numerical Results

Four numerical cases are proposed in this chapter to evaluate the methods that were presented in Chapter 4. The first (case A) is a simple two-dimensional 10×10 grid, intended to verify the implementation of OSSO. The second (case B) uses the same grid to investigate the merits of methods in the presence of output constraints. Two additional cases are included to evaluate the methods on more realistic models. Hereby, a five-spot model proposed by Bellout et al. (2012) (case C), and one layer of a single geological realization of the egg model (case D).

The objective function for all cases is a scaled NPV function, similar to the one given in Equation (4.1). The fixed oil price is set to $r_o = 1$ for each std m^3 oil, water handling cost $r_w = 0.1$, and water injection cost $r_i = 0.1$. The discount factor is set to $b = 0.1$. During optimization, all variables[1] are scaled to prevent bad performance, as the initial Hessian approximation and convergence conditions depends on this (Codas et al., 2015). The scaling factors are manually decided after exploring the problems beforehand. To mimic reality, The NPVs could also easily be scaled to realistic values in terms of dollar per barrel, if this were preferable. The optimality-tolerance for REMSO is set to 10^{-4}, see (Codas et al., 2015, Case and Results) for more information, and 10^{-4} for OSSO and IPOPT.

The simulations are performed on a Dell OptiPlex 9020 workstation, with 16 GB RAM, and an Intel Core i7-4770 quad processor at 3.4 GHz. Simulations are implemented using Matlab2014a and MRST2014b. QP's in OSSO are solved with Matlab's embedded solver `quadprog`.

In figures showing well-trajectories, a red line is used for controlled variables, while a blue line is used for well-variables that are calculated algebraically from well equations. Moreover, a small circle (o) is used to indicate the beginning of a control period, in order to visualize the degrees of freedom for the problem. A small cross (x) is used to indicate each simulation step. Water-rate is denoted by WRAT, oil-rate by ORAT, and control periods by CP. A star behind a number in result-tables indicates that this number was the limiting factor for the optimization

[1] controls, states, well variables, NPV

Table 5.1: The methods used to optimize the numerical examples in Chapter 5.

#	ID	Explanation	Reference
1	IPOPT	Output constrained gradient-based optimization (SS)	Section 4.1.4
2	REMSO	Output constrained gradient-based optimization (MS)	Section 4.1.3
3	OSSO	Output unconstrained gradient-based optimization (SS)	Section 4.3.1
4	RC1	Reactive control with control switching and well-shut in	Section 4.2
5	RC2	Same as RC1, but with the initial control provided by an output unconstrained optimization with OSSO	Section 4.3.3
6	REF1	Reformulation-based heuristic	Section 4.3.2
7	REF2	Same as REF1, but initial controls provided by a constrained optimization by REMSO	Section 4.3.2

procedure.

A summary of the methods that are used to optimize the cases is found in Table 5.1. The organization of simulation steps and control periods is explained for all cases in Appendix B.2.

Remark 5.1. Due to the run-time associated with the optimization on the available equipment, there has not been time to perform Monte-Carlo like simulations of the algorithms. As a valid quantitative analysis should be based on a representative amount of samples, including different parameter-settings and starting points, the numerical examples must rather be interpreted as an aid to prove concepts, in a qualitative fashion.

Remark 5.2. A direct comparison between REMSO/IPOPT and the other approaches is not entirely fair, because REMSO and IPOPT are the only algorithms that are restricted to use the original control-periods. The other algorithms may refine the periods as appropriate during forward simulation, or when reformulating the problem.

Remark 5.3. Plots of well-trajectories optimized with REMSO is obtained by running a forward simulation using the optimized controls, rather then plotting the MS simulation directly.

5.1 Case A: Simple 10x10 Grid - Output Unconstrained

The first case is mainly proposed to ensure correct implementation of the output unconstrained optimizer OSSO. The problem is also solved with IPOPT and REMSO for verification purposes. A simple two-dimensional 10×10 grid is chosen to keep the complexity at a minimum. It is equipped with one producer and one injector, both controlled by BHP, and their location are shown in Figure 5.1. In the base case, the producer is set to operate at 150 bar, and the injector at 240. The horizon is divided into 40 simulation steps and four control periods. See Appendix B.2.

The input constraints for the problem are listed in Table 5.2. The pressure in all grid-blocks is initialized at 234 bar, and the blocks are populated with equal petrophysical properties. The complete dataset for the reservoir is listed in Table B.1. The NPV was scaled with a factor of 10^5 during optimization. The results

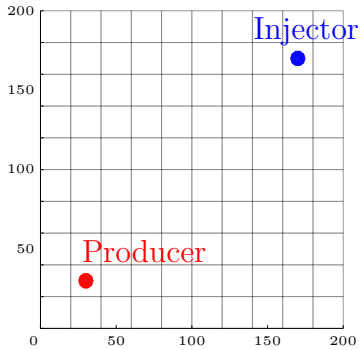


Figure 5.1: Case A: Grid. Also used for Case B

Table 5.2: Case A: Constraints

Wellname	BHP		WRAT		ORAT	
	min	max	min	max	min	max
Producer	150	240	$-\infty$	∞	$-\infty$	∞
Injector	150	240	$-\infty$	∞		

of the experiment are listed in Table 5.3.

Is is straight away observed that all algorithms find the same solution. This indicates that OSSO is implemented correctly.

The well trajectories in Figure 5.2 shows that the producer is kept on the minimum allowed pressure in the optimized case, while the injecting pressure is somewhat

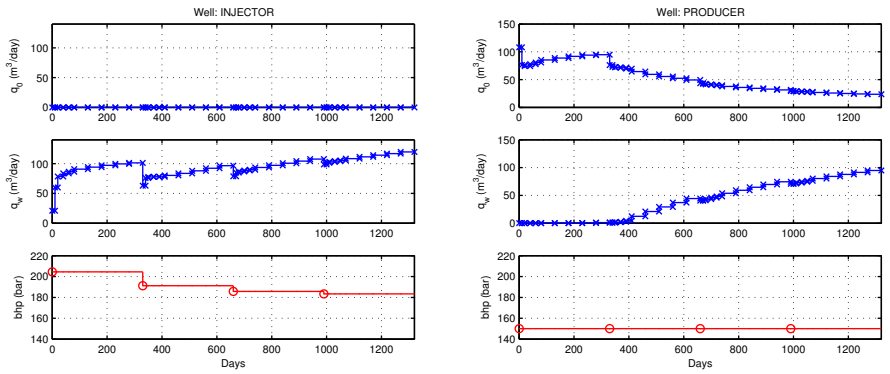


Figure 5.2: Case A: Optimized well trajectories. Equal for REMSO, IPOPT and OSSO.

lowered from the base-case, and decreases further with time. Figure 5.3 illustrates that all algorithms make huge NPV improvement during the first iterations, while smaller steps are taken afterwards.

As expected, the efficiency of both REMSO and OSSO increase as they progress, illustrated in Figure 5.4. Towards the end, when small steps Δu are taken, the mean of the newton iterations closes down to one, indicating almost perfect efficiency. On the contrary, IPOPT needs roughly the same amount of Newton-Iterations for each simulated point.

Table 5.3: Case A: Optimization results

	Algorithm	**NPV** ($\times 10^5$)	**Time** [min]	**Iterations**	**CP**
	Base case	1.1527			
1	REMSO	3.1930	6.5	38	4
2	IPOPT	3.1930	11.3	37	4
3	OSSO	3.1930	6.1	33	4

The distribution of oil saturation and pressure (the reservoir states) for the optimized case is shown in Figure 5.5.

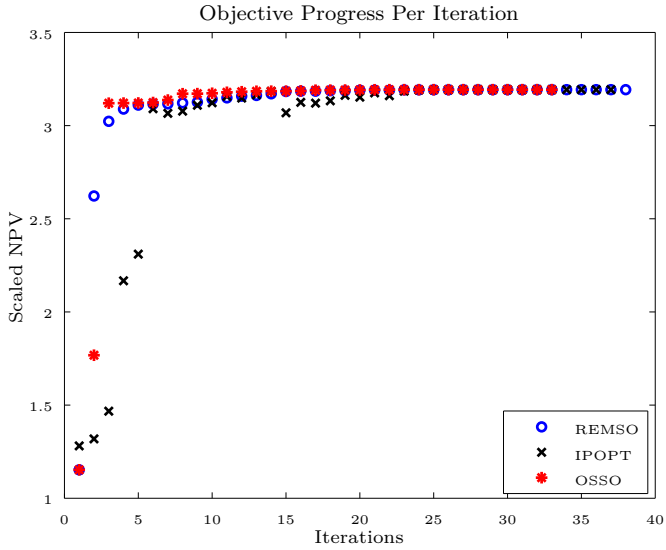


Figure 5.3: Case A: Progress of objective for the iterations

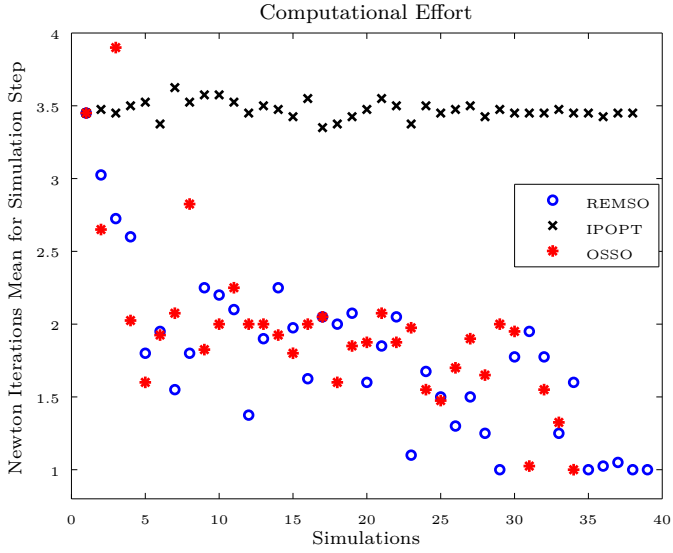


Figure 5.4: Case A: Efficiency of the algorithms. Each marked point indicates the mean number of Newton Iterations (for all timesteps) to solve the reservoir equations.

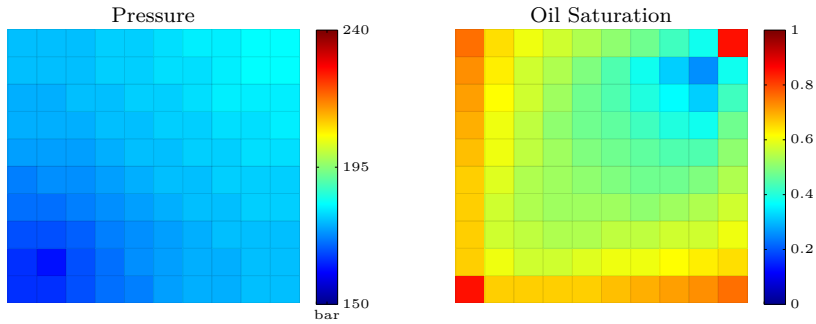


Figure 5.5: Case A: Oil saturation and pressure distribution at the end of the horizon, for the optimized case.

5.2 Case B: Simple 10x10 Grid - Output Constrained

This case is proposed to investigate the methods ability to handle output constraints, on a simple reservoir. To this end, the reservoir from Section 5.1 is reused. Because Figure 5.2 showed that the optimal solution for the output unconstrained case exceeded a production rate of 100 m^3 water per day, the constraint is set to 50 m^3/day. This accounts for 40 constraints in total, one for each time-step. The constraints are listed in Table 5.4. For the base-case, the wells are again operated at 150 and 240 bar. This control is however not feasible w.r.t the output constraints. The organization of the horizon and control periods are the same as in the previous

Table 5.4: Case B: Constraints

Wellname	**BHP**		**WRAT**		**ORAT**	
	min	max	min	max	min	max
Producer	150	240	-∞	50	-∞	∞
Injector	150	240	∞	∞		

case, see Appendix B.2. The results of the experiment are presented in Table 5.5. REMSO and IPOPT finds the same optimal solution, but REMSO used approximately half the time spent by IPOPT. The scaled NPV obtained by RC1 is 96.1% of the solution found by IPOPT and REMSO, and the solution of RC2 is 99.8% of the same figure. However, RC2 needs the additional time required by OSSO to converge, whereas RC1 only need a single forward simulation. Furthermore, REF1 and REF2 were limited to 45 iterations, because they started to alternate between formulations, apparently not making progress towards convergence. Nevertheless, REF1 obtained higher NPV than REMSO and IPOPT. REF2 generated the highest for this case, slightly improving the starting point from REMSO.

Table 5.5: Case B: Optimization results

	Algorithm	**NPV** ($\times 10^5$)	**Time** [min]	**Iterations**	**CP**	**Reformulations**
	Base case	1.1527				
1	REMSO	3.1620	3.1	31	4	∼
2	IPOPT	3.1620	6	18	4	∼
4	RC1	3.0401	0.2	∼	∼	∼
5	RC2	3.1549	6.5	33	∼	∼
6	REF1	3.1715	8.3	45*	24	30
7	REF2	3.1735	6.5	45*	17	12

The well trajectories for REMSO/IPOPT, RC1 and REF1 are shown in Figure 5.7. For REF1 and RC1, this figure illustrates how a well-variable can alternate between being controlled, and calculated from output equations. The well-trajectories for the remaining methods RC2 and REF2 are included in Appendix B.3. The pressure and distribution of oil-saturation in the reservoir is shown in Figure 5.6 for two of the optimized cases.

