Gradient-based Methods for Production Optimization of Oil Reservoirs

Eka Suwartadi

Doctoral Thesis

Submitted for the Partial Fulfillment of the Requirements for the Degree of

philosophiae doctor

Department of Engineering Cybernetics
Faculty of Information Technology, Mathematics and Electrical Engineering
Norwegian University of Science and Technology

March 29, 2012

NTNU
Norwegian University of Science and Technology

Thesis for the degree of philosophiae doctor

Faculty of Information Technology, Mathematics
and Electrical Engineering
Department of Engineering Cybernetics

ISBN 978-82-471-3485-6 (printed version)
ISBN 978-82-471-3486-3 (electronic version)
ISSN 1503-8181

ITK Report 2012-3-W

Doctoral Theses at NTNU, 2012:104

Printed by NTNU-Trykk

To my wife and my parents

Summary

Production optimization for water flooding in the secondary phase of oil recovery is the main topic in this thesis. The emphasis has been on numerical optimization algorithms, tested on case examples using simple hypothetical oil reservoirs. Gradient-based optimization, which utilizes adjoint-based gradient computation, is used to solve the optimization problems.

The first contribution of this thesis is to address output constraint problems. These kinds of constraints are natural in production optimization. Limiting total water production and water cut at producer wells are examples of such constraints. To maintain the feasibility of an optimization solution, a Lagrangian barrier method is proposed to handle the output constraints. This method incorporates the output constraints into the objective function, thus avoiding additional computations for the constraints gradient (Jacobian) which may be detrimental to the efficiency of the adjoint method.

The second contribution is the study of the use of second-order adjoint-gradient information for production optimization. In order to speedup convergence rate in the optimization, one usually uses quasi-Newton approaches such as BFGS and SR1 methods. These methods compute an approximation of the inverse of the Hessian matrix given the first-order gradient from the adjoint method. The methods may not give significant speedup if the Hessian is ill-conditioned. We have developed and implemented the Hessian matrix computation using the adjoint method. Due to high computational cost of the Newton method itself, we instead compute the Hessian-times-vector product which is used in a conjugate gradient algorithm.

Finally, the last contribution of this thesis is on surrogate optimization for water flooding in the presence of the output constraints. Two kinds of model order reduction techniques are applied to build surrogate models. These are proper orthogonal decomposition (POD) and the discrete empirical interpolation method (DEIM). Optimization using a trust-region framework (TRPOD) is then performed on the surrogate models. Furthermore, the output constraints are again handled by the Lagrangian barrier method.

Summary

Preface

The PhD research project presented in this thesis is fully supported by Center for Integrated Operations in Petroleum Industry (IO Center) at NTNU under Program 2 - Reservoir management and production optimization.

Acknowledgment

This thesis has been a wonderful journey. I always wanted to know on what extent systems and control can be applied to industries and real life applications. I was working for IT industry in oil and gas companies back then in 2006 and was wondering how control systems for upstream industry. I was fortunate to meet and have interview with Bjarne Foss, who gave me opportunity to do research in oil and gas areas. Without him, this thesis would be non existent. His guidance, patience, and importance of being critical left a trace in my mind. More importantly, he has introduced me to the beautiful universe of numerical optimization.

The second person whom I would like to thank is Stein Krogstad. Being a second supervisor, Stein has helped me on the technical details. The adjoint codes would not be possible without his supervision. His comments on the drafts of papers are always critical and constructive. My visits to Stein's office at SINTEF ICT in Oslo have been fruitful.

I would like to thank to John Bagterp Jørgensen, Ana Isabel Pereira, and Jan Tommy Gravdahl for their willingness to be on my committee and their valuable comments.

I am grateful for warm interaction at the Department of Engineering Cybernetics. Those at the administrative staff for their help and fellow PhD students for inspiring Wednesday meetings and other social activities.

Not forget to mention, Indonesian community in Trondheim. For the gathering and *makan-makan*, relieving my longings for my homeland.

Finally, thanks to my family in Indonesia for their unconditional support. My wife who always encourages me to achieve the "finish line" of this journey as soon as pos-

sible.

Trondheim, March 2012 *Eka Suwartadi*

Glossary

List of abbreviations

The following abbreviations are used in this thesis:

AD	Automatic Differentiation
BFGS	Broyden-Fletcher-Goldfarb-Shanno
BHP	Bottom-hole pressure
CG	Conjugate Gradient
CLRM	Closed-loop Reservoir Management
CPU	Central Processing Unit
DEIM	Discrete Empirical Interpolation Method
EnKF	Ensemble Kalman Filter
EnOpt	Ensemble Optimization
EOR	Enhanced Oil Recovery
EQP	Equality Quadratic Programming
IEnKF	Iterative Ensemble Kalman Filter
IMPES	Implicit Pressure Explicit Saturation
IQP	Inequality Quadratic Programming
KKT	Karush-Kuhn-Tucker
LP	Linear Programming
mD	milli Darcy
MOR	Model Order Reduction
MPC	Model Predictive Control
NPV	Net Present Value
PCG	Projected Conjugate Gradient
PDE	Partial Differential Equation
POD	Proper Orthogonal Decomposition
PVI	Pore Volume Index
SLQP	Sequential-Linear Quadratic Programming
SQP	Sequential Quadratic Programming

SR1	Symmetric Rank 1
SVD	Singular Value Decomposition
TN	Truncated Newton
TRPOD	Trust-region Proper Orthogonal Decompsition

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

The need for fossil fuel energy has always been increasing due to the rise of human population and the modernization of civilization. Oil reservoirs which were easy to find and drain are now few and far between. Oil fields were initially discovered mainly as onshore reservoirs but offshore areas are becoming more dominating, even moving towards the Arctic region. Given the current situation, a small improvement in enhanced oil recovery (EOR) techniques will have enormous impact on oil production and hence the value of assets. Therefore EOR is a very active research field in the reservoir engineering community.

Processes involved in an oil reservoir production can be classified into upstream and downstream parts as shown in Figure 1.1. The upstream processes deal with subsurface flows, including the subsurface reservoir itself while the downstream part has to do with pipelines, separators, and export facilities. Timescales in the oil reservoir extraction processes vary from the upstream to the downstream part. In the upstream, fluid movement can be very slow due to geological properties and the large-scale nature of oil reservoirs. Fluid movement in a reservoir may be in the order of months or years, e.g., for moving fluid from injector wells to producer wells. The dimension of oil reservoirs, as seen in Figure 1.1, can be in the order of kilometers. In contrary to the upstream part, timescales are much faster in the downstream processes; typically hours or even into minutes.

The focus in this thesis will be on the upstream part of the oil reservoir process. We will apply techniques from systems and control to increase oil recovery. The upstream technologies include modeling of subsurface reservoirs, which is known as reservoir simulation. The simulation uses mathematical models governed by partial differential equations (PDEs) to help reservoir engineers learn the dynamics and support important economic decisions. The decisions are typically made on a long term horizon,

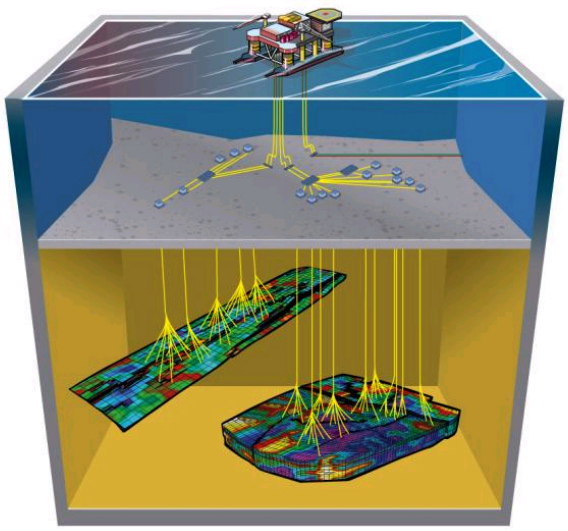

Figure 1.1: Oil reservoir from subsurface to surface facilities. Courtesy of Nexus - Halliburton.

like a five year period or more. As such, the reservoir simulations are normally run offline. Determining fluid injection rates and production targets are examples of decisions supported by reservoir simulators. There also exists lower level control, on a faster timescale, which will execute the actual production decisions, see Foss [2011] for a detailed discussion of process control methods in the upstream petroleum industries. The research topic of systems and control in oil reservoirs has been coined by the term *closed-loop reservoir management*. This expression is further explained in the subsequent section.

1.2 Closed-loop Reservoir Management

The term closed-loop reservoir management (CLRM) appears in Jansen et al. [2005] and Jansen et al. [2009a]. It is also known as real-time reservoir management in Saputelli et al. [2006] or a self-learning reservoir in Saputelli et al. [2005]. The principle of CLRM is to use control and optimization concepts from the fields of process control, and data assimilation from oceanography and meteorology. We depict CLRM in Figure 1.2, consisting of an optimizer, plant (oil reservoir), and state/parameter estimator or observer. Controlling an oil reservoir is a complex time-dependent process because it cannot be seen physically, as it is lying deep down in Mother Earth. Measurement de-

vices ranging from sensors in wells to seismic data acquisition systems are used to infer reservoir conditions. But, still one is not able to describe the reservoir accurately. Large uncertainties related to fluid and rock properties are unavoidable. In addition, measurement devices also introduce noise. The measurement data are used to close the loop (as feedback) since the observer is connected to the optimizer as it uses updated reservoir models. However, reservoir models do still have challenges to fully represent the physics of the reservoirs themselves.

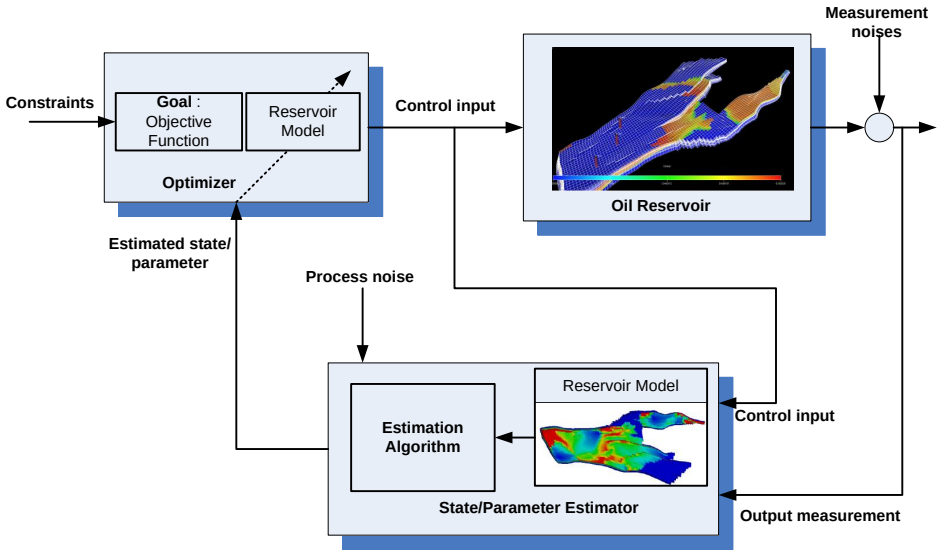

Figure 1.2: Closed-loop control system treating oil reservoir as a plant.

The closed-loop reservoir management approach provides a handful of computational methods. To deal with uncertainties, instead of using one best-effort model and by applying a data assimilation technique, the reservoir model may be realized as a set of ensemble models to include statistical uncertainties. For the optimizer, there exists a wide variety of optimization methods. The most efficient method to deal with large-scale systems, as oil reservoirs, is to use gradient-based optimization provided gradients are available at a reasonable cost. Optimal control theory offers an efficient method to compute gradients since only two simulation runs are necessary regardless of the number of optimization variables. This is the *adjoint method*. Furthermore, due to the large dimension of reservoir models, there exists a scaling technique to reduce the dimension to a reasonable number which is still good enough to preserve physical properties of the reservoir model. This technique is called *model order reduc-*

tion which, once implemented, may give substantial speedup of reservoir simulation runtime. One goal of CLRM is to enable reservoir model update more frequently, as opposed to treat it as a one-time batch process. Ultimately, the use of the CLRM will increase the value of an asset. This has been shown in a case example in Jansen et al. [2009a]. On the other hand, there are limitations of the CLRM strategy as discussed in Foss & Jensen [2011].

From a reservoir engineering point of view the terms optimizer and observer are not customary. Instead of observer, reservoir engineers often use the term *history matching* or reservoir conditioning. Optimizer is commonly referred to as *production optimization*. These two problems, production optimization and history matching, are the main tasks of a reservoir engineer. Well placement, which is also part of the optimizer box, is another important decision problem worked on by reservoir engineers. A further description of each problem will be explained in the next subsections.

It should be noted that CLRM does not mean automated control or automated decision making. There will for the most part be humans in the loop meaning that CLRM systems support decisions which are ultimately made by humans.

1.2.1 Production Optimization

Production optimization is an optimal control strategy to determine injector and producer settings in order to maximize some economical measure of a reservoir. In control terms, production optimization provides a means to decide the optimal control input. The control input for oil reservoirs can be well rates, bottom-hole pressures (BHP), and valve/choke settings. A method that has been commonly used in practise is a reactive control strategy. This method is somewhat similar to '*on-off*' control since it closes a well when water or gas content in a producer well reaches a threshold which makes the producer well uneconomical. An optimal control approach applies a predictive strategy and in this sense, optimal control approach is regarded as proactive control strategy rather than a reactive method.

The production optimization problem can be cast as an optimization problem. The objective function in production optimization is an economic measure, such as net present value (NPV), recovery factor, or sweep efficiency. The number of decision variables, that is, the dimension of control input depends on the length of the production horizon, the number of wells and the parametrization of the well settings. In this work we apply piecewise constant well settings. The number may be fairly large since a typical problem may involve a 5-10 year horizon, more than 10 wells and changes in the well setting every 3 months. As a basic assumption we apply gradient-based optimization methods when applicable, i.e., when gradients are available at a reasonable cost. Hence, the use of derivative-free methods [Conn et al., 2009], which are of interest to reservoir optimization, will not be included in this work. The reservoir optimization

problem is a nonlinear program. Hence, the methods applied herein cannot guarantee to find a global optimum, the best we can hope for is a (good) local optimum. Global optimization [Horst et al., 2000] is in theory an option, but due to large computational costs, it is not viewed as a viable option.

From an optimizer point of view, an oil reservoir model is seen as an implicit constraint in the optimization problem. The derivative-free methods are regarded as black-box optimization approaches since they do not need any details inside a simulator. Such methods just provide an input and use the simulator result based on the given input. This is an iterative algorithm which will terminate after a stopping criterion is satisfied. Gradient methods can apply a finite-difference method to compute the gradients. The finite-difference method perturbs the simulator calculation about a nominal value. However, the method is costly and vulnerable to numerical noise. Computing gradients using the adjoint method usually requires knowledge of the internal of a simulator since the adjoint method seldom is implemented in available simulators. The implementation of the adjoint method is laborious. Applying this line of thought, the derivative-free methods will often be preferable because of their ease of implementation. A recent and quite efficient method is to use ensemble-based gradient computation Su & Oliver [2010].

Once the gradients are available, one needs to decide what kind optimization method to be used. Common approaches in the production optimization literature are steepest descent, conjugate gradient, and quasi-Newton methods. These methods use first-order gradients and quasi-Newton methods to approximate second-order information. In terms of convergence rate, that is, how fast the optimization algorithms converge to a (local) solution, quasi-Newton is faster than the first-order gradient (steepest descent/ascent) and the Newton method is the fastest. Computing the Hessian using an adjoint method with reservoir model dynamics is not yet available in the literature.

In the optimization literature, see e.g, Biegler [2010] and Nocedal & Wright [2006], there exist two types of strategies: line-search and trust-region. So far, gradient-based production optimization has mostly used the line-search strategy. The trust-region method is not that much explored yet in the reservoir simulation research community.

Well-rates, bottom-hole pressures, and valve-settings all have limitation in their operational values. Consequently, there are bound constraints on the control inputs. Further, a possible incompressible fluid assumption introduces an equality constraint to the control input, that is, the fact that total injection rate must equal total producer rate. Other constraints are nonlinear and hence more complex. One example of such constraints is limits on water production rates at producer wells. Another example is conflicting objective functions for short-term and long-term horizons [van Essen et al., 2010]. Typically, an optimization package has an option to input the nonlinear constraint, but one has to supply gradients of the nonlinear constraint (Jacobian). In the

case of adjoints this requires additional adjoint equations. In other words, the presence of nonlinear constraints may be detrimental to the adjoint method. Some work have addressed this concern such as Kraaijevanger et al. [2007], Sarma et al. [2008], and recently by Chen [2011].

One (forward) simulation for the whole life-cycle of a reservoir can be expensive in term of computational time. Moreover, during the course of one optimization run many simulation runs are needed. The idea to use model order reduction technique is really beneficial. Markovinovic [2009] and Cardoso [2009] embed reduced-order models into the iterative algorithm of production optimization. The reduced-order models in that work are built on proper orthogonal decomposition (POD). The speedup factor gained is at least in the order of two.

In addition to the control inputs mentioned above, well location is an important decision variable. This leads to the well placement problem, which also can be cast as an optimization case. Compared to production optimization, which has long-term and short-term horizons, the well placement optimization is normally posed on a long-term horizon. Zandvliet et al. [2008a] uses the adjoint method combined with a pseudowell concept and Zhang et al. [2010], Forouzanfar et al. [2010] regard the well-placement problem as a nonlinear constraints in an adjoint-gradient based optimization algorithm. In addition to the well placement problem, control of gas coning is a special case of the production optimization problem. This problem may occur on short and long-term horizons; from days to years. Work along these lines include Leemhuis et al. [2007], Leemhuis et al. [2008], and Hasan et al. [2011].

In practice the implementation of advanced production optimization techniques described above is eased by the use of smart-well technology, such as inflow control device (ICD) [Brouwer, 2004].

1.2.2 History Matching

History matching is a work process to adjust the parameter values of a reservoir model based on available data such as production data, well logs and/or seismic data. This is known as the parameter estimation problem. The parameters typically include fluid and rock properties like fault location, permeability, porosity, and net-to-gross ratio. Due to the large number of parameters and limited measurement data, this parameter estimation problem is an ill-posed problem in the sense of Hadamard [Hadamard, 1902]. Thus, it will not give a unique solution. After a model has been fitted towards data it can be used for predictive analysis provided its predictive capabilities are reasonable. Reservoir problems are quite similar to weather forecasting. Both oil reservoir and weather models are large-scale systems derived from PDEs. But the timescale of weather forecasting is much faster than that of oil reservoir models. History matchings for reservoir models are therefore usually performed every 18 or 24 months on a cam-

paign basis while the weather forecasting model updating can be as often as 6 hours. This explains why weather forecasting needs a supercomputer to predict the weather.

In general there are two kinds of approaches in dealing with the history matching problem. The first approach is to use optimization techniques, whose objective function is the discrepancy between observation data and computed outputs from a reservoir model. The adjoint method has been used extensively to solve this problem since the 1990's, see e.g., Wu [1999]. The number of optimization variables in such cases is much larger than in the production optimization problem. The dimension of decision variable is proportional to the number of gridblock in a reservoir, which can be of dimension $10^4 - 10^6$. Reduced-order techniques, or a more popular term in history matching known as *parameterization*, has also been applied. Work of Sarma et al. [2007] for an example use a kernel principle component analysis (KPCA) to reduce the number of parameters to be optimized.

The second approach is Kalman-filtering techniques. This method has been actively used in weather forecasting. Initiated by Naevdal et al. [2002], its application to reservoir parameter estimation nowadays has gained a lot of attention [Aanonsen et al., 2009] in particular the Ensemble Kalman Filter (EnKF) [Evensen, 2009] because of its ability to deal with a variety of uncertainties. A mix of EnKF and optimization methods yields an iterative EnKF (IEnKF). The work of Gu & Oliver [2007] and Li & Reynolds [2009] have shown that IEnKF handles strong nonlinearities favorably. A detailed review of recent progress in history matching can be found in Oliver & Chen [2010].

Production optimization and history matching are coupled problems. In a SPE benchmark problem - Brugge Case [Peters et al., 2010], both problems are jointly presented as an exercise to develop a CLRM strategy. A 'true' reservoir model was available to the SPE benchmark organizers only. Interested participants with the benchmark case were given 104 realizations of parameters in the truth reservoir model. For this case the participants using EnKF combined with ensemble-based production optimization (EnOpt) techniques got the best results. The result obtained from the best participant had only 3% lower NPV than that of the true model.

Apart from production optimization and history matching, control analysis of oil reservoir models is also an active research field. The work of Zandvliet et al. [2008b] analyzed controllability, observability, and identifiability of a single-phase fluid system. A follow up work on controllability of a two-phase fluid system can be found in Jansen et al. [2009b] and further detailed analysis is available in van Doren J. F. M. [2010]. These analyses are important and they highlight the fact that the dynamics of reservoirs live in a low-dimensional space as compared to the dimensional of a typical high fidelity reservoir model.

1.3 Research Objectives and The Method of Attack

This thesis has an eye to the production optimization problem especially in dealing with nonlinear output constraints. Several references are given above and much research have been done. Our hypothesis, however, is that there is a significant potential for improved optimization algorithms for reservoir production optimization. This includes both improvements of existing methods as well as applying methods from other parts of the systems and optimization literature. In this work we will to a large extent use existing state of the art optimization packages as part of the algorithms which have been proposed, implemented and tested. In this view, we will utilize available optimization packages/toolboxes to solve production optimization problem.

The use of the adjoint method, despite its efficiency, still has some challenges in reservoir simulation studies. The aim of this work is to preserve the efficiency of the method in the presence of nonlinear output constraints. We will also investigate the use of model-order reduction (MOR) techniques for production optimization problems since the development of MOR algorithms has been vastly progressing during recent years. This gives opportunities for identifying MOR methods that might be suitable for oil reservoir models. To our knowledge the use of MOR methods in production optimization were started with Markovinovic [2009], van Doren et al. [2006], Heijn et al. [2004] and Cardoso [2009].

The development of the adjoint method in other fields is also significant. In the reservoir simulation literature, the adjoint method is used to compute first-order information. An adjoint method for second-order gradient is already available, see e.g. Ito & Kunisch [2008]. We will investigate the use of second-order gradient information for production optimization problems.

Since this thesis focuses on methodology or optimization algorithm development, we have used simple synthetic reservoir models. We do not use real oil reservoir simulators due to proprietary reservoir simulator code, which is hard, if not impossible to analyze. Moreover, we do not investigate gridding techniques for reservoir models. We assume the reservoir models have been properly developed and ready to use in a production optimization problem setting. In all examples the aim has been to evaluate the proposed algorithms in a fair manner through the design of the examples and test scenarios. It should be noted that we use a simplified version of the IO center supported Norne model in the case study in this work.[1]

[1]The model is publicly available at Norne's webpage `http://www.ipt.ntnu.no/~norne` or at the Program 2 - IO Center's webpage `http://www.iocenter.no/doku.php?id=research:program2`

1.4 Contribution

The main contributions in this thesis are:

Nonlinear output constraints

This type of constraints has many appearances in production optimization problems. Based on feasibility of initial controls, we propose to use a Lagrangian-barrier method to handle one-dimensional and multidimensional output constraint problems. We demonstrate the method through case examples. One-dimensional cases include an upper bound on the volume of water production and hierarchial short-time and long-term optimization while an example of a multidimensional problem is constraining water cut at all producer wells. The Lagrangian barrier method is compared to other appropriate optimization methods.

Second-order adjoint-gradient

We have developed second-order adjoint-gradient code, and due to expensive computational cost of the Newton method, we supply instead the Hessian-times-vector product to a conjugate gradient optimizer. The convergence rate of the second-order gradients against a quasi-Newton approach is studied on reasonable examples. Further, optimizations are run with a variety of initial conditions since gradient-based optimization is sensitive to initial guesses. This also applies to the nonlinear output constraint problem above.

Surrogate optimization

We apply a recent model order reduction technique, that is, the discrete empirical interpolation method (DEIM) and use the resulting reduced-order model in a surrogate optimization framework. To ensure the quality of the reduced-order model, we apply a trust-region strategy to check the objective function value towards a high-fidelity model. Furthermore, we also introduce output constraints in the surrogate optimization.

1.5 Thesis Organization

The remainder of the thesis is organized as follows:

- *Chapter 2* - This chapter presents background material for subsequent chapters. The chapter will give an introduction to the reservoir model that has been used during the research period, gradient-based optimization theory, brief description of the adjoint method and its validation, and finally basic numerical linear algebra used in this thesis.

- *Chapter 3* - We explain the Lagrangian-barrier and other similar methods in handling the nonlinear output constraints. Three case examples are presented to demonstrate the performance of the proposed algorithm.

- *Chapter 4* - In this chapter, we present the development of the second-order adjoint-method and further use the gradient for production optimization problem. We use both well-rates and BHP as control inputs in order to test performance of the algorithm in two numerical case examples.

- *Chapter 5* - This chapter builds on Chapter 3 and develops a nonlinear output constraint handling method with reduced-order models. We study and evaluate the DEIM and trust-region framework using three simple reservoir examples.

- *Chapter 6* - Concluding remarks and recommendation for further work are given in this chapter.

1.6 List of publications

The following publications list are the result of this thesis research work.

Journal papers

- [Suwartadi et al., 2012b] - E. Suwartadi, S. Krogstad, and B. Foss. Nonlinear Output Constraints Handling for Production Optimization of Oil Reservoirs. Published in *Computational Geosciences*, Volume 16, Issue 2 (2012), page 499-517.

- [Suwartadi et al., 2012a] - E. Suwartadi, S. Krogstad, and B. Foss. Adjoint-based Surrogate Optimization of Oil Reservoir Water Flooding. Submitted to *Journal of Computational Physics*.

Conference papers

- [Suwartadi et al., 2009]. - E. Suwartadi, S. Krogstad, and B. Foss. On Adjoint State Constraints of Adjoint Optimization in Oil Reservoir Water-flooding. In *Proceeding of SPE Reservoir Simulation and Characteristic Conference, Abu Dhabi*, October 2009.

- [Suwartadi et al., 2010b] - E. Suwartadi, S. Krogstad, and B. Foss. Nonlinear Output Constraints Handling for Production Optimization of Oil Reservoirs. In *Proceeding of European Conference on Mathematical Oil Recovery 2010 (ECMOR XII), Oxford, UK*, September 2010.

- [Suwartadi et al., 2010a] - E. Suwartadi, S. Krogstad, and B. Foss. A Lagrangian-Barrier Function for Adjoint State Constraints Optimization of Oil Reservoir Water Flooding. In *Proceeding of IEEE Conference on Decision and Control 2010, Atlanta, Georgia, USA*, December 2010

- [Suwartadi et al., 2010c] - E. Suwartadi, S. Krogstad, and B. Foss. Second-Order Adjoint-Based Control for Multiphase Flow in Subsurface Oil Reservoirs. In *Proceeding of IEEE Conference on Decision and Control 2010, Atlanta, Georgia, USA*, December 2010

Other papers

During the PhD project, I was involved in developing a unique real-field benchmark case and the result was presented in the following paper.

- [Rwechungura et al., 2010] - R. Rwechungura, E. Suwartadi, M. Dadashpour, J. Kleppe, and B. Foss. The Norne Field Case -A Unique Comparative Case Study. SPE 127538. In *Proceeding of SPE Intelligent Energy Conference and Exhibition, Utrecht, The Netherlands*, March 2010.

Unpublished work

Due to time constraints some research work have not been published yet. Here is the list of the work.

- *Quantitative measure of controllability and identifiability for oil reservoir models.* This topic is inspired by the work of W. Kang in Kang & Xu [2009] and Kang &

Xu [2010] who use dynamic optimization to quantitatively measure observability and controllability for systems governed by partial differential equations. This type of problem is considered as an example of nonlinear output constraint handling. Preliminary results of this work have been presented at ECMOR XII along with the paper Suwartadi et al. [2010b].

- *A trust-region Iterative Ensemble Kalman Filter*. This study tries to use second-order adjoint-gradient information for the history matching problem. As mentioned in Subsection 1.2.2, IEnKF is an optimization version of EnKF which deals with strong nonlinearities in reservoir models. The IEnKF consumes more CPU runtime than the EnKF, but it results in better estimation of parameters and good agreement with measurement data.

Chapter 2

Preliminaries

This chapter provides preliminary materials for the next chapters. We describe the oil reservoir simulator used in this work and formulate the production optimization problem in mathematical programming language. Gradient-based optimization and how to obtain the gradient are also explained in this chapter.

2.1 The reservoir model

Oil reservoirs in porous media constitute a mixture between hydrocarbons and water. Other components like CO_2, H_2S and sulphur may also appear, but they are neglected in this work. The hydrocarbon part typically contains many components but a common simplification is to assume a three phase (three component) description known as the black oil model [Aziz & Settari, 1979]. In this work, we assume the reservoirs are in the secondary recovery phase where the pressures are above the bubble point pressure of the oil phase. Therefore, *two-phase immiscible flow*, that is, no mass transfer between the two liquid phases, is a fair assumption. We focus on water-flooding cases for two-phase (oil and water) reservoirs. Further, we assume incompressible fluids and rocks, no gravity effects or capillary pressure, no-flow boundaries, and finally isothermal conditions.

To implement the reservoir model, we do not develop an oil reservoir simulator from scratch. Instead, we use an open source MATLAB toolbox described in [Lie et al., 2011]. The assumptions in previous paragraph are covered in the toolbox. Interested readers to the toolbox can find more information on how to use the toolbox at its website (http://www.sintef.no/Projectweb/MRST/) along with references therein.

2.1.1 Mass Balance and Constitutive Equations

Let $\Omega \subset \mathbb{R}^d$ ($d \leq 3$) be a porous medium domain with boundary $\partial\Omega$ and let $\mathbf{n}$ be the outward pointing unit normal on the boundary as depicted in Figure 2.1 . The mass conservation in Ω implies that the rate of change inside Ω is equal to the rate of mass entering or leaving through the boundary $\partial\Omega$ and the rate of mass contributed by sources or sinks.

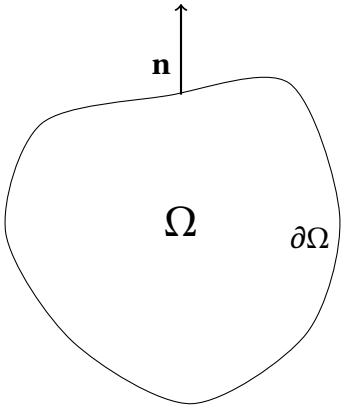

Figure 2.1: Porous media Ω in 2D-space, $d = 2$

Accordingly, the conservation of some quantity c can be described by

$$\frac{\partial}{\partial t}\int_\Omega c\,dx + \oint_{\partial\Omega} \mathbf{F}\cdot\mathbf{n}\,ds = \int_\Omega q\,dx, \tag{2.1}$$

where $\mathbf{F}$ is the mass flux and q is the source or sink term. Using the Gauss theorem this leads to

$$\int_\Omega \left(\frac{\partial}{\partial t}c + \nabla\cdot\mathbf{F}\right) dx = \int_\Omega q\,dx, \tag{2.2}$$

which gives us the continuity equation:

$$\frac{\partial}{\partial t}c + \nabla\cdot\mathbf{F} = q. \tag{2.3}$$

The conserved quantity c in two-phase oil reservoirs is phase mass $\phi s_\alpha \rho_\alpha$, where $\alpha = o$ (oil) or w (water), ϕ is the porosity, s is the saturation, and ρ is the fluid density. The density, and similarly the porosity, is independent of pressure as we assume that the fluid is incompressible. We define $\mathbf{F}_\alpha = \rho_\alpha \vec{v}_\alpha$, where $\vec{v}$ is the velocity. Now, the mass conversion law becomes

$$\frac{\partial}{\partial t}\left(\phi\rho_\alpha s_\alpha\right) + \nabla\cdot\left(\rho_\alpha\vec{v}_\alpha\right) = q_\alpha. \tag{2.4}$$

The velocities obey Darcy's law:

$$\vec{v}_\alpha = -\mathbf{K}\lambda_\alpha \nabla p_\alpha. \tag{2.5}$$

Here $\mathbf{K}$ is the permeability tensor, $\lambda_\alpha = k_{r\alpha}/\mu_\alpha$ is the phase mobility, where $k_{r\alpha}$ is the relative permeability, μ_α is the viscosity, and p_α is the pressure. The relative permeability is the main source of nonlinearity in the model. The relative permeability data are obtained from laboratory experiments using small portions of rock which do not generally represent the rock properties of the whole reservoir. Hence, uncertainties are unavoidable. The final data from the experiments are usually converted into look up tables used in a simulator.

The porosity and density are independent of pressure since we here consider incompressible rock and fluids. Therefore, they are just constants and consequently (2.4) can thus be rewritten as

$$\phi \frac{\partial s_\alpha}{\partial t} + \nabla \cdot \vec{v}_\alpha = \frac{q_\alpha}{\rho_\alpha}. \tag{2.6}$$

Two-phase Flow Formulations

As we assume there is no capillary pressure, that is, $p_o = p_w$, we then define a pressure variable p as $p = p_w = p_o$. The total saturation of the two-phase system is always one, i.e., $s_w + s_o = 1$. It is common to choose the water saturation as the primary unknown variable and define it as a variable s, that is, $s = s_w$. Now, the primary variables are the pressure p and water saturation s. Porosity, mobility, and density are constants since we assume that the flow is incompressible. However, relative permeabilities are not constant.

Relative permeability is a function of water saturation. In this work we assume the rock permeability follows the *Corey* model [Aziz & Settari, 1979];

$$s_{N,\alpha} = \frac{s - s_{\alpha r}}{1 - s_{wr} - s_{or}}, \tag{2.7}$$

$$k_{r\alpha} = k^0_{r\alpha}\, s^{n_\alpha}_{N,\alpha}, \tag{2.8}$$

where $s_{N,\alpha}$ is the normalized water saturation, s_{wr} and s_{or} are the residual water and oil saturation respectively, k^0_{rw} and k^0_{ro} are the end points of relative permeability, and n_w and n_o are empirical coefficients. The typical relative permeability functions is a (quadratic) function of water saturation as shown in Figure 2.2.

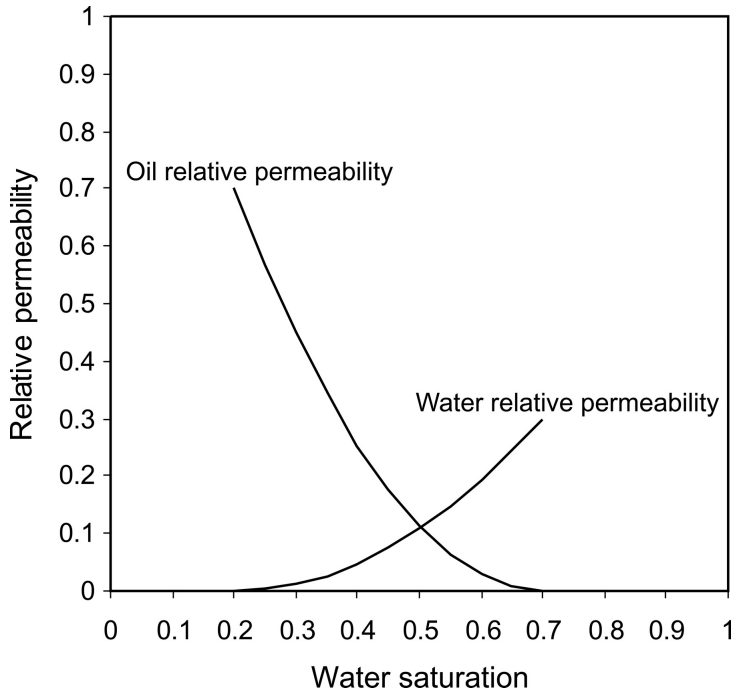

Figure 2.2: Typical relative permeability curves taken from `http://aapgbull.geoscienceworld.org/`

The velocity in (2.4) consists of velocity for oil and water. If we sum the velocity of the two phases and introduce $\vec{v} = \vec{v}_o + \vec{v}_w$ as the total velocity, we end up with

$$\begin{aligned} \vec{v} &= -\mathbf{K}\lambda_t(s)\nabla p, && \text{in } \Omega \\ \nabla \cdot \vec{v} &= q, && \text{in } \Omega \\ \vec{v} \cdot n &= 0, && \text{on } \partial\Omega \end{aligned} \tag{2.9}$$

where q is a volumetric rate, and λ_t is the total mobility, which in this setting is the sum of the water and oil mobility functions,

$$\lambda_t(s) = \lambda_w(s) + \lambda_o(s) = k_{rw}(s)/\mu_w + k_{ro}(s)/\mu_o. \tag{2.10}$$

We refer to the equations and boundary condition of (2.9) as the *pressure equations*, which in this case is an elliptic PDE. Now, we observe the water velocity from the pressure equation, that is

$$\vec{v}_w = -\frac{\lambda_w}{\lambda_t}\mathbf{K}\nabla p = \frac{\lambda_w}{\lambda_w + \lambda_o}\vec{v}, \tag{2.11}$$

and refer to $\frac{\lambda_w}{\lambda_w+\lambda_o}$ as the *water fractional flow* denoted by $f_w(s)$. We substitute the resulting water velocity (2.11) for the water phase of (2.6) to obtain

$$\phi\frac{\partial s}{\partial t} + \nabla \cdot f_w(s)\,\vec{v} = \frac{q_w}{\rho_w}, \tag{2.12}$$

which is the *saturation equation* (a hyperbolic PDE).