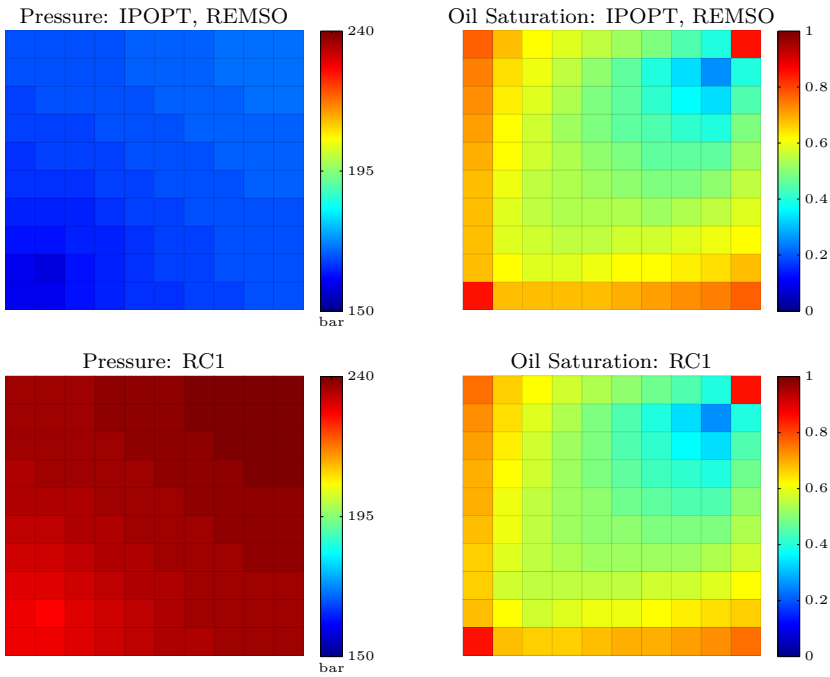
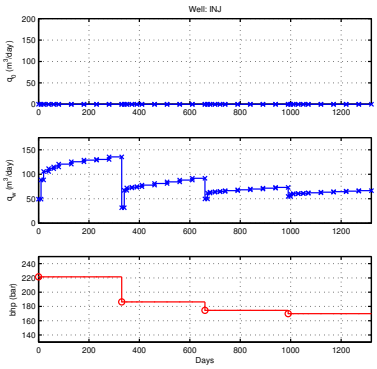
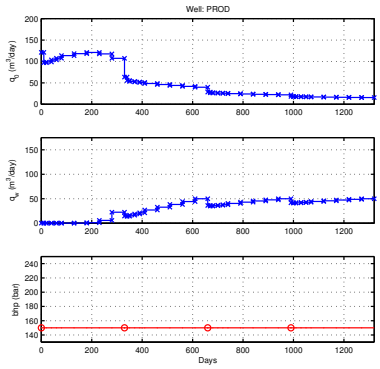


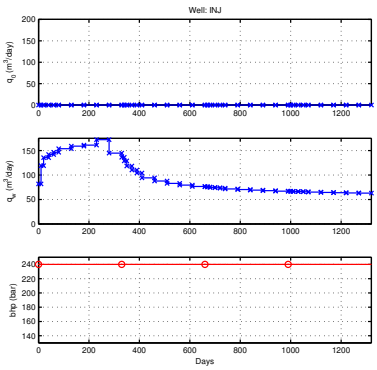
Figure 5.6: Case B: Oil saturation and pressure distribution after optimization



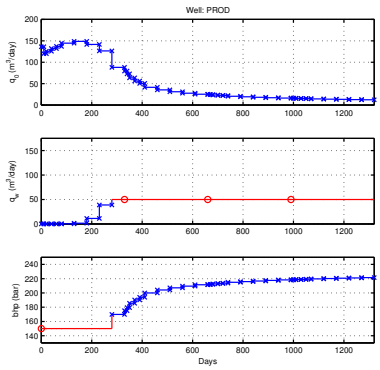
(a) REMSO, IPOPT



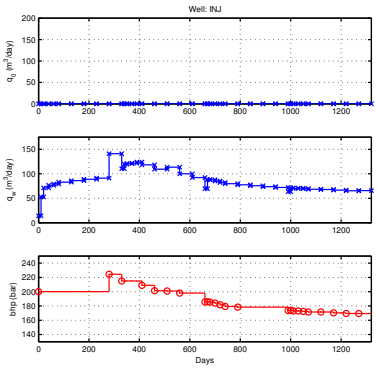
(b) REMSO, IPOPT



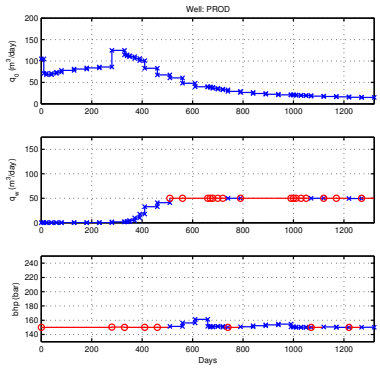
(c) RC1



(d) RC1



(e) REF1



(f) REF1

Figure 5.7: Case B: Optimized well trajectories

5.3 Case C: Five Spot Model

As introduced in Section 1.3, the five spot pattern is frequently used for waterflooding operations. The pattern is in this case applied to a synthetic two-dimensional 60×60 grid proposed by Bellout et al. (2012). The reservoir covers an area of almost 2.1 km^2, with a thickness of 24 m. The reservoir, together with the location of the wells, is shown in Figure 5.8. The reservoir is simulated for 156 steps, divided into 24 control periods, see Appendix B.2. Unlike the previous cases, the porosity and permeability parameters varies throughout the grid, shown in Figure 5.9.

Figure 5.9b illustrates that the injector is placed in a high-permeability zone. Because the producer PROD2 is located nearby the injector, it is expected that this producer will experience early water-breakthrough. On the contrary, due to its location and the permeability-field, PROD1 is expected to produce for a longer time, before the breakthrough occur. The complete dataset for the reservoir is listed in Table B.2 in Appendix B. All wells are controlled by BHP. For the base-case, the

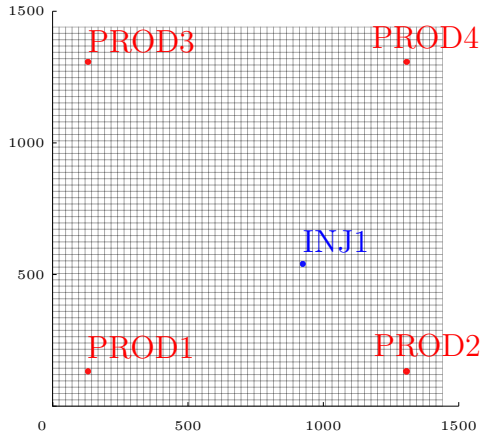


Figure 5.8: Case C: The five spot model

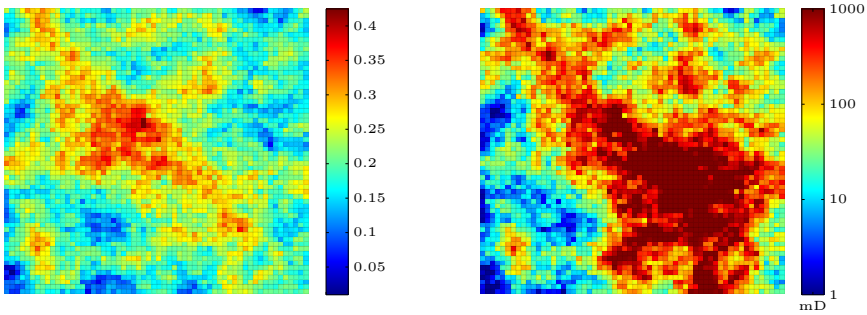
producers are operated at 170 bar, and the injector at 180. Output constraints are included, specifying maximum allowed production rate of water for all producers. The constraints are present for all simulation-steps, which adds up to 624 output constraints in total, summarized in Table 5.6. The results of the optimization are found in Table 5.7.

IPOPT, REMSO, and OSSO[2] were limited to 100 iterations. IPOPT achieved the highest NPV, and converged after 59 iterations. REMSO did not converge, but needed considerably less wall-clock time for the 100 iterations it did perform, than IPOPT used for 59. However, close inspection of the well-trajectories for IPOPT in Figure 5.11b reveals that the producer PROD2 injects oil around day

[2]Needed by RC2

Table 5.6: Case C: Constraints

Wellname	BHP		WRAT		ORAT	
	min	max	min	max	min	max
Prod1	150	200	$-\infty$	100	$-\infty$	∞
Prod2	150	200	$-\infty$	100	$-\infty$	∞
Prod3	150	200	$-\infty$	100	$-\infty$	∞
Prod4	150	200	$-\infty$	100	$-\infty$	∞
Inj1	150	200	$-\infty$	∞		



(a) Case C: Porosity **(b)** Case C: Permeability

Figure 5.9: Case C: Permeability and porosity

2500-3500. This also occur with the controls found by REMSO and REF2. The well-trajectories are infeasible, and is further discussed in the next chapter.

RC1 generated higher NPV than RC2, even though RC2 is based on the optimized control from OSSO.

REF1 was allowed to iterate for a while, but was manually haltered once there was discovered that it alternated between formulations, apparently not making progress. It also expanded the number of control periods from 24 to 61. REF2 slightly managed to improve the solution computed by REMSO, but all the while REMSO was prematurely haltered, this does necessarily not have to mean anything.

All methods managed to more than double the NPV of the base case. The significance of the varying permeability field is visualized in Figure 5.10, showing the distribution of pressure and oil saturation at the end of the horizon. Note that the high-permeability zone around the injector is almost completely saturated with water. All well-trajectories are included in Appendix B.3. Remark that the rate-axis in these figures is restricted to 600 m^3/day, to better reveal the difference between

Table 5.7: Case C: Optimization results

	Algorithm	**NPV** ($\times 10^6$)	**Time** [min]	**Iterations**	**CP**	**Reformulations**
	Base case	4.2639				
1	REMSO	9.5764	76	100*	24	∼
2	IPOPT	9.6353	474	59	24	∼
4	RC1	9.5589	3	∼	∼	∼
5	RC2	9.5131	310	100*	∼	∼
6	REF1	9.5786	470	150*	61	39
7	REF2	9.5851	101	115	30	7

trajectories.

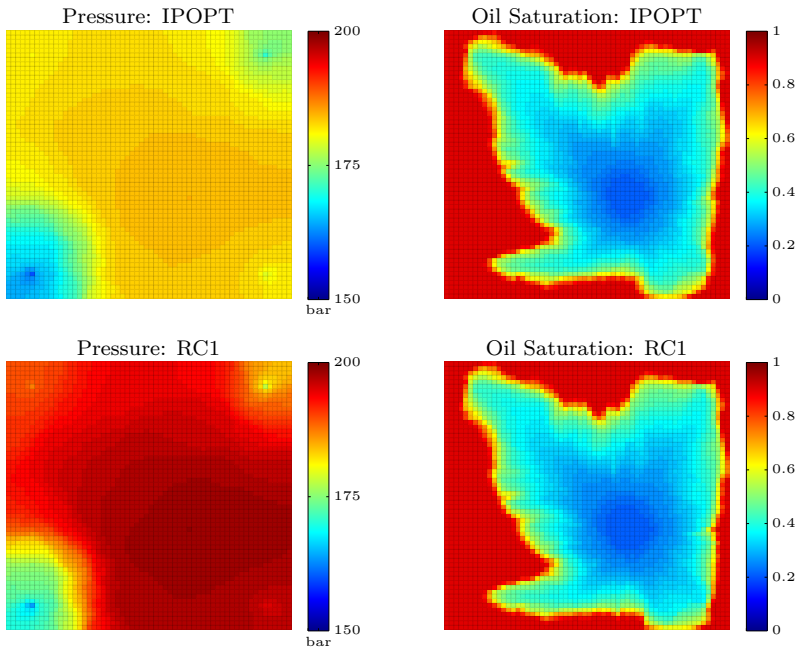
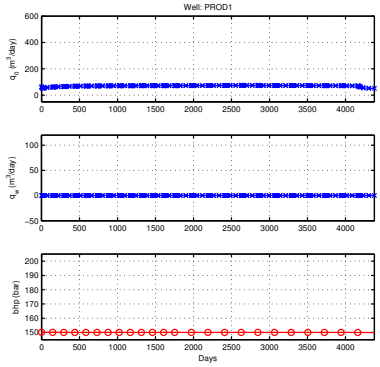
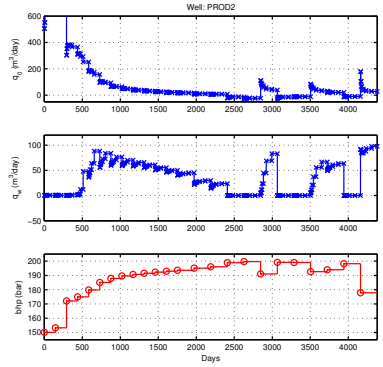


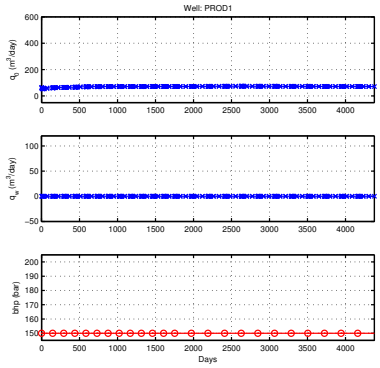
Figure 5.10: Case C: Reservoir states at the end of the horizon.



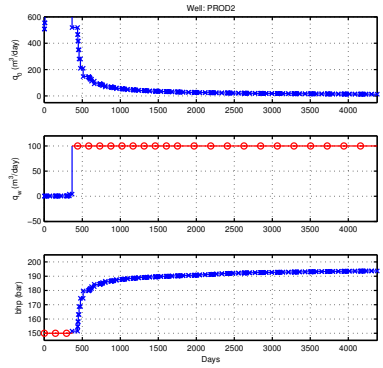
(a) IPOPT



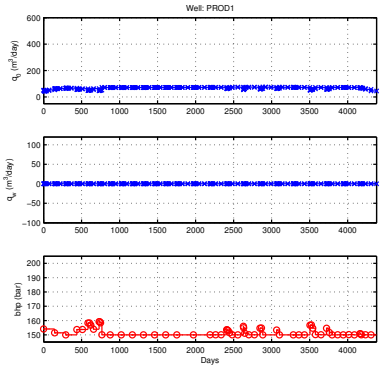
(b) IPOPT



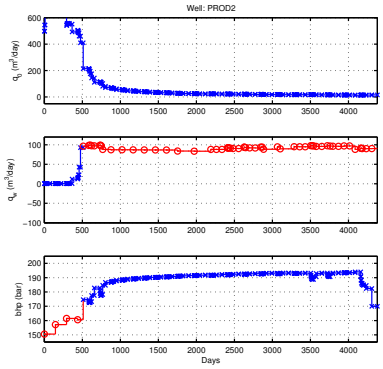
(c) RC1



(d) RC1



(e) REF1



(f) REF1

Figure 5.11: Case C: Optimized well trajectories for PROD1 and PROD2

5.4 Case D: The Egg Model

The Egg Model, which was introduced in Section 4.1.2, is a synthetic reservoir proposed by Jansen et al. (2004). The model consists of $60 \times 60 \times 7 = 25.200$ grid cells, of which 18.553 cells are active. As the non-active cells are at the outside of the model, surrounding the active cells, the active cells forms the shape of an egg - hence its name. In most publications, the model has been used to simulate two-phase (oil-water) flow. Because the model has no aquifer or gas cap, primary production is almost negligible, and the production mechanism is waterflooding, with the aid of eight injectors and four producers. A random permeability realization of the model is shown in Figure 5.12.

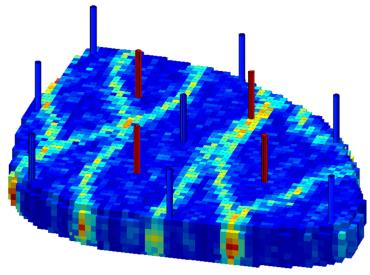
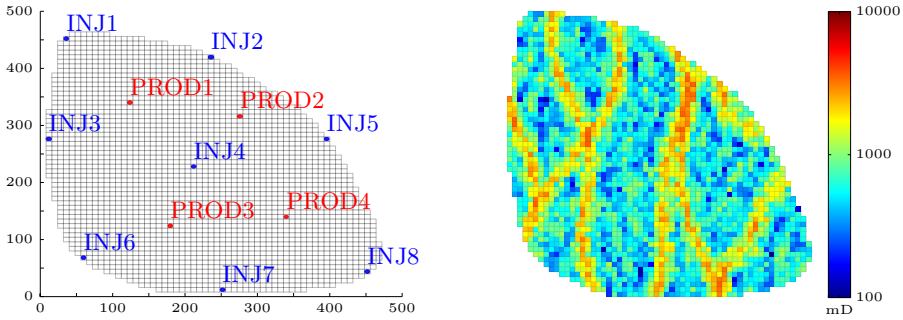


Figure 5.12: A geological realization of the full egg model, with injectors (blue) and producers (red). The vertical scale is exaggerated with a factor two. (Jansen et al., 2004)



(a) Well location **(b)** Permeability field

Figure 5.13: Case D: The egg model implemented MRST

Due to the wall-clock time required to optimize the model on the available equipment, only one of the seven vertical layers is used. This comes with an additional advantage, as cross-flow between layers does not have to be considered for wells with flow-rate near zero. Codas et al. (2015) mentioned this as a problem. The model that is implemented in MRST is shown in Figure 5.13.

The injectors are operated by rate, and the producer are operated by BHP. In order to maintain waterflooding as the primary production method, the pressure-bounds on the producers are set conservatively around the initial reservoir pressure of 400 bar. In the base-case, the producers are set to 395 bar, and the injectors are set to 5 m^3/day. As the previous case showed the importance of bounding rates below for BHP controlled wells, output constraints are incorporated for all producers, for both the oil and the water phase. This accounts for[3] 1200 output constraints. Output constraints on the allowed pressure-region for the injectors are also included, yielding 2400 additional constraints[4]. In total, 3600 output constraints are present. Note that, as outlined in Section 4.3.2, all these constraints cannot be ensured by REF1 and REF2. The constraints are summarized in Table 5.8. The results of the experiment are listed in Table 5.9, and they highlight some

Table 5.8: Case D: Constraints

Wellname	**BHP**		**WRAT**		**ORAT**	
	min	max	min	max	min	max
All 4 producers	390	410	0	∞	0	∞
All 8 injectors	390	410	0	20		

Table 5.9: Case D: Optimization results

	Algorithm	**NPV** ($\times 10^6$)	**Time** [min]	**Iterations**	**CP**	**Reformulations**
	Base Case	2.3186				
1	REMSO	3.1091	265	100*	10	$\sim$
2	IPOPT	3.0027	1180	100*	10	$\sim$
4	RC1	2.4872	2	$\sim$	$\sim$	$\sim$
5	RC2	2.2983	516	100*	$\sim$	$\sim$
6	REF1	2.9213	460	55	47	30*
7	REF2	2.0163	480	141	80	30*

important issues. First, it is again observed that REMSO and IPOPT generates the highest NPV, where REMSO is superior in terms of wall-clock time. RC1 and RC2 only manages to slightly improve the NPV, compared to the base case. Nevertheless, the most interesting result is that REF2 generates a lower NPV compared to the base-case, even though the optimal solution from REMSO is used. Another important observation is that the solution found by REF1 is infeasible (negative rates), see Figure 5.15e and 5.15f. Both issues are further discussed in the next chapter.

[3]Four wells, lower constraints for oil rate and water rate, 150 steps, i.e $4 \times 2 \times 150 = 1200$.
[4]Eight wells, upper and lower constrains, 150 steps, i.e $8 \times 2 \times 150 = 2400$.

Figure 5.15 shows the well trajectories for PROD2 and INJ1, for REMSO, RC2 and REF1. Well-trajectories for IPOPT, RC1 and REF2 are found in Appendix B.3. Distribution of pressure and oil-saturation is shown in Figure 5.14, illustrating the impact of the varying permeability field etc.

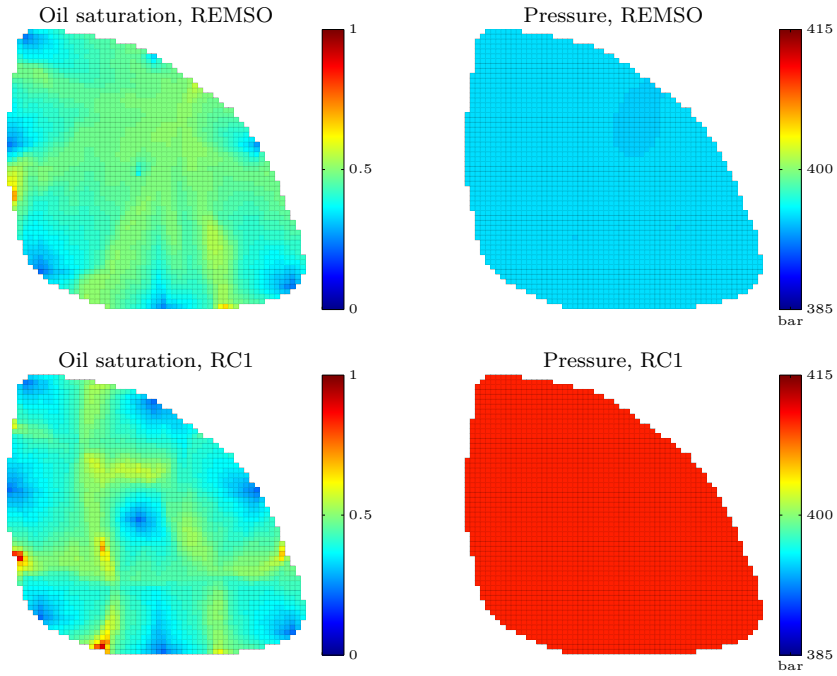
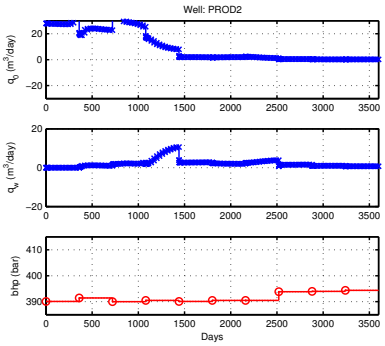
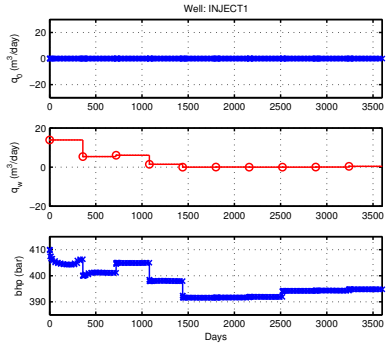


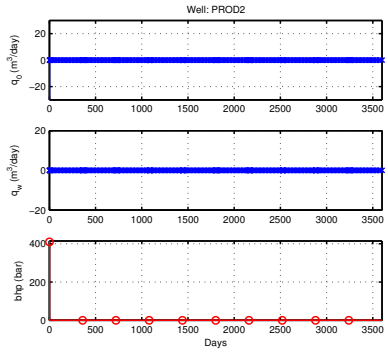
Figure 5.14: Case D: Pressure and oil saturation after 10 years. Optimized with REMSO and RC1



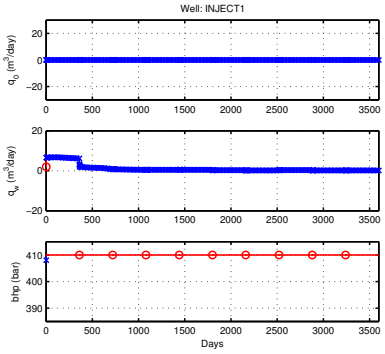
(a) REMSO



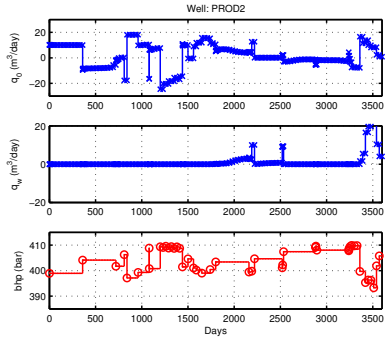
(b) REMSO



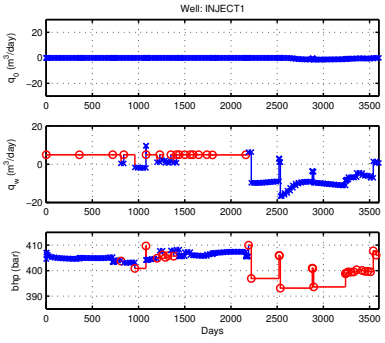
(c) RC2



(d) RC2



(e) REF1



(f) REF1

Figure 5.15: Case D: Well trajectories for REMSO, RC2 and REF1

Chapter 6

Discussion

The merits of the methods that were presented in Chapter 4 are assessed in this chapter, based on the numerical results in Chapter 5. Each case is first discussed individually, and a final comparison follows towards the end.

Case A

The results indicated that OSSO was implemented correctly, as the algorithm found the same control-sequence as REMSO and IPOPT. In terms of runtime, OSSO was the fastest algorithm to converge, slightly faster than REMSO. Their performance relative to IPOPT, are most likely due to the reduced simulation effort, illustrated in Figure 5.4. This highlights the importance of warm-starting Newton-Iterations. Furthermore, as a forward simulation only took a couple of seconds, the parallelism opportunity that comes with the MS-approach in REMSO, did not realize its potential, due to the overhead needed for this small-sized problem. This is emphasized by the fact that the single shooting based OSSO converged faster. To this end, the problem should probably be of a certain size for the parallelism to be beneficial.

The second important result is found in Figure 5.3, which shows that the algorithms make huge NPV improvement during the first iterations. If constraints-violations are put aside, as both IPOPT and REMSO are infeasible-path algorithms, this suggest that good solutions can be found relatively quick. Consequently, there may not always be necessary to let the gradient-based algorithms iterate with very tight convergence criteria.

Case B

This case was intended to investigate the methods ability to handle output constraints. RC1 was of course the fastest method in terms of wall-clock time, as only one forward simulation is required. In addition, the scaled NPV found by RC1

was only 4% lower than the value found by REMSO and IPOPT. This indicates that the simple reactive control heuristic can be used to quickly find better well-trajectories than the base case. Furthermore, the second reactive strategy RC2 slightly improved the NPV compared to RC1, but at the price of the additional run-time required by OSSO.

REF1 generated the highest NPV, but also spent the most amount of time. In fact, as the heuristic seemed to alternate between formulations, it was limited to 45 iterations in total. Already at this stage, this indicates a major drawback for the heuristic, as it did not manage to converge. This can maybe be resolved, or improved, by addressing two of issues outlined in Section 4.3.2, namely how to reformulate the problem, and how to reuse the approximated Hessian. The latter is particularly important, as the matrix is used to determine the search-direction in the restarted problem. Furthermore, the number of control periods were expanded from 4 to 24, drastically increasing the number of decision variables, making the problem harder to solve. Lastly, the reformulation based heuristic REF2 was also tried as means of improving the controls found by REMSO, of which it slightly managed.

Case C

The first key observation in this case is the benefit of the parallelism in the multiple shooting approach, as REMSO only used 79 minutes for 100 iterations. IPOPT needed on the other hand 474 minutes for 59 iterations. This does not tell the entire truth, as IPOPT could be improved by warm-starting Newton-Iterations in MRST. However, based on these observations, it is likely to believe that parallelism is of significant importance for problems where simulations are computationally expensive

The second key observation is that the controls found by REMSO, IPOPT, and REF2 resulted in infeasible well-trajectories. Indeed, the producer PROD2 started to inject oil after around 2500 days. These trajectories can obviously not be achieved on a real field. A lesson learned is that there should be incorporated output constraints on the minimum allowed flow-rate, for wells controlled by BHP. Another approach to prevent this issue is to use a-priori knowledge of the system to adjust the feasible pressure region for BHP-controlled wells. Such knowledge can for instance be obtained by running forward simulations in advance.

RC1 obtained surprisingly satisfactory NPV, all the while it only needs a single forward simulation. The NPV obtained by RC2 was in fact worse, but this can possibly be explained by the fact that also the well-trajectories found by OSSO was infeasible. Nevertheless, Figure B.5 shows that PROD2 was shut after about 1700 days due to the amount of water produced relative to oil. This illustrates the necessity of shutting the well to prevent unprofitable "short-circuiting" between injectors and producers.

REF1 expanded the number of control periods from 24 to 61, more than doubling

the number of decision variables. Consequently, as the number of controls increases, more local minimums can be introduced, and the problem gets gradually harder to solve.

The results also confirmed the expectations, namely that PROD2 experienced early water-breakthrough for all methods, while it never occurred for PROD1. RC2 handled this by shutting PROD2 entirely, while IPOPT and REMSO handled it by increasing the controlled pressure, to lower the flow. This even resulted in the negative flow discussed previously.

Case D

REMSO and IPOPT showed their strength on the Egg-Model, and obtained the highest NPV by far. REMSO was again superior in terms of wall-clock time, using only 265 minutes, whereas IPOPT needed 1180 minutes for the same number of iterations. However, there were two important findings. The first is that REF1 found infeasible well-trajectories for almost all 12 wells. This is because a lower-limit on rate cannot be included as a constraint in MRST, see Section 3.3 and 4.3.2. To this end, positive flow-rate cannot be ensured for BHP-controlled wells. This can also occur if a rate controlled well switches to BHP control. If there had been time, this feature could maybe have been implemented in MRST, as the simulator is open-source. This includes making MRST able to handle minimum rate constraints on producers and injectors, handle maximum pressures for producers, and minimum pressure for injectors. Expansion of control-periods in combination with frequent control-switching make the well trajectories for REF1 and REF2 appear chaotic.

The second important finding was that REF2 generated a lower NPV than the base case, even though the controls optimized by REMSO was used as starting point. This manifests a weakness of the reformulation based heuristic, namely that the outcome is not deterministic.