Discretization Method

The pressure and saturation equations, (2.9) and (2.12), are referred to as the state equations. It is impossible to find an analytical solution of the state equations for typical reservoir models. Hence, one approximates the solutions by some numerical method. In reservoir simulation literature typical methods of choice are the finite-difference, finite-volume, and finite-element methods. We do not differentiate the finite-difference and finite-volume method, in the sense that the finite-volume method is a conservative finite-difference scheme that treats the grid cells as control volumes [Aarnes et al., 2007].

However, before proceeding we need to select how to solve the state equations using a sequential time step scheme. In the reservoir simulation literature both *implicit* and *explicit* methods are applied. The implicit method is unconditionally stable while the explicit is potentially more efficient but restricted by numerical stability conditions. In addition to the most common fully implicit scheme, the solution strategy which can be used in reservoir management is the IMPES (*IM*plicit *P*ressure and *E*xplicit *S*aturation) method. However, in this work we use the other way around, i.e., explicit-pressure and implicit-saturation. First, the strategy computes relative permeabilities using the initial water saturation. Second, the pressure equation is solved using the initial water saturation and the secondary variables values. Third, with the obtained pressure solution, the velocity is computed and is used to solve the saturation equation. This procedure is repeated until the final time is reached.

We begin with the pressure equations (2.9). In this work we use a cell-centred finite-volume method, which is known as the *two-point flux-approximation (TPFA)* scheme. We discretize the domain Ω into a number of grid blocks, k, such that $\Omega_i \in \Omega$ and $i = 1, 2, \ldots, k$. After some rearrangement, (2.9) can be written as

$$\nabla \cdot \left(-\mathbf{K}\lambda_t(s)\,\nabla p\right) = q. \tag{2.13}$$

The left-hand side of (2.13), after discretization, is called the transmissibilities. The equation (2.13) is a linear equation, $\mathbf{A}(s)\,\mathbf{p} = \mathbf{q}$, the left-hand side is represented by $\mathbf{A}$, which is a symmetric matrix with diagonal elements given by

$$a_{ik} = \begin{cases} \sum_j t_{ij} & \text{if } k = i, \\ -t_{ik} & \text{if } k \neq i. \end{cases} \tag{2.14}$$

In a Cartesian grid, the matrix **A** is a tridiagonal matrix for 1D, pentadigonal for 2D, and heptadiagonal for 3D cases.

Note that the discretization above is a spatial discretization of the pressure equation. We need to discretize the equation in time as well. To perform temporal discretization for the pressure equation, we use a finite difference operator and end up with

$$\mathbf{A}(\mathbf{s}^{n-1})\mathbf{p}^n = \mathbf{B}\mathbf{u}^n. \tag{2.15}$$

Here, n represent the time step, $\mathbf{u}$ are the well rates, and $\mathbf{B}$ is the arrangement matrix of the well rates. Now, we intend to discretize the saturation equation, (2.12). We apply the finite difference operator;

$$\frac{\phi_i}{\triangle t}\left(\mathbf{s}_i^{n+1} - \mathbf{s}_i^n\right) + \frac{1}{|\Omega_i|}\sum_{j\neq i} R_{ij}\left(\mathbf{s}^{n+1}\right) = \frac{q_{w,i}\left(\mathbf{s}_i^n\right)}{\rho_w}. \tag{2.16}$$

The porosity in Ω_i is denoted by ϕ_i and R_{ij} is the approximation of the velocity at the edge γ_{ij}, which is

$$R_{ij} \approx \int_{\gamma_{ij}} \left(f_w(s)_{ij}\,\vec{v}_{ij}\right)\cdot n_{ij}\,dS, \tag{2.17}$$

where n_{ij} is the normal vector. The water fractional flow at the edge is approximated by using upstream weighting, such that

$$f_w(s)_{ij} = \begin{cases} f_w(s_i) & \text{if } \vec{v}\cdot n_{ij} \geq 0, \\ f_w\left(s_j\right) & \text{if } \vec{v}\cdot n_{ij} < 0. \end{cases} \tag{2.18}$$

This gives the following discrete form of the saturation equation.

$$\mathbf{s}^n = \mathbf{s}^{n-1} + \triangle t^n \mathbf{D}_{PV}^{-1}\left(\mathbf{R}\left(\mathbf{v}^n\right) f_w\left(\mathbf{s}^n\right) + \mathbf{q}\left(\mathbf{v}^n\right)_+\right). \tag{2.19}$$

Here $\mathbf{D}_{PV}$, $\mathbf{R}(\mathbf{v}^n)$ is a matrix representing $\left[f_w(\mathbf{s}^n)\,q(\mathbf{v}^n)_- - \nabla\cdot\left(f_w(\mathbf{s}^n)\,\mathbf{v}^n\right)\right]$, and $\mathbf{v}^n$ denote the total volume of Ω_i and the velocity, respectively at time instance n.

The discrete state equations (2.15) and (2.19) can be written in an implicit form $F(\widetilde{\mathbf{x}}, \widetilde{\mathbf{u}}) = 0$ as

$$\begin{aligned} \mathbf{F}(\widetilde{\mathbf{x}}, \widetilde{\mathbf{u}}) &= \begin{pmatrix} \mathbf{F}^0\left(\mathbf{p}^1, \mathbf{s}^0, \mathbf{s}^1, \mathbf{u}^1\right) \\ \vdots \\ \mathbf{F}^{N-1}\left(\mathbf{p}^N, \mathbf{s}^{N-1}, \mathbf{s}^N, \mathbf{u}^N\right), \end{pmatrix} \\ \mathbf{x}^{nT} &= (\mathbf{p}^{nT}, \mathbf{s}^{nT}), \quad n = 1, ..., N, \\ \widetilde{\mathbf{x}}^T &= (\mathbf{x}^{1T}, ..., \mathbf{x}^{NT}), \\ \widetilde{\mathbf{u}}^T &= (\mathbf{u}^{1T}, ..., \mathbf{u}^{NT}). \end{aligned} \tag{2.20}$$

The state vectors and control input vectors are stacked for all time instances from $n = 1, \ldots, N$. The control input variables are either bottom-hole pressures (BHP) in the wells or well rates. Both producer wells and injector wells are covered by this formulation. We refer to (2.20) as the *forward model* of the oil reservoir model.

Boundary Conditions

Due to no-flow boundary conditions, the driving forces of the reservoir models are at the wells. Injector wells inject water while producer wells produce both oil and water. At the wells we can either fix the rate or bottom-hole pressure (BHP). The former amounts to the Neumann boundary condition while the latter gives the so-called Dirichlet boundary condition.

In this study, the well rate is implemented by using the Peaceman well model [Peaceman, 1983], that is,

$$q_t = \lambda_t WI \left(p_{wf} - p_{gb}\right), \tag{2.21}$$

where WI denotes the well index, p_{wf} is the BHP, and p_{gb} is the well-block pressure. The well index is described by the following equation

$$WI = 2\pi \frac{dz \sqrt[3]{k_x k_y k_z}}{V_{gb}\left(ln\left(\frac{r_o}{r_w}\right) + S\right)}, \tag{2.22}$$

where dz is the well-segment length, V_{gb} the volume of the well grid block, r_w the radius of the well, k_x, k_y, k_z are the permeability in the x, y, z directions, respectively, and r_o is the effective well radius which is expressed as

$$r_o = 0.28 \frac{\left[\sqrt{\frac{k_x}{k_y}}(\triangle y)^2 + \sqrt{\frac{k_y}{k_x}}(\triangle x)^2\right]^{\frac{1}{2}}}{\sqrt[4]{\frac{k_x}{k_y}} + \sqrt[4]{\frac{k_y}{k_x}}}. \tag{2.23}$$

The grid block length in the x and y direction are denoted by $\triangle x$ and $\triangle y$, respectively.

2.2 Gradient-based optimization methods

Production optimization of oil reservoir is a constrained type mathematical nonlinear programming problem. In general, the problem reads

$$
\begin{aligned}
(\mathscr{P}) \quad & \max_{\tilde{\mathbf{u}} \in \mathbb{R}^{n_{\tilde{\mathbf{u}}}}} \mathscr{J}(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}) \\
\text{subject to}: \quad & \mathbf{F}(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}) = 0, \\
& g(\mathbf{u}^n) \geq 0, \ \forall n = 1, \ldots, N, \\
& h(\mathbf{x}^n, \mathbf{u}^n) \geq 0, \ \forall n = 1, \ldots, N, \\
& \mathbf{x}^0 \text{ is given.}
\end{aligned}
\tag{2.24}
$$

$\mathscr{J}(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})$, $g(\mathbf{u}^n)$, $h(\mathbf{x}^n, \mathbf{u}^n)$ are assumed $\mathscr{C}^2$ or $\mathscr{C}^1$ if one uses quasi-Newton methods. The control input and the output constraints are represented by $g : \mathbb{R}^{n_{\tilde{\mathbf{u}}}} \to \mathbb{R}^{n_g}$ and $h : \mathbb{R}^{n_{\tilde{\mathbf{x}}} \times n_{\tilde{\mathbf{u}}}} \to \mathbb{R}^{n_h}$, respectively. The objective function is given by $\mathscr{J} : \mathbb{R}^{n_{\tilde{\mathbf{x}}}} \times \mathbb{R}^{n_{\tilde{\mathbf{u}}}} \to \mathbb{R}$ and the state equations are posed as implicit constraints. The state variables and the control inputs are dependent, therefore we are able to perform the optimization in the control input space of $\tilde{\mathbf{u}}$ instead of in the full space given by $(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})$. To this end, we denote the objective as $\mathscr{J}(\tilde{\mathbf{u}})$ omitting $\mathscr{J}(\tilde{\mathbf{x}}(\tilde{\mathbf{u}}), \tilde{\mathbf{u}})$. The objective function is an economic measure of the oil reservoir such as net present value (NPV), recovery factor, sweeping efficiency, etc. Therefore, it is natural to maximize the value.

The relation between the optimization problem here, which is solved using the optimizer described in next section, and the reservoir model (simulator) explained in Section 2.1 is depicted in Figure 2.3. The reservoir simulator computes the objective function and its gradient based on the given control input, or decision variables, obtained from the optimizer. The control input must honor the state constraints, which are evaluated in the reservoir simulator as well.

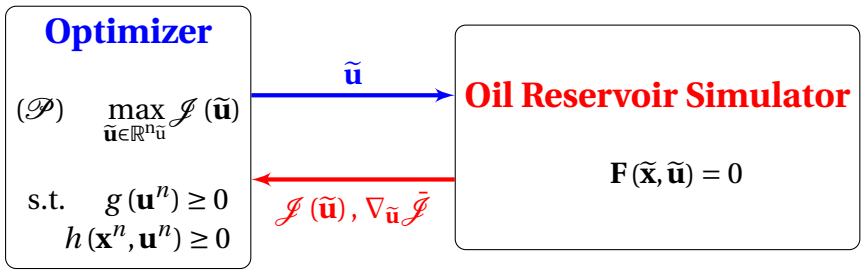

Figure 2.3: Relation between optimizer and oil reservoir simulator.

As in constrained optimization, it is worth mentioning the (first-order stationary) *Karush-Kuhn-Tucker* (KKT) *condition* since gradient-based methods seek to converge to a solution where this condition holds. A solution $\mathbf{u}^*$, which yields optimal states $\mathbf{x}^*$, of the optimization problem $\mathscr{P}$ is said to satisfy the KKT condition if there exist Lagrange multipliers vectors $\boldsymbol{\lambda}_g^*$ and $\boldsymbol{\lambda}_h^*$ such that

$$
\begin{aligned}
g\left(\mathbf{u}^*\right) &\geq 0, \\
h\left(\mathbf{x}^*, \mathbf{u}^*\right) &\geq 0, \\
\frac{\partial \mathscr{J}}{\partial \mathbf{u}^*} &= \frac{dg}{d\mathbf{u}^*}\boldsymbol{\lambda}_g^* + \frac{\partial h}{\partial \mathbf{u}^*}\boldsymbol{\lambda}_h^*, \\
\boldsymbol{\lambda}_g^* &\geq 0, \\
\boldsymbol{\lambda}_h^* &\geq 0, \\
g\left(\mathbf{u}^*\right)\boldsymbol{\lambda}_g^* &= 0, \\
h\left(\mathbf{x}^*, \mathbf{u}^*\right)\boldsymbol{\lambda}_h^* &= 0.
\end{aligned}
\tag{2.25}
$$

The detailed explanation on the KKT condition and its proof can be found in any numerical optimization book such as Nocedal & Wright [2006]. To solve the problem, with the current state-of-the-art constrained optimization methods there are two main solvers, which are based on active-set and interior-point methods. These methods are widely implemented in various optimization packages and one may find a regular benchmark for the packages at http://plato.asu.edu/bench.html. The main nonlinear optimizers include IPOPT, KNITRO, LOQO, PENNON, SNOPT, and CONOPT [1]. These results show that KNITRO and IPOPT which uses an interior point algorithm are the best performing packages, respectively. We have used these packages for production optimization of oil reservoirs, see Suwartadi et al. [2009] and KNITRO based on an active-set method gave better performance than IPOPT in our study. The performance of these packages is of course case dependent. We sketch the active-set algorithm, which is a variant of sequential quadratic programming (SQP) method, in this section.

Algorithm 1 Gradient-based optimization (subscript refers to iteration number)

INPUT: an initial control input $\widetilde{\mathbf{u}}_0$
OUTPUT: solution of the optimization problem $\widetilde{\mathbf{u}}_k$
while (stopping criteria is not met) **do**
  choose direction $\mathbf{d}_k$ and step length α_k
  $\widetilde{\mathbf{u}}_{k+1} = \widetilde{\mathbf{u}}_k + \alpha_k \mathbf{d}_k$
  $k = k + 1$
end while

Gradient based optimization can be structured as in Algorithm 1. As an introductory, we firstly do not take into account the constraints. The control and state constraints handling will be discussed in the next subsection using active-set method. The algorithm consists of the local method for updating the step based on gradient information and globalization strategy. In the literature [Nocedal & Wright, 2006] there are

[1] at http://plato.asu.edu/ftp/ampl-nlp.html

two main globalization strategies: line-search and trust-region methods. The line-search methods separate the computation of direction $\mathbf{d}_k$ and step length α_k. The trust-region methods however compute simultaneously the direction and step length after a trust-region radius is prescribed. Based on gradient information supplied in the direction computation, we can classify the gradient-based methods as follows

- Steepest descent : $\mathbf{d}_k = -\nabla \mathscr{J}(\widetilde{\mathbf{u}}_k)$
- Conjugate Gradient (CG): $\mathbf{d}_k = -\nabla \mathscr{J}(\widetilde{\mathbf{u}}_k) + \beta_k \mathbf{d}_{k-1}$. β_k is a constant chosen such that the direction at the current iteration is orthogonal to the previous one.
- Newton method: $\mathbf{d}_k = -\left(\nabla^2 \mathscr{J}(\widetilde{\mathbf{u}}_k)\right)^{-1} \nabla \mathscr{J}(\widetilde{\mathbf{u}}_k)$
- Quasi-newton method: $\mathbf{d}_k = -\mathbf{B}_k^{-1} \nabla \mathscr{J}(\widetilde{\mathbf{u}}_k)$, where $\mathbf{B}_k$ is an approximation of Hessian.

In the literature, the Hessian approximation mostly uses two types of algorithms, namely, symmetric-rank-one (SR1) and Broyden-Fletcher-Goldfarb-Shanno (BFGS). These algorithm obeys the the secant equation

$$\mathbf{B}_{k+1}\mathbf{s}_k = \mathbf{y}_k, \tag{2.26}$$

where $\mathbf{s}_k = \widetilde{\mathbf{u}}_{k+1} - \widetilde{\mathbf{u}}_k$ and $\mathbf{y}_k = \nabla \mathscr{J}_{k+1} - \nabla \mathscr{J}_k$. The SR1 update enforces symmetry with the rule

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \frac{\left(\mathbf{y}_k - \mathbf{B}_k\mathbf{s}_k\right)\left(\mathbf{y}_k - \mathbf{B}_k\mathbf{s}_k\right)^T}{\left(\mathbf{y}_k - \mathbf{B}_k\mathbf{s}_k\right)^T \mathbf{s}_k}, \tag{2.27}$$

while the BFGS enforces symmetry, positive definiteness, and rank-two update, through the update rule

$$\mathbf{B}_{k+1} = \mathbf{B}_k - \frac{\mathbf{B}_k\mathbf{s}_k\mathbf{s}_k^T\mathbf{B}_k}{\mathbf{s}_k^T\mathbf{B}_k\mathbf{s}_k} + \frac{\mathbf{y}_k\mathbf{y}_k^T}{\mathbf{y}_k^T\mathbf{s}_k}. \tag{2.28}$$

Due to its positive definiteness, the BFGS method is mostly used in line-search methods. Otherwise, a modification of the Hessian is needed. The step length α_k is computed in two steps: bracketing and bisection/interpolation. The bracketing means finding an interval containing a good step length. Afterwards, the bisection will compute a good step within the interval. We suggest interested readers to read chapter 3 of Nocedal & Wright [2006] for further detailed discussion on the step length computation.

As an alternative to line-search methods, the trust-region approach computes the step length and direction at once given a maximum trust-region radius. The method defines a region within which an quadratic approximation of the objective function is *trusted*, then minimize the approximation model within this region. If the step is unacceptable, the size of the trust-region radius is reduced. The quadratic model $m_k(\mathbf{d})$

of the true function $\mathscr{J}(\widetilde{\mathbf{u}}_k + \mathbf{d})$ at the point $\widetilde{\mathbf{u}}_k$ is

$$m_k(\mathbf{d}) = \mathscr{J}_k + \nabla \mathscr{J}_k^T \mathbf{d} + \frac{1}{2}\mathbf{d}^T \mathbf{B}_k \mathbf{d}. \tag{2.29}$$

One might use the exact Hessian for the $\mathbf{B}_k$ term. For every trust-region iteration, the step is found by solving the following constrained optimization problem

$$\underset{\mathbf{d}\in\mathbb{R}^n}{\text{minimize}} \qquad m_k(\mathbf{d}) \quad \text{s.t.} \quad \|\mathbf{d}\| \leq \triangle_k. \tag{2.30}$$

The quality of model approximation $m_k(\mathbf{d})$ is measured by the ratio of actual reduction to the predicted reduction of the objective function, that is,

$$\rho_k = \frac{\mathscr{J}(\widetilde{\mathbf{u}}_k) - \mathscr{J}(\widetilde{\mathbf{u}}_k + \mathbf{d}_k)}{m_k(0) - m_k(\mathbf{d}_k)}. \tag{2.31}$$

If ρ_k is small it implies the model is not a good estimate of the objective function and if ρ_k is large, the model is good enough to approximate the objective function. To wrap-up the trust-region methods, Algorithm 2 describes the methods. The method for solving subproblem (2.30) is explained in the next chapter.

Algorithm 2 Trust-region

INPUT: maximum trust-region radius:$\hat{\triangle} > 0$, initial trust-region radius: $\triangle_0 \in \left(0, \hat{\triangle}\right)$, and $\eta \in [0, \frac{1}{4})$
OUTPUT: solution of the optimization problem $\widetilde{\mathbf{u}}_k$
while (stopping criteria is not met) **do**
  solve (2.30) to obtain $\mathbf{d}_k$
  compute ρ_k from (2.31)
  if $\rho_k < \frac{1}{4}$
    $\triangle_{k+1} = \frac{1}{4}\triangle_k$
  else
    if $\rho_k > \frac{3}{4}$ and $\|\mathbf{d}_k\| = \triangle_k$
      $\triangle_{k+1} = \min\left(2\triangle_k, \hat{\triangle}\right)$
    else
      $\triangle_{k+1} = \triangle_k$
  end if
  if $\rho_k > \eta$
      $\widetilde{\mathbf{u}}_{k+1} = \widetilde{\mathbf{u}}_k + \mathbf{d}_k$
  else
      $\widetilde{\mathbf{u}}_{k+1} = \widetilde{\mathbf{u}}_k$
  end if
end while

Sequential Linear-Quadratic Programming (SLQP)

We now explain briefly the active-set SQP method implemented in KNITRO. The details are described in Bryd et al. [2004], which is also summarized in Chapter 18 of Nocedal & Wright [2006]. It should be noted that there may be notations overlapping in this subsection to the ones in the reservoir model. Readers should read the description of the notations in order to avoid any confusion.

The method belongs to the equality-constrained quadratic program (EQP) type of SQP; another method is based on inequality-constrained (IQP). EQP is more appealing than the IQP since it gives more economical CPU time when dealing with large dimensions of decision variable. IQP-based methods compute the step and estimate the optimal active set at the same time while EQP separates the computation into two stages. First, linear program (LP) with the goal to estimate the optimal active set is performed. Second, an equality-constrained quadratic program is solved to compute the step given the active-set constraint in the first step. The method uses the ℓ_1 merit function to determine the acceptability of a step within a trust-region framework. The second step is solved using a projected conjugate gradient (PCG) which will be explained in the next subsection.

To solve the problem $\mathscr{P}$, we rewrite the equation (2.24) as follows

$$\begin{aligned} \min_{\widetilde{\mathbf{u}}} \quad & -\mathscr{J}(\widetilde{\mathbf{u}}) \doteq f(\widetilde{\mathbf{u}}) \\ \text{subject to}: \quad & c_i(\widetilde{\mathbf{u}}) = 0, \qquad i \in \mathscr{E} \\ & c_i(\widetilde{\mathbf{u}}) \geq 0, \qquad i \in \mathscr{I}. \end{aligned} \tag{2.32}$$

The bound constraints $g(\widetilde{\mathbf{u}})$ are divided into equality and inequality constraints. The inequality constraints are handled explicitly using CG method. The other constraints will be a part of equality constraints $c_i(\widetilde{\mathbf{u}}) = 0, i \in \mathscr{E}$. The state constraints are described as inequality constraints $c_i(\widetilde{\mathbf{u}}) \geq 0, i \in \mathscr{I}$. The ℓ_1 merit function takes the form

$$\phi(\widetilde{\mathbf{u}}; \nu) = f(\widetilde{\mathbf{u}}) + \nu \sum_{i \in \mathscr{E}} |c_i(\widetilde{\mathbf{u}})| + \nu \sum_{i \in I} (\max(0, -c_i(\widetilde{\mathbf{u}}))), \tag{2.33}$$

where ν is the penalty parameter.

In the first stage a LP program, which aims to estimate the optimal active set $\mathscr{W}^*$, reads

$$\begin{aligned} \min_{\mathbf{d}} \quad & f_k + \nabla f_k^T \mathbf{d} \\ \text{subject to}: \quad & c_i(\widetilde{\mathbf{u}}) + \nabla c_i(\widetilde{\mathbf{u}})^T \mathbf{d} = 0, \quad i \in \mathscr{E} \\ & c_i(\widetilde{\mathbf{u}}) + \nabla c_i(\widetilde{\mathbf{u}})^T \mathbf{d} \geq 0, \quad i \in \mathscr{I} \\ & \|\mathbf{d}\|_\infty \leq \triangle_k^{LP}, \end{aligned} \tag{2.34}$$

where $\triangle_k^{LP}$ is the trust-region radius at iteration k and $\mathbf{d}$ is the solution. Because of the linearization of the objective function, this LP formulation can yield infeasible constraints. Thus, a ℓ_1 penalty is added to the linearized objective function such that

$$\begin{aligned} \min_{\mathbf{d}} \quad & \ell_\nu(\mathbf{d}) \doteq f_k + \nabla f_k^T \mathbf{d} + \nu \sum_{i\in\mathscr{E}} \left| c_i(\widetilde{\mathbf{u}}_k) + \nabla c_i(\widetilde{\mathbf{u}})^T \mathbf{d} \right| \\ & \quad + \nu \sum_{i\in\mathscr{I}} \max\left(0, -c_i(\widetilde{\mathbf{u}}) - \nabla c_i(\widetilde{\mathbf{u}})^T \mathbf{d}\right) \\ \text{subject to:} \quad & \|\mathbf{d}\|_\infty \le \triangle_k^{LP}. \end{aligned} \tag{2.35}$$

However, this ℓ function is well-known to be non-differentiable. Therefore, a smooth linear program below is used.

$$\begin{aligned} \min_{\mathbf{d},\mathbf{r},\mathbf{s},\mathbf{t}} \quad & f_k + \nabla f_k^T \mathbf{d} + \nu \sum_{i\in\mathscr{E}} (r_i + s_i) + \nu \sum_{i\in\mathscr{I}} t_i \\ \text{subject to:} \quad & c_i(\widetilde{\mathbf{u}}_k) + \nabla c_i(\widetilde{\mathbf{u}}_k)^T \mathbf{d} = r_i - s_i, \qquad i \in \mathscr{E} \\ & c_i(\widetilde{\mathbf{u}}_k) + \nabla c_i(\widetilde{\mathbf{u}}_k)^T \mathbf{d} \ge -t_i, \qquad i \in \mathscr{I} \\ & \|\mathbf{d}\|_\infty \le \triangle_k^{LP} \\ & \mathbf{r}, \mathbf{s}, \mathbf{t} \ge 0. \end{aligned} \tag{2.36}$$

Here $\mathbf{r}$, $\mathbf{s}$, and $\mathbf{t}$ are vectors of slack variables. To this end, we denote a solution of this LP problem as $\mathbf{d}^{LP}(\nu)$. The resulting estimate of the optimal active set from this solution is

$$\begin{aligned} \mathscr{A}_k\left(\mathbf{d}^{LP}\right) = & \left\{ i \in \mathscr{E} \mid c_i(\widetilde{\mathbf{u}}_k) + \nabla c_i(\widetilde{\mathbf{u}}_k)^T \mathbf{d}^{LP} = 0 \right\} \\ & \cup \left\{ i \in \mathscr{I} \mid c_i(\widetilde{\mathbf{u}}_k) + \nabla c_i(\widetilde{\mathbf{u}}_k)^T \mathbf{d}^{LP} = 0 \right\}, \end{aligned}$$

and the set of constraints violation is

$$\begin{aligned} \mathscr{V}_k\left(\mathbf{d}^{LP}\right) = & \left\{ i \in \mathscr{E} \mid c_i(\widetilde{\mathbf{u}}_k) + \nabla c_i(\widetilde{\mathbf{u}}_k)^T \mathbf{d}^{LP} \neq 0 \right\} \\ & \cup \left\{ i \in \mathscr{I} \mid c_i(\widetilde{\mathbf{u}}_k) + \nabla c_i(\widetilde{\mathbf{u}}_k)^T \mathbf{d}^{LP} < 0 \right\}, \end{aligned}$$

and denote its complement by $\mathscr{V}_k^c$. The working active set, $\mathscr{W}_k$, is defined as linearly independent subset of the active set $\mathscr{A}_k\left(\mathbf{d}^{LP}\right)$. Furthermore, we define a Cauchy point $\mathbf{d}^c$ to ensure the algorithm has favorable global convergence properties, by the following equation

$$\mathbf{d}^c = \alpha^{LP} \mathbf{d}^{LP}, \tag{2.37}$$

where $\alpha^{LP} \in (0,1]$ is the steplength computed using a line search backtracking method in a decreasing direction of the quadratic approximation objective function (2.33).

Given the solution $\mathbf{d}^{LP}$ and working set $\mathcal{W}_k$, we now attempt to solve the EQP problem

$$\begin{aligned} \min_{\mathbf{d}} \quad & f_k + \tfrac{1}{2}\mathbf{d}^T \mathbf{H}_k^{EQP}(\tilde{\mathbf{u}}, \boldsymbol{\lambda})\,\mathbf{d} + \nabla\left(f_k^{EQP}\right)^T \quad \mathbf{d} \\ \text{subject to:} \quad & c_i(\tilde{\mathbf{u}}_k) + \nabla c_i(\tilde{\mathbf{u}}_k)^T \mathbf{d} = 0, \qquad i \in \mathcal{E} \cap \mathcal{W}_k \\ & c_i(\tilde{\mathbf{u}}_k) + \nabla c_i(\tilde{\mathbf{u}}_k)^T \mathbf{d} = 0, \qquad i \in \mathcal{I} \cap \mathcal{W}_k \\ & \|\mathbf{d}\|_2 \le \triangle_k. \end{aligned} \tag{2.38}$$

The $\mathbf{H}^{EQP}$ and f^{EQP} are defined as follows:

$$\begin{aligned} H^{EQP}(\tilde{\mathbf{u}}, \boldsymbol{\lambda}) \;=\; & \nabla^2 f(\tilde{\mathbf{u}}) + \nu \sum_{i \in \mathcal{V} \cap \mathcal{E}} \operatorname{sign}\left(c_i + \nabla c_i(\tilde{\mathbf{u}})^T \mathbf{d}\right) \nabla^2 c_i(\tilde{\mathbf{u}}) \\ & -\nu \sum_{i \in \mathcal{V} \cap \mathcal{I}} \nabla^2 c_i(\tilde{\mathbf{u}}) - \sum_{i \in \mathcal{V}^c \cap \mathcal{E}} \lambda_i \nabla^2 c_i(\tilde{\mathbf{u}}) - \sum_{i \in \mathcal{V}^c \cap \mathcal{I}} \lambda_i \nabla^2 c_i(\tilde{\mathbf{u}}). \end{aligned} \tag{2.39}$$

$$\begin{aligned} f^{EQP} \;=\; & \mathbf{H}^{EQP}(\tilde{\mathbf{u}}, \boldsymbol{\lambda})\,\mathbf{d} + \nabla f(\tilde{\mathbf{u}}) \\ & +\nu \sum_{i \in \mathcal{V} \cap \mathcal{E}} \operatorname{sign}\left(c_i(\tilde{\mathbf{u}}) + \nabla c_i(\tilde{\mathbf{u}})^T \mathbf{d}\right) \nabla c_i(\tilde{\mathbf{u}}) - \nu \sum_{i \in \mathcal{V} \cap \mathcal{I}} \nabla c_i(\tilde{\mathbf{u}}). \end{aligned} \tag{2.40}$$

We denote a solution of (2.38) solved by the projected-CG (PCG) as $\mathbf{d}^{EQP}$. The total step $\mathbf{d}_k$ of the SLQP algorithm is

$$\mathbf{d}_k = \mathbf{d}^c + \alpha^{EQP}\left(\mathbf{d}^{EQP} - \mathbf{d}^c\right). \tag{2.41}$$

Here, the steplength $\alpha^{EQP} \in [0,1]$ is similar to α^{LP} solved by the line-search backtracking algorithm. The SLQP algorithm relies on the penalty parameter ν, two trust-region radii $\triangle^{LP}$ and $\triangle$. We refer to Bryd et al. [2004] for discussion on how to choose and update these parameters values.

Projected Conjugate Gradient (PCG)

We now describe the PCG-algorithm used for solving (2.38). We use Steihaug conjugate gradient [Steihaug, 1983] to compute a solution of the trust-region subproblem

$$\begin{aligned} \min_{\mathbf{x} \in \mathbb{R}^n} \quad & m_k(\mathbf{x}) \doteq f_k + \nabla f_k^T \mathbf{x} + \tfrac{1}{2}\mathbf{x}^T \mathbf{B}_k \mathbf{x} \\ \text{subject to:} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \|\mathbf{x}\| \le \triangle_k \end{aligned} \tag{2.42}$$

where $\mathbf{B}_k$ is the Hessian or its approximation. The Steihaug-PCG is described in Algorithm 3 below. The inequality constraints, that is, $l \leq g(\tilde{\mathbf{u}}) \leq u$ is denoted by $l \leq \mathbf{x} \leq u$ affects the trust-region radius constraint such that it now becomes a bound constraint: $\max(l, \mathbf{x}_k - \triangle_k \mathbf{e}) \leq \mathbf{x} \leq \min(u, \mathbf{x}_k + \triangle_k \mathbf{e})$, where $\mathbf{e} = (1, 1, \ldots, 1)^T$.

Algorithm 3 PCG-Steihaug

INPUT: tolerance $\epsilon_k > 0$, initial $\mathbf{x}_0$ satisfying $\mathbf{A}\mathbf{x}_0 = \mathbf{b}$, $\mathbf{P}$ projection operator.
OUTPUT: solution $\mathbf{x}_k$
Initialization: Set $\mathbf{z}_0 = \mathbf{P}\mathbf{x}_0$, $\mathbf{r}_0 = \mathbf{B}\mathbf{x}_0 + \nabla f_0$, $\mathbf{d}_0 = -\mathbf{r}_0$ and iteration $j = 0$.
if $\|\mathbf{r}_0\| < \epsilon_k$
  $\mathbf{x}_k = \mathbf{z}_0$
end if
while (stopping tolerance is not met) **do**
  if $\mathbf{d}_j^T \mathbf{B}_k \mathbf{d}_j \leq 0$
    compute τ such that $\mathbf{x}_k = \mathbf{P}(\mathbf{z}_j + \tau \mathbf{d}_j)$ minimizes $m_k(\mathbf{d}_k)$ in (2.42)
    and satisfies $\|\mathbf{x}_k\| = \triangle_k$
    stop
  end if
  set $\alpha_j = \mathbf{r}_j^T \mathbf{r}_j / \mathbf{d}_j^T \mathbf{B}_k \mathbf{d}_j$
  set $\mathbf{z}_{j+1} = \mathbf{z}_j + \alpha_j \mathbf{d}_j$
  if $\|\mathbf{z}_{j+1}\| \geq \triangle_k$
    find $\tau \geq 0$ such that $\mathbf{x}_k = \mathbf{P}(\mathbf{z}_j + \tau \mathbf{d}_j)$ satisfies $\|\mathbf{x}_k\| = \triangle_k$
    stop
  end if
  set $\mathbf{r}_{j+1} = \mathbf{r}_j + \alpha_j \mathbf{B}_k \mathbf{d}_j$
  if $\|\mathbf{r}_{j+1}\| < \epsilon_k$
    $\mathbf{x}_k = \mathbf{P}(\mathbf{z}_{j+1})$
    stop
  end if
  set $\beta_{j+1} = \mathbf{r}_{j+1}^T \mathbf{P}(\mathbf{r}_{j+1}) / \mathbf{r}_j^T \mathbf{r}_j$
  set $\mathbf{d}_{j+1} = -\mathbf{P}(\mathbf{r}_{j+1}) + \beta_{j+1} \mathbf{d}_j$
  set $j = j + 1$
end while

2.3 Computing gradients

The conventional way of computing gradients is to use finite-difference methods. These methods are easy to implement because they treat the system simulator, e.g., an oil

reservoir simulator, as a black box. However, when the simulator is comprehensive and the number of control variables is large, these methods become prohibitive. Another way is to use sensitivity equations, which is still quite expensive. It should be noted however that there exists interesting options which exploits the structural similarities between the sensitivity equations and the model equations [Gunzburger, 2003]. The modest CPU time of both methods is $\mathcal{O}(n_{\tilde{\mathbf{u}}}+1)$ times an evaluation of objective function value, where $n_{\tilde{\mathbf{u}}}$ is the dimensional of control input. The finite differences methods require even more CPU time to increase gradient accuracy by for instance using a central difference scheme instead of a one-sided method. An accurate gradient approximation using a complex method is used in Kim et al. [2006]. The gradient is approximated by

$$\nabla_{\tilde{\mathbf{u}}}\mathcal{J}(\tilde{\mathbf{u}})\big|_{\tilde{\mathbf{u}}^*} = \frac{\mathrm{Im}\left(\mathcal{J}\left(\mathbf{z}_{\tilde{\mathbf{u}}}^k\right)\right)}{h} + O\left(h^2\right), \tag{2.43}$$

where $\mathrm{Im}(\cdot)$ is the imaginary part of the argument and $\mathbf{z}_{\tilde{\mathbf{u}}}^k$ is the perturbation representation of the k-th element in the form of

$$\mathbf{z}_{\tilde{\mathbf{u}}}^k = \begin{bmatrix} \mathbf{u}_1^* & \mathbf{u}_2^* & \dots & \mathbf{u}_{k-1}^* & \left(\mathbf{u}_k^* + ih\right) & \mathbf{u}_{k+1}^* & \dots & \mathbf{u}_N^* \end{bmatrix}^T$$

with $i = \sqrt{-1}$. This approximation results in similar accuracy to that of central finite-difference but with the same CPU time as forward finite-difference, that is $\mathcal{O}(n_{\tilde{\mathbf{u}}}+1)$. Because there is no subtraction operation in the approximation, h can be chosen to a very small value, e.g., $h = 10^{-100}$. Another practical approach for constructing the gradient (Jacobian) would be to use automatic differentiation (AD) tools [Griewank & Walther, 2008].

In this thesis adjoint-gradient computation is utilized, which requires two times evaluation of the objective function regardless of the dimension of control input. Detailed explanation of the adjoint method can be found in the subsequent chapters. We have implemented a first-order and second-order adjoint-gradient for the reservoir simulation model used in this work.

In order to validate the adjoint-gradient implementation, one obviously can compare the gradients produced by the adjoint methods against those from the finite-difference computation. In other fields, such as in meteorology, a Taylor expansion method is used to check the adjoint-gradient [Li et al., 1993]. The objective function can be described as

$$\mathcal{J}(\tilde{\mathbf{u}}+\delta\tilde{\mathbf{u}}) = \mathcal{J}(\tilde{\mathbf{u}}) + \delta\tilde{\mathbf{u}}^T\nabla\mathcal{J} + h.o.t.^1 \tag{2.44}$$

[1] high order terms

or rewritten it as

$$\frac{\mathscr{J}(\widetilde{\mathbf{u}}+\delta\widetilde{\mathbf{u}})-\mathscr{J}(\widetilde{\mathbf{u}})}{\delta\widetilde{\mathbf{u}}\nabla\mathscr{J}}=1+h.o.t. \tag{2.45}$$

The gradient obtained from the adjoint method is $\nabla\mathscr{J}$ and the goal is now to make the left hand side of (2.45) approach 1 as $\delta\widetilde{\mathbf{u}}$ approaches zero. If we define $\delta\widetilde{\mathbf{u}}$ by

$$\delta\widetilde{\mathbf{u}}=\alpha\frac{\nabla\mathscr{J}}{\left\|\nabla\mathscr{J}\right\|}, \tag{2.46}$$

which is in the direction of $\nabla\mathscr{J}$ and the variation of the control input gives a consistent scaling, then (2.46) in (2.45) will result in

$$\phi(\alpha)=\frac{\mathscr{J}\left(\widetilde{\mathbf{u}}+\alpha\frac{\nabla\mathscr{J}}{\left\|\nabla\mathscr{J}\right\|}\right)-\mathscr{J}(\widetilde{\mathbf{u}})}{\alpha\left(\frac{\nabla\mathscr{J}}{\left\|\nabla\mathscr{J}\right\|}\right)^{T}\cdot\nabla\mathscr{J}}=1+O(\alpha). \tag{2.47}$$

If the obtained gradient is correct, small values of α, ϕ should linearly approach 1 as α is reduced. Another method to check the adjoint-gradient implementation is to use the checkpointing techniques, see, e.g., Walther & Griewank [2004].

2.4 Numerical Linear Algebra Stuff

The pressure and adjoint equations in this work are linear equations. The saturation equation is non-linear, but requires multiple linear equations. To solve solutions of the equations, we use a direct sparse solver which is the default blackslash (\) operator in MATLAB. An explanation of the sparse linear solver approach can be found in Davis [2006].

Another concept which will be used in the next chapters, particularly in the reduced-order model part, is singular value decomposition (SVD). SVD is used to compute singular values and vectors of a matrix which will be basis functions for a reduced-order model. Let $\mathbf{A}\in\mathbb{R}^{\mathrm{m}\times\mathrm{n}}$, $m\geq n$ and $rank(\mathbf{A})=r\leq n$. Then, the matrix $\mathbf{A}$ can be decomposed as

$$\mathbf{A}=\mathbf{U}\mathbf{\Sigma}\boldsymbol{V}^{T}, \tag{2.48}$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthonormal matrices, $\mathbf{U}\in\mathbb{R}^{\mathrm{m}\times\mathrm{m}}$ and $\mathbf{V}\in\mathbb{R}^{\mathrm{n}\times\mathrm{n}}$. $\mathbf{\Sigma}=\begin{pmatrix}\mathbf{\Sigma}_r & 0\\ 0 & 0\end{pmatrix}$, where $\mathbf{\Sigma}_r=diag(\sigma_1,\ldots,\sigma_r)\in\mathbb{R}^{\mathrm{r}\times\mathrm{r}}$ with $\sigma_1\geq\ldots\geq\sigma_r>0$. The decomposition (2.48) is called *singular value decomposition* (SVD) and the diagonal entries of $\mathbf{\Sigma}_r$ are the *singular values*. Matrix $\mathbf{A}$ will be obtained through snapshots of the dynamic response of reservoir systems in a later section of this thesis. It should be noted that reduced

SVD is useful when computing reduced-order basis, that is, the same decomposition but for $\mathbf{U} \in \mathbb{R}^{m \times n}$, $\mathbf{\Sigma} = \mathbf{\Sigma}_r$, and $\mathbf{V} \in \mathbb{R}^{n \times n}$.

Chapter 3

Nonlinear Output Constraints Handling for Production Optimization of Oil Reservoirs

This chapter is based on Suwartadi et al. [2012b] which is published in Computational Geoscience Vol.16:2, 2012. Earlier papers on the same topic are Suwartadi et al. [2009], Suwartadi et al. [2010b] and Suwartadi et al. [2010a].

Adjoint-based gradient computations for oil reservoirs have been increasingly used in closed-loop reservoir management optimization. Most constraints in the optimizations are linked to the control inputs. These are either linear bound constraints or linear equality constraints. This paper addresses (nonlinear) output constraints and proposes to use a (interior) barrier function approach, where the output constraints are added as a barrier term to the objective function. As we assume there always exist feasible initial control inputs, the method maintains the feasibility of the constraints. Three case examples are presented. One of these examples resembles the Norne field geometry. The results show that the proposed method is able to preserve the computational efficiency of the adjoint method.

3.1 Introduction

The use of optimization is steadily expanding to evermore demanding applications. Hence, the need for robust and computationally efficient algorithms is critical. Demands increase due to factors like system size, the nature of nonlinearities, integer decision variables, stochasticity and reduced computation time. In this work we focus on large-scale systems, which typically originates from discretized dynamic nonlinear

PDE-models, and the use of adjoint-based techniques for gradient calculations. In all high-standard gradient-based algorithms efficient computation of the gradient itself is a critical ingredient. Before presenting the problem in more detail, a review of adjoint based methods and its relevance to our application area will be given.

Computing the gradient of the objective function with respect to the decision variables in large-scale systems can be computationally prohibitive. To compute the gradient, there are essentially three approaches; finite differences, the forward sensitivity equation and the (backwards) adjoint approach. For n_u decision variables the finite difference method requires $n_u + 1$ simulation runs when applying the forward difference approach. This method might be impractical, in the context of PDE-constrained optimization if the number of decision variables is large. For example, history matching optimization in reservoir engineering has a large number of decision variables. The sensitivity equation method requires one simulation run of the discretized PDE in addition to n_u sensitivity models, each with a complexity similar to the discretized PDE itself. A challenge for large-scale systems is the sheer size of the sensitivity information which has to be stored to compute the gradient of the objective function. The storage requirement amounts to $n_u \cdot n_x \cdot N$, where n_x is the state dimension and N is the number of time steps in the simulation horizon. In addition to the above, the emerging ensemble-based methods (see Chen et al. [2008]) enables gradient computations in a stochastic framework although the computational time remains a challenge.

The beauty of the adjoint method is the fact that the gradient of the objective function can be computed by two simulations runs only, applying the discretized PDE in the normal forward mode and subsequently its adjoint in a backward mode Bryson & Ho [1975]. This is potentially highly efficient for computationally expensive models. This advantage does however add some challenges. First, it is necessary to store the complete solution of the forward simulation, that is, all the state vectors at each solution time, since they are used as inputs to the adjoint model. Second and more importantly, the presence of output constraints, or path constraints, may be detrimental to the efficiency of the adjoint method. The reason is that constraint gradient information is required. Therefore each new constraint requires one additional Lagrange multiplier, and, hence a new adjoint system. This observation has been made by many authors (see e.g., Mehra & Davis [1972]; Hargraves & Paris [1987]; Bloss et al. [1999]) as a starting point for proposing changes to maintain efficiency of the adjoint approach in the output constrained case. Finally, the adjoint implementation itself is a challenging task. The adjoint method is straightforward, but the implementation is a completely different matter.

3.1.1 Background

In the optimal control literature (see e.g., Bryson & Ho [1975]; Becerra [2004]) practical methods to handle path constraints are typically collocation methods, which convert optimal control problems to nonlinear programming problems. The analytical method described in Bryson & Ho [1975] requires prior knowledge of the number and sequence of path-constrained arcs, which might not be possible to obtain. The use of the generalized reduced gradient (GRG) method proposed in Mehra & Davis [1972] and SQP-based optimization in Hargraves & Paris [1987] are in the class of collocation methods, where the path constraints are posed as inequality constraints at each collocation point in time. Collocation implies that both the control inputs and states are discretized. Hence, the GRG method consists of independent and dependent variables where the former are referred to as the decision variables. A change is enforced if the method yields an infeasible solution. This implies that one of the independent variables is changed to a dependent variable and vice versa. Moreover, in this method, the gradient is projected onto the constraint set and if some of the dependent variables hit the constraint boundary, the variables will be changed as independent variables in the next iteration. The changes from independent to dependent variables may be difficult to implement and become practically infeasible in large-scale optimization problems.

In the nonlinear optimization literature, in addition to the SQP method, penalty and barrier methods Fiacco & McCormick [1968] are often used to handle the path constraints. In these methods the constraints are augmented to the objective function equipped with a parameter known as barrier or penalty parameter. Hence, the path constraints are removed. A solution is found as a sequence of minimizations (or maximizations), where the penalty or barrier parameter is varied between iterations. However, the methods might cause numerical difficulty as they approach the solution, that is ill-conditioning of the Hessian matrix may occur which affects the convergence rate. This gives rise to the augmented Lagrangian method and more recently Primal Dual logarithmic barrier methods. In practice the ill-conditioned Hessian matrix, however, can be benign if the methods are implemented properly. An excellent survey of the class of interior point methods, which also includes classical penalty and barrier methods, can be found in Forsgren et al. [2002].

Path constraints are notoriously difficult to handle and there are several approaches to mitigate the problems. One of the most used methods is based on constraint lumping, also known as ℓ_1 constraints. The idea is to lump controller coefficients in a way that allows solution of a scalar optimization function, which is known as the *KS (Kreisselmeier-Steinhauser)* function approach Kreisselmeier & Steinhauser [1979]. The KS function is used to aggregate all the path constraints into one constraint. The aggregation is in the form of an exponential term. Although it seems appealing, the KS function does not augment the path constraints to the objective function. Therefore, it

requires the Jacobians of the path constraints, which is expensive for large-scale problems. On the other hand, there are also methods trying to approximate the Jacobians, for example, the TR2 method Griewank & Korzec [2005]. In the PDE-constrained optimization literature (see e.g., Hinze et al. [2009]), Moreau-Yosida regularization and Lavrentiev regularization are usually used to handle the path constraints. The Moreau-Yosida regularization is similar to a penalty method while the Lavrentiev regularization is also a penalty-based method with a slightly modification in the penalty term. The Lavrentiev regularization introduces a parameter as a new artificial control.

In reservoir management research, the adjoint method has been used in oil reservoir production optimization problems since the 1980s and 1990s (see e.g., Virnovsky [1991]; Zakirov et al. [1996]) and recently it was reviewed in Jansen [2011]. Applications of the adjoint method have further been suggested for history matching [Wu, 1999; Rommelse, 2009] and well placement optimization Handels & Zandvliet [2007]. The development of path constraints handling in adjoint-based optimization for reservoir management has been considered in Montleau et al. [2006]; Sarma et al. [2008]; Kraaijevanger et al. [2007]. In Sarma et al. [2008] a feasible direction optimization method combined with a lumping (state) constraints strategy is proposed. The lumping of constraints in Sarma et al. [2008] is the same as the KS function approach. However, the proposed method in Sarma et al. [2008] may converge to an infeasible solution. This was clearly shown in one of the case examples where the solution violated the constraint. The use of the GRG method was proposed in Montleau et al. [2006] and Kraaijevanger et al. [2007] with a certain control input and state variables combination, that is, bottom hole pressure (BHP) controlled wells with rate constraints and vice versa. It cannot easily be extended to other control input and state variable combinations, such as constraints on saturation or total water production, because these cannot be considered independent variables.

In this paper we propose the use of a Lagrangian barrier method for reservoir management optimization problems. The reason to use this method instead of using the existing optimization packages is that one needs to supply the Jacobians of the output constraints which will give an overhead to the computational time. This leads to the development of a customized optimization package. The proposed method is different from the usual barrier method Fiacco & McCormick [1968] in the sense that the constraints are treated differently. The Lagrangian barrier method uses Lagrangian multiplier estimates as a device to identify active constraints asymptotically and to prevent numerical problems, that is, an ill-conditioned Hessian matrix. Although in practice this rarely occurs and with the advent of trust region method, the ill-conditioned problem can be circumvented Forsgren et al. [2002]. The method is presented and compared to a pure barrier method and a standard nonlinear programming method by applying it on three case examples. This provides a basis for drawing conclusions.

3.2 Forward Model

In this work, we assume the reservoirs are in the secondary recovery phase where the pressures are above the bubble point pressure of the oil phase. Therefore, *two-phase immiscible flow*, that is, without mass transfer between the two phases, is a reasonable assumption. We focus on water-flooding cases for two-phase (oil and water) reservoirs. Further, we assume incompressible fluids and rocks, no gravity effects or capillary pressure, no-flow boundaries, and finally isothermal conditions.

The state equations in an oil reservoir Ω, with boundary $\partial\Omega$ and outward facing normal vector $\mathbf{n}$, can be represented by pressure and saturation equations. The pressure equation is described as

$$\begin{aligned} \vec{v} &= -\mathbf{K}\lambda_t(s)\nabla p, && \text{in}\,\Omega \\ \nabla\cdot\vec{v} &= q, && \text{in}\,\Omega \\ \vec{v}\cdot\mathbf{n} &= 0, && \text{on}\,\partial\Omega \end{aligned} \tag{3.1}$$

where $\vec{v}$ denotes the total velocity, $\mathbf{K}$ is the permeability, p is the pressure, q is the volumetric well rate, and λ_t is the total mobility, which in this setting is the sum of the water and oil mobility functions,

$$\lambda_t(s) = \lambda_w(s) + \lambda_o(s) = k_{rw}(s)/\mu_w + k_{ro}(s)/\mu_o. \tag{3.2}$$

The saturation equation is given by

$$\phi\frac{\partial s}{\partial t} + \nabla\cdot f_w(s)\,\vec{v} = q_w, \tag{3.3}$$

where ϕ is the porosity, s is the saturation, $f_w(s)$ is the *water fractional flow* which is defined as $\frac{\lambda_w}{\lambda_w+\lambda_o}$, and q_w is the volumetric water rate at the well.

In this work we solve the pressure and saturation equations sequentially, and after spatial and temporal discretizations, the pressure equation, can be written as

$$\mathbf{A}(\mathbf{s}^{n-1})\mathbf{p}^n = \mathbf{B}\mathbf{u}^n, \tag{3.4}$$

where the matrix $\mathbf{A}\left(\mathbf{s}^{n-1}\right)$ represents the two-point flux transmissibilities Aziz & Settari [1979], and depend on the previous time step water saturation $\mathbf{s}^{n-1}$. Furthermore $\mathbf{p}$ is the vector of grid block pressures, $\mathbf{u}$ is the control input which contains either well rates or BHPs, and $\mathbf{B}$ is the arrangement matrix for the control input. The superscript n indicates discretized time. The saturation equation is discretized using an implicit finite volume scheme

$$\mathbf{s}^n = \mathbf{s}^{n-1} + \triangle t^n \mathbf{D}_{PV}^{-1}\left(\mathbf{R}\left(\mathbf{v}^n\right) f_w\left(\mathbf{s}^n\right) + \mathbf{q}\left(\mathbf{v}^n\right)_+\right), \tag{3.5}$$

where $\triangle t^n$ is the time step and $\mathbf{D}_{PV}$ is the diagonal matrix containing the grid block pore volume. The matrix $\mathbf{R}(\mathbf{v}^n)$ is the sparse flux matrix based on an upstream weighted discretization scheme, and $\mathbf{q}(\mathbf{v}^n)_+$ is the vector of positive sources (in this setting, water injection rates).

The discrete state equations (3.4) and (3.5) can be rewritten in an implicit form $F(\widetilde{\mathbf{x}}, \widetilde{\mathbf{u}}) = 0$ as

$$\begin{aligned} \mathbf{F}(\widetilde{\mathbf{x}}, \widetilde{\mathbf{u}}) &= \begin{pmatrix} \mathbf{F}^1\left(\mathbf{p}^1, \mathbf{s}^0, \mathbf{s}^1, \mathbf{u}^1\right) \\ \vdots \\ \mathbf{F}^N\left(\mathbf{p}^N, \mathbf{s}^{N-1}, \mathbf{s}^N, \mathbf{u}^N\right), \end{pmatrix} \\ \mathbf{x}^{nT} &= (\mathbf{p}^{nT}, \mathbf{s}^{nT}), \quad n = 1, ..., N, \\ \widetilde{\mathbf{x}}^T &= (\mathbf{x}^{1T}, ..., \mathbf{x}^{NT}), \\ \widetilde{\mathbf{u}}^T &= (\mathbf{u}^{1T}, ..., \mathbf{u}^{NT}). \end{aligned} \tag{3.6}$$

An initial condition on the saturations, i.e. $\mathbf{s}^0$, is also needed. The state vectors and control input vectors are stacked for all time instances from $n = 1, \ldots, N$. Both producer wells and injector wells are covered by this formulation. We refer to equation (4.9) as the *forward model* of the oil reservoir model.

3.3 Problem Formulation

An oil reservoir should be exploited by maximizing its value while honoring all key constraints. This goal is usually manifested as an objective function in production optimization problems. In oil reservoirs this can be Net Present Value (NPV), or other key performance indices such as recovery factor, sweeping efficiency or accumulated oil production. The production optimization problem can be cast as a nonlinear programming problem, which is described as follows;

$$\begin{aligned} (\mathscr{P}) \quad & \max_{\tilde{\mathbf{u}} \in \mathbb{R}^{n_{\tilde{\mathbf{u}}}}} \mathscr{J}(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}) \\ \text{subject to}: \quad & \mathbf{F}(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}) = 0 \\ & g(\mathbf{u}^n) \geq 0, \ \forall n = 1, \ldots, N \\ & h(\mathbf{x}^n, \mathbf{u}^n) \geq 0, \ \forall n = 1, \ldots, N \\ & \mathbf{x}^0 \text{ is given.} \end{aligned}$$

$\mathscr{J}(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})$, $g(\mathbf{u}^n)$, $h(\mathbf{x}^n, \mathbf{u}^n)$ are assumed $\mathscr{C}^1$. The control input and the output constraints are represented by $g : \mathbb{R}^{n_{\tilde{\mathbf{u}}}} \to \mathbb{R}^{n_g}$ and $h : \mathbb{R}^{n_{\tilde{\mathbf{x}}}} \times \mathbb{R}^{n_{\tilde{\mathbf{u}}}} \to \mathbb{R}^{n_h}$, respectively. The objective function is given by $\mathscr{J} : \mathbb{R}^{n_{\tilde{\mathbf{x}}}} \times \mathbb{R}^{n_{\tilde{\mathbf{u}}}} \to \mathbb{R}$ and the state equations are posed as implicit constraints. The state variables and the control inputs are dependent variables, therefore we are able to perform the optimization in the control input space of $\tilde{\mathbf{u}}$ instead of

in the space of $(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})$. To this end, we denote the objective objective as $\mathscr{J}(\tilde{\mathbf{u}})$ omitting $\mathscr{J}(\tilde{\mathbf{x}}(\tilde{\mathbf{u}}), \tilde{\mathbf{u}})$. In this paper we use NPV as the objective function

$$\begin{aligned} \mathscr{J}(\tilde{\mathbf{u}}) &= \sum_{n=0}^{N-1} \left[\sum_{j=1}^{\mathscr{N}_{prod}} \left(r_o q_{o,j}^n - r_w q_{w,j}^n \right) - \sum_{l=1}^{\mathscr{N}_{inj}} \left(r_{inj} q_l^n \right) \right] \frac{\triangle t^n}{(1+d)^{t_n}} \\ &= \sum_{n=0}^{N-1} \mathscr{J}^n, \end{aligned} \tag{3.7}$$

where r_o, r_w, r_{inj} represent the oil price, the water separation cost, and the water injection cost, respectively. The well rate at injector wells is denoted by q_l, the water rate at producer wells q_w, the oil rate at producers q_o, d is the discount factor, $\triangle t^n$ is the time interval, and N is the total simulation time. In addition, $\mathscr{N}_{prod}, \mathscr{N}_{inj}$ denote the number of producer and injector wells. Note that in the special case when d = 0 and water injection and separation costs are set to zero, the NPV is proportional to the recovery factor in the sense that the optimal control inputs are equal.

The control input, either BHP or well rate, is always bounded. BHP is constrained by well and reservoir conditions, for example, a requirement to keep flowing BHP above the dew point. This leads to constraints on $g(\mathbf{u}^n)$. Moreover, since we assume the flow is incompressible, this also introduces a constraint since the total injection rate must equal the total production rate at all times. A common control input setting in reservoir management is BHP control at producer wells and well-rate control at injector wells. However, in this study we will use only well rate-controlled wells.

The output constraints might have different guises. We classify the constraints into two classes. One class deals with one dimensional output constraints ($n_h = 1$) and another class poses multidimensional constraints, $n_h > 1$. Examples of the one dimensional class are minimum total oil production, maximum water production, hierarchial optimization as described in van Essen et al. [2010], or other approaches where the output constraints are projected onto one measure. The multidimensional class appears when several output constraints are treated individually. Examples are water cut or water saturation constraints at each producer well.

Since we use gradient-based optimization, the adjoint method is used to compute gradients with respect to the objective function $\mathscr{J}(\tilde{\mathbf{u}})$. In this method, an augmented objective function (or Hamiltonian) $\mathscr{H} = \sum \mathscr{L}^n$, where $\mathscr{L}^n$ is defined as

$$\mathscr{L}^n\left(\mathbf{x}^n, \mathbf{x}^{n-1}, \mathbf{u}^n\right) = \mathscr{J}^n\left(\mathbf{x}^n, \mathbf{u}^n\right) + \boldsymbol{\lambda}^{nT} F\left(\mathbf{x}^n, \mathbf{x}^{n-1}, \mathbf{u}^n\right) \quad \text{for } n = 1, \ldots, N. \tag{3.8}$$

Taking the derivative of the augmented objective function, $\mathscr{H}$, with respect to the state variables, $\mathbf{x}^n$, leads us to the adjoint equation

$$\left(\frac{\partial F\left(\mathbf{x}^n,\mathbf{x}^{n-1},\mathbf{u}^n\right)}{\partial \mathbf{x}^n}\right)^T \boldsymbol{\lambda}^n + \left(\frac{\partial F\left(\mathbf{x}^{n+1},\mathbf{x}^n,\mathbf{u}^{n+1}\right)}{\partial \mathbf{x}^n}\right)^T \boldsymbol{\lambda}^{n+1} \tag{3.9}$$
$$= -\left(\frac{\partial \mathscr{J}^n(\mathbf{x}^n,\mathbf{u}^n)}{\partial \mathbf{x}^n}\right)^T \quad \text{for } n = N,\ldots,1,$$

with $\boldsymbol{\lambda}^{N+1} = \mathbf{0}$. The adjoint equation above is a linear equation where the solution are the Lagrange multipliers $\boldsymbol{\lambda}^n$. We then use the Lagrange multipliers to compute the gradient with respect to the controls $\tilde{\mathbf{u}}$, which is

$$\nabla_{\tilde{\mathbf{u}}}\mathscr{L}^n = \frac{\partial \mathscr{J}^n(\mathbf{x}^n,\mathbf{u}^n)}{\partial \mathbf{u}^n} + \boldsymbol{\lambda}^{nT}\frac{\partial F\left(\mathbf{x}^n,\mathbf{x}^{n-1},\mathbf{u}^n\right)}{\partial \mathbf{u}^n}. \tag{3.10}$$

Implementing the adjoint equations correctly can be cumbersome. Hence, to validate the adjoint implementation, we have compared the computed gradient to the gradient obtained from a finite difference method (for a few control inputs). An example of the validation check is shown in Figure 3.1.

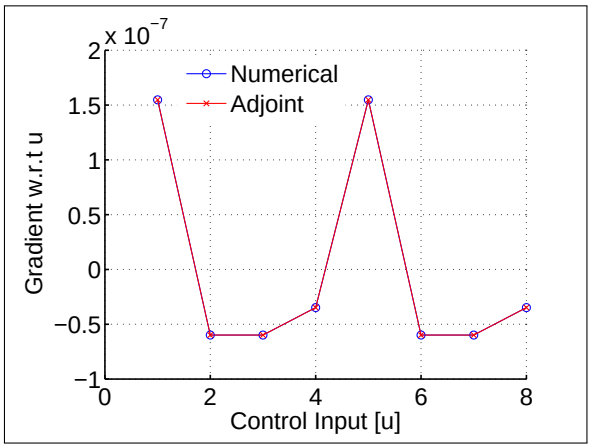

Figure 3.1: An example of gradient comparison between the numerical and adjoint gradients. The numerical gradient is computed using forward finite difference with relative perturbation size 10^{-5}.

As seen, in order to compute the derivative $\nabla_{\tilde{u}}\mathscr{L}$ of an objective function with respect to the decision variables, the adjoint method requires the partial derivatives; $\frac{\partial \mathbf{F}^n}{\partial \mathbf{x}^n}$, $\frac{\partial \mathbf{F}^{n+1}}{\partial \mathbf{x}^n}$, $\frac{\partial \mathbf{F}^n}{\partial \mathbf{u}^n}$, and $\frac{\partial \mathscr{J}^n}{\partial \mathbf{u}^n}$. Since we use a simplified model in this work, we derive all the partial derivatives analytically. For a general simulator, however, this can be cumbersome. As noted in Jansen [2011]; Sarma et al. [2008] the partial derivative $\frac{\partial \mathbf{F}^n}{\partial \mathbf{x}^n}$ is exactly what is used in the Newton-Raphson iteration of a fully implicit simulator, so in this

case these matrices can be stored during the forward simulation, and reused in the adjoint simulation. Alternatively, techniques such as *Automatic Differentiation* (AD) (see e.g., Griewank & Walther [2008]), can be utilized to compute the derivatives efficiently. The computational bottlenecks are in the linear solvers for the pressure equation and Lagrangian multipliers. They account for more than 80 percent of the runtime in the examples we consider.

3.4 Solution Method

In this section we will explain in detail how to the handle the output constraints. We begin by explaining the motivation behind the barrier methods. Subsequently, we will discuss the methods for handling the control input and output constraints. Finally, we shall present some methods for performance benchmark purposes.

3.4.1 Motivation

To solve the optimization problem $\mathscr{P}$, the Lagrangian barrier method, as in Conn et al. [1997], uses the following barrier function

$$\Psi\left(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}, \mu, \boldsymbol{\lambda}_h\right) = \mathscr{J}\left(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}\right) + \mu \sum_{i=1}^{m} \lambda_{h,i} \log\left(h_i\left(\tilde{\mathbf{x}}\right) + \mu \lambda_{h,i}\right), \tag{3.11}$$

where $\boldsymbol{\lambda}_h$ is the Lagrangian multiplier estimates, $s_{h,i} = \mu\lambda_{h,i}$ is the shift parameter which is positive, and μ denotes the barrier parameter. The (primal) barrier method, as in Fiacco & McCormick [1968], uses the barrier function

$$\Psi\left(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}, \mu\right) = \mathscr{J}\left(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}\right) + \mu \sum_{i=1}^{m} \log\left(h_i\left(\tilde{\mathbf{x}}\right)\right). \tag{3.12}$$

The differences between the barrier and Lagrangian barrier methods are the shift vector and Lagrangian multiplier estimates. The shift parameter allows the Lagrangian barrier to be started from an infeasible initial solution. However, the shift will relax the output constraint, hence the solution may become infeasible. On the other hand, the barrier method must be initiated from a feasible initial solution. Furthermore, as seen in (3.12), the barrier function treats all the output constraints equally, while the Lagrangian barrier method uses the Lagrangian multiplier estimates to differentiate each of the path constraints. This provides a potential benefit in the case of multidimensional path constrained problems.

In addition to the barrier methods, penalty methods are also possible to use to solve optimization problem $\mathscr{P}$. Referring to the penalty method in Fiacco & McCormick

[1968], the quadratic penalty function is

$$\Psi(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}, \nu) = \mathscr{J}(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}) - \frac{1}{\nu} \sum_{i=1}^{m} (h_i(\tilde{\mathbf{x}}))^2 . \tag{3.13}$$

Further, the augmented Lagrangian function, which is an enhancement of the classical penalty method, has the following form

$$\Psi(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}, \nu, \mathbf{s}_h) = \mathscr{J}(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}) - \frac{1}{\nu} \sum_{i=1}^{m} \left(h_i(\tilde{\mathbf{x}}) + s_{h,i}\right)^2 . \tag{3.14}$$

Note that the ν in the penalty methods is referred to as the penalty parameter. In both types of methods, penalty and barrier, a solution is found as $\mu \to 0$ and $\nu \to 0$. Further, the penalty methods have no requirement to start from a feasible set. However, both of the methods have the well-known ill-conditioned Hessian matrix. We explain this by studying (3.12) closer. The method approaches a solution of $\tilde{\mathbf{u}}(\mu) \to \tilde{\mathbf{u}}^*$ as $\mu \to 0$. Furthermore, if we define $r_i(\mu) \equiv \mu / h_i(\tilde{\mathbf{x}})$, we end up with a Hessian matrix

$$\mathbf{H} = \nabla^2 \mathscr{J}(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}) + \sum_{i=1}^{m} r_i \nabla^2 h_i(\tilde{\mathbf{x}}) - \frac{1}{\mu} \sum_{i=1}^{m} r_i^2 \nabla h_i(\tilde{\mathbf{x}}) \nabla h_i^T(\tilde{\mathbf{x}}) .$$

Notice that if a constraint is active at the solution, the corresponding r_i is non-zero, that will result in a condition number r_i^2/μ as $\mu \to 0$.

The Lagrangian barrier method, proposed in Conn et al. [1997], is a combination of barrier and penalty approaches. The method uses a barrier function and a penalty parameter, and allows one to start from an infeasible initial solution. The method has several advantages. First, the Lagrangian barrier method is able to asymptotically identify inactive constraints. This can be done by utilizing the Lagrangian multiplier estimates, that is, if any value of them is nonzero it implies an active constraint. Second, the method is less sensitive to *degenerate* problems, namely problems where the corresponding Lagrangian multipliers are zero for active constraints at an optimal solution. Finally, there are no slack variables introduced in this method which keep the number of unknown variables limited.

As mentioned above, with the presence of the shift parameter the Lagrangian barrier can be started from infeasible initial solutions. However, the choice of initial value of shift parameter is a challenge. Therefore, we follow an earlier development of barrier function classes, that is, the so-called modified barrier function (see Jittorntrum & Osborne [1980]). The details of this method will be explained in the subsequent subsection. The idea to use the barrier method is that, in principle, we want to keep the solution feasible. In addition, the modified barrier function has the other nice properties of the Lagrangian barrier mentioned above. Furthermore, the Lagrangian multiplier estimates are also useful to prevent ill-conditioned Hessian matrices since μ is bounded away from zero.

The barrier method consists of two iteration levels; inner and outer iterations. The inner iteration deals with the control input constraints and the outer iteration treats the nonlinear output constraints. The two iteration levels will be explained in the subsections below.

3.4.2 Control Input Constraints Handling

To handle the control input constraints, which appear as equality or bound constraints, we use the projected conjugate gradient method for the equality constraints and a barrier method for the bound constraints. These methods are implemented in the optimization package KNITRO, see Byrd et al. [2006]. Apart from the control input constraints handling, most of the runtime is spent in the inner iteration. Therefore, the choice of globalization strategies is crucial. Given the current state-of-the-art of gradient-based methods, there are two major choices; line search and trust region methods.

In general, the solution of the optimal control input is iteratively computed in the following form

$$\tilde{\mathbf{u}}_{k+1} = \tilde{\mathbf{u}}_k + \alpha \mathbf{s}_k. \tag{3.15}$$

Here $\mathbf{s}_k$ is the search direction. This line search process may take many function evaluations or forward simulation runs. Among many methods, two methods for computing α are the golden ratio search method and backtracking methods.

Contrary to the line search approaches, the trust region methods iteratively find the optimal solution of control inputs using

$$\widetilde{\mathbf{u}}_{k+1} = \widetilde{\mathbf{u}}_k + \mathbf{s}_k. \tag{3.16}$$

The step $\mathbf{s}_k$ simultaneously determines the direction and step length. This might give significant reduction in the number of function evaluations. More importantly, these methods still work in the presence of indefinite of Hessian matrices Conn et al. [2000]. For these reasons, we choose a trust region method in the inner iteration, see Algorithm 1. The trust region method is presented as a macro code below.

The step $\mathbf{s}_k$ is found by solving a quadratic approximation of the objective function, which is called a trust region subproblem. We use a conjugate gradient (CG) method to solve the step $\mathbf{s}_k$, that is, Steihaug-CG [Steihaug, 1983]. Moreover, since the quadratic approximation requires second order gradient (Hessian matrix) information we use the BFGS method to numerically approximate the Hessian. Algorithm 2 describes the *standard* Steihaug-CG method. To this end, scaling the objective function or control input is important to speed up the optimization.

In brief, the trust region method measures the objective function decrease (in a minimization problem) by using a ratio ρ. If the ratio is larger than (typically) 0.75 or

Algorithm 4 Trust Region Method

1.**Initialization**:

- Set initial control input $\widetilde{\mathbf{u}}_0$ and initial trust region radius Δ_0.
- Set constants η_1, η_2, γ_1, γ_2 which satisfy $0 < \eta_1 \leq \eta_2 < 1$ and $0 < \gamma_1 \leq \gamma_2 < 1$.
- Set gradient convergence tolerance $\omega_* \ll 1$

2. For $k = 0, 1, \ldots$

- If $\left\| \nabla_{\widetilde{\mathbf{u}}_k} \mathscr{J}\left(\widetilde{\mathbf{u}}_k\right) \right\| \leq \omega_*$, then **stop**.
- Compute a step $\mathbf{s}_k$ by solving trust region sub-problem :

$$\min_{\mathbf{s}} \quad q_k(\mathbf{s}) \qquad \text{s.t} : \|\mathbf{s}\| \leq \Delta_k,$$

where $q_k(\mathbf{s}) = \nabla \mathscr{J}\left(\widetilde{\mathbf{u}}_k\right)^T \mathbf{s} + \frac{1}{2}\mathbf{s}^T \nabla^2 \mathscr{J}\left(\widetilde{\mathbf{u}}_k\right) \mathbf{s}$, a quadratic approximation of the objective function $\mathscr{J}\left(\widetilde{\mathbf{u}}_k\right)$.