OSSO also computed infeasible well trajectories, but it was expected that this might happen, as output constraints are not present. This affects the RC2 heuristic, as negative production rate implies that the corresponding producers are shut immediately, shown in Figure 5.15c. It even resulted in a NPV that were lower than for the base-case. This is a drawback for the RC2 heuristic that was not discussed by Kourounis et al. (2014), which originally proposed the heuristic. This could to a large degree been prevented by choosing the input BHP constraints wisely, or by using strictly rate-controlled wells.

Final Comparison

The adjoint-based methods REMSO and IPOPT consistently found good solutions in terms of NPV. In particular, Case D showed their strength in presence of multiple output constraints. Moreover, case A indicated that the overhead with the MS

approach means that parallelism is probably not beneficial if simulations are very cheap. Furthermore, case C and D showed REMSO's ability to efficiently handle output constraints on slightly larger problems. This is one of the advantages of the MS-approach, because constraints on algebraic output variables and reservoir states are easily included as bounds on decision variables in each shooting interval. Lastly, the benefit of warm-starting Newton-Iterations in MRST was confirmed for REMSO and OSSO.

The reactive approach RC1 were in all cases able to improve the NPV for the base case. In case B and C, the NPV was also surprisingly close to those obtained by REMSO and IPOPT. Considering that the approach is model-independent, and only require one forward simulation when implemented on a computer, it is easy to understand its popularity in the industry. Moreover, RC2 provided higher NPV than RC1 in case B and D. However, as revealed in case D, the controls obtained by output unconstrained optimization may yield infeasible well-trajectories, if not input constraints for the initial optimization are carefully chosen. This again may impact RC2 unnecessarily. A final remark on the reactive control approach, supported by Figure 5.6, 5.10, and 5.14 is that it often leaves the reservoir in a state of higher pressure, compared to gradient-based procedures. This suggest that the initial energy in the reservoir is not utilized as efficient as with other approaches.

The major workload during this work was to develop, and especially implement the reformulation-based heuristic. However, case D stated a major drawback, namely that the outcome of the approach is completely indeterministic. Moreover, in order to provide feasible results on realistically scaled problems, minimum rate constraints should be implemented in MRST, ensuring that producers can not act as injectors, and vice versa. For adjoint-based optimization, it is usual to use more refined simulation steps at the beginning of each control-period. This may not be appropriate for the reformulation based heuristic, as control-periods are most likely changed anyway. It also remain to be figured out how different reformulations can be applied, and how the Hessian can be initialized properly, to improve the heuristics convergence properties.

Chapter 7

Conclusion

This thesis have showed that it is possible to solve an output **constrained** control optimization problem for reservoir waterflooding, with a heuristic that continuously reformulates the problem. To this end, an output **unconstrained** NLP algorithm is used. However, this method achieved lower NPV than adjoint-based optimization approaches for the realistically sized problems. Besides the relatively low complexity, it was not discovered any significant advantages. It might be that some positive aspects can be revealed in the future, but this would require that the three issues at the current stage are addressed and resolved. These issues centers around how to reformulate the problem, and how reuse the approximated Hessian, in addition to equip MRST with additional functionality to handle more types of constraints.

It was also observed that the simple reactive control in many cases perform adequately compared to gradient-based optimization algorithm, especially when accounting for its simplicity and model-independence.

If aspects regarding reservoir uncertainty are put aside, then the adjoint-based optimization algorithms IPOPT and REMSO generated higher NPV than reactive control for all cases considered in this work. They appear as the preferred methods, especially in the presence of multiple output constraints.

7.1 Future Work

A natural extension of this work is to address the three major issues related to the reformulation based heuristic. It can be investigated how different methods to initialize the Hessian affect the heuristic, and possible strategies for how to reformulate the problem. In addition, MRST can be equipped with the functionality to handle additional constraints, see Section 3.3. If these issues are resolved, an analysis of how the heuristic handles different problems can be considered.

Another interesting possibility, which there already has been conducted studies on, is how gradient-based approaches are affected by different types of problems and constraints. For instance, when should wells be operated by BHP, and when by rate. Advantages and drawback can be classified.

There has also been thought of a way to extend the NLP into a MINLP. If the reservoir is explored beforehand, one can estimate when water-breakthrough is likely to occur. To this end, discrete decision-variables can be included at certain points in the horizon, deciding if a well should be entirely shut or remain open.

Bibliography

Albersmeyer, J. and Diehl, M. (2010). The Lifted Newton Method and Its Application in Optimization. *SIAM Journal on Optimization*, 20(3):1655–1684.

Asheim, H. (1988). Maximization of water sweep efficiency by controlling production and injection rates. In *Paper SPE 18365 presented at the SPE European Petroleum Conference, London, UK, October 16-18*, number 3.

Avigad, J. and Donnelly, K. (2004). Formalizing O notation in Isabelle/HOL. *Automated Reasoning*.

Bellout, M. C., Echeverría Ciaurri, D., Durlofsky, L. J., Foss, B., and Kleppe, J. (2012). Joint optimization of oil well placement and controls. *Computational Geosciences*, 16(4):1061–1079.

Biegler, L. T. (2007). An overview of simultaneous strategies for dynamic optimization. *Chemical Engineering and Processing: Process Intensification*, 46(11):1043–1053.

Biegler, L. T. (2010). *Nonlinear programming : concepts, algorithms, and applications to chemical processes*. Mos-Siam series on optimization. Mathematical Optimization Society and the Society for Industrial and Applied Mathematics.

Biegler, L. T., Cervantes, A. M., and Wächter, A. (2002). Advances in simultaneous strategies for dynamic process optimization. *Chemical Engineering Science*, 57(4):575–593.

Binder, T., Blank, L., Bock, H. G., Bulirsch, R., Dahmen, W., Diehl, M., Kronseder, T., Marquardt, W., Schlöder, J. P., and Stryk, O. (2001). Introduction to Model Based Optimization of Chemical Processes on Moving Horizons. In Grötschel, M., Krumke, S., and Rambau, J., editors, *Online Optimization of Large Scale Systems*, chapter 3, pages 295–339. Springer Berlin Heidelberg.

Bradley, A. M. (2013). Pde-constrained optimization and the adjoint method.

Brouwer, D. and Jansen, J. (2004). Dynamic Optimization of Waterflooding With Smart Wells Using Optimal Control Theory. *SPE Journal*, 9(4):29–31.

Brouwer, R. (2004). *Dynamic water flood optimization with smart wells using optimal control theory*. PhD thesis, TU Delft.

Carlson, M. (2003). *Practical Reservoir Simulation*. PennWell Boks.

Codas, A. (2015). Reservoir multiple shooting optimization code and cases, [Online]. Available: https://github.com/iocenter/remso.

Codas, A., Foss, B., and Camponogara, E. (2015). Output-Constraint Handling and Parallelization for Oil-Reservoir Control Optimization by Means of Multiple Shooting. *SPE Journal*, (SPE-174094-PA).

Conti, J. (2014). *International Energy Outlook 2014: World Petroleum and Other Liquid Fuels With Projections to 2040*. U.S. Energy Information Administration (EIA), Washington.

Dolle, N., Brouwer, D., and Jansen, J. (2002). Dynamic optimization of water flooding with multiple injectors and producers using optimal control theory. In *Proc. XIV International Conference on Computational Methods in Water Resources, Delft (2002) June 23-28*.

Echeverría Ciaurri, D., Isebor, O. J., and Durlofsky, L. J. (2010). Application of derivative-free methodologies to generally constrained oil production optimization problems. *Procedia Computer Science*, 1(1):1301–1310.

Foss, B. and Heirung, T. (2013). Merging Optimization and Control. pages 1–57.

Foss, B. and Jensen, J. (2011). Performance Analysis for Closed-Loop Reservoir Management. *SPE Journal*, 16(1).

Gill, P. E., Murray, W., and Saunders, M. A. (2002). SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization. *SIAM Journal on Optimization*, 12(4):979–1006.

Gravdahl, J. T. and Egeland, O. (2002). *Modeling and Simulation for Automatic Control*. Marine Cybernetics.

Gunnerud, V. (2011). *On decomposition and piecewise linearization in petroleum production optimization*. PhD thesis, Norwegian University of Science and Technology.

Jahn, F., Cook, M., and Graham, M. (2008). *Hydrocarbon Exploration and Production*. Second edition.

Jansen, J. D. (2011). Adjoint-based optimization of multi-phase flow through porous media - A review. *Computers and Fluids*, 46(1):40–51.

Jansen, J. D. (2013). *A systems description of flow through porous media*. Springer.

Jansen, J. D., Fonseca, R. M., Kahrobaei, S., Siraj, M. M., Essen, G. M. V., and den Hof, P. M. J. V. (2004). The egg model – a geological ensemble for reservoir simulation. *Geoscience Data Journal*, 1(2):192–195.

Kawajir, Y., Laird, C. D., and Waechter, A. (2010). Introduction to IPOPT: A tutorial for downloading, installing, and using IPOPT .

Kourounis, D., Durlofsky, L. J., Jansen, J. D., and Aziz, K. (2014). Adjoint formulation and constraint handling for gradient-based optimization of compositional reservoir flow. *Computational Geosciences*, 18:117–137.

Kraaijevanger, J., Egberts, P., Valstar, J., and Buurman, H. (2013). Optimal waterflood design using the adjoint method. In *SPE 105764*.

Krogstad, S., Lie, K. A., Moyner, O., Nilsen, H. M., Raynaud, X., Skaflestad, B., and SINTEF, I. (2015). MRST-AD – an Open-Source Framework for Rapid Prototyping and Evaluation of Reservoir Simulation Problems. In *Society of Petroleum Engineers*. Society of Petroleum Engineers.

Kühl, P., Diehl, M., Kraus, T., Schlöder, J. P., and Bock, H. G. (2011). A real-time algorithm for moving horizon state and parameter estimation. *Computers and Chemical Engineering*, 35:71–83.

Leineweber, D. B., Bauer, I., Bock, H. G., and Schlöder, J. P. (2003). An efficient multiple shooting based reduced sqp strategy for large-scale dynamic process optimization. part 1: Theoretical aspects. *Computers and Chemical Engineering*, 27(4):157–166.

Lie, K. A. (2014). *An Introduction to Reservoir Simulation Using MATLAB: User guide for the Matlab Reservoir Simulation Toolbox (MRST)*. SINTEF ICT, Departement of Applied Mathematics, Oslo, Norway.

Lie, K. A., Krogstad, S., Ligaarden, I. S., Natvig, J. R., Nilsen, H. M., and Skaflestad, B. (2012). Open-source MATLAB implementation of consistent discretisations on complex grids. *Computational Geosciences*, 16:297–322.

Lie, K. A. and Mallison, B. T. (2013). Mathematical models for oil reservoir simulation. In *Encyclopedia of Applied and Computational Mathematics*, pages 1–8. Springer-Verlag Berlin Heidelberg.

Lynch, D. R. (2005). *Numerical partial differential equations for environmental scientists and engineers: A first practical course*. Springer US.

Navasca, C. and Krener, A. (2000). Solution of Hamilton Jacobi Bellman equations. In *Proceedings of the 39th IEEE Conference on Decision and Control*, volume 1.

Nelder, J. A. and Mead, R. (1965). A Simplex Method for Function Minimization. *The Computer Journal*, 7(4):308–313.

Nelson, S. A. (2012). Physical Geology: http://www.tulane.edu/ ˜sanelson/eens1110/energy.htm.

Nocedal, J. and Wright, S. J. (1999). *Numerical Optimization*, volume 43.

Oljedirektoratet (2015). Fordeling av oljereserver og ressurser for de største oljefeltene i drift per 31.12.2014: http://npd.no/Tema/Ressursregnskap-og-analyser/Temaartikler/Norsk-sokkel-i-tall-kart-og-figurer/Fordeling-av-oljereserver/.

Peters, L., Arts, R., Brouwer, G., Geel, C., Cullick, S., Lorentzen, R., Chen, Y., Dunlop, N., Vossepoel, F., Xu, R., Sarma, P., Alhuthali, A., and Reynolds, A. (2010). Results of the Brugge Benchmark Study for Flooding Optimization and History Matching. *SPE Reservoir Evaluation and Engineering*, 13(3).

Pontryagin, L. S. (1987). *L.S. Pontryagin Selected Works - Mathematical Theory of Optimal Processes, Volume 4*. Gordon and Breach Science Publishers.

Rawlings, J. and Mayne, D. W. (2009). *Model Predictive Control: Theory and Design*. Nob Hill Pub.

Sargent, R. and Sullivan, G. (1978). The development of an efficient optimal control package. In Stoer, J., editor, *Optimization Techniques*, volume 7 of *Lecture Notes in Control and Information Sciences*, pages 158–168. Springer Berlin Heidelberg.

Schittkowski, K. (1986). NLPQL: A fortran subroutine solving constrained nonlinear programming problems. *Annals of Operations Research*, 5(1-4):485–500.

Schmid, C. and Biegler, L. (1994). Quadratic programming methods for reduced hessian SQP. *Computers & Chemical Engineering*, 18(9):817–832.

SPE (2014). Petrowiki: Waterflooding: http://petrowiki.org/Waterflooding.

Sudaryanto, B. and Yortsos, Y. C. (2000). Optimization of fluid front dynamics in porous media using rate control. I. Equal mobility fluids. *Physics of Fluids*, 12(7):1656–1670.

Sudaryanto, B. and Yortsos, Y. C. (2001). Optimization of displacements in porous media using rate control. *Paper SPE 71509 presented at the SPE Annual Technical Conference and Exhibition, New Orleans*, 30.

Thakur, G. (1996). What Is Reservoir Management? *Journal of Petroleum Technology*, 48(6):520–525.

U.S. Energy Information Agency (2013). International Energy Outlook 2013. *Outlook 2013*.

van Essen, G., Zandvliet, M., Van den Hof, P., Bosgra, O., and Jansen, J.-D. (2009). Robust Waterflooding Optimization of Multiple Geological Scenarios. *SPE Journal*, 14(1):24–27.

Virnovsky, G. A. (1991). Waterflooding strategy design using optimal control theory. In *IOR 1991 - 6th European Symposium on Improved Oil Recovery*.

Zakirov, I., Aanonsen, S., Zakirov, E., and Palatnik, B. (1996). Optimizing Reservoir Performance by Automatic Allocation of Well Rates. *Proceedings of the 5th European Conference on the Mathematical Oil Recovery*, pages 3–5.