- Compute a ratio ρ_k :

$$\rho_k = \frac{\mathscr{J}\left(\widetilde{\mathbf{u}}_k + \mathbf{s}_k\right) - \mathscr{J}\left(\widetilde{\mathbf{u}}_k\right)}{q_k(\mathbf{s}_k)}.$$

- Update $\mathbf{u}_k$ based on ρ_k value :

$$\widetilde{\mathbf{u}}_{k+1} = \begin{cases} \widetilde{\mathbf{u}}_k + \mathbf{s}_k & \text{if } \rho_k \geq \eta_1 \\ \widetilde{\mathbf{u}}_k & \text{if } \rho_k < \eta_1 \end{cases}.$$

- Update trust region radius :

$$\Delta_{k+1} \in \begin{cases} [\Delta_k, \infty) & \text{if } \rho_k \geq \eta_2, \\ [\gamma_2 \Delta_k, \Delta_k] & \text{if } \rho_k \in [\eta_1, \eta_2), \\ [\gamma_1 \Delta_k, \gamma_2 \Delta_k] & \text{if } \rho_k < \eta_1. \end{cases}$$

close to one, the step **s** will be updated and accordingly the trust region radius will be enlarged. If not, the ratio is larger than 0.01 (or slightly larger than 0), the step will be updated and the trust region radius will either be kept or decreased. Otherwise, if the ratio less than 0.01, the increment will be rejected and the radius will be shrunk. The parameters $\eta_1, \eta_2, \gamma_1, \gamma_2$ in Algorithm 1 control the changes of the ratio and trust region radius. The detailed description of this method can be found in Conn et al. [2000].

Algorithm 5 Steihaug Conjugate Gradient Method

1. Set $\eta_k < 1$, $\Delta_k > 0$, $\underline{\mathbf{s}}_0 = \mathbf{0}$, $\mathbf{r}_0 = -\nabla \mathscr{J}(\tilde{\mathbf{u}}_k)$, and $\mathbf{d}_0 = \mathbf{r}_0$.
2. For $i = 0, 1, \ldots$
   - If $\|\mathbf{r}_i\| \leq \eta_k \left\|\nabla \mathscr{J}(\tilde{\mathbf{u}}_k)\right\|$, then $\mathbf{s}_k = \underline{\mathbf{s}}_i$ and **stop**.
   - If $\mathbf{d}_i^T \mathbf{H} \mathbf{d}_i \leq 0$, compute τ such that $\left\|\underline{\mathbf{s}}_i + \tau \mathbf{d}_i\right\| = \Delta_k$, then $\mathbf{s}_k = \underline{\mathbf{s}}_i + \tau \mathbf{d}_i$ and **stop**.
   - $\vartheta_i = \|\mathbf{r}_i\|^2 / \mathbf{d}_i^T \mathbf{H} \mathbf{d}_i$
   - $\underline{\mathbf{s}}_{i+1} = \underline{\mathbf{s}}_i + \vartheta_i \mathbf{d}_i$
   - If $\left\|\underline{\mathbf{s}}_{i+1}\right\| \geq \Delta_k$, compute τ such that $\left\|\underline{\mathbf{s}}_i + \tau \mathbf{d}_i\right\| = \Delta_k$, then $\mathbf{s}_k = \underline{\mathbf{s}}_i + \tau \mathbf{d}_i$ and **stop**.
   - $\mathbf{r}_{i+1} = \mathbf{r}_i - \vartheta_i \mathbf{H} \mathbf{d}_i$
   - $\phi_i = \|\mathbf{r}_{i+1}\|^2 / \|\mathbf{r}_i\|^2$
   - $\mathbf{d}_{i+1} = \mathbf{r}_{i+1} + \phi_i \mathbf{d}_i$.

3.4.3 Nonlinear Constraints Handling

Now, we are in a position to explain the output constraint handling and we use the Lagrangian barrier method for this purpose. The barrier method requires us to construct a composite function consisting of the objective function and the output constraints with a barrier parameter. This parameter will vary from one iteration to the next. In the spirit of the Lagrangian barrier function of Conn et al. [1997] and Jittorntrum & Osborne [1980], we use the following function

$$\Psi\left(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}, \mu, \boldsymbol{\lambda}_h\right) = \mathscr{J}(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}) + \mu \sum_{i=1}^{m} \lambda_{h_i} \log\left(h_i(\tilde{\mathbf{x}})\right), \tag{3.17}$$

where λ_{h_i}, $h_i(\tilde{\mathbf{x}})$ are the Lagrange multiplier estimates and the output constraints, all in componentwise form, respectively. The barrier parameter is denoted by μ. The index m denotes the number of output constraints. Note that the control input constraints $g(\mathbf{u}^n)$ are not included. Further, we define the Lagrangian multiplier estimates as

$$\bar{\lambda}_{h_i}\left(\tilde{\mathbf{x}}, \lambda_{h_i}, \mu\right) := \mu \frac{\lambda_{h_i}}{h_i(\tilde{\mathbf{x}})}. \tag{3.18}$$

This identity is derived by taking the first derivative of the barrier function,

$$\nabla_{\tilde{\mathbf{u}}} \Psi\left(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}, \mu, \boldsymbol{\lambda}_{\boldsymbol{h}}\right) = \nabla_{\tilde{\mathbf{u}}} \mathscr{J}(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}) + \mu \sum_{i=1}^{m} \frac{\lambda_{h_i}}{h(\tilde{\mathbf{x}})} \nabla_{\tilde{\mathbf{u}}} h(\tilde{\mathbf{x}}). \tag{3.19}$$

We aim to estimate $\boldsymbol{\lambda}_h^*$ (the optimal Lagrangian multiplier) such that $\boldsymbol{\lambda}_h^* \approx \bar{\boldsymbol{\lambda}}_h$ and drive $\nabla_{\tilde{\mathbf{u}}}\Psi\left(\tilde{\mathbf{x}},\tilde{\mathbf{u}},\mu,\boldsymbol{\lambda}_{\boldsymbol{h}}\right)$ as close to zero as demanded by the first order necessary optimality condition. We then use Algorithm 3 to handle the output constraints.

Algorithm 6 Lagrangian Barrier Method

Step 0: **Initialization**
- Set feasible initial solution of control input $\tilde{\mathbf{u}}_0$ and positive initial multiplier $\boldsymbol{\lambda}_0$
- Set convergence tolerances:
  - gradient tolerance $\omega_* \ll 1$
  - constraint violation $\eta_* \ll 1$
  - objective function changes ϵ_*
- Set positive barrier parameter μ_0 and $\tau < 1$
- Set initial convergence tolerance: ω_0.

Perform iteration: $k = 0, 1, 2, \ldots$

Step 1: **Inner iteration**
- Find $\tilde{\mathbf{u}}_k$ such that
  $\left\|\nabla_{\tilde{\mathbf{u}}_k}\Psi\left(\tilde{\mathbf{x}}_k,\tilde{\mathbf{u}}_k,\mu,\boldsymbol{\lambda}_k\right)\right\| \leq \omega_k$

Step 2: **Test for convergence**
- If $\left\|\nabla_{\tilde{\mathbf{u}}_k}\Psi\left(\tilde{\mathbf{x}}_k,\tilde{\mathbf{u}}_k,\mu,\boldsymbol{\lambda}_k\right)\right\| \leq \omega_*$
  or $\left\|[h_i(\tilde{\mathbf{x}}_k)]_{i=1}^m\right\| \leq \eta_*$
  or $\left|\mathscr{J}_{k+1}(\widetilde{\mathbf{x}},\widetilde{\mathbf{u}}) - \mathscr{J}_k(\widetilde{\mathbf{x}},\widetilde{\mathbf{u}})\right| \leq \epsilon_*$ then **stop**.
- Check $\left\|\bar{\boldsymbol{\lambda}}_{h,k} - \boldsymbol{\lambda}_{h,k}\right\| \leq \frac{\tau_k}{\mu_k}$. If this holds continue to **Step 3**. Otherwise go to **Step 4**.

Step 3: **Update Lagrangian Multipliers**
- $\mu_{k+1} = \mu_k$
- $\omega_{k+1} = \tau_k\omega_k$
- $\boldsymbol{\lambda}_{h,k+1} = \bar{\boldsymbol{\lambda}}_h\left(\tilde{\mathbf{x}}_k,\boldsymbol{\lambda}_{h,k},\mu_{k+1}\right)$
- Continue to **Step 1**.

Step 4: **Update Barrier parameter**
- $\mu_{k+1} = \tau_k\mu_k$
- $\tau_{k+1} = \mu_k^{0.5}$
- Continue to **Step 1**.

Remarks:
- The control input constraint $g(\mathbf{u}^n)$ is handled during the inner iteration. Any kind of gradient based optimization method can be used in the inner iteration.
- When the output constraints are violated, the logarithmic term in the objective function will give a meaningless value (not a real-valued number). To mitigate

this, we use the following modified objective function

$$J_b(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}) = \begin{cases} \Psi\left(\widetilde{\mathbf{x}}_k, \widetilde{\mathbf{u}}_k, \mu, \boldsymbol{\lambda}_k\right) & , \text{if } \forall h_i > 0 \\ -\infty & , \text{if } \exists h_i < 0 \end{cases}$$

- Given the three kinds of stopping criteria in the algorithm, there exists a finite iteration number $\bar{k}$ such that the Step 3 is executed in $\bar{k}$ times and a threshold $\underline{\mu} > 0$. Therefore, the algorithm manages to prevent the ill-conditioned Hessian matrix problem.
- The value of the Lagrangian multiplier estimates λ_{h_i} gives information on which constraints are close to becoming active. This is contrary to the pure barrier method which treats all the constraints equally.
- It should be noted that the Lagrange multiplier and barrier parameter are not updated every outer iteration, as opposed to the pure barrier method described in Algorithm 4. The Lagrange multiplier is updated when the output constraint violation, which is shown in the algorithm above by the difference between Lagrange multiplier values and their estimates, is small and the barrier parameter is fixed, and the gradient tolerance is tightened. When the constraint output constraint is large, the barrier parameter is updated and the Lagrange multiplier is fixed, and the gradient tolerance is kept from the previous outer iteration. The consequence of this condition is that the Lagrangian barrier method may stop earlier than the pure barrier method since the gradient stopping criterion is not updated every outer iteration. Hence, the Lagrangian barrier method requires fewer number of inner iteration. This stopping criterion is a modification from our previous work in Suwartadi et al. [2010a].
- The algorithm enjoys a superlinear convergence rate, see Jittorntrum & Osborne [1980] and Conn et al. [1997].

3.4.4 Alternative methods

In addition to the Lagrangian barrier method, in this work we also implement the pure barrier method as well as an existing optimization option in KNITRO (SLQP). The aim is to compare their performance relative to the Lagrangian barrier method. The pure barrier method is described in algorithm 4 while the algorithm in KNITRO is based on the SQP method. The SLQP method uses a linear programming solver to detect active constraints in a way resembles the active-set method Byrd et al. [2006].

In the next section, we demonstrate and compare the algorithms for some simple reservoir management cases with output constraints.

Algorithm 7 Pure Barrier Method

Step 0: **Initialization**

- Set feasible initial solution of control input $\tilde{\mathbf{u}}_0$
- Set convergence tolerances: gradient $\omega_* \ll 1$, constraint violation $\eta_* \ll 1$, objective function changes ϵ_*
- Set positive barrier parameter μ_0 and $\tau < 1$
- Set initial convergence tolerance: ω_0.

Perform iteration: $k = 0, 1, 2, \ldots$

Step 1: **Inner iteration**

- Find $\tilde{\mathbf{u}}_k$ such that
  $\left\| \nabla_{\tilde{\mathbf{u}}_k} \Psi \left(\tilde{\mathbf{x}}_k, \tilde{\mathbf{u}}_k, \mu \right) \right\| \leq \omega_k$

Step 2: **Test for convergence**

- If $\left\| \nabla_{\tilde{\mathbf{u}}_k} \Psi \left(\tilde{\mathbf{x}}_k, \tilde{\mathbf{u}}_k, \mu \right) \right\| \leq \omega_*$
  or $\left\| [h_i(\tilde{\mathbf{x}}_k)]_{i=1}^m \right\| \leq \eta_*$
  or $\left| \mathscr{J}_{k+1} (\widetilde{\mathbf{x}}, \widetilde{\mathbf{u}}) - \mathscr{J}_k (\widetilde{\mathbf{x}}, \widetilde{\mathbf{u}}) \right| \leq \epsilon_*$ then **stop**.
- otherwise execute Step 3

Step 3: **Updates and tighten convergence tolerances**

- $\mu_{k+1} = \tau_k \mu_k$
- $\omega_{k+1} = \tau_k \omega_k$
- $\tau_{k+1} = \tau_k \omega_k$

Continue to Step 1.

3.5 Numerical Cases

In this section three cases demonstrate the proposed method for handling output constraints. In these cases the output constraints appear as total water production, water cut (water fractional flow) constraints, and an objective function constraint. The first and third cases represent one dimensional problems, while the second case demonstrates the multidimensional output constraint problem. All the cases are open loop optimization problems, that is, there is no use of real-time data for feedback purposes. To measure the performance of the proposed method, we compare the Lagrangian barrier method with a pure barrier method and a *Sequential Quadratic Programming* (SQP)-based method, SLQP, which is one algorithm option in the KNITRO solver. The pure barrier is the classical barrier method in the sense that it uses no Lagrangian multiplier estimates. The simulations for the case examples are done on a Linux 64-bit machine with CPU specification Intel Xeon(R) 3.00 GHz and 16 GB RAM. The oil reservoir models in this work are all implemented by using *MATLAB Reservoir Simulation Toolbox* [Lie et al., 2011].

3.5.1 Numerical Case 1

In this case, we constrain the maximum total water production. First, we run an optimization with maximum water production constraint for the first layer of SPE 10th Comparative Study [Christie & Blunt, 2001]. Later, we will run the optimization for 10 selected layers of the SPE 10th model. The wells follow a 5-spot pattern consisting of one injector well at the middle and four producer wells at the corners, see Figure 3.2.

The grid has 60×220 blocks where a grid block has physical dimension 10ft$\times$ 20ft $\times$ 2ft. Figure 3.2 displays a heterogeneous permeability field while the porosity, for simplicity, is set to 0.3 in all gridblocks. The water-to-oil mobility ratio is set to 5 and initial water saturation is 0.2. The objective function is *normalized* net present value (NPV) with a zero discount rate, where oil price is 1 and zero water separation cost and zero water injection cost. The control inputs are injector and producer well-rates. Note that the total injector rate must equal the total production rate due to incompressible fluids and formation. We divide the control inputs into 12 equal intervals for a 360-day simulation time. In other words, we change the well-rates every month. The initial water injection is 0.1 of total pore volume at constant rate. To prevent producer wells from becoming injector wells and vice versa we confine their values to be positive.

In this case the output constraint is maximum water production. We limit the maximum total water production to $2.24 \times 10^4\,\mathrm{m}^3$ or 0.5 of the pore volume of the reservoir, and define this as $Q_{w,max}$. The barrier term for this output constraint is

$$\psi\left(\widetilde{\mathbf{x}}, \widetilde{\mathbf{u}}, \mu, \boldsymbol{\lambda}\right) = \sum_{n=1}^{N} \mathscr{J}^n\left(\mathbf{x}^n, \mathbf{u}^n\right) + \mu\lambda \log\left(Q_{w,max} - \sum_{n=1}^{N} Q_{w,prd}^n\right). \tag{3.20}$$

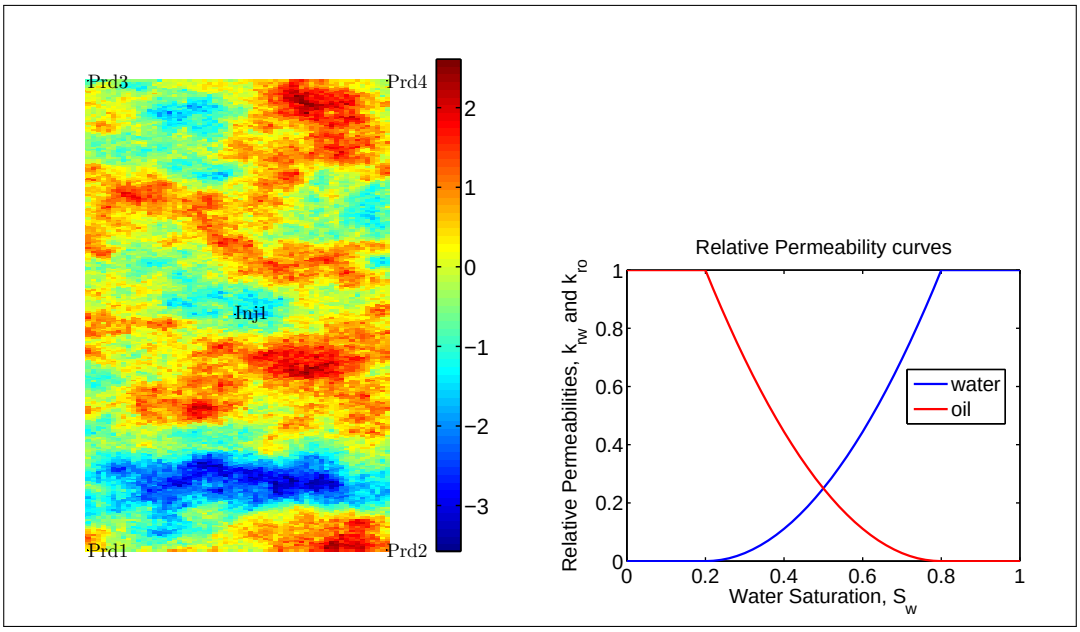

Figure 3.2: Permeability field, well location and relative permeability curves for Case 1. The color bar shows the logarithm of the permeability of layer 1 of the SPE 10 model in milidarcy. The well locations follow the 5-spot pattern in which 4 producers are placed in the corners and 1 injector in the middle.

The initial barrier parameter is set to $\mu_0 = 100$, other parameters are set as follows: $\tau = 0.1$, $\omega_0 = 0.1$, $\boldsymbol{\lambda}_0 = \mathbf{1} \in \mathbb{R}^{12\times 1}$. The stopping criteria parameters are the relative gradient tolerance $\omega_* = 10^{-4}$, absolute maximum total water rate producers tolerance $\eta_* = 10^{-6}$, and absolute objective function changes $\epsilon_* = 10^{-2}$ percent of NPV.

The optimization results for the first layer is described in Table 3.1. The barrier methods terminate due to the absolute objective function change criterion. The pure and Lagrangian barrier methods result in almost the same NPV and total water injected. Both methods require the same number of outer iteration as seen in Figure 3.3. Since the constraint violation is quite large, the Lagrangian barrier method does not tighten the gradient stopping criterion. Therefore, the method takes fewer inner iterations to converge than those of the pure barrier method. The optimized well-rate for each method is depicted in Figure 3.4. The Lagrangian barrier and pure barrier methods produce very similar optimized control inputs.

We also run optimization for various layers of the SPE 10th model. Table 3.2 summarizes the results. Similarly to the results of the first layer, the Lagrangian barrier method is the fastest in terms of CPU time. The Lagrangian and pure barrier methods achieve almost the same NPV, but the Lagrangian barrier method is slightly faster. Note that in all cases the barrier methods stop due to the objective function change

Table 3.1: Optimization results of the three methods for the first layer SPE 10th model.

Aspects	SLQP	Lag. Barrier	Pure Barrier
NPV (in 10^4)	1.69	1.73	1.73
Total Water Production ($10^4 m^3$)	2.20	2.22	2.22
CPU time (in seconds)	2260	1320	1444
Total Water Injected (in PVI*)	0.8780	0.8709	0.8709

PVI stands for Pore Volume Injected.

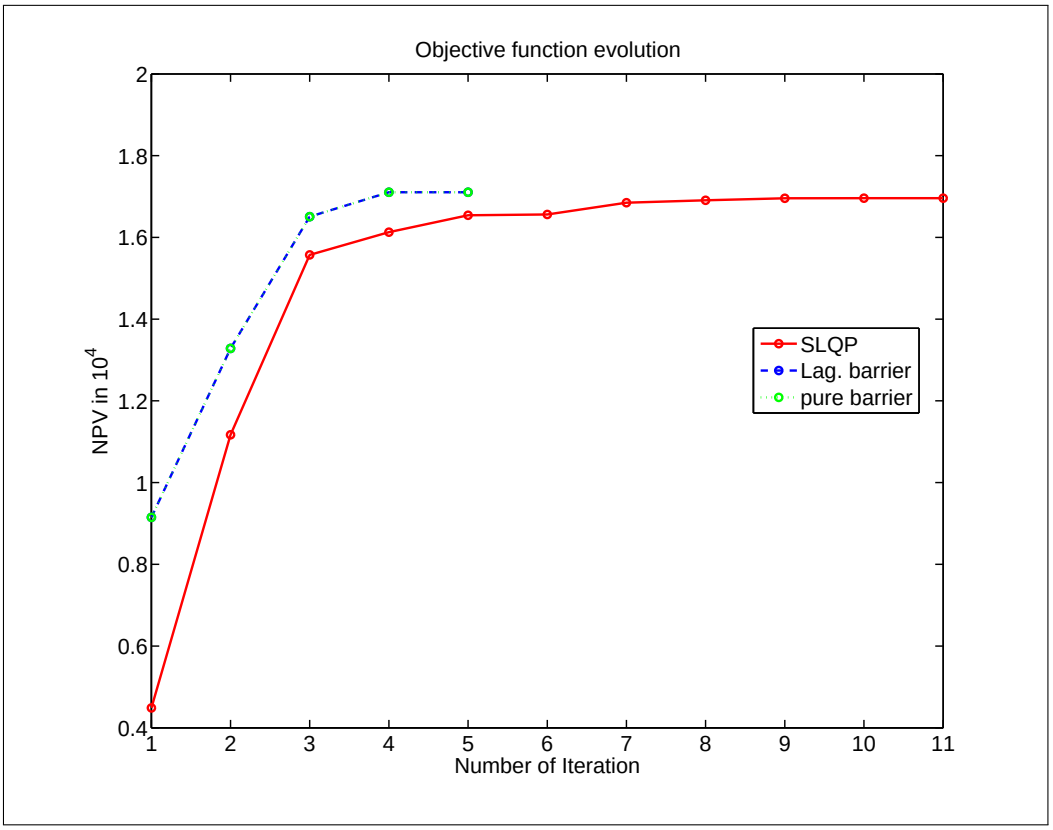

Figure 3.3: Objective function evolution for each iteration. The iterations for the barrier methods refer to outer iteration, while SLQP represents to the number of inner iteration.

criterion.

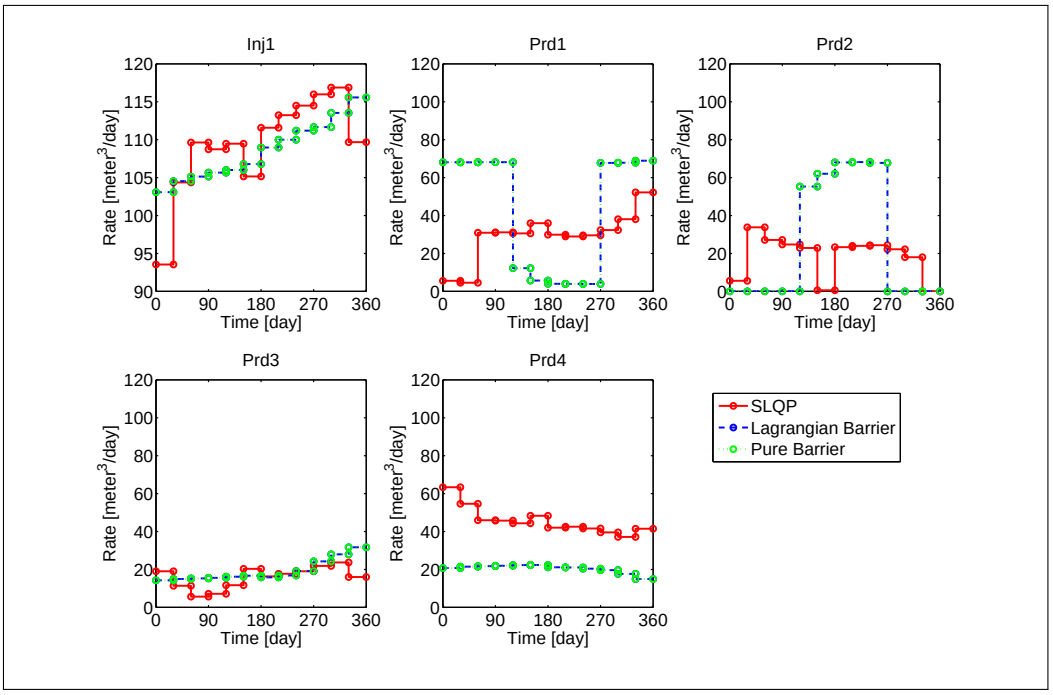

Figure 3.4: Optimized control input (well-rates) for each method. The Lagrangian barrier and pure barrier methods yield almost the same control inputs. Therefore the blue colors of the Lagrangian barrier are overlapped by the green colors of the pure barrier.

Table 3.2: Optimization results of the three methods for 10 layers of SPE 10th model. NPV is given in 10^4, TWI stands for Total Water Injected in PVI, CPU Time is in seconds.

Layer	Method								
	SLQP			Lag. Barrier			Pure Barrier		
	NPV	TWI.	CPU Time	NPV	TWI.	CPU Time	NPV	TWI.	CPU Time
5	1.62	0.86	1513	1.68	0.85	905	1.68	0.85	1001
9	1.57	0.79	589	1.63	0.84	870	1.63	0.84	924
13	1.36	0.80	995	1.41	0.78	916	1.41	0.78	1006
17	1.60	0.85	1566	1.65	0.84	833	1.65	0.84	889
21	1.51	0.84	1441	1.54	0.82	897	1.54	0.82	999
25	1.52	0.73	314	1.62	0.85	885	1.62	0.85	1059
29	1.68	0.87	1497	1.73	0.85	895	1.73	0.86	1020
33	1.45	0.82	1421	1.46	0.81	956	1.46	0.81	1036
37	1.02	0.73	1173	1.11	0.68	895	1.11	0.72	1024
41	1.01	0.72	929	1.04	0.71	950	1.04	0.71	954
Average	1.43	0.80	1144	1.48	0.80	900	1.48	0.81	991

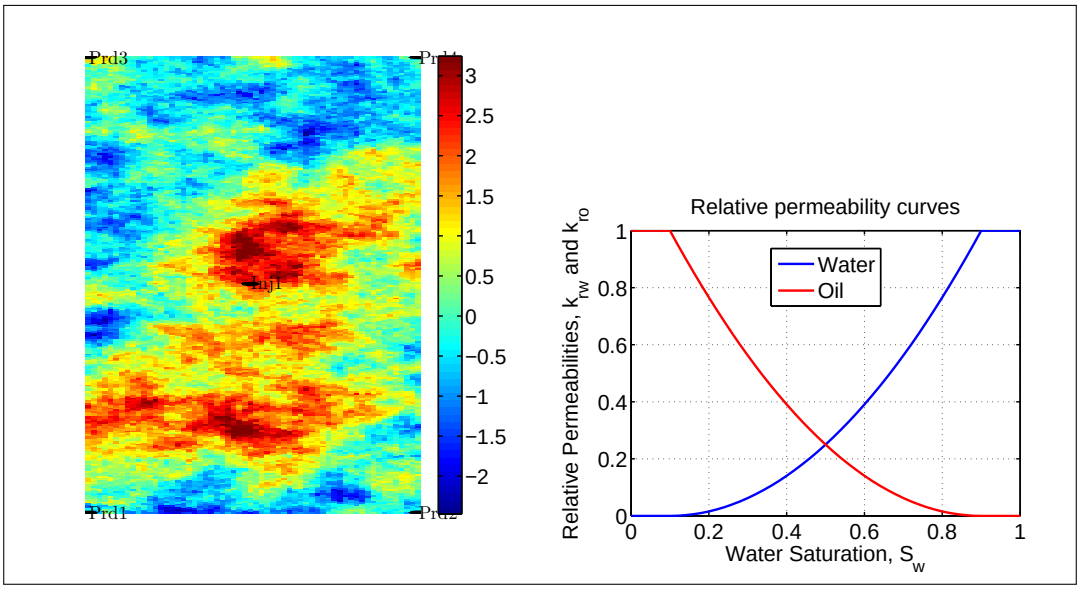

Figure 3.5: The logarithm of permeability field in milidarcy, well location and relative permeability curves for Case 2. The well locations follow the 5-spot pattern in which 4 producers are placed in the corners and 1 injector in the middle.

3.5.2 Numerical Case 2

This case is again taken from the SPE 10th Comparative Study; layer 10, as seen in Figure 3.5. The reservoir has 60×220 gridblocks, initial water saturation 0.1, connate water saturation and residual oil saturation 0.1, and water-to-oil mobility ratio 5. For simplicity, we use constant porosity 0.2. The gridblock dimensions are 10ft$\times$ 20ft $\times$ 2ft. Water cut at each producer well at the end time of the simulation (N) is the output constraint, which is described in the following barrier function

$$\Psi\left(\tilde{\mathbf{x}},\tilde{\mathbf{u}},\mu,\boldsymbol{\lambda}\right)=\sum_{n=1}^{N}\mathscr{J}^{n}\left(\mathbf{x}^{n},\mathbf{u}^{n}\right)+\mu\sum_{i=1}^{m}\lambda_{i}\log\left(f_{w,max}-f_{w\,prod,i}^{N}\right). \tag{3.21}$$

The maximum allowable water cut, $f_{w,max}$, at producer wells is set to 0.9. The objective function is recovery factor and the control inputs are well rates. The total simulation time is divided into 5 control intervals where each interval equals 100 days. In this case, there is no upper bound constraint on the control inputs.

The parameter settings in this case are: $\boldsymbol{\lambda}_0 = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}^T$, $\tau = 0.1$, $\mu_0 = 100$, $\omega_0 = 0.1$, relative gradient tolerance $\omega_* = 10^{-4}$, absolute maximum water cut tolerance $\eta_* = 10^{-6}$, and absolute objective function changes $\epsilon_* = 10^{-2}$ percent of recovery factor. The initial total injection is 0.2 of the total pore volume at constant and equal rates for all the wells. We chose an active set algorithm in KNITRO to the handle control

input constraints $g(\tilde{\mathbf{u}})$.

Table 3.3 and Figure 3.6, 3.7, and 3.8 show the performance comparison of the methods and constraints satisfaction of each method. The final barrier parameter, μ, is for both the barrier methods set to 10^{-2}. Interestingly, the Lagrangian multiplier estimates are able to asymptotically identify the active constraints by their values, which are

$$\boldsymbol{\lambda} = \begin{bmatrix} 786.2 & 29.8 & 22.3 & 8.9 \end{bmatrix}^T$$

The values imply that all constraints active.

Table 3.3: Comparison of the methods in Case 2.

Method	SLQP	Lag. Barrier	Pure Barrier
Recovery Factor (%)	52.74	56.00	56.10
CPU Time (seconds)	556	504	734

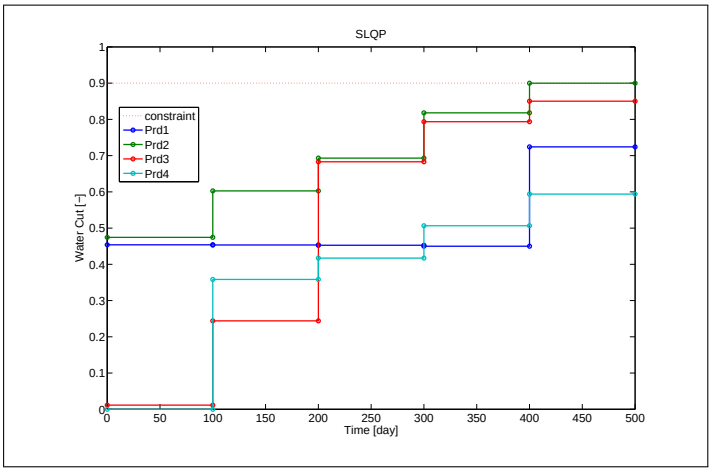

Figure 3.6: SLQP - The output constraints satisfaction. Notice that the output constraints are almost active at the final control interval, i.e., between 400 and 500 days in some producer wells

The optimized production profile is depicted in Figure 3.10. In Figures 3.6, 3.7, and 3.8, we observe that the constraints for $Prd1$ and $Prd4$ do not become active for the SLQP method. The reason probably is that, to use the SLQP algorithm in KNITRO, we aggregate the output constraints in the following way,

$$h(\tilde{\mathbf{x}}) = \sum_{i=1}^{4} \max\left(\left(f_{w,max} - f^N_{w,prd\,i}\right), 0\right), \tag{3.22}$$

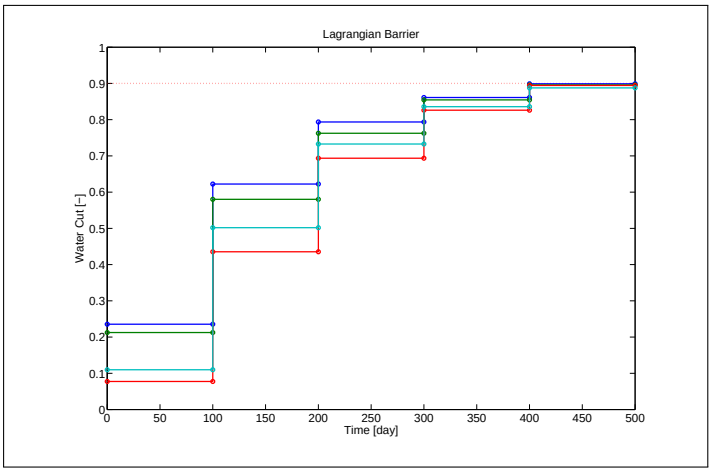

Figure 3.7: Lagrangian Barrier

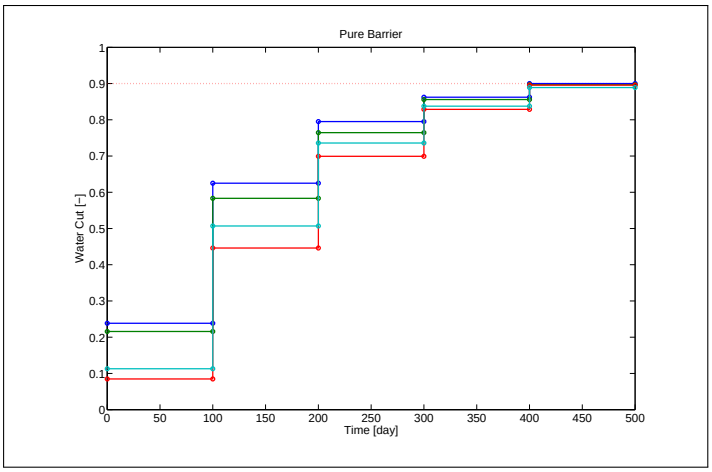

Figure 3.8: Pure Barrier

which yields non-smoothness, affecting convergence rate of the optimizer. This also effects the objective function evolution seen in Figure 3.9. Furthermore, for this particular choice of control inputs, the SLQP method gives the lowest recovery factor, see Table 3.3. The pure barrier leads to a slightly higher recovery factor than the Lagrange barrier method at the expense of a longer runtime. The reason for this is similar to the first case, that is, the Lagrangian barrier method does not tighten the gradient tolerance. As seen in Figure 3.9 both the barrier methods take the same number of outer iteration but the gradient tolerance for the Lagrangian barrier method is not changed for each inner iteration.

We then run 10 simulations with different initial values since the gradient-based

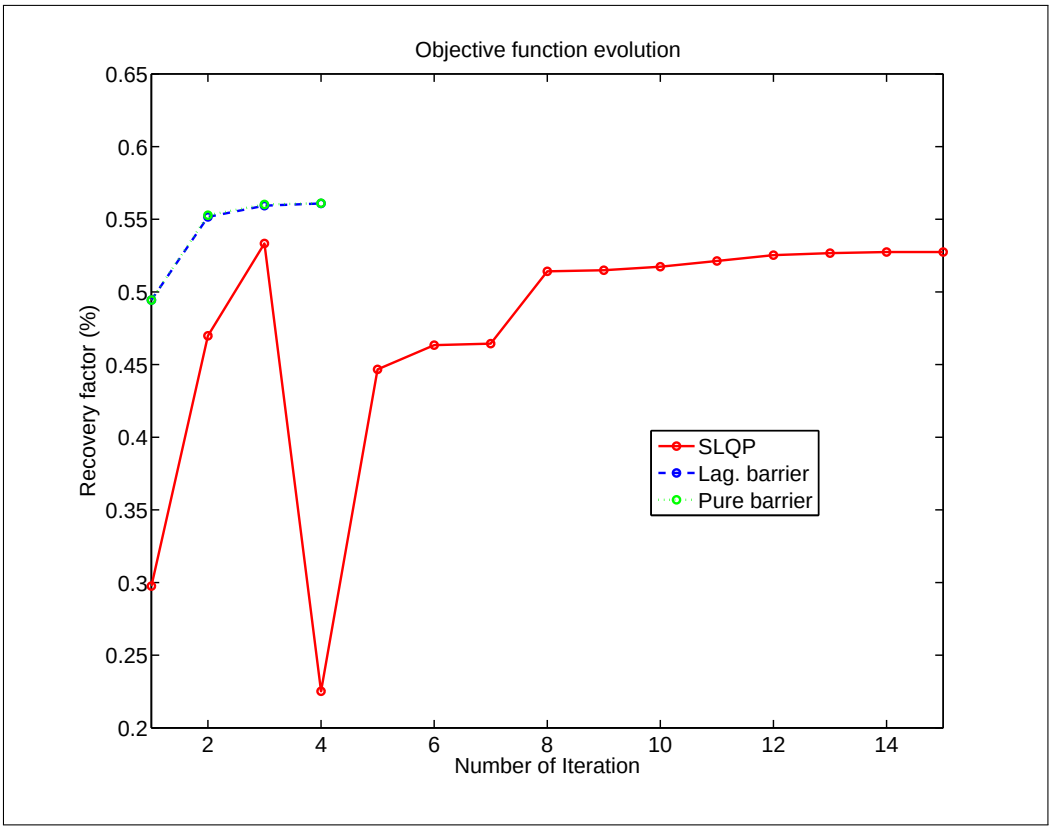

Figure 3.9: Objective function evolution for each iteration. The iterations for the barrier methods refer to outer iteration, while SLQP represents to the number of inner iteration.

optimizations are sensitive to initial solutions. Table 3.4 describes the results. On average the Lagrangian barrier method gives almost identical results as the pure barrier method but with a 29% runtime reduction.

Table 3.4: Simulation results from 10 different initial control inputs

	SLQP			Lag. Barrier			Pure Barrier		
	min	max	average	min	max	average	min	max	average
Recovery Factor (%)	48.36	55.50	51.98	55.06	56.43	55.83	55.06	56.44	55.86
CPU time [sec]	133	627	364	188	259	225	273	427	319

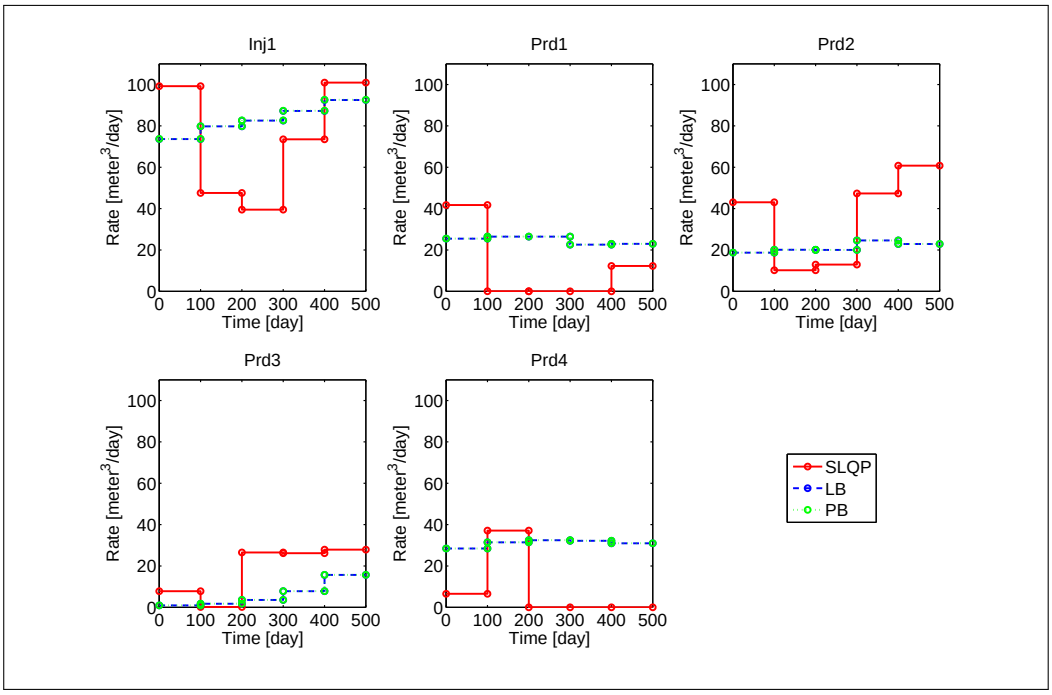

Figure 3.10: Comparison of optimal control inputs from the methods.

3.5.3 Numerical Case 3

This case uses the objective function (NPV) as a constraint and originates from van Essen et al. [2010]. The constrained NPV is a result of an optimal control strategy which has lower value in the short term horizon compared to the NPV of reactive control strategy. Hence, a second stage optimization is needed to enhance the resulting optimal control strategy. In the second stage optimization, the optimized control inputs are re-optimized such that the NPV in the short term horizon is improved. The long term NPV value in the second stage optimization is maintained by constraining its value so that it approaches to the NPV of first stage optimization.

In addition to the use of Hessian information in van Essen et al. [2010] to solve this problem, the use of an augmented Lagrangian method has been proposed in Chen et al. [2011]. In that work, as mentioned in Section 4, the augmented Lagrangian method can be started from any initial guess. Furthermore, the authors in Chen et al. [2011] also incorporate model uncertainties by introducing an ensemble of oil reservoir realizations into the optimization algorithm. Hence, the objective function is an expected value (average) of NPV from the realizations.

We use a real reservoir geometry, from the Norne field case Rwechungura et al. [2010] as shown in Figure 3.11. The reservoir consists of $46 \times 112 \times 22$ gridblocks with 4 injector and 8 producer wells. We perform 5 years simulation divided into 20 control

intervals. Hence, in total we have 240 well-rates controlled. We use the same relative permeability in this case as in case 1. The permeability of the field is depicted in Figure 3.11 and water-to-oil mobility ratio is set to 5. The oil price is set to $\$100/m^3$, water separation cost $\$10/m^3$, and water injection cost $\$1/m^3$. In the first stage optimization the objective function is NPV with a zero discount rate and in the second stage NPV has a discount rate of 25 percent aiming to emphasize production on the short term horizon. The rates at the producer and injector wells are constrained downwards by zero, that is, producer and injector wells cannot turn into injector and producer wells, respectively.

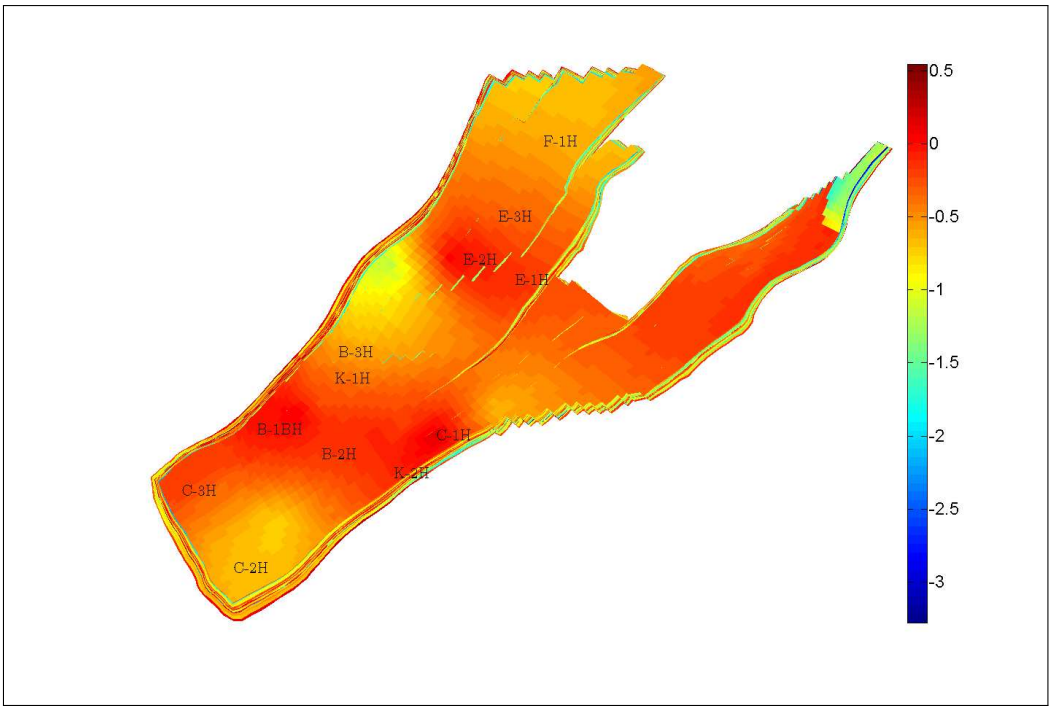

Figure 3.11: Permeability field and well location of Norne field for case 3. Color bar in the picture displays logarithm of permeability in millidarcy. Injector wells are C-1H, C-2H, C-3H, and F-1H. The remaining wells are production wells.

As seen in Figure 3.12, the reactive control results in better NPV in the short term horizon while in the long run, the optimal control strategy yields a significantly higher NPV. The output constraint in this case will be defined as the long term NPV of the optimal control strategy. After finishing the long term optimization, we run the second stage optimization in order to increase NPV of the optimal control result. To summarize, the following is the optimization procedure.

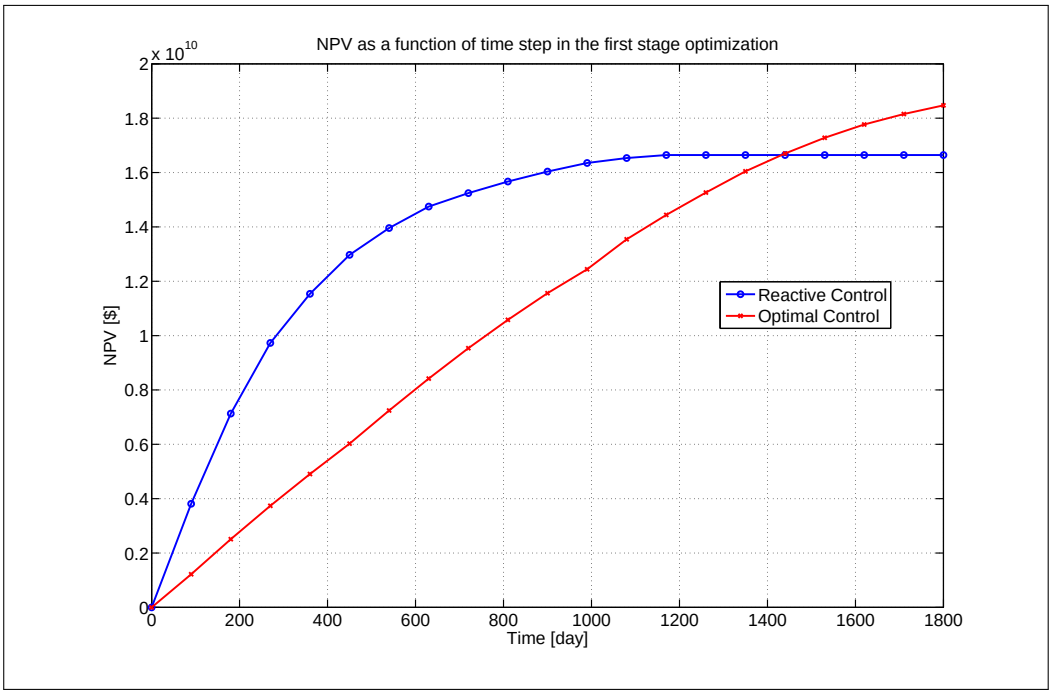

Figure 3.12: Comparison of long term optimization using optimal control and reactive control

Hence, in this case the barrier function is

$$\Psi\left(\tilde{\mathbf{x}},\tilde{\mathbf{u}},\mu,\boldsymbol{\lambda}\right)=\sum_{n=1}^{N_{short}}\mathscr{J}_s^n\left(\mathbf{x}^n,\mathbf{u}^n\right)-\mu\lambda\log\left(\left(\mathscr{J}^*-\epsilon\right)-\sum_{n=1}^{N}\mathscr{J}_s^n\left(\mathbf{x}^n,\mathbf{u}^n\right)\right). \tag{3.23}$$

The ϵ in the barrier function accounts for a small reduction of the long term NPV ($\mathscr{J}^*$) in the second stage optimization. The objective function for the second stage optimization is $\mathscr{J}_s^n$ with a non-zero discount rate.

The parameter settings in this case are: $\lambda_0 = 1$, $\tau = 0.1$, $\mu_0 = 100$, $\omega_0 = 0.1$, relative gradient tolerance $\omega_* = 10^{-8}$, absolute constraint violation $\eta_* = 10^{-6}$\$, absolute

Algorithm 8 Hierarchical Optimization Procedure

- Run the long term (or first stage) optimization to obtain: $\mathscr{J}^*$ and $\tilde{\mathbf{u}}^*$,
- Use the $\tilde{\mathbf{u}}^*$ as initial solution,
- Use the $\mathscr{J}^*$ as an output constraint,
- Set an NPV with a non-zero discount rate to emphasize the short term production horizon,
- Run the short term (second stage) optimization.

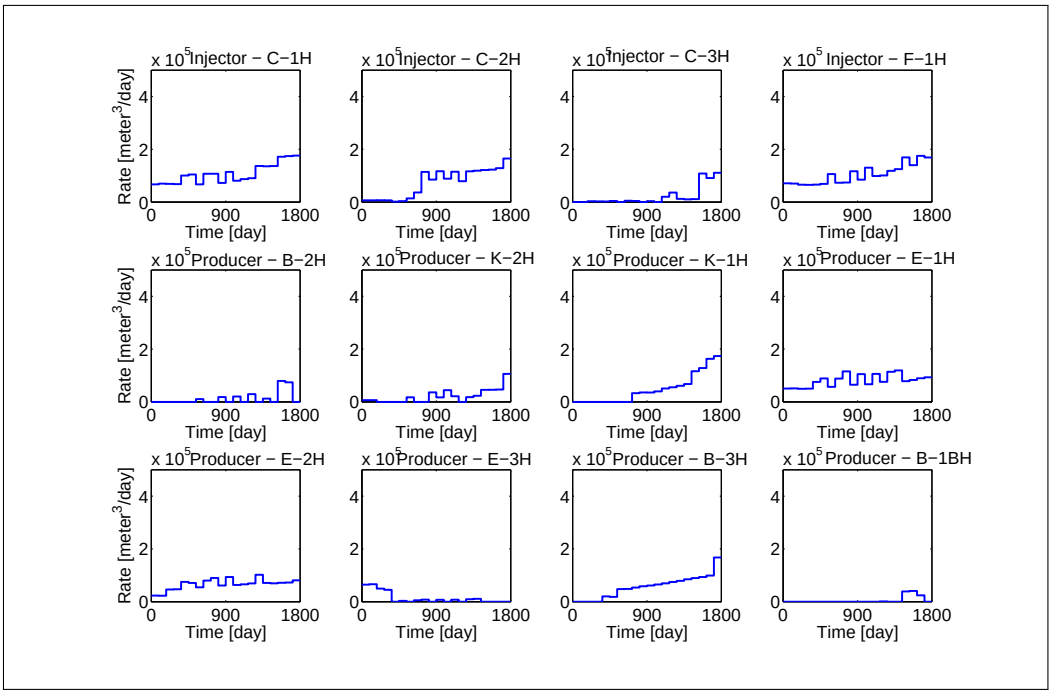

Figure 3.13: Optimized well rates for the 4 injectors and 8 producers.

objective function tolerance changes $\epsilon_* = 1000\$$, reduction of long term NPV $\epsilon = 10^9\$$.

Reactive control and an optimized production strategy are compared in Figure 3.12. In the reactive control case equal and constant injection rates are used. A producer well is shut in when the water cut exceeds 0.9. The optimized production strategy, shown in Figure 3.13, is in this case computed using the SQP method; with a discount factor of zero. The result indicates a significant potential for optimal control.

Next we apply the three optimization methods on the short optimization problem. The time horizon for the short optimization problem is the same as the first stage optimization. The results are depicted in Figure 3.14, 3.16 and Table 3.5. As seen in the table, the pure barrier and SLQP methods yield the highest NPV. Interestingly, these methods lead to 2.7% increasement of long term NPV. The Lagrangian barrier method results in the fastest CPU times but with the lowest long term NPV. The evolution of the objective function changes is shown in Figure 3.15. The Lagrangian barrier terminates just after two outer iterations. This is due to the unchanged value of the gradient stopping criterion. Hence, the method requires a lower number of inner iterations. As a result, the objective function at the second outer iteration is almost the same as the previous iteration. Then, the method stops because of the objective function changes criterion. The pure barrier method, with the gradient tolerance changed every outer iteration, results in better objective function value but takes more inner iterations to

converge.

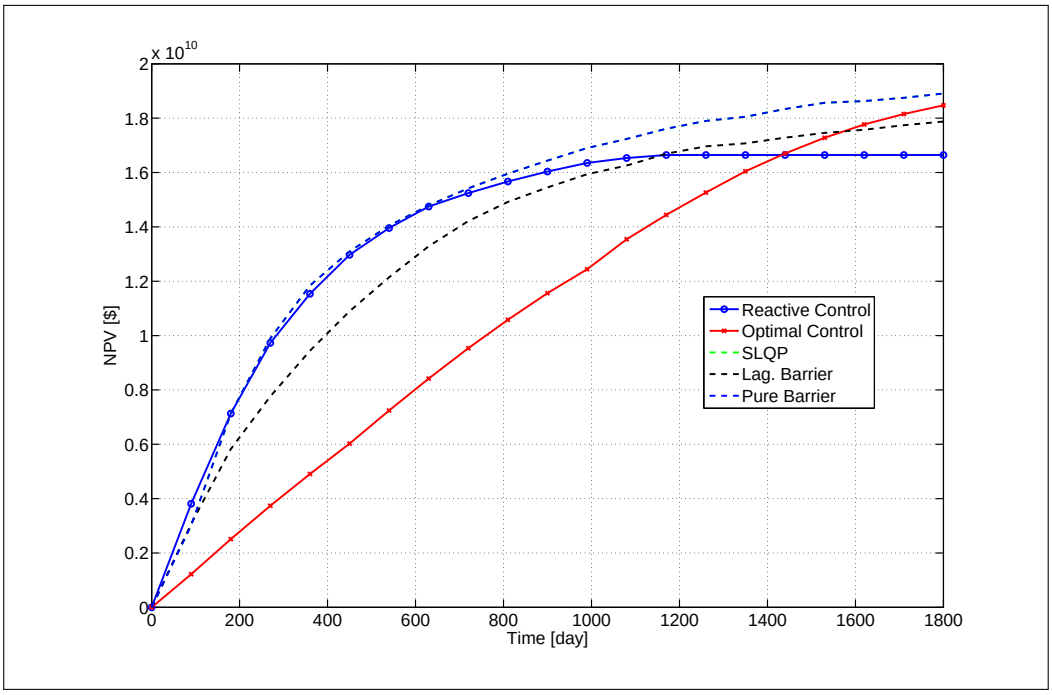

Figure 3.14: Hierarchial optimization enhancing the short term optimal control strategy. The SLQP and pure barrier methods have almost the same NPV, therefore the green dash line of SLQP is overlapped by the blue dash line of pure barrier.

Table 3.5: Comparison of the short term NPV

Method	SLQP	Lag. barrier	Pure barrier
NPV (in $\$10^{10}$)	1.89	1.79	1.89
CPU Time (in minutes)	433	97	540

In this case we have deliberately chosen initial values in which the short term optimization result is below the reactive control NPV for the same horizon. This is reasonable since this situation may well occur due to the fact that there are multiple local solutions of this problem.

3.5.4 Discussion of results

Three cases have been presented. As described, the barrier methods need proper initial parameters, for example the barrier parameter, μ. For the Lagrangian barrier method,

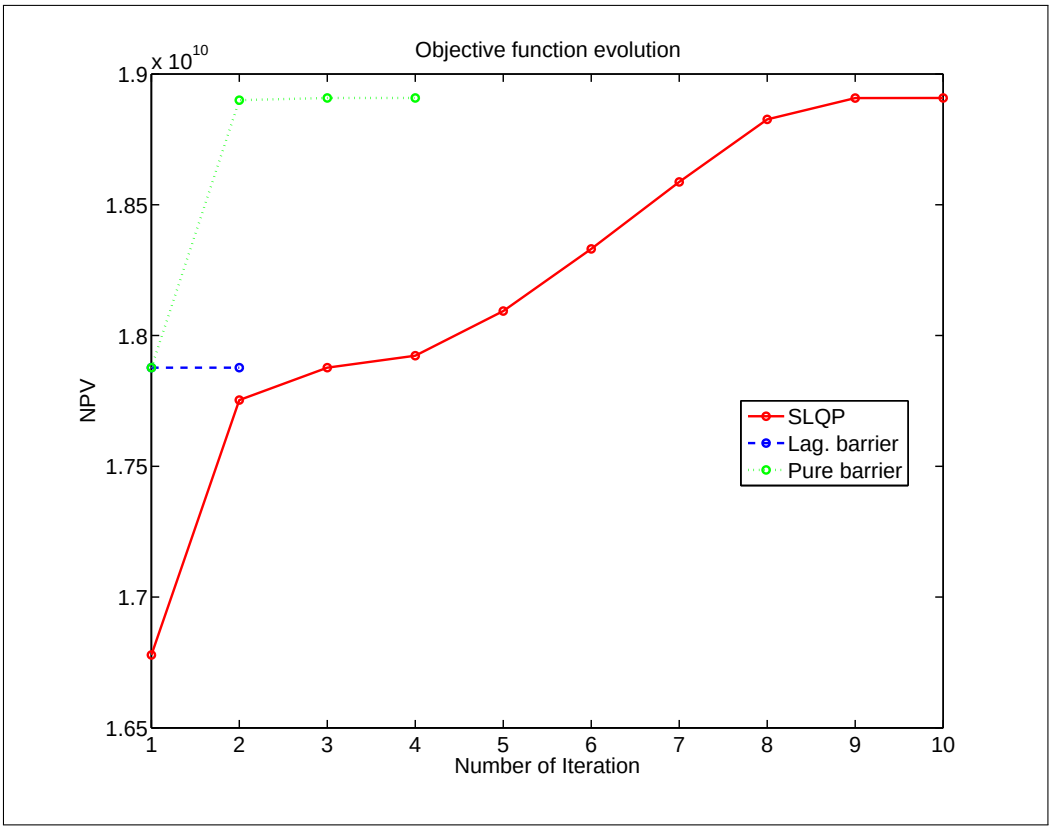

Figure 3.15: Objective function evolution for each iteration. The iterations for the barrier methods refer to outer iteration, while SLQP represents to the number of inner iteration.

we need also to supply initial Lagrangian barrier estimates. This did not complicate matters in the two earlier examples. In the latter case however it seems like more care needs to be taken in order to initialize the Lagrangian barrier method. For the choice of parameter values used in the two first case examples, the barrier methods seem robust to a variety of initial values of control inputs. However, we have not presented the effect of varying the initial parameter values to the optimization performance. This will be subject for future research. Furthermore, the stopping criteria is also an important consideration. All the barrier methods in the case examples terminate due to the objective function changes criteria.

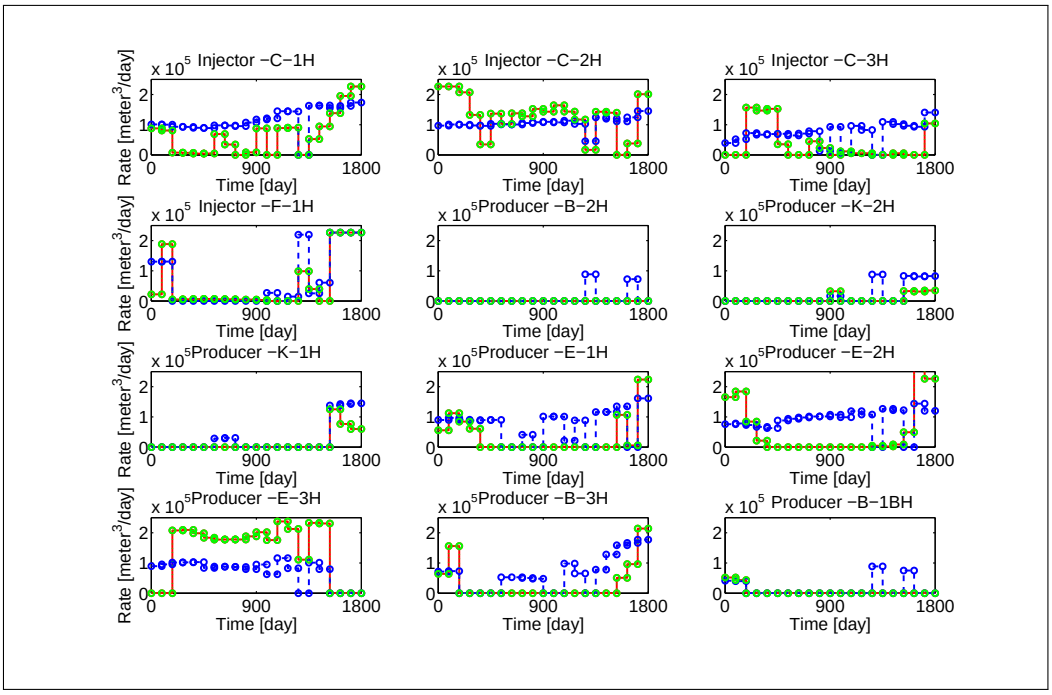

Figure 3.16: Comparison of the methods, SLQP, Lagrangian barrier, and pure barrier. The colors represent; SLQP-red, pure barrier-green, and Lagrangian barrier-blue. Notice that pure barrier and SLQP have almost the same optimized control inputs. Therefore the red color is overlapped by the blue color.

3.6 Conclusion

In this paper we have proposed the use of a Lagrangian barrier method for the handling of output constraints, where the dimensions of the output constraints appear in one and multidimensional cases. The proposed method is compared to a pure barrier and SLQP methods. The proposed Lagrangian barrier method is able to honor the nonlinear output constraints. The performance of the Lagrangian and barrier methods are very similar for the first two case examples presented, while for the third case they are slightly different. In the examples presented, the Lagrangian barrier method required fewer function evaluations than the other two approaches, and hence gave a faster CPU time. Furthermore, the use of Lagrangian multiplier estimates give information on which constraints are active.

The assumption of a feasible initial guess is needed for the use of the Lagrangian barrier method. We have used the modified barrier function, that is, without a shift parameter. This shift parameter allows an infeasible initial guess. An idea for future direction is to use the shift parameter when the solution is infeasible and set the shift

parameter to zero when the solution is feasible.

Chapter 4

Second-Order Adjoint-Based Control for Multiphase Flow in Subsurface Oil Reservoirs

A short version of this chapter was presented in Suwartadi et al. [2010c].

This paper presents an efficient way to compute second-order gradients by using the adjoint method for PDE-constrained optimization. The gradient thus obtained is used in an optimization algorithm. We propose a conjugate gradient approach combined with the trust-region method, which, ultimately, may exhibit a quadratic convergence like the Newton's method. Furthermore, we compare the proposed algorithm to a quasi-Newton method (BFGS), and apply the method for production optimization in two-phase oil reservoirs. Two numerical cases are presented, showing that our proposed method requires fewer function and gradient evaluations than the quasi-Newton approach even though the CPU time does not decrease.

4.1 Introduction

Oil reservoirs are modeled by partial differential equations (PDEs) and are implemented as a reservoir simulator. The models are used by reservoir engineers to get insight into the physics and to help deciding reservoir management strategy. Due to complex time-dependent processes and uncertainties in rock and fluid properties of subsurface reservoirs, the reservoir simulator may not accurately describe the physics. Hence, one-best-effort model is not preferable rather a set of ensemble models are usually employed, representing spread of the uncertainties. The reservoir management decisions include how to sweep remaining oil efficiently (production optimization), use

of the models for future asset calculation (parameter estimation/reservoir conditioning), and determine where a injector or producer well should be placed. These types of decisions can be regarded as numerical optimization problems.

PDE-constrained optimization is currently an active research field. Books of Hinze et al. [2009] and Troltzsch [2010] give details introduction and applications of the optimization. In the literature, one can observe two different approaches. The first approach is to use derivative-free methods, for instance Hooke-Jeeves's direct-search [Hooke & Jeeves, 1961] and genetic algorithm [Goldberg, 1989]. This approach treats the reservoir simulator as a black box and gives an ease of implementation. However, for large-scale systems such as oil reservoir, which is discretized into 10^4 - 10^6 grid blocks, will result in many simulation runs which consequently be prohibitive in term of CPU time. The second approach is gradient-based optimization. From optimal control theory perspective, the adjoint method is readily available for computing the gradient. Although the implementation is a challenging task, at the end the optimization will be more computationally efficient. The adjoint method is focus of the topics in the books [Hinze et al., 2009; Troltzsch, 2010]. Production optimization of oil reservoir in PDE-constrained optimization is seen as an example of *boundary control* problem since the control inputs are either well-rate or bottom hole pressure at the producer wells. A well in oil reservoir model is a boundary condition of the PDEs. While parameter estimation for oil reservoir, which is known as history matching by reservoir engineers, is a type of *distributed control*. The parameters (rock or fluid properties) are assigned to each and every grid block of reservoir model.

In the reservoir simulation literature, the use of adjoint method for production optimization has been started since early 1980s see for example Asheim [1986]. The method has been revived in Brouwer & Jansen [2004] and recently reviewed in Jansen [2011]. The current state-of-the-art industrial oil simulator, such as Eclipse E300 Schlumberger [2009] provides the ability for computing gradient using the adjoint methods. The simulator also incorporates steepest descent and conjugate gradient optimizers. The adjoint-gradient that is supplied to the optimizers is first-order gradient. In the research literature, one can find the use of quasi-Newton, e.g., BFGS method used in order to speedup the optimization.

In this work we are interested in the use of second-order gradient or the Hessian. The development of the Hessian matrix in the framework of adjoint methods can be traced to Raffard & Tomlin [2005], where an auxiliary convex quadratic optimization was used to compute the Newton step. In that work, the Hessian matrix was not computed explicitly. In Raffard et al. [2008], the use of finite difference methods to compute the Hessian was proposed. Similarly, van Essen et al. [2010] presented a hierarchical optimization approach for oil reservoirs that used the central finite difference method to estimate the Hessian. In the optimization community, an analysis of the Newton method was presented by, among other works, Ito & Kunisch [2000]. In Heinkenschloss

[2008] an inexact Newton method was applied, that is, Truncated Newton or Newton Conjugate Gradient, to a PDE-constrained optimization.

In this paper, in the spirit of Ito & Kunisch [2000] and Heinkenschloss [2008], we implement the adjoint method for the Hessian matrix. A Hessian-times-vector, as in Heinkenschloss [2008], is derived for production optimization of an oil reservoir model. In addition, we use the conjugate gradient algorithm, following Steihaug [1983], to perform the optimization using the gradients (first and second order) from the adjoint method. Here, we do not consider constraint on the state variables. We only put constraints on the control inputs, which can be inequality and equality constraints. Interested readers on state constraints for production optimization problem may refer to our work in Suwartadi et al. [2012b].

The outline of this paper is the following. We discuss the second-order adjoint algorithm in Section 4.2. In Section 4.3 we explain our incompressible oil reservoir model along with the optimization goal. Two numerical cases demonstrating our proposed algorithm are presented in Section 4.4. In Section 4.5 we discuss the results of the numerical cases, and in Section 4.6 we present our conclusions.

4.2 The Adjoint Gradient For The Hessian

In general, we consider an optimization problem,

$$\begin{aligned} &\min_{\mathbf{x}\in\mathbb{R}^{n_x},\mathbf{u}\in\mathbb{R}^{n_u}} \mathscr{J}(\mathbf{x},\mathbf{u}) \\ &\text{subject to} \quad c(\mathbf{x},\mathbf{u}) = 0, \quad (\mathbf{x},\mathbf{u}) \in W_{\text{ad}}, \end{aligned} \tag{4.1}$$

where $\mathscr{J} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ is the objective function, $c : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_c}$ is the implicit constraint representing a dynamic model, and $W_{ad} \subset W := \mathbb{R}^{n_x} \times \mathbb{R}^{n_u}$ is a nonempty closed set. $\mathscr{J}$ and c are continuously Fréchet-differentiable. We assume that for a given control input $\mathbf{u}$, there is always a unique solution of the state variable $\mathbf{x}$, i.e., $\mathbf{u} \in \mathbb{R}^{n_u} \mapsto \mathbf{x} \in \mathbb{R}^{n_x}$. Moreover, we also assume that $c_{\mathbf{x}}(\mathbf{x}(\mathbf{u}),\mathbf{u})$ is continuously invertible. Hence, the implicit function theorem guarantees that $\mathbf{x}(\mathbf{u})$ is continuously differentiable. Since $\mathbf{x}$ is dependent on $\mathbf{u}$, we obtain the reduced problem

$$\begin{aligned} \min_{\mathbf{u}\in\mathbb{R}^{n_u}} \widehat{\mathscr{J}}(\mathbf{u}) : \quad &= \quad \mathscr{J}(\mathbf{x}(\mathbf{u}),\mathbf{u}), \\ \text{subject to} \quad &\mathbf{u} \in \widehat{U}_{ad} := \left\{\mathbf{u} \in \mathbb{R}^{n_u} : (\mathbf{x}(\mathbf{u}),\mathbf{u}) \in W_{ad}\right\}. \end{aligned} \tag{4.2}$$

Furthermore, an equation for $\mathbf{x}_{\mathbf{u}}(\mathbf{u})$ is obtained by taking the derivative of $c(\mathbf{x},\mathbf{u})$:

$$c_{\mathbf{x}}(\mathbf{x},\mathbf{u})\,\mathbf{x}_{\mathbf{u}}(\mathbf{u}) + c_{\mathbf{u}}(\mathbf{x},\mathbf{u}) = 0. \tag{4.3}$$

We will use this derivative for gradient computation which will be explained shortly.