Appendix A

Sensitivities

A key ingredient for every gradient-based optimization algorithm is quite obviously the gradient itself. These algorithms typically requires gradients with respect to both the objective and the constraining functions. Sufficiently accurate gradients are necessary in order for the algorithms to converge successfully, while the time spent on obtaining them is crucial for the overall run-time performance.

A.1 Background

The gradient is defined

Definition A.1.1 (gradient). The gradient is a generalization of the derivative for a one-dimensional scalar function, extended to multidimensional functions. For a function $f : \mathbb{R}^N \to \mathbb{R}$ in a rectangular coordinate system, the gradient yields

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \dots & \frac{\partial f}{\partial x_N} \end{bmatrix} \tag{A.1}$$

This appendix is concerned with three methods in particular for calculating gradients, namely the method of finite differences, the forward method, and the adjoint method. Whereas the first method only computes an approximation of the gradient, the forward and the adjoint method differentiate the objective analytically, by application of the chain rule. The theory for the method of finite differencing is widespread, and can be found in almost every book that deals with numerical analysis. However, because Kraaijevanger et al. (2013) presents all methods of interest in a concise manner, suitable for dynamic optimization, it is used as the main reference for this appendix.

Recall the semi-explicit system defined in Section 2.1. The parameter vector p, and the algebraic variables v are neglected throughout this Appendix for simplicity. Let the discrete state transition function be given by

$$x^{i+1} = f^i(x^i, u^i) \quad , \quad i = 0, 1, \dots, N-1 \tag{A.2}$$

with the states $x^i \in \mathbb{R}^{n_x}$ and the controls $u^i \in \mathbb{R}^{n_u}$. Next, define the following vectors

$$x = \begin{bmatrix} x^1 \\ x^2 \\ \vdots \\ x^N \end{bmatrix} \quad , \quad u = \begin{bmatrix} u^0 \\ u^1 \\ \vdots \\ u^{N-1} \end{bmatrix} \tag{A.3}$$

such that the discrete system (A.2) can be compressed into one "super" function

$$F(x, x^0, u) = 0 \tag{A.4}$$

with initial conditions x^0. More specifically, the super function F represents

$$F(x, x^0, u) = \begin{bmatrix} x^1 - f^0(x^0, u^0) \\ x^2 - f^1(x^1, u^1) \\ \vdots \\ x^N - f^{N-1}(x^{N-1}, u^{N-1}) \end{bmatrix} \tag{A.5}$$

In either case of optimal control or parameter estimation, the objective is to minimize an objective-function. For discrete systems this function is separable, and given by

$$\min \ J(x, u) = J(x(u), u) = \sum_{i=1}^{N} J^i(x^i, u^i) \tag{A.6}$$

The first step is to differentiate the cost-function with respect to the controls, by the use of the rule for total derivatives, which states

Definition A.1.2 (Total derivative). For a function $M(t, p_1, p_2, \ldots, p_n)$, the total derivative rule is given by

$$\frac{\mathrm{d}M}{\mathrm{d}t} = \frac{\partial M}{\partial t} + \sum_{i=1}^{n} \frac{\partial M}{\partial p_i} \frac{\mathrm{d}p_i}{\mathrm{d}t} \tag{A.7}$$

such that (A.6) becomes

$$\frac{\mathrm{d}}{\mathrm{d}u} J(x(u), u) = \frac{\partial J}{\partial u} + \frac{\partial J}{\partial x} \frac{\mathrm{d}x}{\mathrm{d}u} = J_u + J_x \frac{\mathrm{d}x}{\mathrm{d}u} \tag{A.8}$$

where J_u and J_x denotes the Jacobian of J with respect to u and x, respectively. Furthermore, the matrix

$$\frac{\mathrm{d}x}{\mathrm{d}u} = \left(\frac{\partial x_i}{\partial u_j} \right) \tag{A.9}$$

is the matrix of partial derivatives of the states x with respect to the controls u. This matrix is significant for both the forward and the adjoint method. In the following sections, the three mentioned methods are presented and evaluated. However, only a discrete derivation is included for the forward and the adjoint method, see for instance (Bradley, 2013) for a continuous derivation.

A.2 Finite Differences

The method of finite differences is by far the simplest approach for approximating derivatives. This method treats the objective function as a black-box that is capable of computing the function values $J(u)$, given the input u. Unlike exact methods, the content of J itself is not required, only its value subject to the argument.

In short, the gradient is approximated by perturbing J in the direction of interest, and the difference of the objective value is divided by the size of the perturbation. The method is closely related to the definition of a derivative, which is given by

Definition A.2.1 (Derivative). The derivative of a function f is defined by the limit

$$\frac{\mathrm{d}f}{\mathrm{d}u} = f'(u) = \lim_{h \to 0} \frac{f(u+h) - f(u)}{h} \tag{A.10}$$

The concept of Big Oh is in addition required to classify the advantages and drawbacks for the method. Let the Big Oh be denoted by $\mathcal{O}$. Avigad and Donnelly (2004) proposes the following definition

Definition A.2.2 ($\mathcal{O}$ notation). If f and g are functions, the notation $f(u) = \mathcal{O}(g(u))$ is used to express the fact that f's rate of growth is no bigger than that of g, in the following sense

$$f(u) = \mathcal{O}(g(u)) \quad \text{means } \exists\, c \ \forall\, u \text{ such that } (|f(u)| \leq c \cdot |g(u)|) \tag{A.11}$$

Loosely speaking, $f = \mathcal{O}(u)$, indicates that the growth rate of f is no bigger than the asymptotic growth rate of u, which is linear. Likewise, $g = \mathcal{O}(u^2)$ indicates that the growth rate of g is no bigger than the growth rate of u^2, which is quadratic. Now, let the forward difference formula be given by the following Theorem A.2.1.

Theorem A.2.1. *(Forward Difference) The forward difference method approximates the derivative of a function f by the expression*

$$f'(u_i) \approx \frac{f(u_i + \delta u_i) - f(u_i)}{\delta u_i}$$

with approximation error

$$e = \mathcal{O}(\delta u_i)$$

Proof. Expand the function f at u_{i+1} about u_i by it's Taylor Series

$$f(u_i + \delta u_i) = f(u_i) + \delta u_i \left.\frac{\partial f}{\partial u}\right|_{u_i} + \left.\frac{\delta u_i^2}{2!}\frac{\partial^2 f}{\partial u^2}\right|_{u_i} + \left.\frac{\delta u_i^3}{3!}\frac{\partial^3 f}{\partial u^3}\right|_{u_i} + \dots$$

by rearranging terms and dividing by δx, this Taylor Series can be rearranged into the following

$$\frac{f(u_i + \delta u_i) - f(u_i)}{\delta u_i} - \left.\frac{\partial f}{\partial u}\right|_u = \underbrace{\left.\frac{\delta u_i}{2!}\frac{\partial^2 f}{\partial u^2}\right|_u + \left.\frac{\delta u_i^2}{3!}\frac{\partial^3 f}{\partial u^3}\right|_u + \dots}_{\text{Truncation Error}}$$

For smooth functions, where higher order derivatives exists, the first term of the truncation error characterizes the order of magnitude of the error, as this term dominates the other terms if the step size δu_i is sufficiently small. This implies that the error is $\mathcal{O}(\delta u)$. Then

$$f'(u_i) = \left.\frac{\partial f}{\partial u}\right|_{u_i} = \frac{f(u_i + \delta u_i) - f(u_i)}{\delta u_i} + \mathcal{O}(\delta u_i)$$

□

An important implication from Theorem A.2.1 is that the approximation error of the forward difference approximation increases with the step-size δu. Indeed, if $\delta u \to 0$, then the approximated gradient by the forward difference equals its derivative, stated in definition A.2.1.

The last step is to extend the forward difference formula for scalar functions in Theorem A.2.1, to fit dynamic optimization problems. Let the unit vector for all the controls $j = 1, 2, \dots, n_u$ be defined $\hat{u}_1$, together with an appropriate sized scalar $\delta_j \in \mathbb{R}^+$. The forward difference method then reads

$$\frac{\mathrm{d}J}{\mathrm{d}u} \approx \left[\frac{J(u+\delta_1\hat{u}_1)-J(u)}{\delta_1} \quad \frac{J(u+\delta_2\hat{u}_2)-J(u)}{\delta_2} \quad \dots \quad \frac{J(u+\delta_{n_u}\hat{u}_{n_u})-J(u)}{\delta_{n_u}}\right] \in \mathbb{R}^{n_u} \tag{A.12}$$

Similarly, the objective J can be perturbed in the opposite direction, resulting in the *backward difference* method

$$\frac{\mathrm{d}J}{\mathrm{d}u} \approx \left[\frac{J(u)-J(u-\delta_1\hat{u}_1)}{\delta_1} \quad \frac{J(u)-J(u-\delta_2\hat{u}_2)}{\delta_2} \quad \dots \quad \frac{J(u)-J(u-\delta_{n_u}\hat{u}_{n_u})}{\delta_{n_u}}\right] \tag{A.13}$$

The last method to be considered is the *central difference* method, which perturbs the function of interest in both forward and backward direction

$$\frac{\mathrm{d}J}{\mathrm{d}u} \approx \left[\frac{J(u+\frac{1}{2}\delta_1\hat{u}_1)-J(u-\frac{1}{2}\delta_1\hat{u}_1)}{\delta_1} \quad \dots \quad \frac{J(u+\frac{1}{2}\delta_{n_u}\hat{u}_{n_u})-J(u-\frac{1}{2}\delta_{n_u}\hat{u}_{n_u})}{\delta_{n_u}}\right] \tag{A.14}$$

Lynch (2005) shows that the approximation error of the backward difference method also is $\mathcal{O}(\delta u_i)$, while the approximation error for the central method is $\mathcal{O}(\delta u_i^2)$. Both the forward and backward method require $(n_u + 1)$ simulator runs, while the central method require $(2n_u + 1)$. However, the latter usually results in more accurate approximation. The computational cost grows asymptotically linear with $\dim(u)$ for all methods.

An important drawback for all finite difference method is the trickiness of selecting

proper perturbation-sizes δ_j. At one hand, the perturbation size must be chosen big enough to overcome numerical noise, while at the same time be kept small enough to make satisfactory approximations. On the contrary, as stated in the beginning, the advantage for this class of methods is its simplicity and ease of implementation. An illustration of the three finite difference methods is shown in Figure A.1. Note that one should chose the particular finite difference method

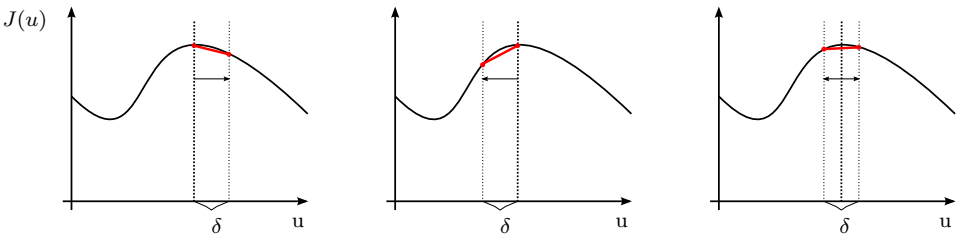


Figure A.1: A scalar illustration of the forward, backward and central finite difference method, respectively. The derivative is approximated as the slope of the red line.

with care for constrained optimization. For instance, if the the input u represents the water-height in an open tank of water with wall-height $\bar{x}$, forward difference cannot be applied if $x = \bar{x}$, as $\bar{x}$ represents a hard-bound. In this case, backward difference must be applied.

A.3 Forward Sensitivity

Unlike the finite difference methods, the forward sensitivity method utilizes analytical differentiation of the system equations (A.4) in order to compute the derivatives. More specifically, it solves the *sensitivity equation* forward in time. The key element in this approach is the construction of the sensitivity matrix defined in (A.9), namely

$$S = \frac{\mathrm{d}x}{\mathrm{d}u}$$

As the goal is to construct the sensitivity matrix S, the first step is to differentiate (A.4)

$$F_x \mathrm{d}x + F_u \mathrm{d}u = 0$$

where F_x and F_u denotes the Jacobians of the super-function F. Then, by rearranging (A.4), it can be observed that S must satisfy the sensitivity equation

$$F_x S = -F_u$$

Assuming that the matrix F_x is invertible, the sensitivity equation can be solved for S

$$S = -F_x^{-1} F_u$$

and finally, an expression for the total derivative is obtained by substituting the latter expression of S into equation (A.8)

$$\frac{\mathrm{d}J}{\mathrm{d}u} = J_u + J_x S = J_u - J_x F_x^{-1} F_u \tag{A.15}$$

For ease of notation, define the following expressions

$$A^k = \frac{\partial f^k}{\partial x^k} \quad , \quad B^k = \frac{\partial f^k}{\partial x^{k-1}}$$

such that F_x can be compactly formulated in the following manner, due to its special structure

$$F_x = \begin{bmatrix} A^1 & 0 & \dots & 0 \\ B^2 & A^2 & & \\ & \ddots & \ddots & \\ 0 & & B^N & A^N \end{bmatrix}$$

Similarly, define

$$D^k = \frac{\partial f^k}{\partial u^k}$$

In order to express the jacobian F_u as the block-diagonal matrix

$$F_u = \begin{bmatrix} D^1 & 0 & \dots & 0 \\ 0 & D^2 & & \\ \vdots & & \ddots & \\ 0 & & & D^N \end{bmatrix}$$

The sensitivity matrix S can now be seen as a block matrix consisting of block components $S^{k,l}$, where

$$S^{k,l} = \frac{\partial x^k}{\partial u^l} \quad k, l = 1, 2, \dots, N$$

x^k is independent of u^l when $l > k$, thus

$$S^{k,l} = 0 \quad \forall l > k$$

such that S has a lower triangular block structure, hence

$$S = \begin{bmatrix} S^{1,1} & S^{1,2} & \dots & S^{1,N} \\ S^{2,1} & S^{2,2} & & \\ \vdots & & \ddots & \\ S^{N,1} & & & S^{N,N} \end{bmatrix} = \begin{bmatrix} S^{1,1} & 0 & \dots & 0 \\ S^{2,1} & S^{2,2} & & \\ \vdots & & \ddots & 0 \\ S^{N,1} & & & S^{N,N} \end{bmatrix}$$

When constructing S for a given index k, firstly the matrices A^k, B^k and D^k are constructed to obtain $S^{k,l}$ for all $1 \leq l \leq k$. This is obtained by the following equations

$$A^k S^{k,k} = -D^k \tag{A.16a}$$

$$B^k S^{k-1,l} + A^k S^{k,l} = 0 \;\; 1 \leq l < k \tag{A.16b}$$

The computational effort of this method is proportional $\dim(u)$.

A.4 Adjoint Sensitivity

The adjoint sensitivity, or backward sensitivity, is also based on analytical differentiation by the chain rule. The first step is to define

$$\lambda = J_x F_x^{-1} \tag{A.17}$$

then (A.17) is inserted into equation (A.15) to obtain

$$\frac{\mathrm{d}J}{\mathrm{d}u} = J_u - \lambda F_u$$

The row vector λ is commonly called the adjoint vector, and can be viewed as the solution of the *adjoint equation*

$$\lambda F_x = J_x \tag{A.18}$$

As both sides of (A.18) are row vectors, they are often transposed into more convenient column vectors, thus

$$F_x^T \lambda^T = \nabla_x J \tag{A.19}$$

And it can be observed that once F_x and $\nabla_x J$ are specified, Equation (A.19) represents a linear system on the common form $Ax = b$. However, the linear system indicates that the computational effort of (A.19) is independent of the dimension of u. This suggests that the adjoint method is very efficient when the number of controls are large compared to the number of states in the system.

Due to the block-structure of F_x, the solution of (A.18) reduces to the following recurrence. A derivation is attached in Appendix A.4.1.

$$\lambda^N A^N = \frac{\partial J}{\partial x^N} \tag{A.20a}$$

$$\lambda^i A^i + \lambda^{i+1} B^{i+1} = \frac{\partial J}{\partial x^i} \qquad i = N-1, \ldots, 1 \tag{A.20b}$$

in the recurrence above, λ^i is the adjoint vector λ that corresponds to the state-transition function at timestep i. When solving the recurrence given in (A.20), one should denote that this must be performed backwards in time, because the set of equations are specified at the boundary N. Indeed, once (A.20a) is solved for the last adjoint vector λ^N, one can successively solve (A.20b) for the remaining adjoint vectors λ^i, starting at index $i = N-1$, and setting $i = i-1$ until $i = 1$. This approach for assembling the adjoint-vectors reflects why this method often is referred to as the *backward sensitivity method*. An additional note regarding the adjoint-method, is that it requires that all state vectors x^i must be stored in a table during a forward simulation run. Then, during the "backward"-run, all adjoint-vectors are assembled from Equation A.20, and finally the gradients are calculated from Equation (A.19).

A.4.1 Derivation

Let the states be denoted by $x^i \in \mathbb{R}^{n_x}$, where x^0 denotes the initial condition. Furthermore, let the transition function at each step be defined by $x^i = f^i(x^{i-1}, u^i)$, with u as the manipulated variable. Given initial state x_0, and a set of control-vectors u^i, the final state x^N at the end of the horizon is reached by successively computing the step-transition function

$$\begin{aligned} x^1 &= f^1(x^0, u^1) \\ x^2 &= f^2(x^1, u^2) \\ &\vdots \\ x^N &= f^N(x^{N-1}, u^N) \end{aligned}$$

Let the Lagrangian[1] $\mathcal{L}$ be defined by

$$\mathcal{L} = \sum_{i=1}^{N} J_i(x^i, u^i) - \lambda_i^T (x^i - f^i(x^{i-1}, u^i)) \tag{A.21}$$

where $\lambda_i^T \in \mathbb{R}^{1 \times n_x}$ is the vector of Lagrange multipliers, in this setting referred to as Adjoints. The next step is to take the total derivative of (A.21) with respect to a control vector u^j

$$\frac{\mathrm{d}}{\mathrm{d}u^j}\mathcal{L} = \sum_{i=1}^{N} \frac{\partial J^i}{\partial x^i}\frac{\mathrm{d}x^i}{\mathrm{d}u^j} + \frac{\partial J^i}{\partial u^i}\frac{\mathrm{d}u^i}{\mathrm{d}u^j} - \lambda_i^T \left(\frac{\mathrm{d}x^i}{\mathrm{d}u^j} - \frac{\partial f^i}{\partial x^{i-1}}\frac{\mathrm{d}x^{i-1}}{\mathrm{d}u^j} - \frac{\partial f^i}{\partial u^i}\frac{\mathrm{d}u^i}{\mathrm{d}u^j} \right)$$

the term $\frac{\mathrm{d}u^i}{\mathrm{d}u^j}$ is equal to zero unless $i = j$. In that case, $\frac{\mathrm{d}u^i}{\mathrm{d}u^j} = I$, the identity matrix with proper dimension. Thus, the equation is restated

$$\frac{\mathrm{d}}{\mathrm{d}u^j}\mathcal{L} = \sum_{i=1}^{N} \left[\frac{\partial J^i}{\partial x^i}\frac{\mathrm{d}x^i}{\mathrm{d}u^j} - \lambda_i^T \left(\frac{\mathrm{d}x^i}{\mathrm{d}u^j} - \frac{\partial f^i}{\partial x^{i-1}}\frac{\mathrm{d}x^{i-1}}{\mathrm{d}u^j} \right) \right] + \frac{\partial J^j}{\partial u^j} + \lambda_j^T \frac{\partial f^j}{\partial u^j}$$

rearranging the terms

$$\frac{\mathrm{d}}{\mathrm{d}u^j}\mathcal{L} = \sum_{i=1}^{N} \left[\left(\frac{\partial J^i}{\partial x^i} - \lambda_i^T \right) \frac{\mathrm{d}x^i}{\mathrm{d}u^j} + \lambda_i^T \frac{\partial f^i}{\partial x^{i-1}}\frac{\mathrm{d}x^{i-1}}{\mathrm{d}u^j} \right] + \frac{\partial J^j}{\partial u^j} + \lambda_j^T \frac{\partial f^j}{\partial u^j}$$

and splitting up into two sums

$$\frac{\mathrm{d}}{\mathrm{d}u^j}\mathcal{L} = \sum_{i=1}^{N} \left[\left(\frac{\partial J^i}{\partial x^i} - \lambda_i^T \right) \frac{\mathrm{d}x^i}{\mathrm{d}u^j} \right] + \sum_{i=1}^{N} \left[\lambda_i^T \frac{\partial f^i}{\partial x^{i-1}}\frac{\mathrm{d}x^{i-1}}{\mathrm{d}u^j} \right] + \frac{\partial J^j}{\partial u^j} + \lambda_j^T \frac{\partial f^j}{\partial u^j}$$

[1] denote that $x^i - f^i(x^{i-1}, u^i) = 0$ by its definition, such that this term always adds exactly zero to the Lagrangian

Next, the last index from the first sum is extracted

$$\frac{\mathrm{d}}{\mathrm{d}u^j}\mathcal{L} = \sum_{i=1}^{N-1}\left[\left(\frac{\partial J^i}{\partial x^i} - \lambda_i^T\right)\frac{\mathrm{d}x^i}{\mathrm{d}u^j}\right] + \left(\frac{\partial J^N}{\partial x^N} - \lambda_N^T\right)\frac{\mathrm{d}x^N}{\mathrm{d}u^j} + \sum_{i=1}^{N}\left[\lambda_i^T\frac{\partial f^i}{\partial x^{i-1}}\frac{\mathrm{d}x^{i-1}}{\mathrm{d}u^j}\right] + \frac{\partial J^j}{\partial u^j} + \lambda_j^T\frac{\partial f^j}{\partial u^j}$$

Shifting index in the second summation

$$\frac{\mathrm{d}}{\mathrm{d}u^j}\mathcal{L} = \sum_{i=1}^{N-1}\left[\left(\frac{\partial J^i}{\partial x^i} - \lambda_i^T\right)\frac{\mathrm{d}x^i}{\mathrm{d}u^j}\right] + \left(\frac{\partial J^N}{\partial x^N} - \lambda_N^T\right)\frac{\mathrm{d}x^N}{\mathrm{d}u^j} + \sum_{i=0}^{N-1}\left[\lambda_{i+1}^T\frac{\partial f^{i+1}}{\partial x^i}\frac{\mathrm{d}x^i}{\mathrm{d}u^j}\right] + \frac{\partial J^j}{\partial u^j} + \lambda_j^T\frac{\partial f^j}{\partial u^j}$$

extracting the first index from the second sum

$$\frac{\mathrm{d}}{\mathrm{d}u^j}\mathcal{L} = \sum_{i=1}^{N-1}\left[\left(\frac{\partial J^i}{\partial x^i} - \lambda_i^T\right)\frac{\mathrm{d}x^i}{\mathrm{d}u^j}\right] + \left(\frac{\partial J^N}{\partial x^N} - \lambda_N^T\right)\frac{\mathrm{d}x^N}{\mathrm{d}u^j} + \sum_{i=1}^{N-1}\left[\lambda_{i+1}^T\frac{\partial f^{i+1}}{\partial x^i}\frac{\mathrm{d}x^i}{\mathrm{d}u^j}\right] + \lambda_1^T\frac{\partial f^1}{\partial x^0}\frac{\mathrm{d}x^0}{\mathrm{d}u^j} + \frac{\partial J^j}{\partial u^j} + \lambda_j^T\frac{\partial f^j}{\partial u^j}$$

here it can be denoted that x^0 and u^j are independent, such that the term $\lambda_1^T\frac{\partial f^1}{\partial x^0}\frac{\mathrm{d}x^0}{\mathrm{d}u^j} = 0$ for all j - and this term can safely be eliminated. Then

$$\frac{\mathrm{d}}{\mathrm{d}u^j}\mathcal{L} = \sum_{i=1}^{N-1}\left[\underbrace{\left(\frac{\partial J^i}{\partial x^i} - \lambda_i^T + \lambda_{i+1}^T\frac{\partial f^{i+1}}{\partial x^i}\right)}_{\theta_{1,i}}\frac{\mathrm{d}x^i}{\mathrm{d}u^j}\right] + \underbrace{\left(\frac{\partial J^N}{\partial x^N} - \lambda_N^T\right)}_{\theta_2}\frac{\mathrm{d}x^N}{\mathrm{d}u^j} + \frac{\partial J^j}{\partial u^j} + \lambda_j^T\frac{\partial f^j}{\partial u^j} \tag{A.22}$$

Computing the total derivative $\mathrm{d}x^i/\mathrm{d}u^j$ can indeed be troublesome. The idea with the adjoint-method is to construct all λ_i such that $\theta_{1,i}$ and θ_2 is equal to zero $\forall\ i$. This eliminates the need of computing $\mathrm{d}x^i/\mathrm{d}u^j$ in the first place. These conditions are formally stated

$$\frac{\partial J^N}{\partial x^N} - \lambda_N^T = 0 \tag{A.23a}$$

$$\frac{\partial J^i}{\partial x^i} - \lambda_i^T + \lambda_{i+1}^T\frac{\partial f^{i+1}}{\partial x^i} = 0 \tag{A.23b}$$

The boundary of (A.23) is specified at the endpoint N - thus λ_N^T is the first multiplier to be computed - using (A.23a). Then, the remaining multipliers λ_i^T are

computed backwards in time, for $i = N-1, N-2, \ldots, 2, 1$ with equation (A.23b).

Finally, the gradient are computed from the remaining terms of (A.22). This corresponds to solving the linear system

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}u^1}\mathcal{L} &= \frac{\partial J^1}{\partial u^1} + \lambda_1^T \frac{\partial f^1}{\partial u^1} \\ &\vdots \\ \frac{\mathrm{d}}{\mathrm{d}u^N}\mathcal{L} &= \frac{\partial J^N}{\partial u^N} + \lambda_N^T \frac{\partial f^N}{\partial u^N} \end{aligned}$$

which in a general form yields

$$\frac{\mathrm{d}}{\mathrm{d}u^j}\mathcal{L} = \frac{\partial J^j}{\partial u^j} + \lambda_j^T \frac{\partial f^j}{\partial u^j} \tag{A.24}$$

Comment

This approach requires the construction of the matrices A^i, B^i, J_x^i and J_u^i, which are given by

$$A^i = \frac{\partial f^{i+1}}{\partial x^i} = \begin{bmatrix} \frac{\partial x_1^{i+1}}{\partial x_1^i} & \frac{\partial x_1^{i+1}}{\partial x_2^i} & \cdots & \frac{\partial x_1^{i+1}}{\partial x_{n_x}^i} \\ \frac{\partial x_2^{i+1}}{\partial x_1^i} & \frac{\partial x_2^{i+1}}{\partial x_2^i} & & \frac{\partial x_2^{i+1}}{\partial x_{n_x}^i} \\ \vdots & & \ddots & \vdots \\ \frac{\partial x_{n_x}^{i+1}}{\partial x_1^i} & \cdots & \cdots & \frac{\partial x_{n_x}^{i+1}}{\partial x_{n_x}^i} \end{bmatrix} \in \mathbb{R}^{n_x \times n_x} \quad , \quad i = 1, 2, \ldots, N-1$$

$$B^i = \frac{\partial f^i}{\partial u^i} = \begin{bmatrix} \frac{\partial x_1^i}{\partial u_1^i} & \frac{\partial x_1^i}{\partial u_2^i} & \cdots & \frac{\partial x_1^i}{\partial u_{n_u}^i} \\ \frac{\partial x_2^i}{\partial u_1^i} & \frac{\partial x_2^i}{\partial u_2^i} & & \frac{\partial x_2^i}{\partial u_{n_u}^i} \\ \vdots & & \ddots & \vdots \\ \frac{\partial x_{n_x}^i}{\partial u_1^i} & \cdots & \cdots & \frac{\partial x_{n_x}^i}{\partial u_{n_u}^i} \end{bmatrix} \in \mathbb{R}^{n_x \times n_u} \quad , \quad i = 1, 2, \ldots, N \tag{A.25}$$

$$J_x^i = \frac{\partial J^i}{\partial x^i} = \begin{bmatrix} \frac{\partial J^i}{\partial x_1^i} & \frac{\partial J^i}{\partial x_2^i} & \cdots & \frac{\partial J^i}{\partial x_{n_x}^i} \end{bmatrix} \in \mathbb{R}^{1 \times n_x} \quad , \quad i = 1, 2, \ldots, N \tag{A.26}$$

$$J_u^i = \frac{\partial J^i}{\partial u^i} = \begin{bmatrix} \frac{\partial J^i}{\partial u_1^i} & \frac{\partial J^i}{\partial u_2^i} & \cdots & \frac{\partial J^i}{\partial u_{n_u}^i} \end{bmatrix} \in \mathbb{R}^{1 \times n_u} \quad , \quad i = 1, 2, \ldots, N$$

To conclude, by constructing the matrices in (A.25) and (A.26), preferably by utilizing automatic differentiation, one is able to compute the solution of (A.24) by first computing all λ_i from (A.23).