4.2.1 The Adjoint Method

Following Heinkenschloss [2008], we now derive the first-order gradient using the adjoint method. We construct a Lagrangian:

$$\mathscr{L}(\mathbf{x},\mathbf{u},\boldsymbol{\lambda}) = \mathscr{J}(\mathbf{x},\mathbf{u}) + \boldsymbol{\lambda}^T c(\mathbf{x},\mathbf{u}). \tag{4.4}$$

By the optimality condition $\nabla_{\mathbf{x}}\mathscr{L}(\mathbf{x},\mathbf{u},\boldsymbol{\lambda}) = 0$, we can solve the following linear equation

$$c_{\mathbf{x}}(\mathbf{x}(\mathbf{u}),\mathbf{u})^T\boldsymbol{\lambda} = -\nabla_{\mathbf{x}}\mathscr{J}(\mathbf{x}(\mathbf{u}),\mathbf{u}). \tag{4.5}$$

Denote the solution as the Lagrangian multiplier $\boldsymbol{\lambda}(\mathbf{u})$ and the first-order gradient as

$$\nabla_{\mathbf{u}}\mathscr{J}(\mathbf{x}(\mathbf{u}),\mathbf{u}) = \nabla_{\mathbf{u}}\mathscr{L}(\mathbf{x},\mathbf{u},\boldsymbol{\lambda})|_{\mathbf{x}=\mathbf{x}(\mathbf{u}),\boldsymbol{\lambda}=\boldsymbol{\lambda}(\mathbf{u})}. \tag{4.6}$$

If one is interested in the use of a quasi-Newton method, for example, the BFGS algorithm, (4.6) will be supplied to the algorithm to numerically approximate the inverse of the Hessian matrix. Now, we continue to derive the adjoint for the Hessian matrix.

For brevity, we denote the Hessian $\nabla^2\mathscr{J}(\mathbf{u})$ referring to $\nabla_{\mathbf{uu}}\mathscr{J}(\mathbf{x}(\mathbf{u}),\mathbf{u})$, and let $\mathbf{x}$ and $\boldsymbol{\lambda}$ denote $\mathbf{x}(\mathbf{u})$ and $\boldsymbol{\lambda}(\mathbf{u})$, respectively. From (4.6), we take the derivative with respect to $\mathbf{x}$, $\mathbf{u}$, $\boldsymbol{\lambda}$, yielding the Hessian, as follows:

$$\begin{aligned}\nabla^2\mathscr{J}(\mathbf{u}) &= \nabla_{\mathbf{ux}}\mathscr{L}(\mathbf{x},\mathbf{u},\boldsymbol{\lambda})\,\mathbf{x}_{\mathbf{u}}(\mathbf{u}) + \nabla_{\mathbf{uu}}\mathscr{L}(\mathbf{x},\mathbf{u},\boldsymbol{\lambda}) \\ &\quad +\nabla_{\mathbf{u}\boldsymbol{\lambda}}\mathscr{L}(\mathbf{x},\mathbf{u},\boldsymbol{\lambda})\,\boldsymbol{\lambda}_{\mathbf{u}}(\mathbf{u}).\end{aligned} \tag{4.7}$$

Furthermore, if we differentiate $\nabla_{\mathbf{x}}\mathscr{L}(\mathbf{x},\mathbf{u},\boldsymbol{\lambda}) = 0$ we obtain

$$\begin{aligned}&\nabla_{\mathbf{xx}}\mathscr{L}(\mathbf{x},\mathbf{u},\boldsymbol{\lambda})\,\mathbf{x}_{\mathbf{u}}(\mathbf{u}) \\ &\quad +\nabla_{\mathbf{xu}}\mathscr{L}(\mathbf{x},\mathbf{u},\boldsymbol{\lambda}) \\ &+\nabla_{\mathbf{x}\boldsymbol{\lambda}}\mathscr{L}(\mathbf{x},\mathbf{u},\boldsymbol{\lambda})\,\boldsymbol{\lambda}_{\mathbf{u}}(\mathbf{u}) = 0.\end{aligned} \tag{4.8}$$

Using the implicit function theorem, we take the derivative of the Lagrangian (4.4) and obtain

$$\nabla_{\mathbf{x}\boldsymbol{\lambda}}\mathscr{L}(\mathbf{x},\mathbf{u},\boldsymbol{\lambda}) = c_{\mathbf{x}}(\mathbf{x},\mathbf{u})^T \tag{4.9}$$

$$\nabla_{\mathbf{u}\boldsymbol{\lambda}}\mathscr{L}(\mathbf{x},\mathbf{u},\boldsymbol{\lambda}) = c_{\mathbf{u}}(\mathbf{x},\mathbf{u})^T. \tag{4.10}$$

Substitute (4.9) into (4.8) with the aim of computing the derivative of the Lagrangian multiplier $\lambda_{\mathbf{u}}(\mathbf{u})$ will lead to

$$\begin{aligned}\boldsymbol{\lambda}_{\mathbf{u}}(\mathbf{u}) &= -c_{\mathbf{x}}(\mathbf{x},\mathbf{u})^{-T} \\ &\quad (\nabla_{\mathbf{xx}}\mathscr{L}(\mathbf{x},\mathbf{u},\boldsymbol{\lambda})\,\mathbf{x}_{\mathbf{u}}(\mathbf{u}) + \nabla_{\mathbf{xu}}\mathscr{L}(\mathbf{x},\mathbf{u},\boldsymbol{\lambda})).\end{aligned} \tag{4.11}$$

Equation (4.3) also implies

$$\mathbf{x_u}(\mathbf{u}) = -c_\mathbf{x}(\mathbf{x},\mathbf{u})^{-1} c_\mathbf{u}(\mathbf{x},\mathbf{u}). \tag{4.12}$$

Then, we substitute (4.10), (4.11), and (4.12) to the Hessian matrix in (4.7). After some rearrangement, we get

$$\begin{aligned} \nabla^2 \mathscr{J}(\mathbf{u}) &= \nabla_{\mathbf{ux}} \mathscr{L}(\mathbf{x},\mathbf{u},\boldsymbol{\lambda})\,\mathbf{x_u}(\mathbf{u}) + \nabla_{\mathbf{uu}} \mathscr{L}(\mathbf{x},\mathbf{u},\boldsymbol{\lambda}) \\ &\quad -\mathbf{x_u}(\mathbf{u})^T \nabla_{\mathbf{xx}} \mathscr{L}(\mathbf{x},\mathbf{u},\boldsymbol{\lambda})\,\mathbf{x_u}(\mathbf{u}) \\ &\quad -\mathbf{x_u}(\mathbf{u})^T \nabla_{\mathbf{xu}} \mathscr{L}(\mathbf{x},\mathbf{u},\boldsymbol{\lambda}). \end{aligned} \tag{4.13}$$

If we define $W(\mathbf{x},\mathbf{u}) = \begin{pmatrix} \mathbf{x_u}(\mathbf{u}) \\ \mathbf{I} \end{pmatrix}$, then (4.13) will become

$$\begin{aligned} \nabla^2 \mathscr{J}(\mathbf{u}) &= W(\mathbf{x},\mathbf{u})^T \\ &\quad \begin{pmatrix} -\nabla_{\mathbf{xx}} \mathscr{L}(\mathbf{x},\mathbf{u},\boldsymbol{\lambda}) & -\nabla_{\mathbf{xu}} \mathscr{L}(\mathbf{x},\mathbf{u},\boldsymbol{\lambda}) \\ \nabla_{\mathbf{ux}} \mathscr{L}(\mathbf{x},\mathbf{u},\boldsymbol{\lambda}) & \nabla_{\mathbf{uu}} \mathscr{L}(\mathbf{x},\mathbf{u},\boldsymbol{\lambda}) \end{pmatrix} \\ &\quad W(\mathbf{x},\mathbf{u}). \end{aligned} \tag{4.14}$$

The identity (4.14) can be used to compute the Hessian matrix. However, in practice, the pure Newton's method is prohibitive in terms of computational cost. Therefore, we instead use a Hessian-times-vector product. This Hessian-times-vector product will be used in a conjugate gradient-based method, which we will describe in the next section. Let us summarize the procedure in the following steps.

1. Given the control input $\mathbf{u}$, solve the system model $c(\mathbf{x},\mathbf{u}) = 0$. Denote the solution by the state variable $\mathbf{x}$.
2. Solve the adjoint equation $c_\mathbf{x}(\mathbf{x}(\mathbf{u}),\mathbf{u})^T \boldsymbol{\lambda} = -\nabla_\mathbf{x} \mathscr{J}(\mathbf{x}(\mathbf{u}),\mathbf{u})$. The solutions are the Lagrangian multipliers $\boldsymbol{\lambda}$.
3. Compute the derivative of the state variable in the direction of $\mathbf{s}_1$

$$\mathbf{x_u}(\mathbf{u})\,\mathbf{s}_1 = -c_\mathbf{x}(\mathbf{x},\mathbf{u})^{-1} c_\mathbf{u}(\mathbf{x},\mathbf{u})\,\mathbf{s}_1$$

4. Also compute the derivative of the Lagrangian multiplier in the direction of $\mathbf{s}_1$

$$\begin{aligned} \boldsymbol{\lambda}_\mathbf{u}(\mathbf{u})\,\mathbf{s}_1 &= -\mathbf{c_x}(\mathbf{x},\mathbf{u})^{-T} \\ &\quad (\nabla_{\mathbf{xx}} \mathscr{L}(\mathbf{x},\mathbf{u},\boldsymbol{\lambda})\,\mathbf{x_u}(\mathbf{u}) + \nabla_{\mathbf{xu}} \mathscr{L}(\mathbf{x},\mathbf{u},\boldsymbol{\lambda})\,\mathbf{s}_1) \end{aligned}$$

5. Compute the Hessian-times-vector product

$$\begin{aligned} \nabla^2 \mathscr{J}(\mathbf{u})\,\mathbf{s}_1 &= \nabla_{\mathbf{ux}} \mathscr{L}(\mathbf{x},\mathbf{u},\boldsymbol{\lambda})\,\mathbf{x_u}(\mathbf{u})\,\mathbf{s}_1 \\ &+ \nabla_{\mathbf{uu}} \mathscr{L}(\mathbf{x},\mathbf{u},\boldsymbol{\lambda})\,\mathbf{s}_1 \\ &+ c_\mathbf{u}(\mathbf{x},\mathbf{u})^T \boldsymbol{\lambda}_\mathbf{u}(\mathbf{u})\,\mathbf{s}_1 \end{aligned}$$

Note that steps 3 and 4 are linear equations that supplement the adjoint equation, which is also a linear equation.

4.2.2 Truncated Newton Method

Newton's method, $\nabla^2 \mathscr{J}(\mathbf{u}_k)\,\mathbf{s}_k = -\nabla \mathscr{J}(\mathbf{u}_k)$, is approximately solved using the conjugate gradient (CG) method. The method consists of two layers of iterations, i.e., an inner and an outer iteration. The inner iteration finds a Newton step $\mathbf{s}_k$ at iteration k, while the outer iteration is a trust-region globalization strategy. Algorithm 9 below describes the outer iteration. The algorithm setting here is similar to Steihaug [1983].

Algorithm 9 Trust-region Method (Outer iterations)

1. Initialization:

- Set the initial control input $\mathbf{u}_0$ and initial trust region radius Δ_0.
- Set constants η_1,η_2, γ_1, γ_2, which satisfy $0 < \eta_1 \leq \eta_2 < 1$ and $0 < \gamma_1 \leq \gamma_2 < 1$.
- Set the gradient convergence tolerance $\omega_* \ll 1$

2. For $k = 0, 1, \ldots$

- If $\left\| \nabla_{\mathbf{u}_k} \mathscr{J}(\mathbf{u}_k) \right\| \leq \omega_*$, then **stop**
- Compute a step $\mathbf{s}_k$ by solving the trust region sub-problem:

$$\min_{\mathbf{s}} \quad q_k(\mathbf{s}) \qquad \text{s.t} : \|\mathbf{s}\| \leq \Delta_k,$$

where $q_k(\mathbf{s}) = \nabla \mathscr{J}(\mathbf{u}_k)^T \mathbf{s} + \frac{1}{2}\mathbf{s}^T \nabla^2 \mathscr{J}(\mathbf{u}_k)\,\mathbf{s}$, a quadratic approximation of the objective function $\mathscr{J}(\mathbf{u}_k)$.

- Compute a ratio ρ_k :

$$\rho_k = \frac{\mathscr{J}(\mathbf{u}_k + \mathbf{s}_k) - \mathscr{J}(\mathbf{u}_k)}{q_k(\mathbf{s}_k)}.$$

- Update $\mathbf{u}_k$ based on ρ_k value:

$$\mathbf{u}_{k+1} = \begin{cases} \mathbf{u}_k + \mathbf{s}_k & \text{if } \rho_k \geq \eta_1 \\ \mathbf{u}_k & \text{if } \rho_k < \eta_1 \end{cases}.$$

- Update the trust region radius:

$$\Delta_{k+1} \in \begin{cases} [\Delta_k, \infty) & \text{if } \rho_k \geq \eta_2, \\ [\gamma_2 \Delta_k, \Delta_k] & \text{if } \rho_k \in [\eta_1, \eta_2), \\ [\gamma_1 \Delta_k, \gamma_2 \Delta_k] & \text{if } \rho_k < \eta_1. \end{cases}$$

The trust region sub-problem in Algorithm 9 is solved using Steihaug's Conjugate Gradient method Steihaug [1983]. We choose this method because the Hessian matrix may be negative definite. Hence, if we use line search methods, we will not succeed in finding a descent direction (minimization problem). The method is described in Algorithm 10. The algorithm is terminated if one of the following criteria is met: residual tolerance, crossing the boundary of the trust region, or negative curvature. Furthermore, since the Conjugate Gradient algorithm might have lengthy iterations, the iteration is "truncated" in the early stages of the inner iteration. This algorithm is also

known as the Truncated Newton or Inexact Newton method. The truncation is done according to the residual, which is described by the following inequality

$$\mathbf{r}_i = \left\| \nabla \mathscr{J}(\mathbf{u}_k) + \nabla^2 \mathscr{J}(\mathbf{u}_k)\, \underline{\mathbf{s}}_i \right\| \leq \eta_k \left\| \nabla \mathscr{J}(\mathbf{u}_k) \right\|. \tag{4.15}$$

The subscript i represents the iteration within the Conjugate Gradient algorithm. The $\underline{\mathbf{s}}$ is the approximate solution of the Newton step.

Algorithm 10 Steihaug Conjugate Gradient (Inner iterations)

1. Set $\eta_k < 1$, $\Delta_k > 0$, $\underline{\mathbf{s}}_0 = \mathbf{0}$, $\mathbf{r}_0 = -\nabla \mathscr{J}(\mathbf{u}_k)$, and $\mathbf{d}_0 = \mathbf{r}_0$.
2. For $i = 0, 1, \ldots$
   - If $\|\mathbf{r}_i\| \leq \eta_k \left\| \nabla \mathscr{J}(\mathbf{u}_k) \right\|$, then $\mathbf{s}_k = \underline{\mathbf{s}}_i$ and **stop**
   - $\mathbf{H}\mathbf{v}_i = \nabla^2 \mathscr{J}(\mathbf{u}_k)\, \mathbf{d}_i$
   - If $\mathbf{d}_i^T \mathbf{H}\mathbf{v}_i \leq 0$, compute τ such that $\left\| \underline{\mathbf{s}}_i + \tau \mathbf{d}_i \right\| = \Delta_k$, then $\mathbf{s}_k = \underline{\mathbf{s}}_i + \tau \mathbf{d}_i$ and **stop**
   - $\vartheta_i = \|\mathbf{r}_i\|^2 / \mathbf{d}_i^T \mathbf{H}\mathbf{v}_i$
   - $\underline{\mathbf{s}}_{i+1} = \underline{\mathbf{s}}_i + \vartheta_i \mathbf{d}_i$
   - If $\left\| \underline{\mathbf{s}}_{i+1} \right\| \geq \Delta_k$, compute τ such that $\left\| \underline{\mathbf{s}}_i + \tau \mathbf{d}_i \right\| = \Delta_k$, then $\mathbf{s}_k = \underline{\mathbf{s}}_i + \tau \mathbf{d}_i$ and **stop**
   - $\mathbf{r}_{i+1} = \mathbf{r}_i - \vartheta_i \mathbf{H}\mathbf{v}_i$
   - $\phi_i = \|\mathbf{r}_{i+1}\|^2 / \|\mathbf{r}_i\|^2$
   - $\mathbf{d}_{i+1} = \mathbf{r}_{i+1} + \phi_i \mathbf{d}_i$.

The Hessian-times-vector ($\mathbf{H}\mathbf{v}_i$) in Algorithm 10 is obtained from the adjoint method explained in the previous section. The theorem below further explains the relationship between the convergence rate and the residual value in (4.15).

Theorem 1. *Assume that $\nabla \mathscr{J}(\mathbf{u})$ is continuously differentiable in a neighborhood of a local solution $\mathbf{u}^*$ of (4.1). In addition, assume that $\nabla^2 \mathscr{J}(\mathbf{u}^*)$ is nonsingular and that it is Lipschitz continuous at $\mathbf{u}^*$. Assume that iteration k of the truncated-Newton method computes a step $\mathbf{s}_k$ that satisfies*

$$\left\| \nabla \mathscr{J}(\mathbf{u}_k) + \nabla^2 \mathscr{J}(\mathbf{u}_k)\, \mathbf{s}_k \right\| \leq \eta_k \left\| \nabla \mathscr{J}(\mathbf{u}_k) \right\|$$

for a specified value of η_k ; the new estimate of the solution is computed using $\mathbf{u}_k + \mathbf{s}_k \rightarrow \mathbf{u}_{k+1}$. If $\mathbf{u}_0$ is sufficiently close to $\mathbf{u}^*$ and $0 \leq \eta_k \leq \eta_{max} < 1$, then $\{\mathbf{u}_k\}$ converges to $\mathbf{u}^*$ q-linearly in the norm $\|\cdot\|_*$, defined by $\|v\|_* \equiv \left\| \nabla^2 \mathscr{J}(\mathbf{u}^*) v \right\|$, with asymptotic rate constant no greater than η_{max}. If $lim_{k\rightarrow\infty} \eta_k = 0$, then the convergence is q-superlinear. If $\eta_k = O\left(\left\| \nabla \mathscr{J}(\mathbf{u}_k) \right\|^r \right)$ for $0 < r \leq 1$, then the convergence is of order at least $(1 + r)$.

Proof: see Dembo et al. [1982]. □

The η_k is called the *forcing sequence*, which, as suggested in Dembo et al. [1982], has practical value $\eta_k = \min\left\{ \frac{1}{2}, \kappa \left\| \nabla \mathscr{J}(u_k) \right\|^a \right\}$, where κ is a positive constant and $0 < a \leq 1$.

The algorithms above have been described without constraints handling. Later, we will present case examples with constraints. To handle the constraints, we use an active set Sequential Linear-Quadratic Programming (SLQP) method, as implemented in the software package KNITRO [Byrd et al., 2006]. This method is based on the projected Conjugate Gradient (PCG). In short explanation, the constraints are represented by equality constraints g_E and inequality constraints g_I. The objective function will be in the form

$$P(\mathbf{u};\nu) = \widehat{\mathscr{J}}(\mathbf{u}) + \nu \sum_{i\in\mathscr{E}} \left|g_i(\mathbf{u})\right| + \nu \sum_{i\in\mathscr{I}} \left(\max\left(0,-g_i(\mathbf{u})\right)\right). \tag{4.16}$$

Here, ν is the penalty parameter and $i \in \mathscr{E}$ and $i \in \mathscr{I}$ represent the vectors g_E and g_I, respectively. The Hessian-times-vector derived in the previous section is supplied to the optimizer. Refer to Byrd et al. [2006] for further details of constraints handling in the SLQP method.

4.3 Production Optimization Of Oil Reservoirs

The problem that we are interested in for demonstrating the algorithms is that of oil reservoirs. An oil reservoir model is posed as the implicit constraint $c(\mathbf{x},\mathbf{u})$ in (4.1). An economic objective function is selected to represent the objective function $\mathscr{J}(\mathbf{x},\mathbf{u})$. Furthermore, our problem here is an open-loop optimal control setting, with the goal being to find the best possible solution for production strategy.

4.3.1 Oil Reservoir Model

We focus on the optimal production case for two-phase (oil and water) reservoirs. Furthermore, we assume immiscible and incompressible fluids and rocks, no gravity effects or capillary pressure, no-flow boundaries, and finally isothermal conditions. Let Ω be a porous media domain with boundary $\partial\Omega$. The corresponding state equations are referred to as the *pressure equation* and the *saturation equation*. The pressure equation is given by

$$\vec{v} = -\mathbf{K}\lambda_t(s)\nabla p, \quad \nabla\cdot\vec{v} = q \quad \text{in} \quad \Omega, \tag{4.17}$$

where $\vec{v}$ is the Darcy velocity, $\mathbf{K}$ is the permeability tensor, and q is the volumetric source/sink term. Finally, λ_t is the total mobility, which in this setting is the sum of the water and oil mobility functions,

$$\lambda_t(s) = \lambda_w(s) + \lambda_o(s) = k_{rw}(s)/\mu_w + k_{ro}(s)/\mu_o. \tag{4.18}$$

Here, k_{rw}, k_{ro} and μ_w, μ_o are relative permeabilities and viscosities for water and oil, respectively. Assuming no-flow boundaries mean that the normal component of the

Darcy velocity across boundaries is zero. The saturation equation is given by

$$\phi \frac{\partial s}{\partial t} + \nabla \cdot \left(f_w(s) \vec{v} \right) = q_w \quad \text{in} \quad \Omega, \tag{4.19}$$

where ϕ is the porosity and q_w is the volumetric water source. Finally, f_w is the water fractional flow function $f_w(s) = \lambda_w(s)/\lambda_t(s)$. The nonlinear behavior of the above equations is mainly dictated by the shape of the relative permeability functions, which in this paper are taken to be quadratic.

The reservoir simulation model is typically given as a (non-matching) hexahedral grid, where each grid block is assigned reservoir properties such as permeability and porosity. For ease of exposition, we assume here that the pressure equation (4.17) is discretized using a *two-point flux approximation* (TPFA) (see, e.g., Aziz & Settari [1979]), although the MATLAB implementation employed in this work Aziz & Settari [1979] is more general in the sense that a wider range of discretization schemes can be used. For a given saturation field, the TPFA assumes that the Darcy flux from one grid block i to its neighbor j is proportional to the pressure drop between the two blocks, i.e.,

$$v_{ij} = t_{ij}(s_i, s_j)(p_i - p_j). \tag{4.20}$$

In the above equation, t_{ij} is referred to as the transmissibility, which in this formulation is taken to be dependent on the saturation in the two blocks (or rather, $\lambda(s_i)$ and $\lambda(s_j)$). Wells are implemented using the Peaceman well model [Peaceman, 1983]

$$q_i^w = -WI_i^w(s_i)\left(p^w - p_i\right). \tag{4.21}$$

Here, q_i^w is the flow rate from well w into grid block i, and p^w is the wellbore pressure (assumed to be constant since we neglect gravity and wellbore flow effects). Finally, $WI_i^w(s_i)$ is the Peaceman well-index as a function of the grid block saturation s_i. The implementation uses a sequential splitting, that is, the pressure field at time-step n is calculated based on the saturation at time step $n-1$, and the saturation at time step n is calculated based on the pressure field at time step n. Let $\mathbf{p}^n$ denote the vector containing the grid block pressures and unknown wellbore pressures at time-step n. Similarly, let $\mathbf{s}^{n-1}$ denote the grid block saturations at time step $n-1$. Enforcing volume balance, i.e., setting the sum of all out-fluxes (expressed as (4.20) and (4.21)) from each block equal to the source, leads to a positive definite matrix $\mathscr{A}(\mathbf{s}^{n-1})$ in a linear system equation

$$\mathscr{A}\left(\mathbf{s}^{n-1}\right)\mathbf{p}^n = \mathscr{B}\mathbf{u}^n, \tag{4.22}$$

or in more detailed form described as

$$\begin{pmatrix} \mathbf{B}^n\left(\mathbf{s}^{n-1}\right) & \mathbf{0} & \mathbf{C} & \mathbf{D} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_w^n\left(\mathbf{s}^{n-1}\right) & \mathbf{C}_w & \mathbf{0} & \mathbf{D}_{w,N} \\ \mathbf{C}^T & \mathbf{C}_w^T & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{D}^T & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_{w,N}^T & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{v}^n \\ -\mathbf{q}_w^n \\ -\mathbf{p}^n \\ \boldsymbol{\pi}^n \\ \mathbf{p}_{w,N}^n \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ -\mathbf{D}_{w,D}\mathbf{p}_{w,D}^n\left(\mathbf{u}^n\right) \\ \mathbf{0} \\ \mathbf{0} \\ -\mathbf{q}_{tot,N}^n\left(\mathbf{u}^n\right) \end{pmatrix}. \tag{4.23}$$

Here, we have taken the right-hand-side as a function of a control input vector $\mathbf{u}^n$ for time step n, which can either be well rates or well pressures (bottom hole pressure/BHP). We note that the right-hand-side is linear in $\mathbf{u}^n$ and refer to (4.22) as the discretized pressure equation. The solution vectors $\begin{bmatrix} \mathbf{v}^n & -\mathbf{q}_w^n & -\mathbf{p}^n & \boldsymbol{\pi}^n & \mathbf{p}_{w,N}^n \end{bmatrix}^T$ are the fluxes, the well rates, grid the block pressures, the face and well pressures, and the wellbore pressure, respectively.

We discretize the saturation equation (4.19) using a standard upstream weighted implicit finite volume method to form

$$\mathbf{s}^n = \mathbf{s}^{n-1} + \triangle t^n \mathbf{D}_{PV}^{-1}\left(\mathbf{A}\left(\mathbf{v}^n\right) f_w\left(\mathbf{s}^n\right) + \mathbf{q}\left(\mathbf{v}^n\right)_+\right). \tag{4.24}$$

Here, $\triangle t^n$ is the time step and $\mathbf{D}_{PV}$ is the diagonal matrix containing the grid block pore volumes. The matrix $\mathbf{A}(\mathbf{v}^n)$ is the sparse flux matrix based on the upstream weighted discretization scheme, and $\mathbf{q}(\mathbf{v}^n)_+$ is the vector of positive sources (in this setting, water injection rates). We note that the matrix $\mathbf{A}$ and vector $\mathbf{q}$ are linear functions of $\mathbf{v}^n$, while $\mathbf{v}^n = \mathbf{T}(\mathbf{s}^{n-1})\mathbf{p}^n$, where $\mathbf{T}(\mathbf{s}^{n-1})$ is a matrix containing the transmissibilities and well indices based on $\mathbf{s}^{n-1}$. We refer to (4.24) as the discretized saturation equation.

The discrete state equations (4.22) and (4.24) can be written in an implicit form $F(\mathbf{x},\mathbf{u}) = 0$ as

$$\begin{aligned} \mathbf{F}(\widetilde{\mathbf{x}},\widetilde{\mathbf{u}}) &= \begin{pmatrix} \mathbf{F}^1\left(\mathbf{p}^1,\mathbf{s}^0,\mathbf{s}^1,\mathbf{u}^1\right) \\ \vdots \\ \mathbf{F}^N\left(\mathbf{p}^N,\mathbf{s}^{N-1},\mathbf{s}^N,\mathbf{u}^N\right), \end{pmatrix} \\ \mathbf{x}^{nT} &= (\mathbf{p}^{nT},\mathbf{s}^{nT}), \quad n = 1,...,N, \\ \widetilde{\mathbf{x}}^T &= (\mathbf{x}^{1T},...,\mathbf{x}^{NT}), \\ \widetilde{\mathbf{u}}^T &= (\mathbf{u}^{1T},...,\mathbf{u}^{NT}). \end{aligned} \tag{4.25}$$

The state vectors and control input vectors are stacked for all time instances from $n = 1,\ldots,N$.

In this study, we use Net Present Value (NPV) as the objective function, having the following formula

$$\begin{aligned}\mathscr{J}(\tilde{\mathbf{u}}) &= \sum_{n=0}^{N-1}\left[\sum_{j=1}^{\mathscr{N}_{prod}}\left(r_o q_{o,j}^n - r_w q_{w,j}^n\right) - \sum_{l=1}^{\mathscr{N}_{inj}}\left(r_{inj} q_l^n\right)\right]\frac{\triangle t^n}{(1+d)^{t_n}} \\ &= \sum_{n=0}^{N-1}\mathscr{J}^n, \end{aligned} \tag{4.26}$$

where r_o, r_w, r_{inj} are the represented oil price, water separation cost, and water injection cost, respectively. The well rate at the injector wells is denoted by q_l, the water rate at the producer wells is q_w, the oil rate at the producers is q_o, d is the discount factor, $\triangle t^n$ is the time interval, and N is the total simulation time. In addition, $\mathscr{N}_{prod}$,$\mathscr{N}_{inj}$ denote the number of producer and injector wells, respectively.

4.3.2 Adjoint Implementation

In this section, we follow the procedure in Section 4.2 to compute the first and second order gradients. The first order adjoint-gradient derivation in this work is the same as that presented in Krogstad et al. [2011]. Let $\mathscr{J}(\tilde{\mathbf{x}},\tilde{\mathbf{u}}) = \sum_{n=1}^{N}\mathscr{J}^n(\mathbf{x}^n,\mathbf{u}^n)$ be an objective function and the gradient with respect to the control input $\tilde{\mathbf{u}}$ is $\nabla_{\tilde{\mathbf{u}}}\mathscr{J}$. The procedure is the following:

1. Given the control input $\tilde{\mathbf{u}}$, we are able to compute the state variables $\tilde{\mathbf{x}}$ forward in time, i.e., for $n = 1,\ldots,N$. It should be noted that the state variables consist of pressure and saturation equations. In the later steps, the equations always correspond to the pressure and saturation parts.
2. We then construct the Lagrangian

$$\begin{aligned}\mathscr{L}(\tilde{\mathbf{x}},\tilde{\mathbf{u}},\boldsymbol{\lambda}) &= \mathscr{J}(\tilde{\mathbf{x}},\tilde{\mathbf{u}}) + \boldsymbol{\lambda}^T\mathbf{F}(\tilde{\mathbf{x}},\tilde{\mathbf{u}}) \\ &= \sum_{n=1}^{N}\left(\mathscr{J}^n + \boldsymbol{\lambda}^{nT}\mathbf{F}^n\right), \end{aligned} \tag{4.27}$$

   where

$$\begin{aligned}\boldsymbol{\lambda}^{nT}\mathbf{F}^n &= \boldsymbol{\lambda}_v^{nT}\left(\mathbf{B}^n\mathbf{v}^n - \mathbf{C}\mathbf{p}^n + \mathbf{D}\boldsymbol{\pi}^n\right) \\ &+ \boldsymbol{\lambda}_{q_w}^{nT}\left(-\mathbf{B}_w^n\mathbf{q}_w^n - \mathbf{C}_w\mathbf{p}^n + \mathbf{D}_{w,N}\mathbf{p}_{w,N}^n + \mathbf{D}_{w,D}\mathbf{p}_{w,D}^n(\mathbf{u}^n)\right) \\ &+ \boldsymbol{\lambda}_p^{nT}\left(\mathbf{C}^T\mathbf{v}^n - \mathbf{C}_w^T\mathbf{q}_w^n\right) \\ &+ \boldsymbol{\lambda}_\pi^{nT}\mathbf{D}^T\mathbf{v}^n \\ &+ \boldsymbol{\lambda}_{p_{w,N}}^{nT}\left(-\mathbf{D}_{w,N}^T\mathbf{q}_w^n + \mathbf{q}_{tot,N}^n(\mathbf{u}^n)\right) \\ &+ \boldsymbol{\lambda}_s^{nT}\left(\mathbf{s}^n - \mathbf{s}^{n-1} - \triangle t^n\mathbf{i}^n\right). \end{aligned}$$

For sake of brevity, we refer to $\mathbf{i}^n = \mathbf{D}_{PV}^{-1}\left(\mathbf{A}(\mathbf{v}^n) f_w(\mathbf{s}^n) + \mathbf{q}(\mathbf{v}^n)_+\right)$. The Lagrange multipliers denoted by $\begin{bmatrix} \boldsymbol{\lambda}_v & \boldsymbol{\lambda}_{q_w} & \boldsymbol{\lambda}_p & \boldsymbol{\lambda}_\pi & \boldsymbol{\lambda}_{p_{w,N}} & \boldsymbol{\lambda}_s \end{bmatrix}^T$ are the solution of these adjoint equations. By taking the first order optimality condition, we are able to solve the adjoint equations from $n = N, \dots 1$, given that the Lagrangian multipliers are zero at $N+1$. This equals

$$\left(\mathbf{I} - \triangle t \left(\frac{\partial \mathbf{i}^n}{\partial \mathbf{s}^n}\right)^T\right) \boldsymbol{\lambda}_s^n = \boldsymbol{\lambda}_s^{n+1} - \left(\frac{\partial \mathscr{J}^n}{\partial \mathbf{s}^n}\right)^T - \left(\frac{\partial}{\partial \mathbf{s}^n}\left(\mathbf{B}^{n+1}\mathbf{v}^{n+1}\right)\right)^T \boldsymbol{\lambda}_v^{n+1}, \tag{4.28}$$

$$\begin{pmatrix} \mathbf{B}^n\left(\mathbf{s}^{n-1}\right) & \mathbf{0} & \mathbf{C} & \mathbf{D} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_w^n\left(\mathbf{s}^{n-1}\right) & \mathbf{C}_w & \mathbf{0} & \mathbf{D}_{w,N} \\ \mathbf{C}^T & \mathbf{C}_w^T & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{D}^T & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_{w,N}^T & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\lambda}_v^n \\ \boldsymbol{\lambda}_{q_w}^n \\ \boldsymbol{\lambda}_p^n \\ \boldsymbol{\lambda}_\pi^n \\ \boldsymbol{\lambda}_{p_{w,N}}^n \end{pmatrix} = \begin{pmatrix} \left(\frac{\partial \mathbf{i}^n}{\partial \mathbf{v}^n}\right)^T \boldsymbol{\lambda}_s^n - \left(\frac{\partial J^n}{\partial \mathbf{v}^n}\right)^T \\ \left(\frac{\partial J^n}{\partial \mathbf{q}_w^n}\right)^T \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix}. \tag{4.29}$$

Using the obtained Lagrangian multipliers in (4.28) and (4.29), the first order gradient is

$$\frac{\partial \mathscr{L}}{\partial \mathbf{u}^n} = \frac{\partial \mathscr{J}^n}{\partial \mathbf{u}^n} + \boldsymbol{\lambda}_{q_w}^{nT} \mathbf{D}_{w,D} \frac{\partial \mathbf{p}_{w,D}^n(\mathbf{u}^n)}{\partial \mathbf{u}^n} + \boldsymbol{\lambda}_{p_{w,N}}^{nT} \frac{\partial \mathbf{q}_{tot,N}^n(\mathbf{u}^n)}{\partial \mathbf{u}^n}. \tag{4.30}$$

3. Now we compute the linearized state equations such that

$$\frac{\partial \mathbf{F}^n}{\partial \tilde{\mathbf{x}}} \boldsymbol{\alpha} = -\frac{\partial \mathbf{F}^n}{\partial \tilde{\mathbf{u}}} \boldsymbol{\beta}$$

from $n = 1, \dots, N$. This will lead to analogous equations to (4.23) and (4.24), namely

$$\begin{pmatrix} \mathbf{B}^n\left(\mathbf{s}^{n-1}\right) & \mathbf{0} & \mathbf{C} & \mathbf{D} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_w^n\left(\mathbf{s}^{n-1}\right) & \mathbf{C}_w & \mathbf{0} & \mathbf{D}_{w,N} \\ \mathbf{C}^T & \mathbf{C}_w^T & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{D}^T & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_{w,N}^T & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha}_v^n \\ -\boldsymbol{\alpha}_{q_w}^n \\ -\boldsymbol{\alpha}_p^n \\ \boldsymbol{\alpha}_\pi^n \\ \boldsymbol{\alpha}_{p_{w,N}}^n \end{pmatrix} = \begin{pmatrix} -\frac{\partial}{\partial \mathbf{s}^{n-1}}\left(\mathbf{B}^n\left(\mathbf{s}^{n-1}\right)\right)\boldsymbol{\alpha}_s^{n-1} \\ -\frac{\partial}{\partial \mathbf{s}^{n-1}}\left(\mathbf{B}_w^n\left(\mathbf{s}^{n-1}\right)\right)\boldsymbol{\alpha}_s^{n-1} - \mathbf{D}_{w,D}\mathbf{p}_{w,D}^n(\boldsymbol{\beta}) \\ \mathbf{0} \\ \mathbf{0} \\ -\mathbf{q}_{tot,N}^n(\boldsymbol{\beta}) \end{pmatrix}, \tag{4.31}$$

$$\left(\mathbf{I} - \frac{\partial \mathbf{i}^n}{\partial \mathbf{s}^n}\right)\boldsymbol{\alpha}_s^n = \boldsymbol{\alpha}_s^{n-1} + \frac{\partial \mathbf{i}^n}{\partial \mathbf{v}^n}\boldsymbol{\alpha}_v^n. \tag{4.32}$$