Appendix B

Additional Experiment Information

B.1 Dataset Tables

Table B.1: Case A and B: Reservoir and fluid properties

Variable	Value	Unit
Grid block height	20	m
Grid block width	20	m
Grid block length	20	m
Porosity	0.3	-
Permeability	0.9869	-
Initial reservoir pressure	234	bar
Water density	1080	kg/m^3
Oil density	962	kg/m^3
Oil compressibility	6.65×10^{-10}	Pa^{-1}
Water compressibility	4.28×10^{-10}	Pa^{-1}
Rock compressibility	3×10^{-10}	Pa^{-1}
Reference pressure	234	bar
Initial water saturation	0.15	-
Well-bore radius	0.15	m
Simulation time	1835	day
NPV Scaling	10^{-4}	-

Table B.2: Case C: Reservoir and fluid properties

Variable	**Value**	**Unit**
Grid dimension (w × l × h)	$60 \times 60 \times 1$	-
Grid block width	24	m
Grid block length	24	m
Grid block height	24	m
Initial reservoir pressure	170.4947	bar
Water density	1037.84	kg/m^3
Oil density	786.5	kg/m^3
Oil compressibility	-	Pa^{-1}
Water compressibility	4.5998×10^{-10}	Pa^{-1}
Rock compressibility	4.408×10^{-10}	Pa^{-1}
Reference pressure	1	atm
Initial water saturation	0.1030	-
Well-bore radius	0.0953	m
Simulation time	4380	day
NPV Scaling	10^{-7}	-

Table B.3: Case D: Reservoir and fluid properties

Variable	**Value**	**Unit**
Grid block height	4	m
Grid block length/width	8	m
Porosity	0.2	-
Oil compressibility	1×10^{-10}	Pa^{-1}
Rock compressibility	0	Pa^{-1}
Water compressibility	1×10^{-10}	Pa^{-1}
Oil dynamic viscosity	5×10^{-3}	Pa s
Water dynamic viscosity	1×10^{-3}	Pa s
Water saturation	0.1	-
Initial reservoir pressure	400	bar
Initial water saturation	0.1	-
Well-bore radius	0.1	m
Simulation time	3600	day
NPV Scaling	10^{-5}	-

B.2 Simulation Periods

Case A & B

The horizon is divided into 40 simulation steps, accounting for 1320 days, with four control periods ($n_u = 4$). The simulation steps and control periods are organized in the following manner

Time step k	$(1:2)+\alpha$	$(3:5)+\alpha$	$(6:10)+\alpha$
Step duration (days)	10	20	50
Total duration	20	60	250
Control period j	β	β	β

where α and β is such that

$$\alpha \in \{0, 10, 20, 30\}$$
$$\beta \in \{1, 2, 3, 4\}$$

and the pairing

$$(\alpha, \beta) \in \{\,(0,1)\,,\,(10,2)\,,\,(20,3)\,,\,(30,4)\,\}$$

Case C

The reservoir is simulated for 12 years, divided into 156 simulation periods and 24 control periods. The first 12 control periods are organized such that they each cover 6 simulation steps

Time step k	$1+\alpha$	$2+\alpha$	$3+\alpha$	$4+\alpha$	$5+\alpha$	$6+\alpha$
Step duration (days)	4	5	10	18	36	73
Control period j	β	β	β	β	β	β

where

$$\alpha \in \{0, 6, 12, \ldots, 66\}$$
$$\beta \in \{1, 2, 3, \ldots, 12\}$$

such that the pair

$$(\alpha, \beta) \in \{\,(0,1)\,,\,(6,2)\,,\,(12,3)\,,\,\ldots\,,\,(66,12)\,\}$$

Each of the first 12 control period cover $4+5+10+18+36+73 = 146$ days, which sums up to $146 \times 12 = 1752$ days for all 12 profiles. In this way, the beginning of each control period is simulation with higher resolution.

Similarly, the remaining 12 control periods each cover seven simulation steps

Time step k	$1+\gamma$	$2+\gamma$	$3+\gamma$	$4+\gamma$	$5+\gamma$	$6+\gamma$	$7+\gamma$
Step duration (days)	4	5	10	18	36	73	73
Control period j	κ	κ	κ	κ	κ	κ	κ

where

$$\gamma \in \{72, 79, 86, \ldots, 149\}$$
$$\kappa \in \{13, 14, 15, \ldots, 24\}$$

such that the pair

$$(\gamma, \kappa) \in \{\,(72, 13)\,,\,(79, 14)\,,\,(86, 15)\,,\,\ldots\,,\,(149, 24)\,\}$$

Each of the latter 12 control period cover $4+5+10+18+36+73+73 = 219$ days, which adds up to 2628 days for all twelve periods. All 24 control profiles accounts for $1752 + 2628 = 4380$ days in total, i.e. 12 years ($4380/365 = 12$).

Case D

The horizon is divided into 150 simulation periods with 10 control periods. Each control period cover 15 simulation steps, in the following manner

Time step k	$1+\alpha$	$2+\alpha$	$3+\alpha$	$(4:15)+\alpha$
Step duration (days)	1	4	10	30
Total duration	1	4	10	330
Control period j	β	β	β	β

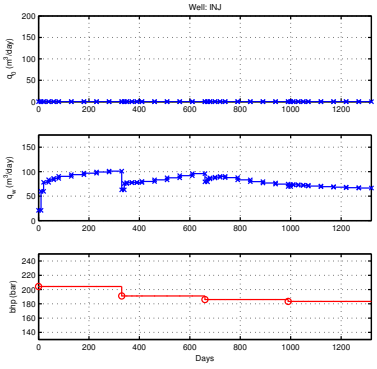
where

$$\alpha \in \{0, 15, 30, \ldots, 135\}$$
$$\beta \in \{1, 2, 3, \ldots, 15\}$$

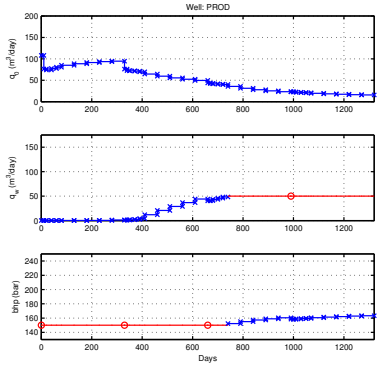
such that the pair

$$(\alpha, \beta) \in \{\,(0, 1)\,,\,(15, 2)\,,\ldots,\,(135, 15)\}$$

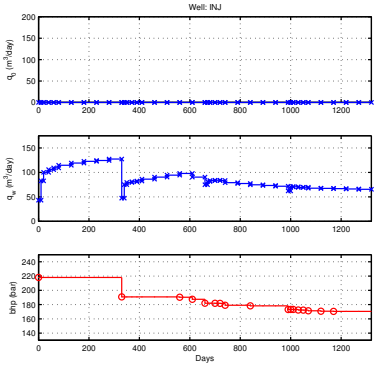
B.3 Figures



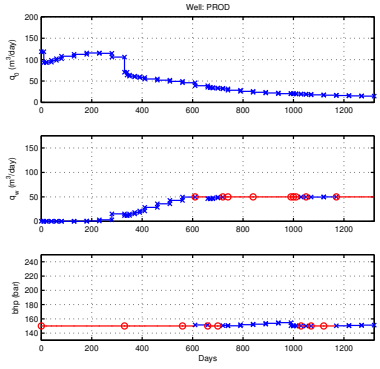
(a) RC2



(b) RC2



(c) REF2



(d) REF2

Figure B.1: Case B: Additional well trajectories

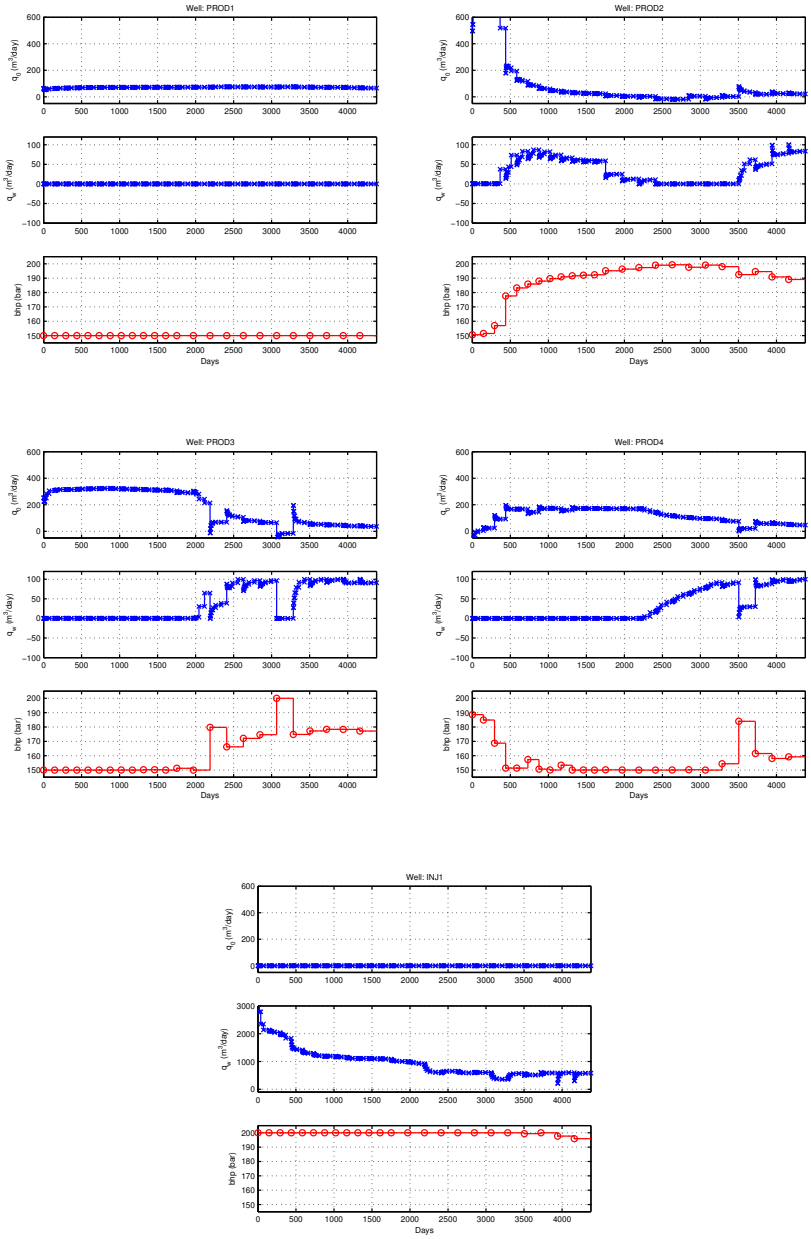


Figure B.2: Case C: REMSO

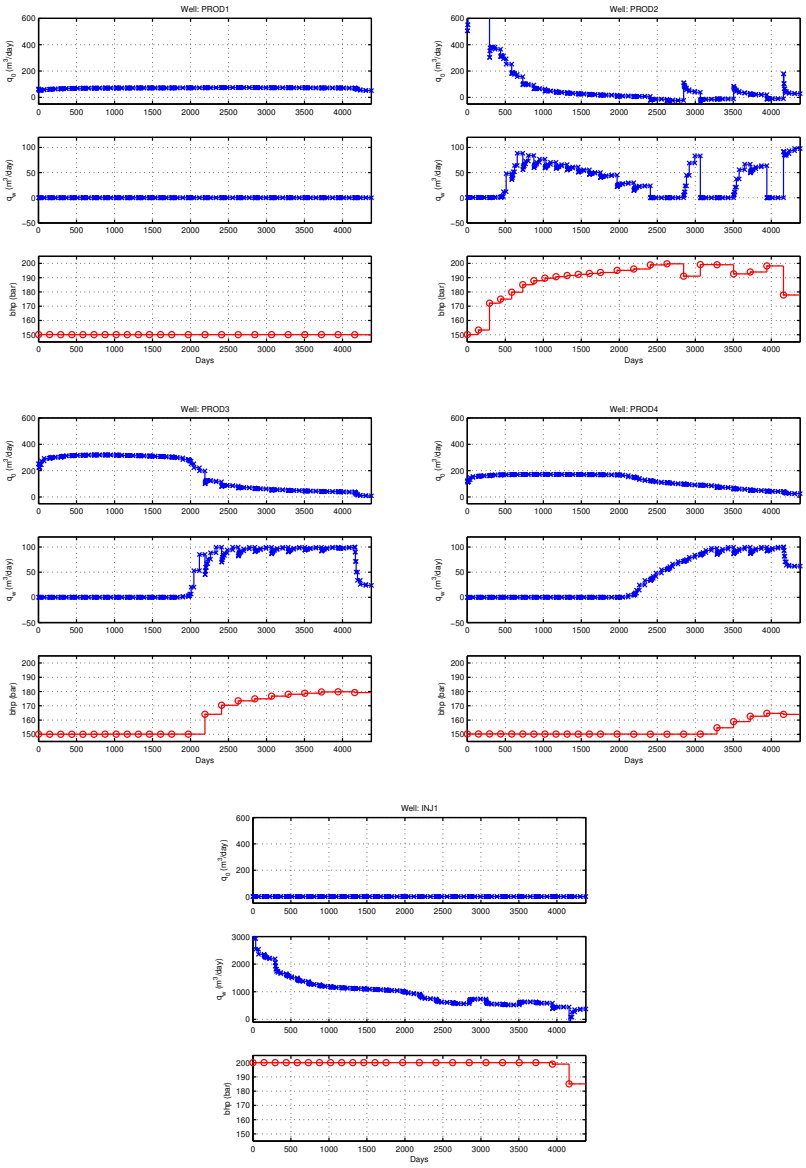


Figure B.3: Case C: IPOPT

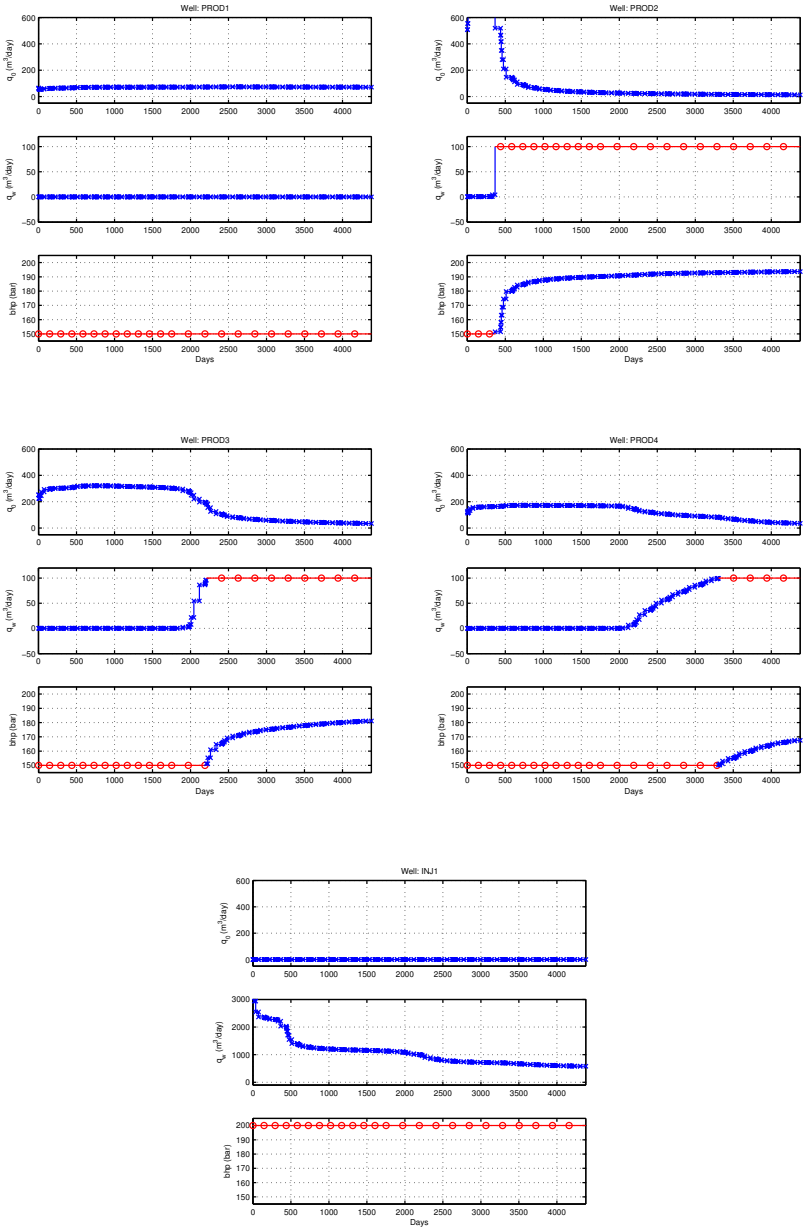


Figure B.4: Case C: RC1

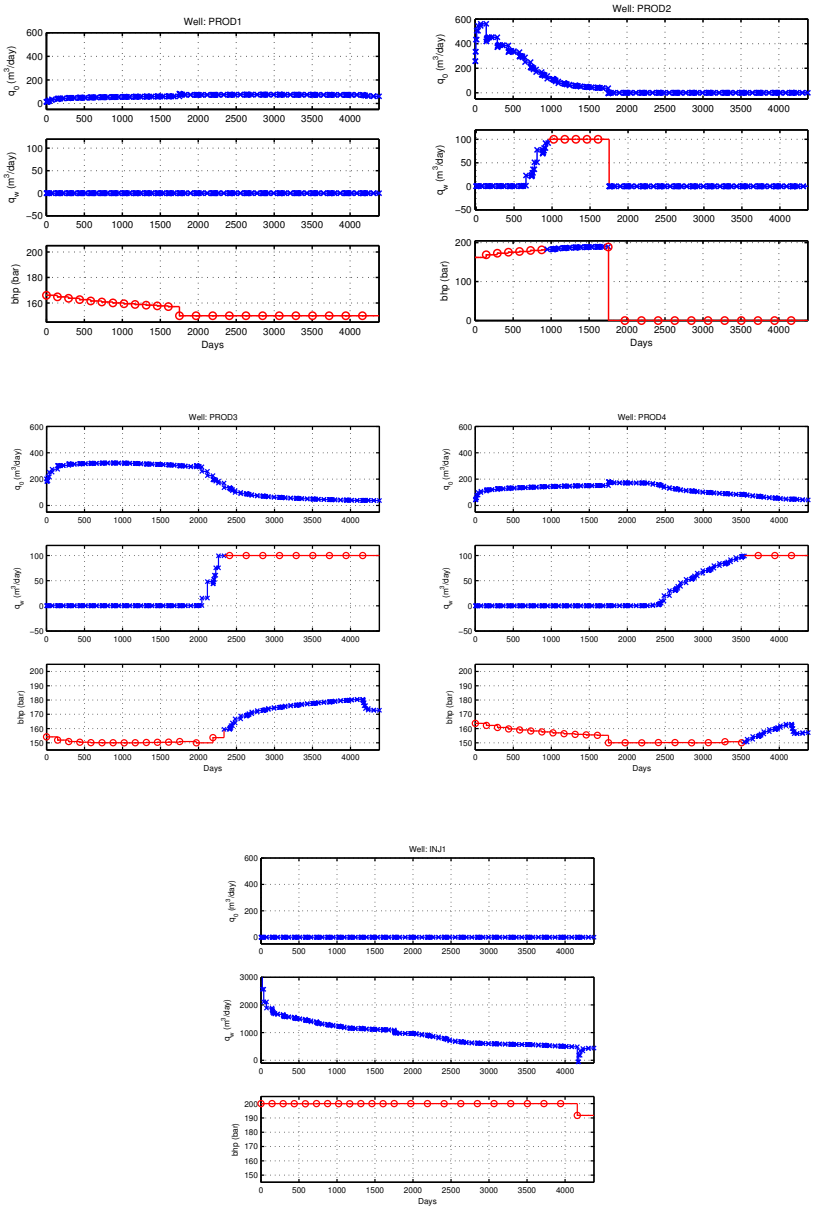


Figure B.5: Case C: RC2

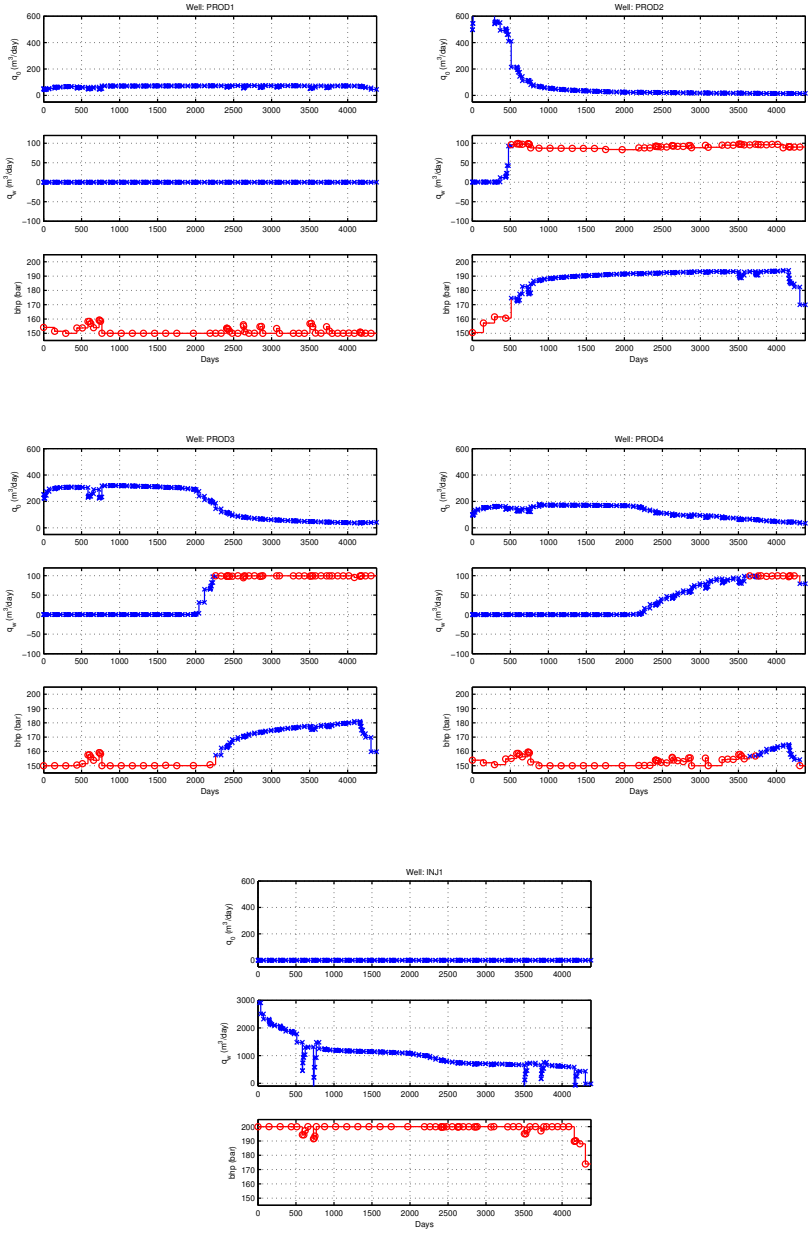


Figure B.6: Case C: REF1

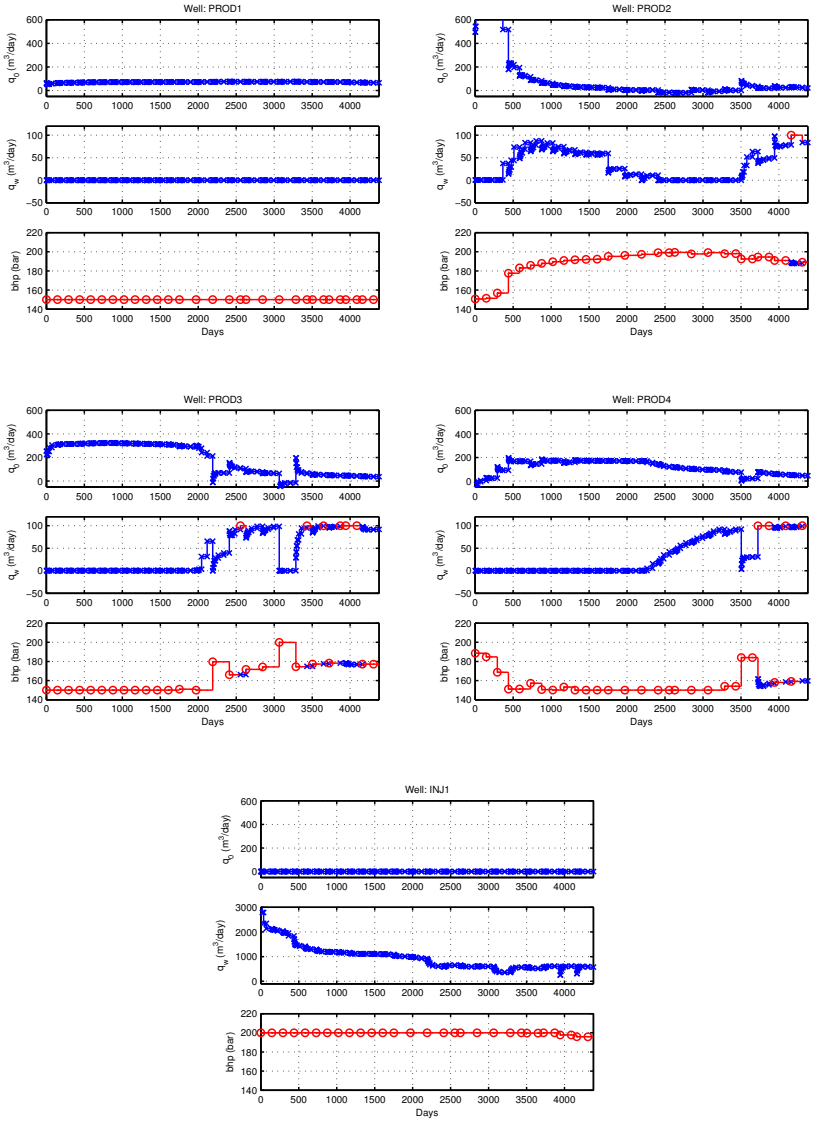
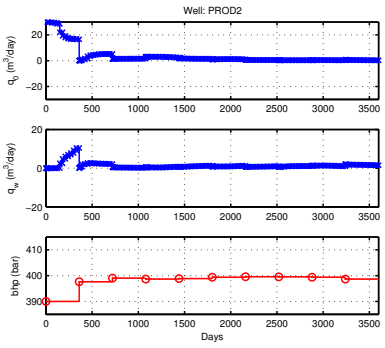
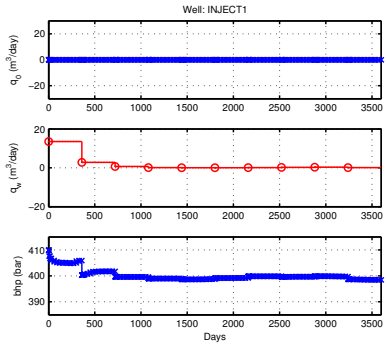


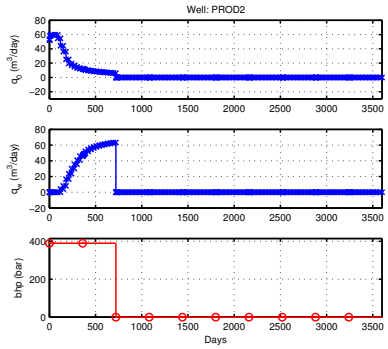
Figure B.7: Case C: REF2



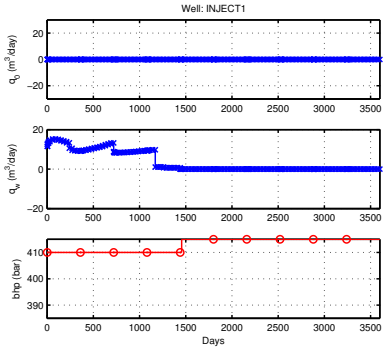
(a) IPOPT



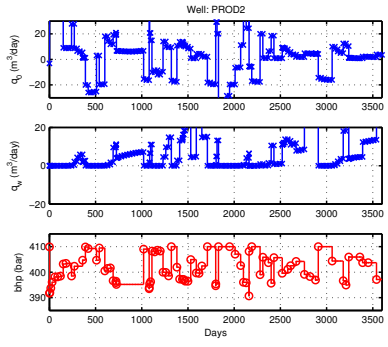
(b) IPOPT



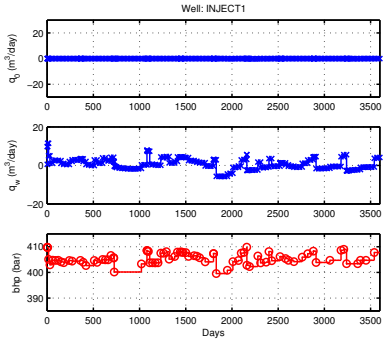
(c) RC1



(d) RC1



(e) REF2



(f) REF2

Figure B.8: Case D: Well trajectories for IPOPT, RC2 and REF2