4. Then we compute the derivative of the Lagrangian multipliers, denoted by

$$\begin{bmatrix} \boldsymbol{\varphi}_v & \boldsymbol{\varphi}_{q_w} & \boldsymbol{\varphi}_p & \boldsymbol{\varphi}_\pi & \boldsymbol{\varphi}_{p_{w,N}} & \boldsymbol{\varphi}_s \end{bmatrix}^T$$

, which are also analogous to the adjoint equation (4.28) and (4.29). The following equations are solved backward in time, i.e., from $n = N, \ldots 1$.

$$\begin{pmatrix} \mathbf{B}^n\left(\mathbf{s}^{n-1}\right) & \mathbf{0} & \mathbf{C} & \mathbf{D} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_w^n\left(\mathbf{s}^{n-1}\right) & \mathbf{C}_w & \mathbf{0} & \mathbf{D}_{w,N} \\ \mathbf{C}^T & \mathbf{C}_w^T & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{D}^T & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_{w,N}^T & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\varphi}_v^n \\ \boldsymbol{\varphi}_{q_w}^n \\ \boldsymbol{\varphi}_p^n \\ \boldsymbol{\varphi}_\pi^n \\ \boldsymbol{\varphi}_{p_{w,N}}^n \end{pmatrix} =$$

$$\begin{pmatrix} -\frac{\partial}{\partial \mathbf{s}^{n-1}}\left(\mathbf{B}^n\left(\mathbf{s}^{n-1}\right)\boldsymbol{\lambda}_v^n\right)\boldsymbol{\alpha}_s^{n-1} + \left(\frac{\partial \mathbf{i}^n}{\partial \mathbf{v}^n}\right)^T\boldsymbol{\varphi}_s + \frac{\partial}{\partial \mathbf{s}^n}\left(\frac{\partial \mathbf{i}^{nT}}{\partial \mathbf{v}^n}\boldsymbol{\lambda}_s^n\right)\boldsymbol{\alpha}_s^n + \frac{\partial}{\partial \mathbf{v}^n}\left(\frac{\partial \mathbf{i}^{nT}}{\partial \mathbf{v}^n}\boldsymbol{\lambda}_s^n\right)\boldsymbol{\alpha}_v^n \\ -\frac{\partial}{\partial \mathbf{s}^{n-1}}\left(\mathbf{B}_w^n\left(\mathbf{s}^{n-1}\right)\boldsymbol{\lambda}_{q_w}^n\right)\boldsymbol{\alpha}_s^{n-1} + \frac{\partial^2 \mathscr{L}^n}{\partial \mathbf{s}^n \partial \mathbf{q}_w^n}\boldsymbol{\alpha}_s^n \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix} \tag{4.33}$$

$$
\begin{aligned}
\left(\mathbf{I}-\frac{\partial \mathbf{i}^{nT}}{\partial \mathbf{s}^n}\right)\boldsymbol{\varphi}_s^n &= \boldsymbol{\varphi}_s^{n+1}+\frac{\partial}{\partial \mathbf{s}^n}\left(\frac{\partial \mathbf{i}^{nT}}{\partial \mathbf{s}^n}\boldsymbol{\lambda}_s^n\right)\boldsymbol{\alpha}_s^n+\frac{\partial}{\partial \mathbf{v}^n}\left(\frac{\partial \mathbf{i}^{nT}}{\partial \mathbf{s}^n}\boldsymbol{\lambda}_s^n\right)\boldsymbol{\alpha}_v^n \\
&- \left(\frac{\partial^2 \mathscr{J}^n}{\partial^2 \mathbf{s}^n}\right)\boldsymbol{\alpha}_s^n-\left(\frac{\partial}{\partial \mathbf{s}^n}\left(\mathbf{B}^{n+1}\left(\mathbf{s}^n\right)\mathbf{v}^{n+1}\right)\right)^T\boldsymbol{\varphi}_v^{n+1} \\
&- \frac{\partial}{\partial \mathbf{s}^n}\left(\left(\frac{\partial}{\partial \mathbf{s}^n}\left(\mathbf{B}^{n+1}\left(\mathbf{s}^n\right)\mathbf{v}^{n+1}\right)^T\right)\boldsymbol{\lambda}_v^{n+1}\right)\boldsymbol{\alpha}_s^n \\
&- \frac{\partial}{\partial \mathbf{v}^{n+1}}\left(\left(\frac{\partial}{\partial \mathbf{s}^n}\left(\mathbf{B}^{n+1}\left(\mathbf{s}^n\right)\mathbf{v}^{n+1}\right)\right)^T\boldsymbol{\lambda}_v^{n+1}\right)\boldsymbol{\alpha}_v^{n+1} \\
&+ \left(\frac{\partial}{\partial \mathbf{s}^n}\left(\mathbf{B}_w^{n+1}\left(\mathbf{s}^n\right)\mathbf{q}_w^{n+1}\right)\right)^T\boldsymbol{\varphi}_{q_w}^{n+1} \\
&+ \frac{\partial}{\partial \mathbf{s}^n}\left(\frac{\partial}{\partial \mathbf{s}^n}\left(\mathbf{B}_w^{n+1}\left(\mathbf{s}^n\right)\mathbf{q}_w^{n+1}\right)^T\boldsymbol{\lambda}_{q_w}^{n+1}\right)\boldsymbol{\alpha}_s^n \\
&+ \frac{\partial}{\partial \mathbf{q}_w^{n+1}}\left(\frac{\partial}{\partial \mathbf{s}^n}\left(\mathbf{B}_w^{n+1}\left(\mathbf{s}^n\right)\mathbf{q}_w^{n+1}\right)^T\boldsymbol{\lambda}_{q_w}^{n+1}\right)\boldsymbol{\alpha}_{q_w}^{n+1}. \qquad (4.34)
\end{aligned}
$$

5. Finally, the Hessian-times-vector product is given by

$$
\frac{\partial^2 \mathscr{L}}{\partial^2 \mathbf{u}^n}\boldsymbol{\beta}=\frac{\partial^2 \mathscr{J}}{\partial^2 \mathbf{u}^n}\boldsymbol{\beta}+\boldsymbol{\varphi}_{q_w}^{nT}\mathbf{D}_{w,D}\frac{\partial \mathbf{p}_{w,D}^n\left(\mathbf{u}^n\right)}{\partial \mathbf{u}^n}+\boldsymbol{\varphi}_{p_{w,N}}^{nT}\frac{\partial \mathbf{q}_{tot,N}^n\left(\mathbf{u}^n\right)}{\partial \mathbf{u}^n}. \qquad (4.35)
$$

To check the adjoint implementation, we compare the obtained gradients with those of the finite-difference method. The gradient from forward finite-difference method is computed as follows

$$
\nabla_{\tilde{\mathbf{u}}}\mathscr{J}=\frac{\mathscr{J}\left(\tilde{\mathbf{u}}+\varepsilon\mathbf{v}\right)-\mathscr{J}\left(\tilde{\mathbf{u}}\right)}{\varepsilon}. \qquad (4.36)
$$

Here, ε is a small value representing the perturbation size, $\mathbf{v}$ is the arbitrary vector with the same dimension as the control input and it is generated randomly. We compare the adjoint-gradient and the gradient from finite-difference in Table 4.1, where $<\mathbf{grad},\mathbf{v}>$ represents the inner product between the adjoint-gradient and the vector $\mathbf{v}$. The absolute error is the absolute value of the difference between adjoint-gradient and gradient using finite-differences. This comparison is performed using an initial control value (well-rate controlled) described in Section 4.4.
We also check the Hessian-times-vector product against its finite-difference approximation. The finite-difference of Hessian-times-vector is defined by

$$
\mathbf{Hv}=\frac{\mathbf{g}_1-\mathbf{grad}}{\varepsilon}, \qquad (4.37)
$$

where $\mathbf{g}_1$ is gradient using finite-differences, **grad** is the adjoint-gradient. If these two gradients in (4.37) are computed by finite-difference methods and the Hessian-times-vector is supplied to a conjugate gradient algorithm, this will lead to a *Hessian-free*

Table 4.1: Comparison of first-order gradient algorithms: Adjoint and finite-difference (FD) approximation

ε-size	Adjoint $<\mathbf{grad},\mathbf{v}>$	FD approximation	Absolute error
$1.000000e-003$	$-2.321109e+005$	$-2.321951e+005$	$8.418134e+001$
$1.000000e-004$	$-2.321109e+005$	$-2.321109e+005$	$2.050601e-005$
$1.000000e-005$	$-2.321109e+005$	$-2.321109e+005$	$1.696753e-007$
$1.000000e-006$	$-2.321109e+005$	$-2.321109e+005$	$1.079170e-006$
$1.000000e-007$	$-2.321109e+005$	$-2.321109e+005$	$4.377827e-006$

Newton method [see Nocedal & Wright, 2006, chap. 7]. The comparison of the Hessian-times-vector product is shown in Table 4.2. The error absolute is inner product of the difference of adjoint and finite-difference approximation.

Table 4.2: Comparison of Hessian-times-vector : Adjoint and finite-difference (FD) approximation

ε-size	absolute error
$1.000000e-003$	$8.940078e+011$
$1.000000e-004$	$5.928105e+000$
$1.000000e-005$	$2.261278e-004$
$1.000000e-006$	$1.597134e-006$
$1.000000e-007$	$2.757622e-008$

Furthermore, the Hessian must be a symmetric matrix since it has the *self-adjoint* property. This is described in Table 4.3, where $\mathbf{v}_1$ and $\mathbf{v}_2$ are generated randomly.

Table 4.3: Self-adjoint test with $\varepsilon = 1.000000e-006$

$<\mathbf{H}\mathbf{v}_1,\mathbf{v}_2>$	$<\mathbf{H}\mathbf{v}_2,\mathbf{v}_1>$	$\lvert<\mathbf{H}\mathbf{v}_1,\mathbf{v}_2>-<\mathbf{H}\mathbf{v}_2,\mathbf{v}_1>\rvert$
$4.974882e-002$	$4.950165e-002$	$2.471707e-004$

4.4 Numerical Cases

In this section, we use a five-spot 2-dimensional oil reservoir consisting of oil and water. The reservoir is discretized into 60×60 grid blocks and it originates from layer 65 of the SPE 10th comparative study [Christie & Blunt, 2001]. The grid block dimension is $10\,\text{ft}\times10\,\text{ft}\times2\,\text{ft}$. There are four producer wells at the corners and one injector well in the middle. The porosity is set homogenously to 0.3 in the reservoir, but the permeability

is heterogenous, as depicted in Figure 4.1. The mobility ratio between oil and water is one. The initial water saturation is 0.2, and pressure in the reservoir is 400 bar. The oil price is set to 126 \$/m^3, the water separation cost to 19 \$/m^3, and the water injection cost to 6 \$/m^3. We set up two examples by distinguishing the control input. Furthermore, we parameterize the control input into five control intervals, where a control interval is equal to 30 days.

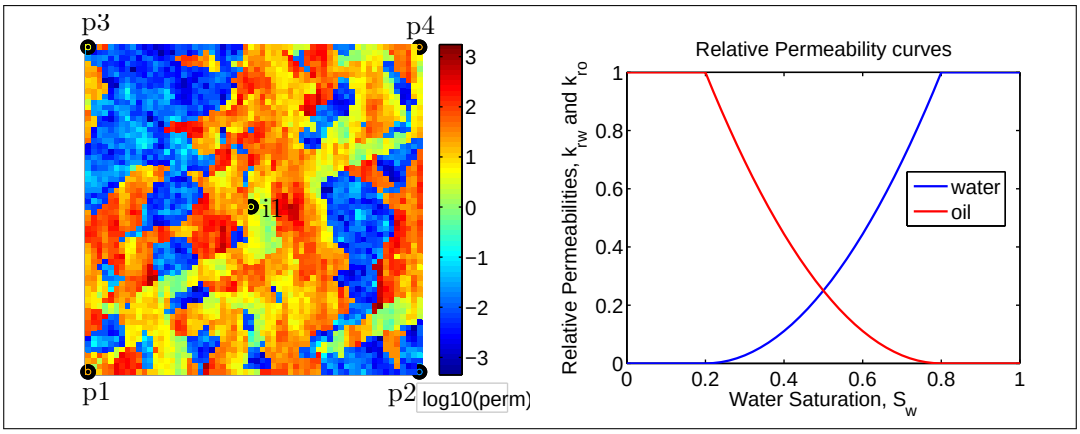

Figure 4.1: Five-spot well pattern with a heterogenous permeability field (with units in milidarcy), 2-dimensional 60 × 60 grid block. $i1$ is the injector well in the middle, and $p1$, $p2$, $p3$, $p4$ are the producer wells at the corners.

4.4.1 Case 1

In this case, we use well rates as our control input, namely

$$\mathbf{u}^n = \begin{bmatrix} q^n_{i_1} & q^n_{p_1} & q^n_{p_2} & q^n_{p_3} & q^n_{p_4} \end{bmatrix}^T.$$

Hence, we have five control intervals, which means that this case has 25 decision variables. Since we assume incompressible flow, the total injection rate is equal to the total producer rates, that is, $q^n_{i1} = \sum_{j=1}^{4} q^n_{pj}$. Moreover, the well rate is bounded to be greater than 0 m^3/day. The initial injection rate is 10 m^3/day.

4.4.2 Case 2

Now, we use BHP in the producer wells as the control input and set a fixed BHP in the injector well. The control input is

$$\mathbf{u}^n = \begin{bmatrix} p^n_{w,p_1} & p^n_{w,p_2} & p^n_{w,p_3} & p^n_{w,p_4} \end{bmatrix}^T.$$

In total, the number of control inputs is 20. We fix the BHP at the injector well to 411 bar, and the initial BHP at each producer well is 408 bar. Furthermore, we constrain BHP below 410 bar.

4.5 Results and Discussion

We compare our proposed method, the Truncated Newton (TN), to a quasi-Newton method, that is, BFGS. The BFGS method is also used in a conjugate gradient algorithm. Table 4.4 summarizes the performance of the methods using the given initial control input. We run each case for 100 different initial control inputs, yielding the statistics presented in Tables 4.5 and 4.6. The simulations were performed on a Linux 64-bit machine with an Intel Xeon(R) 3.00 GHz and 16 GB of RAM. The constraints appear as equality and bound constraints in the first case, and only as bound constraints in the second case. Moreover, we set the number of inner conjugate gradient iterations to 5.

4.5.1 Case 1

The stopping criteria for the optimizer are relative gradient and step size, which are set to 10^{-3} and 10^{-15}, respectively. Both methods terminate due to the gradient tolerance and lead to the same value of NPV. The TN method excels in the number of iterations and the number of function evaluations. The optimized control inputs are displayed in Figure 4.3, where the results from both methods almost have comparable well rates.

4.5.2 Case 2

We use the same stopping criteria as in Case 1 and optimized control inputs are shown in Figure 4.5. In this case, both methods stop due to the gradient criterion. The TN method yields higher NPV than the BFGS method. However, the TN method is more CPU-intensive, as can be seen in Table 4.4. This can be explained from the Hessian-times-vector procedure in Section II, particularly steps 3 and 4, which require the solving of linear equations. To solve the linear equations, we use the built-in solver in MATLAB which is a direct sparse method (see Davis [2006]).

4.6 Conclusion

As shown, the TN method requires fewer function and gradient evaluations. Furthermore, in Case 2, the TN method results in better objective function values than the BFGS method. However, the TN method requires more CPU time. This is due to the

Table 4.4: Performance comparison of BFGS and TN in terms of: number of iterations; number of function, gradient, and Hessian-vector evaluations; CPU time; and objective function values

Method	Case 1		Case 2	
	BFGS	TN	BFGS	TN
# of iteration	54	16	111	21
# of function evals	112	34	112	25
# of grad evals	55	17	112	22
# of Hessian-vec evals	-	250	-	275
CPU time (in sec)	170	370	280	415
NPV (in 10^5)	2.42	2.42	2.24	2.32

Table 4.5: Statistical performance comparison of BFGS and TN for Case 1 from 100 different initial control inputs.

Method	Case 1					
	BFGS			TN		
	min	max	average	min	max	average
# iteration	35	74	46	14	50	23
# function evals.	74	145	97	30	100	50
# grad. evals.	36	75	47	15	50	24
# Hessian-vec. evals.	-	-	-	218	669	365
CPU Time (in sec)	114	228	150	323	990	546
NPV (in $\$10^5$)	2.42	2.42	2.42	2.42	2.42	2.42

Table 4.6: Statistical performance comparison of BFGS and TN for Case 2 from 100 different initial control inputs.

Method	Case 2					
	BFGS			TN		
	min	max	average	min	max	average
# iteration	87	133	113	16	43	25
# function evals.	107	144	119	22	81	42
# grad. evals.	88	134	114	17	44	26
# Hessian-vec. evals.	-	-	-	214	894	474
CPU Time (in sec)	214	301	331	320	1412	686
NPV (in $\$10^5$)	1.23	2.28	2.16	2.28	2.32	2.31

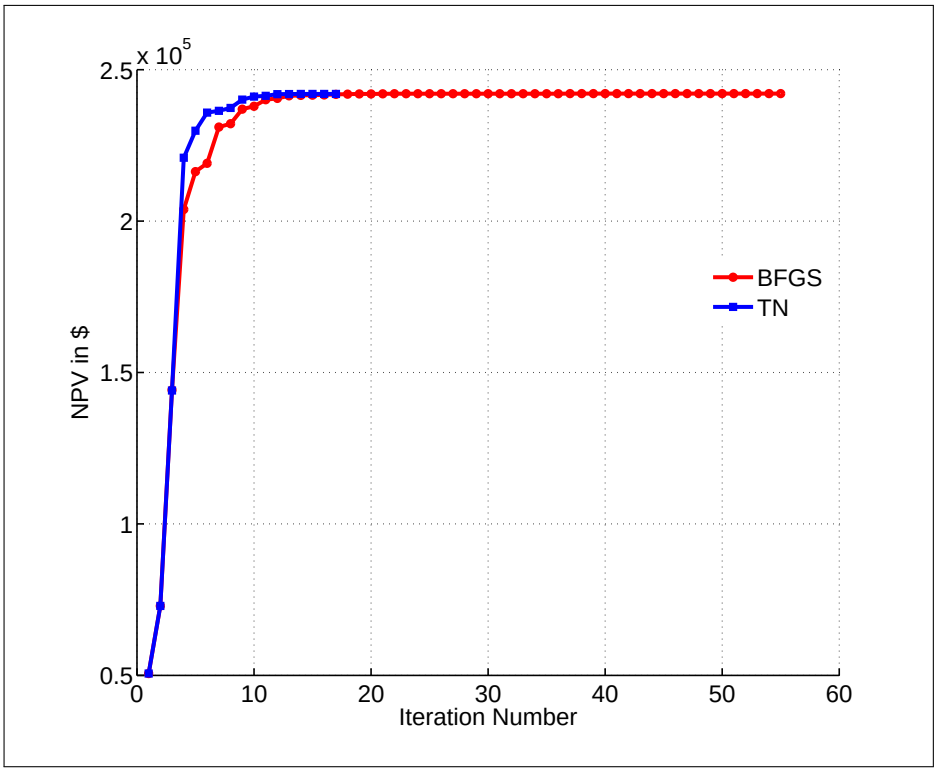

Figure 4.2: Comparison of the objective function (NPV) evaluation between BFGS and Truncated Newton (TN) for the first case. The control inputs are well rates at the injector and producer wells.

fact that the computation of the Hessian-times-vector product requires more linear equations to be solved. The Hessian-times-vector procedure requires 4 simulations: 2 forward simulations and 2 backward simulations. The forward simulations are needed to compute the state variables and the linearized state variables. The backward simulations are used to obtain the Lagrangian multiplier and its derivative.

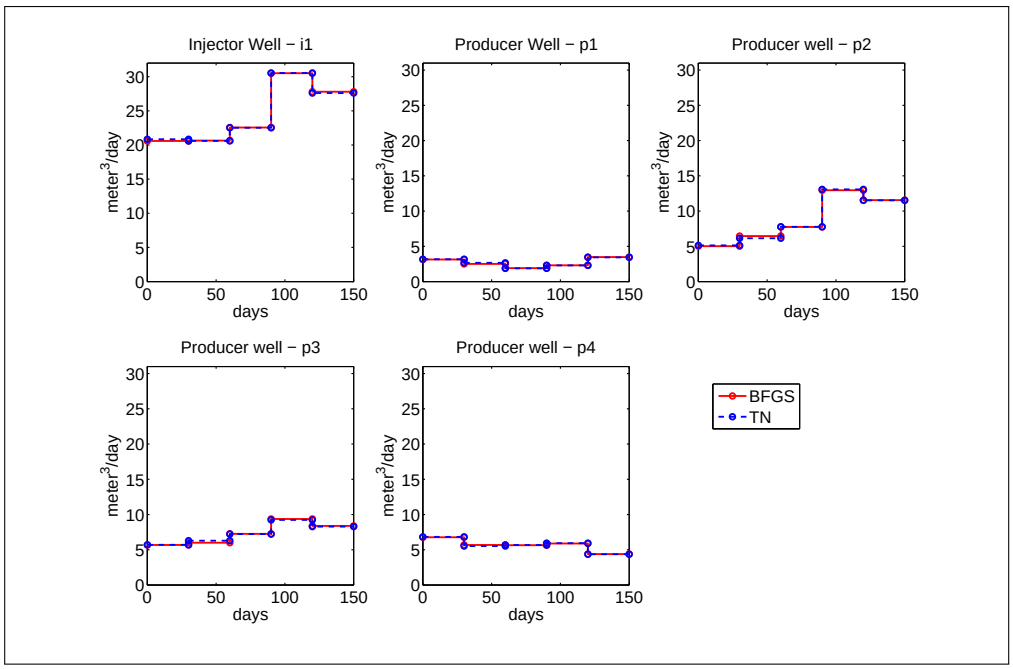

Figure 4.3: Comparison of optimized control inputs, BFGS and TN methods.

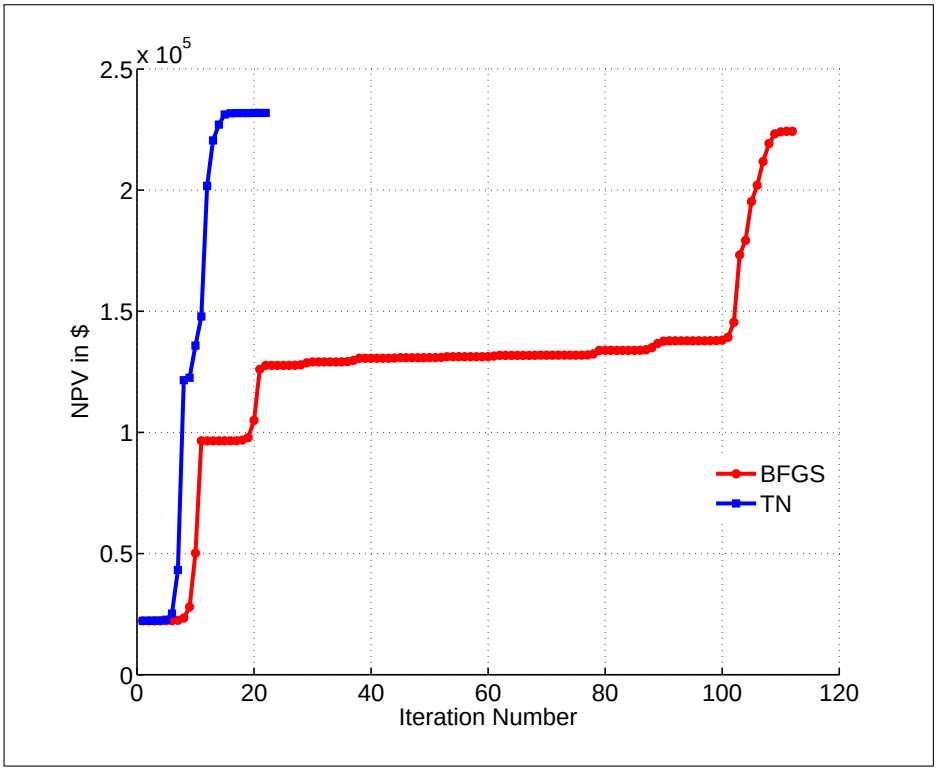

Figure 4.4: Comparison of the objective function (NPV) evaluation between BFGS and TN for the second case. The control inputs are BHP at the producer wells.

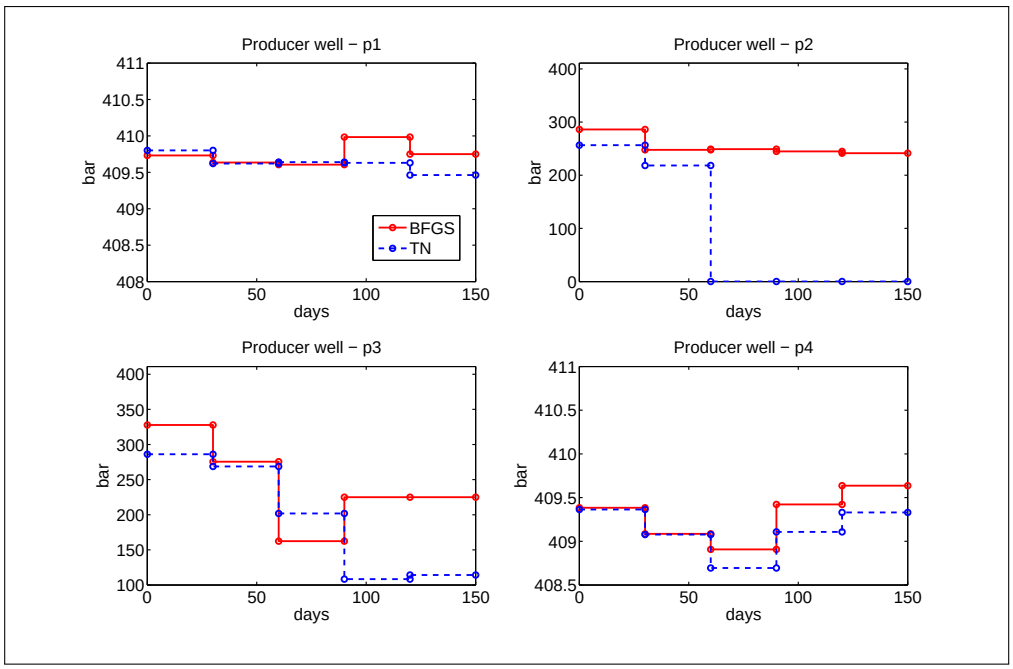

Figure 4.5: Comparison of optimized control inputs, BFGS and TN methods.

Chapter 5

Adjoint-based Surrogate Optimization of Oil Reservoir Water Flooding

This chapter is based on Suwartadi et al. [2012a].

A novel method which uses reduced-order models for solving optimization problems with nonlinear inequality constraints is proposed and tested. The Lagrangian barrier method is used to handle the nonlinear inequality constraints. The optimization with reduced-order models is done by employing a trust-region proper orthogonal decomposition (TRPOD) algorithm. In addition to the POD method, we also build a reduced-order model based on a discrete empirical interpolation method (DEIM). In the algorithm, the first-order gradient of the objective function is computed by using the adjoint method, while the inverse of the second-order gradient is approximated using the BFGS method. The reduced-order models involve both the forward (state) and backward (adjoint) equations. Three optimization case examples are used to study the method. It shows that the TRPOD method is very efficient while simultaneously honoring nonlinear constraints. Furthermore, the contribution of this work is to introduce a novel application of TRPOD combined with a DEIM-based reduced-order model since it is applied successfully to reservoir engineering problems.

5.1 Introduction

Oil reservoirs are usually modeled by partial differential equations (PDEs), where the geological model is in the order of 10^6 to 10^9 discretized grids. Oil reservoir models are used in many important applications in the reservoir engineering domain, such as for determining optimal injection and drainage strategies (production optimization), and for well placement. These applications can be cast into optimization problems. In this paper, we focus on the production optimization application with emphasis on water

flooding. The injected water aims to sweep remaining oil efficiently. With the current state-of-the-art computing power, reservoir simulation models are usually reduced to the order between 10^4 and 10^6 grid blocks. This process is known as *upscaling* since it creates a coarse model from the geological model. Upscaling is done based on geophysical interpretation by reservoir engineers. This involves heuristics and can be a time-consuming task.

Model order reduction techniques can be used to facilitate the upscaling process. The use of model order reduction techniques has been around in the reservoir simulation research since the early years of 2000's, see e.g., Markovinovic et al. [2002b] and Markovinovic et al. [2002a]. The work of Heijn et al. [2004] compared methods for reduced-order modeling which treat oil reservoirs both as linear and nonlinear models. The methods originate from systems and control theory. Balanced truncation, subspace identification, and proper orthogonal decomposition (POD) were compared. It was shown that POD gave the best approximation of the oil reservoir dynamics. In follow up work van Doren et al. [2006]; Markovinovic & Jansen [2006], POD method was used for gradient-based production optimization. The adjoint equations were derived using reduced-order models of the state equations. POD generates reduced-order models with global basis functions. Another approach was presented in Krogstad et al. [2011] where a multiscale method was applied to compute local basis function. In that work an optimization problem using a real geometry of an oil reservoir was solved in 15 minutes, compared to a normal length of hours or even days. In a more recent work a combination of multiscale and POD methods was presented in Krogstad [2011], yielding local POD basis function. The selection of local basis functions in a multiscale method is done by considering physical aspects such as fault locations and flux boundaries, which is more intuitive to the reservoir engineers since the reservoir model is divided into some coarsened segments, where each of the segment has its own local basis function.

The use of the trajectory-piecewise-linearization (TPWL) method, which models the oil reservoir as a linear time varying (LTV) system along selected operating points, was proposed in Cardoso & Durlofsky [2010a]. The same authors also proposed the use of missing point estimation (MPE)-POD in Cardoso et al. [2009] and further used the TPWL-based model order reduction for production optimization in Cardoso & Durlofsky [2010b]. Two optimization methods were presented in Cardoso & Durlofsky [2010b]. There were the gradient-based and generalized pattern search methods. None of the production optimization papers mentioned above discuss the state or nonlinear output constraint problem. In more recent work, the use of approximate dynamic programming combined with POD method was proposed in Wen et al. [2011]. This work used the penalty method to handle the state constraint problem.

In other areas, the POD method has been used for constraints handling in low-fidelity model optimization. Among this, the trust-region POD (TRPOD), originally

proposed in Fahl [2000] for unconstrained optimization problems, was further developed for constrained optimization. The idea is that POD gives a good approximation of the high-fidelity model by updating POD basis functions in limited (or "*trusted*") operating points. During the course of optimization the decision variables are always changed, therefore the POD basis functions need to be updated using the new update of decision variables. Without this updating, the POD basis functions represent the previous/old decision variables, which are no longer valid and give a poor approximation of the high-fidelity model. The constrained TRPOD, which means optimization using reduced-order models in the presence of (equality/inequality) constraints, was initiated in Alexandrov et al. [2001]. The authors developed penalty, augmented Lagrangian, and SQP-like methods. A similar approach was used in Robinson [2007], where POD, space mapping methods, and their combination were proposed for constructing the reduced-order models. Furthermore, the use of the filter method, for nonlinear constraints handling, in low-fidelity models optimization along with TRPOD was presented in Agarwal [2010]; Agarwal & Biegler [2011].

In this work, we follow the TRPOD method and to handle the state constraints we use the Lagrangian barrier method, which is a continuation of our work in Suwartadi et al. [2010a]. To best of our knowledge, the TRPOD method has not been applied to the reservoir simulation problem. Hence, the contribution of this work is to apply TRPOD method to the production optimization of oil reservoirs. Furthermore, we consider nonlinear inequality constraints. Our method is a gradient-based optimization method which uses POD for computing basis functions for state and adjoint equations. Since we have implemented the adjoint method in the high-fidelity model and to avoid the difficulty of re-implementating the adjoint-based gradient in the reduced-order model, we take snapshots of the adjoint equations as well. Thus, the reduced-order models in this work consist of reduced-order state and adjoint equations.

It should be noted that there are many variants of POD methods in addition to the vanilla POD and MPE-POD mentioned above. A combined POD and discrete empirical interpolation method (DEIM), where DEIM is a variant of EIM Barrault et al. [2004], was recently proposed in Chaturantabut & Sorensen [2010]. This work pointed out that the vanilla POD is only good for approximating linear or bi-linear terms of equations. As shown in an example in Chaturantabut & Sorensen [2011], for nonlinear systems, POD in conjunction with DEIM gives considerable CPU time speedup compared to the vanilla POD. Since the oil reservoir models contain nonlinear terms, in this work we also compare the vanilla POD and POD-DEIM methods. This is also a new application of DEIM to oil reservoir model as well as the use of DEIM to a challenging optimization problem. We note this as another contribution of this work.

The outline of this paper is the following. In Section 5.2 we describe the oil reservoir model which consists of pressure and saturation equations representing the state variables. We refer to these state equations as forward equations. In this section we

also derive the adjoint equations and the reduced-order models. The production optimization problem is explained in Section 5.3, which basically is an economic optimization problem. In Section 5.4, we present the algorithms for TRPOD and the Lagrangian barrier method for nonlinear constraint handling. The algorithms use TRPOD in the inner iteration and Lagrangian barrier in the outer iteration. This means TRPOD is used within the Lagrangian barrier iteration. Case examples that use 2D oil reservoirs are presented and the results are discussed in Section 5.5. Finally, based on the case example results we conclude this paper in Section 5.6.

5.2 Oil Reservoir Model

Water flooding is the most common secondary recovery technique for oil reservoirs. During early stages of oil reservoir production, the pressure in the reservoir is high enough to flow naturally. However, water is often injected to provide pressure support in the reservoir and thereby increase recovery.

We assume the reservoir is above the *bubble point* so that the oil component is in liquid form only. Furthermore, we assume the process is isothermal, the liquids are incompressible, immiscible (water and oil cannot be mixed), no capillary pressure between oil and water, no gravity effect, and no-flow at the boundary of the reservoirs.

5.2.1 Forward Model

The oil reservoir is governed by the continuity equation which expresses conservation of mass. We refer the model exposition here to Aarnes et al. [2007]. The state equations consist of *pressure* and *saturation* equations. Let Ω be a porous media domain with boundary $\partial\Omega$. The pressure equation is given by

$$\vec{v} = -\mathbf{K}\lambda_t(s)\nabla p, \quad \nabla \cdot \vec{v} = q \quad \text{in} \quad \Omega, \tag{5.1}$$

where $\vec{v}$ is the Darcy velocity, $\mathbf{K}$ is the permeability tensor, p is the pressure, and q is the volumetric source/sink term. Finally λ_t is the total mobility, which in this setting is the sum of the water and oil mobility functions,

$$\lambda_t(s) = \lambda_w(s) + \lambda_o(s) = \frac{k_{rw}(s)}{\mu_w} + \frac{k_{ro}(s)}{\mu_o}. \tag{5.2}$$

Here, k_{rw}, k_{ro} and μ_w, μ_o are the water and oil relative permeabilities and viscosities, respectively. Assuming no-flow boundaries means that the normal component of the Darcy velocity across boundaries is zero.

The saturation equation is given by

$$\phi \frac{\partial s}{\partial t} + \nabla \cdot \left(f_w(s) \vec{v} \right) = q_w \quad \text{in} \quad \Omega, \tag{5.3}$$

where ϕ is the porosity and q_w is the volumetric water source. Finally, f_w is the water fractional flow function $f_w(s) = \lambda_w(s)/\lambda_t(s)$. The nonlinear behavior of the above equations is mainly dictated by the shape of the relative permeability functions, which in this paper are taken to be quadratic. The relative permeability data are obtained from laboratory experiments using small portions of rocks which do not generally represent the rock properties of the whole reservoir. Hence, uncertainties are unavoidable.

(5.1) and (5.3), which are elliptic and parabolic PDEs respectively, are solved numerically for typical oil reservoir simulation. Hence, we need to discretize the equations. We discretize the domain Ω into a set of polyhedral grid blocks $\{E_i\}$, where a grid block E contains faces e_k, $k = 1, \ldots, n_E$. Let $\mathbf{v}_E$ be the outward pointing flux vectors corresponding to the faces of E, p_E the pressure at the grid block center, and $\boldsymbol{\pi}_E$ the pressures at the grid faces. Then, the discretized pressure equation for a single grid-block is

$$\begin{aligned} \mathbf{v}_E &= \lambda(\mathbf{s}_E)\, \mathbf{T}_E \left(p_E - \boldsymbol{\pi}_E \right) \\ \sum \mathbf{v}_E &= q_E, \end{aligned} \tag{5.4}$$

where $\mathbf{T}_E$ is the transmissibility matrix, and q_E is the source/sink term in block E. Here, we discretize according to the two-point flux-approximation (TPFA) (see e.g., Aziz & Settari [1979]), which will result in diagonal transmissibility matrices.

The boundary conditions are only located at wells since we assume no-flow boundary. As in (5.1) the sink/source terms represent injector/producer wells. The wells are modeled by the Peaceman equation Peaceman [1983] as follows

$$q_E^w = -\lambda(s_E)\, WI_E^w \left(p_E - p_E^w \right). \tag{5.5}$$

Here q_E^w is the flow rate from well w into grid block E and p_E^w is the wellbore pressure (assumed to be constant since we neglect gravity and wellbore flow effects). WI_E^w is the Peaceman well-index as a function of the grid block saturation s_E.

The discretized pressure equation (5.4) and the well equation (5.5) can be combined such that they construct the following linear equation

$$\begin{pmatrix} \mathbf{B}^n\left(\mathbf{s}^{n-1}\right) & \mathbf{0} & \mathbf{C} & \mathbf{D} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_w^n\left(\mathbf{s}^{n-1}\right) & \mathbf{C}_w & \mathbf{0} & \mathbf{D}_{w,N} \\ \mathbf{C}^T & \mathbf{C}_w^T & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{D}^T & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_{w,N}^T & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{v}^n \\ -\mathbf{q}_w^n \\ -\mathbf{p}^n \\ \boldsymbol{\pi}^n \\ \mathbf{p}_{w,N}^n \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ -\mathbf{D}_{w,D}\mathbf{p}_{w,D}^n(\mathbf{u}^n) \\ \mathbf{0} \\ \mathbf{0} \\ -\mathbf{q}_{tot,N}^n(\mathbf{u}^n) \end{pmatrix}. \tag{5.6}$$

Superscript n represents the time step and $\mathbf{u}^n$ is the control input at time step n, which could be either bottom-hole pressure (BHP) or well rate. The solution vector

$$\begin{bmatrix} \mathbf{v}^n & -\mathbf{q}_w^n & -\mathbf{p}^n & \boldsymbol{\pi}^n & \mathbf{p}_{w,N}^n \end{bmatrix}^T$$

include the fluxes, the well rates, the grid-block pressures, the face and well pressures, and the wellbore pressure, respectively. (5.6) is solved for time step n using the default linear solver in MATLAB which is a direct sparse method (see Davis [2006]). We note that when TPFA is used, the pressure equation (5.6) can be reduced to a system of cell pressure-unknowns only, while the current implementation uses a mixed formulation where fluxes and cell pressures are solved for simultaneously.

We discretize the saturation equation (5.3) using a standard upstream weighted implicit finite volume method to form

$$\mathbf{s}^n = \mathbf{s}^{n-1} + \triangle t^n \mathbf{D}_{PV}^{-1}\left(\mathbf{A}\left(\mathbf{v}^n\right) f_w\left(\mathbf{s}^n\right) + \mathbf{q}\left(\mathbf{v}^n\right)_+\right). \tag{5.7}$$

Here, $\triangle t^n$ is the time step and $\mathbf{D}_{PV}$ is the diagonal matrix containing the grid block pore volumes. The matrix $\mathbf{A}(\mathbf{v}^n)$ is the sparse flux matrix based on the upstream weighted discretization scheme, and $\mathbf{q}(\mathbf{v}^n)_+$ is the vector of positive sources (in this setting, water injection rates). We note that the matrix $\mathbf{A}$ and vector $\mathbf{q}$ are linear functions of $\mathbf{v}^n$. The discretized saturation equation (5.7) is solved implicitly for the current time step $n+1$ using a Newton-Raphson method.

As seen (5.6) and (5.7) are coupled. The solution strategy to solve these equations is first solving the discretized pressure equation (5.6) using initial water saturation values, and then solve the discretized saturation equation (5.7). This procedure is repeated forward in time until the final time is reached. This kind of solution strategy is known as a sequential-splitting method Aarnes et al. [2007]. The model used in this work is implemented in Lie et al. [2011]. For convenience, we write the discrete state equations (5.6) and (5.7) in an implicit form $\mathbf{F}(\mathbf{x},\mathbf{u})$ as

$$\begin{aligned}
\mathbf{F}(\widetilde{\mathbf{x}},\widetilde{\mathbf{u}}) &= \begin{pmatrix} \mathbf{F}^1\left(\mathbf{p}^1,\mathbf{s}^0,\mathbf{s}^1,\mathbf{u}^1\right) \\ \vdots \\ \mathbf{F}^N\left(\mathbf{p}^N,\mathbf{s}^{N-1},\mathbf{s}^N,\mathbf{u}^N\right), \end{pmatrix} \\
\mathbf{x}^{nT} &= (\mathbf{p}^{nT},\mathbf{s}^{nT}), \quad n=1,...,N, \\
\widetilde{\mathbf{x}}^T &= (\mathbf{x}^{1T},...,\mathbf{x}^{NT}), \\
\widetilde{\mathbf{u}}^T &= (\mathbf{u}^{1T},...,\mathbf{u}^{NT}).
\end{aligned} \tag{5.8}$$

The state vectors and control input vectors are stacked for all time instances from $n = 1,\ldots,N$.

5.2.2 Adjoint Equations

Let $\mathscr{J}(\widetilde{\mathbf{x}},\widetilde{\mathbf{u}}) = \sum_{n=1}^{N} \mathscr{J}^n(\mathbf{x}^n,\mathbf{u}^n)$ be an objective function denoted $\nabla_{\widetilde{\mathbf{u}}}\mathscr{J}$ the gradient with respect to a control input $\widetilde{\mathbf{u}}$. The detailed description of the objective function $\mathscr{J}(\widetilde{\mathbf{x}},\widetilde{\mathbf{u}})$ will be explained in Section 5.3. We then construct an augmented objective function or Lagrangian functional

$$\begin{aligned}
\mathscr{L}(\widetilde{\mathbf{x}},\widetilde{\mathbf{u}},\boldsymbol{\lambda}) &= \mathscr{J}(\widetilde{\mathbf{x}},\widetilde{\mathbf{u}}) + \boldsymbol{\lambda}^T\mathbf{F}(\widetilde{\mathbf{x}},\widetilde{\mathbf{u}}) \\
&= \sum_{n=1}^{N}\left(\mathscr{J}^n\left(\mathbf{x}^n,\mathbf{u}^n\right) + \boldsymbol{\lambda}^{nT}\mathbf{F}\left(\mathbf{x}^n,\mathbf{x}^{n-1},\mathbf{u}^n\right)\right),
\end{aligned} \tag{5.9}$$

for $n = 1,\ldots,N$, where

$$\begin{aligned}
\boldsymbol{\lambda}^{nT}F &= \boldsymbol{\lambda}_v^{nT}\left(\mathbf{B}^n\mathbf{v}^n - \mathbf{C}\mathbf{p}^n + \mathbf{D}\boldsymbol{\pi}^n\right) \\
&+ \boldsymbol{\lambda}_{q_w}^{nT}\left(-\mathbf{B}_w^n\mathbf{q}_w - \mathbf{C}_w\mathbf{p}^n + \mathbf{D}_{w,N}\mathbf{p}_{w,N}^n\left(\mathbf{u}^n\right)\right) \\
&+ \boldsymbol{\lambda}_p^{nT}\left(\mathbf{C}^T\mathbf{v}^n - \mathbf{C}_w^T\mathbf{q}_w^n\right) \\
&+ \boldsymbol{\lambda}_\pi^{nT}\mathbf{D}^T\mathbf{v}^n \\
&+ \boldsymbol{\lambda}_{p_{w,N}}^{nT}\left(-\mathbf{D}_{w,N}^T\mathbf{q}_w^n + \mathbf{q}_{tot,N}^n\left(\mathbf{u}^n\right)\right) \\
&+ \boldsymbol{\lambda}_s^{nT}\left(\mathbf{s}^n - \mathbf{s}^{n-1} - \triangle t^n\mathbf{i}^n\right).
\end{aligned}$$

Here $\mathbf{i}^n = \mathbf{D}_{PV}^{-1}\left(\mathbf{A}(\mathbf{v}^n)\, f_w(\mathbf{s}^n) + \mathbf{q}(\mathbf{v}^n)_+\right)$. By choosing $\boldsymbol{\lambda}$ that makes $\nabla_{\widetilde{\mathbf{x}}}\mathscr{L} = \mathbf{0}$, we arrive at the adjoint equations

$$\left(\frac{\partial F\left(\mathbf{x}^n,\mathbf{x}^{n-1},\mathbf{u}^n\right)}{\partial\mathbf{x}^n}\right)^T\boldsymbol{\lambda}^n + \left(\frac{\partial F\left(\mathbf{x}^{n+1},\mathbf{x}^n,\mathbf{u}^{n+1}\right)}{\partial\mathbf{x}^n}\right)^T\boldsymbol{\lambda}^{n+1} = -\left(\frac{\partial\mathscr{J}^n(\mathbf{x}^n,\mathbf{u}^n)}{\partial\mathbf{x}^n}\right)^T, \tag{5.10}$$

for $n = N, \ldots, 1$. The details of (5.10) are

$$\begin{pmatrix} \mathbf{B}^n\left(\mathbf{s}^{n-1}\right) & \mathbf{0} & \mathbf{C} & \mathbf{D} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_w^n\left(\mathbf{s}^{n-1}\right) & \mathbf{C}_w & \mathbf{0} & \mathbf{D}_{w,N} \\ \mathbf{C}^T & \mathbf{C}_w^T & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{D}^T & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_{w,N}^T & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\lambda}_v^n \\ \boldsymbol{\lambda}_{q_w}^n \\ \boldsymbol{\lambda}_p^n \\ \boldsymbol{\lambda}_\pi^n \\ \boldsymbol{\lambda}_{p_{w,N}}^n \end{pmatrix} = \begin{pmatrix} \left(\frac{\partial \mathbf{i}^n}{\partial \mathbf{v}^n}\right)^T \boldsymbol{\lambda}_s^n - \left(\frac{\partial \mathscr{J}}{\partial \mathbf{v}^n}\right)^T \\ \left(\frac{\partial \mathscr{J}^n}{\partial \mathbf{q}_w^n}\right)^T \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \tag{5.11}$$

for the corresponding pressure equation and the following equation for the saturation

$$\begin{aligned} \left(\mathbf{I} - \triangle t \left(\frac{\partial \mathbf{i}^n}{\partial \mathbf{s}^n}\right)^T\right) \boldsymbol{\lambda}_s^n &= \boldsymbol{\lambda}_s^{n+1} - \left(\frac{\partial \mathscr{J}^n}{\partial \mathbf{s}^n}\right)^T \\ &- \left(\frac{\partial}{\partial \mathbf{s}^n}\left(\mathbf{B}^{n+1}\mathbf{v}^{n+1}\right)\right)^T \boldsymbol{\lambda}_v^{n+1} \\ &+ \left(\frac{\partial}{\partial \mathbf{s}^n}\left(\mathbf{B}_w^{n+1}\mathbf{q}_w^{n+1}\right)\right)^T \boldsymbol{\lambda}_{q_w}^{n+1}. \end{aligned} \tag{5.12}$$

Using the fact that at the final time $\boldsymbol{\lambda}_\alpha^N = \mathbf{0}$ for $\alpha = \{v, q_w, p, \pi, p_{w,N}, s\}$, we are able to compute the Lagrangian multipliers for each time step backward in time. It should be noted that (5.11) and (5.12) are linear equations and they are solved using the direct sparse method as well. Finally using the obtained Lagrangian multipliers values, the gradient with respect to $\widetilde{\mathbf{u}}$ is

$$\nabla_{\widetilde{\mathbf{u}}} \mathscr{L}^n = \frac{\partial \mathscr{J}^n(\mathbf{x}^n, \mathbf{u}^n)}{\partial \mathbf{u}^n} + \boldsymbol{\lambda}^{nT} \frac{\partial F\left(\mathbf{x}^n, \mathbf{x}^{n-1}, \mathbf{u}^n\right)}{\partial \mathbf{u}^n}. \tag{5.13}$$

5.2.3 Reduced-order Models

Vanilla POD method

In order to build a reduced-order model based on the POD method, we need to take snapshots of the high-fidelity model described in (5.6) and (5.7). Let $\mathbf{x} = \begin{bmatrix} \mathbf{q}_w & \mathbf{p} & \mathbf{p}_{w,N} & \mathbf{s} \end{bmatrix} \in \mathbb{R}^{n_x}$ be the snapshot of the solution of the forward equations with n_x as the dimension

of the solution. Given a set of snapshots $\{\mathbf{x}_1,\ldots,\mathbf{x}_\Xi\}\in\mathbb{R}^{n_x\times\Xi}$ and let $\mathcal{V}=\text{span}\,\{\mathbf{x}_1,\ldots,\mathbf{x}_\Xi\}$, the POD basis function is a solution of an optimization problem for finding orthonormal vectors $\{\boldsymbol{\psi}_i\}_{i=1}^{\ell}$, where $\ell\le\text{rank}\,(\mathcal{V})$. The optimization formulation is

$$\begin{aligned}\min_{\{\boldsymbol{\psi}_i\}_{i=1}^{\ell}}\mathcal{J}\left(\boldsymbol{\psi}_1,\ldots,\boldsymbol{\psi}_\ell\right):=\quad&\sum_{j=1}^{\Xi}\left\|\mathbf{x}_j-\sum_{i=1}^{\ell}\left(\mathbf{x}_j^T\boldsymbol{\psi}_i\right)\boldsymbol{\psi}_i\right\|_2^2\\ \text{subject to}\quad&\boldsymbol{\psi}_i^T\boldsymbol{\psi}_j=\boldsymbol{\delta}_{ij}=\begin{cases}1 & \text{if } i=j\\ 0 & \text{otherwise}\end{cases}.\end{aligned}\tag{5.14}$$

We define a Lagrangian functional

$$\mathcal{L}\left(\boldsymbol{\psi}_1,\ldots,\boldsymbol{\psi}_\ell,\boldsymbol{\lambda}_{11},\ldots,\boldsymbol{\lambda}_{\ell\ell}\right)=\mathcal{J}\left(\boldsymbol{\psi}_1,\ldots,\boldsymbol{\psi}_\ell\right)+\sum_{i,j=1}^{\ell}\boldsymbol{\lambda}_{ij}\left(\boldsymbol{\psi}_i^T\boldsymbol{\psi}_j-\boldsymbol{\delta}_{ij}\right).\tag{5.15}$$

The necessary optimality conditions, $\frac{\partial\mathcal{L}}{\partial\boldsymbol{\psi}_i}=0$ and $\frac{\partial\mathcal{L}}{\partial\boldsymbol{\lambda}_{ij}}=0$, give us an eigenvalue problem

$$\sum_{j=1}^{\Xi}\mathbf{x}_j\left(\mathbf{x}_j^T\boldsymbol{\psi}_i\right)=\boldsymbol{\lambda}_{ii}\boldsymbol{\psi}_i,\ \text{for}\ i=1,\ldots,\ell\tag{5.16}$$

or by setting $\boldsymbol{\lambda}_i=\boldsymbol{\lambda}_{ii}$ and $\mathbf{X}=[\mathbf{x}_1,\ldots,\mathbf{x}_\Xi]\in\mathbb{R}^{\text{n}_\text{x}\times\Xi}$, then the problem reads

$$\mathbf{X}\mathbf{X}^T\boldsymbol{\psi}_i=\boldsymbol{\lambda}_i\boldsymbol{\psi}_i,\ \text{for}\ i=1,\ldots,\ell.\tag{5.17}$$

To compute the solution of (5.17), we decompose the vector $\mathbf{X}$ using singular value decomposition (SVD), that is,

$$\mathbf{X}=\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T,\tag{5.18}$$

where $\mathbf{U}=\left[\mathbf{u}_1,\ldots,\mathbf{u}_{n_x}\right]\in\mathbb{R}^{\text{n}_\text{x}\times\text{n}_\text{x}}$ and $\mathbf{V}=[\mathbf{v}_1,\ldots,\mathbf{v}_\Xi]\in\mathbb{R}^{\Xi\times\Xi}$ are orthogonal matrices, and $\boldsymbol{\Sigma}\in\mathbb{R}^{\text{n}_\text{x}\times\Xi}$ is the pseudo-diagonal matrix with diagonal arranged in a decreased order, that is, $\sigma_1\ge\sigma_2\ge\ldots\ge\sigma_\Xi\ge0$. In other words,

$$\mathbf{U}^T\mathbf{X}\mathbf{V}=\boldsymbol{\Sigma}.\tag{5.19}$$

Moreover, it can be shown for $1\le i\le\Xi$ that

$$\mathbf{X}\mathbf{v}_i=\sigma_i\mathbf{u}_i,\ \mathbf{X}^T\mathbf{u}_i=\sigma_i\mathbf{v}_i,\ \mathbf{X}\mathbf{X}^T\mathbf{u}_i=\sigma_i^2\mathbf{u}_i.\tag{5.20}$$

The solution of problem (5.17) is a POD basis $\boldsymbol{\psi}_i=\mathbf{u}_i$ and $\lambda_i=\sigma_i^2>0$ for $i=1,\ldots,\ell\le d=\dim\mathcal{V}$. The minimized objective function (5.14) is then

$$\mathcal{J}\left(\boldsymbol{\psi}_1,\ldots,\boldsymbol{\psi}_\ell\right):=\sum_{j=1}^{\Xi}\left\|\mathbf{x}_j-\sum_{i=1}^{\ell}\left(\mathbf{x}_j^T\boldsymbol{\psi}_i\right)\boldsymbol{\psi}_i\right\|_2^2=\sum_{i=\ell+1}^{d}\lambda_i.\tag{5.21}$$

To determine the dimension of ℓ, the singular value is cut according to the following '*energy*' truncation

$$E = \frac{\sum_{i=1}^{\ell} \sigma_i}{\sum_{i=1}^{\Xi} \sigma_i} < \alpha, \tag{5.22}$$

where typically $0.9 \leq \alpha < 1$. This choice of truncation is a rather heuristic consideration Volkwein [2003]. We follow what is commonly used in the literature. In other work, one may use quadratic summation of the singular value, see e.g., van Doren et al. [2006]; Markovinovic & Jansen [2006].

The POD method is applied to the state and adjoint equations. Let ℓ_p, ℓ_s, and n_p, n_s be the dimension of the pressure and saturation equations in reduced-order and high-fidelity models respectively, where $\ell_p < n_p$ and $\ell_s < n_s$. Then transformation from the reduced-order to the high-fidelity model is

$$\begin{aligned} \mathbf{x}_p &= \mathbf{V}_p \hat{\mathbf{x}}_p + \bar{\mathbf{x}}_p, \\ \mathbf{x}_s &= \mathbf{V}_s \hat{\mathbf{x}}_s + \bar{\mathbf{x}}_s. \end{aligned} \tag{5.23}$$

The high-fidelity model is represented by $\mathbf{x}_p \in \mathbb{R}^{n_p}$, $\mathbf{x}_s \in \mathbb{R}^{n_s}$ and their respective averages during the snapshots $\bar{\mathbf{x}}_p \in \mathbb{R}^{n_p}$ and $\bar{\mathbf{x}}_s \in \mathbb{R}^{n_s}$. In the reduced-order space, $\hat{\mathbf{x}}_p \in \mathbb{R}^{\ell_p}$ and $\hat{\mathbf{x}}_s \in \mathbb{R}^{\ell_s}$, the forward equations now become

$$\begin{aligned} \mathbf{V}_p^T \begin{pmatrix} \mathbf{B}^n\left(\mathbf{s}^{n-1}\right) & \mathbf{0} & \mathbf{C} & \mathbf{D} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_w^n\left(\mathbf{s}^{n-1}\right) & \mathbf{C}_w & \mathbf{0} & \mathbf{D}_{w,N} \\ \mathbf{C}^T & \mathbf{C}_w^T & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{D}^T & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_{w,N}^T & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix} \mathbf{V}_p \hat{\mathbf{x}}_p^n &= \\ \mathbf{V}_p^T \left(\begin{pmatrix} \mathbf{0} \\ -\mathbf{D}_{w,D} \mathbf{p}_{w,D}^n\left(\mathbf{u}^n\right) \\ \mathbf{0} \\ \mathbf{0} \\ -\mathbf{q}_{tot,N}^n\left(\mathbf{u}^n\right) \end{pmatrix} - \begin{pmatrix} \mathbf{B}^n\left(\mathbf{s}^{n-1}\right) & \mathbf{0} & \mathbf{C} & \mathbf{D} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_w^n\left(\mathbf{s}^{n-1}\right) & \mathbf{C}_w & \mathbf{0} & \mathbf{D}_{w,N} \\ \mathbf{C}^T & \mathbf{C}_w^T & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{D}^T & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_{w,N}^T & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix} \bar{\mathbf{x}}_p \right) \end{aligned} \tag{5.24}$$

$$\hat{\mathbf{s}}^n = \hat{\mathbf{s}}^{n-1} + \triangle t \mathbf{V}_s^T \mathbf{D}_{PV}^{-1} \left(\mathbf{A}\left(\mathbf{v}^n\right) f_w\left(\mathbf{V}_s \hat{\mathbf{s}}^n + \bar{\mathbf{s}}\right) + \mathbf{q}\left(\mathbf{v}^n\right)_+ \right). \tag{5.25}$$

Similarly, we also take snapshots of the adjoint equations (5.11) and (5.12) and obtain reduced-order adjoint equations. The reduced-order corresponding adjoint pressure and saturation respectively are

$$\mathbf{V}_{ap}^T \begin{pmatrix} \mathbf{B}^n\left(\mathbf{s}^{n-1}\right) & \mathbf{0} & \mathbf{C} & \mathbf{D} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_w^n\left(\mathbf{s}^{n-1}\right) & \mathbf{C}_w & \mathbf{0} & \mathbf{D}_{w,N} \\ \mathbf{C}^T & \mathbf{C}_w^T & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{D}^T & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_{w,N}^T & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix} \mathbf{V}_{ap} \begin{pmatrix} \hat{\boldsymbol{\lambda}}_v^n \\ \hat{\boldsymbol{\lambda}}_{q_w}^n \\ \hat{\boldsymbol{\lambda}}_p^n \\ \hat{\boldsymbol{\lambda}}_\pi^n \\ \hat{\boldsymbol{\lambda}}_{p_{w,N}}^n \end{pmatrix} =$$

$$\mathbf{V}_{ap}^T \left\{ \begin{pmatrix} \left(\frac{\partial \mathbf{i}^n}{\partial \mathbf{v}^n}\right)^T \boldsymbol{\lambda}_s^n - \left(\frac{\partial \mathscr{J}}{\partial \mathbf{v}^n}\right)^T \\ \left(\frac{\partial \mathscr{J}^n}{\partial \mathbf{q}_w^n}\right)^T \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix} - \begin{pmatrix} \mathbf{B}^n\left(\mathbf{s}^{n-1}\right) & \mathbf{0} & \mathbf{C} & \mathbf{D} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_w^n\left(\mathbf{s}^{n-1}\right) & \mathbf{C}_w & \mathbf{0} & \mathbf{D}_{w,N} \\ \mathbf{C}^T & \mathbf{C}_w^T & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{D}^T & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_{w,N}^T & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \bar{\boldsymbol{\lambda}}_v \\ \bar{\boldsymbol{\lambda}}_{q_w} \\ \bar{\boldsymbol{\lambda}}_p \\ \bar{\boldsymbol{\lambda}}_\pi \\ \bar{\boldsymbol{\lambda}}_{p_{w,N}} \end{pmatrix} \right\}, \quad (5.26)$$

$$\begin{aligned} \mathbf{V}_{as}^T \left(\mathbf{I} - \triangle t \left(\frac{\partial \mathbf{i}^n}{\partial \mathbf{s}^n}\right)^T\right) \mathbf{V}_{as} \hat{\boldsymbol{\lambda}}_s^n &= \mathbf{V}_{as}^T \left\{ \boldsymbol{\lambda}_s^{n+1} - \left(\frac{\partial \mathscr{J}^n}{\partial \mathbf{s}^n}\right)^T - \left(\frac{\partial}{\partial \mathbf{s}^n}\left(\mathbf{B}^{n+1}\mathbf{v}^{n+1}\right)\right)^T \boldsymbol{\lambda}_v^{n+1} \right\} \\ &+ \mathbf{V}_{as}^T \left\{ \left(\frac{\partial}{\partial \mathbf{s}^n}\left(\mathbf{B}_w^{n+1}\mathbf{q}_w^{n+1}\right)\right)^T \boldsymbol{\lambda}_{q_w}^{n+1} \right\} \\ &- \mathbf{V}_{as}^T \left\{ \left(\mathbf{I} - \triangle t \left(\frac{\partial \mathbf{i}^n}{\partial \mathbf{s}^n}\right)^T\right) \bar{\boldsymbol{\lambda}}_s \right\}, \end{aligned} \quad (5.27)$$

where $\mathbf{V}_{ap}$ and $\mathbf{V}_{as}$ are the basis functions for the corresponding adjoint pressure and saturation equations, and $\bar{\boldsymbol{\lambda}}$ is the average snapshot of Lagrangian multipliers of adjoint equation solutions. As seen in all reduced-order equations (5.11), (5.12), (5.26) and (5.27), the solution of full-space equations are needed in order to solve the reduced-order solutions. Hence, after solving the reduced-order equations we reconstruct the full-space solution through the transformation (5.23). This will inevitably give an overhead in the computational time. After all, we still gain computational reduction in CPU time compare to the high-fidelity model run.

Empirical Interpolation Method (EIM)

We now describe the EIM method and its discrete variant EIM based on the work of Chaturantabut & Sorensen [2010] and Barrault et al. [2004]. The empirical interpolation method (EIM) was first proposed in Barrault et al. [2004] and later is more elaborated in Grepl et al. [2007] within the context of the reduced-basis (RB) method (see http://augustine.mit.edu/). The method seeks to approximate a function in spatial domain by selecting interpolation points which represent the largest errors between a given set of basis functions and their normalized values.

To explain the idea, we consider a non-affine parameter dependent function $d(\cdot;\theta) \in L^\infty(\Omega)$, where Ω is a spatial domain and θ is a parameter in the domain. Our goal is to approximate the function d by $\hat{d}$ given a set of basis functions $\{q_1,\ldots,q_m\}$, where m is the dimension of a low-dimensional space defined by $\mathscr{M}^d \equiv \{d(\cdot;\theta) : \theta \in \mathscr{D}\}$. $\mathscr{D}$ is the parameter domain. The variation in $\mathscr{M}^d$ can be represented in some selected points $\{z_1,\ldots,z_m\} \in \Omega$. The approximation of d by $\hat{d}$ is

$$\begin{aligned} \hat{d}(x;\theta) &= \sum_{\ell=1}^{m} q_\ell(x)\,\beta_\ell(\theta), \\ \hat{d}(z_i;\theta) &= d(z_i;\theta) \qquad i = 1,\ldots,m. \end{aligned} \tag{5.28}$$

The coefficient β in the form $\beta(\theta) = \left[\beta_1(\theta),\ldots,\beta_m(\theta)\right]^T$ is determined in the following way. Given basis function $Q(x) = \left[q_1(x),\ldots,q_m(x)\right]$ and interpolation points $\mathbf{z} = [z_1,\ldots,z_m]^T \in \Omega^m$. Function evaluation at $\mathbf{z}$ is $d(\mathbf{z};\theta) = [d(z_1;\theta),\ldots,d(z_m,\theta)]^T$, and $Q(\mathbf{z}) = \begin{bmatrix} q_1(z_1) & q_2(z_1) & \ldots & q_m(z_1) \\ q_1(z_2) & q_2(z_2) & \ldots & q_m(z_2) \\ \vdots & \vdots & \ddots & \vdots \\ q_1(z_m) & q_2(z_m) & \ldots & q_m(z_m) \end{bmatrix} \in \mathbb{R}^{m\times m}$. From the relation $\hat{d}(\mathbf{x};\theta) = Q(x)\,\beta(\theta)$ and $\hat{d}(\mathbf{z};\theta) = d(\mathbf{z};\theta)$, the coefficient β is $\beta(\theta) = (Q(\mathbf{z}))^{-1} d(\mathbf{z};\theta)$. Finally, the approximated function is $\hat{d}(\mathbf{x};\theta) = Q(\mathbf{x})\,(Q(\mathbf{z}))^{-1} d(\mathbf{z};\theta)$.

Now, in case the interpolation points are not given, the work of Barrault et al. [2004] proposes the EIM Algorithm 11 below to select the interpolation points $\mathbf{z}$.

Algorithm 11 Empirical Interpolation Method

INPUT: A set of basis functions $\{\xi_i\}_{i=1}^m$
OUTPUT: EIM points $\{z_1,\ldots,z_m\}$ and normalized basis functions $\{q_1,\ldots,q_m\}$
1. Set $z_1 = \arg\operatorname{ess}\, sup_{x\in\Omega} |\xi_1(x)|$.
   $q_1 = \frac{\xi_1(x)}{\xi_1(z_1)}$;
   $\mathbf{Q}^1\left(\mathbf{Z}^1\right) = q_1(z_1) = 1$.
2. **for** $L = 2$ to m **do**
   Solve ρ^{L-1} from: $\mathbf{Q}^{L-1}\left(\mathbf{z}^{L-1}\right)\rho^{L-1} = \xi_L\left(\mathbf{z}^{L-1}\right)$
   Define $r_L(x) = \xi_L(x) - \mathbf{Q}^{L-1}(x)\,\rho^{L-1}$
   Set $z_L = \arg\operatorname{ess}\, sup_{x\in\Omega} |r_L(x)|$
   $q_L(x) = \frac{r_L(x)}{r_L(z_L)}$.
end for

For clarity, Figure 5.1 describes how the interpolation points are selected. Notice that the current EIM point in the figure located when the residual between the original (input given) to its normalized value is the largest.

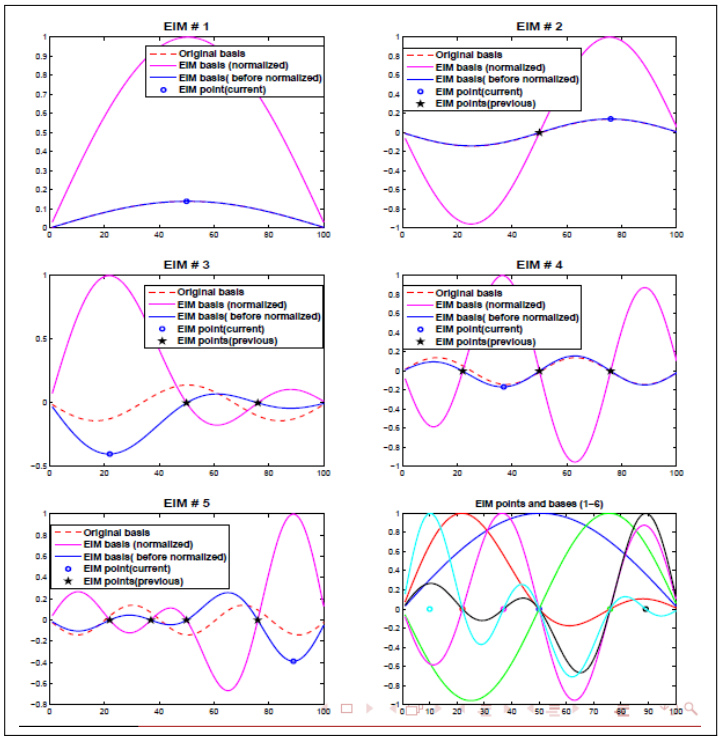

Figure 5.1: This figure shows how the interpolation points are selected. This figure is taken from Chaturantabut & Sorensen [2010] without any modification.

POD-DEIM

The set of basis functions used in the EIM method can be some choices. In the RB method, a greedy basis function algorithm is usually used. The work of Chaturantabut & Sorensen [2010] uses the POD basis functions as the given basis. Furthermore, the POD-DEIM applies specifically the EIM method to systems which contain nonlinear terms. How the POD-DEIM works is described in this subsection.

Let us consider the water saturation equation (5.7) in the following form

$$\mathbf{s}^{n+1} = \mathbf{s}^n + \triangle t^n \mathbf{D}_{PV}^{-1} \left(\mathbf{A}\left(\mathbf{v}^n\right) f_w\left(\mathbf{s}^n\right) + \mathbf{q}\left(\mathbf{v}^n\right)_+ \right). \tag{5.29}$$

This equation is solved for next time step $n+1$ implicitly using Newton-Raphson method, that is,

$$0 \equiv G\left(\mathbf{s}^{n+1}\right) = \mathbf{s}^{n+1} - \mathbf{s}^n - \triangle t^n \mathbf{D}_{PV}^{-1} \left(\mathbf{A}\left(\mathbf{v}^n\right) f_w\left(\mathbf{s}^n\right) + \mathbf{q}\left(\mathbf{v}^n\right)_+ \right). \tag{5.30}$$

Given an initial guess $\tilde{\mathbf{s}}$, then by Taylor expansion, the equation above is approximated by

$$0 = G\left(\mathbf{s}^{n+1}\right) \approx G\left(\tilde{\mathbf{s}}\right) + G'\left(\tilde{\mathbf{s}}\right)\left(\mathbf{s}^{n+1} - \tilde{\mathbf{s}}\right), \tag{5.31}$$

where the solution $\mathbf{s}^{n+1}$ is obtained through $\mathbf{s}^{n+1} = \tilde{\mathbf{s}} + d\tilde{\mathbf{s}}$. The changes $d\tilde{\mathbf{s}} = \mathbf{s}^{n+1} - \tilde{\mathbf{s}}$ satisfies the linear equation $-G'(\tilde{\mathbf{s}})\, d\tilde{\mathbf{s}} = G(\tilde{\mathbf{s}})$. The Jacobian $G'(\tilde{\mathbf{s}})$ is

$$G'(\tilde{\mathbf{s}}) = \mathbf{I} - \triangle t^n \mathbf{D}_{PV}^{-1} \mathbf{A}\left(\mathbf{v}^n\right) f_w^{'}\left(\mathbf{s}^n\right)$$

Note that the terms $f_w^{'}(\mathbf{s}^n)$ and $f_w\left(\mathbf{s}^{n+1}\right)$ are evaluated componentwise, which means that they are evaluated at each gridblock.

As seen in (5.25), even in the reduced-order space we still need to evaluate the nonlinear term, which is in this case is water cut, in full-space. Similarly, in solving (5.31) in the reduced-order space, we evaluate the Jacobian in full-space. To mitigate this, we construct another reduced-order model for the water cut term. This is when the POD-DEIM [Chaturantabut & Sorensen, 2010] comes into play. The method projects the nonlinear term onto a lower dimension, such that

$$\mathbf{f}(\tau) \simeq \mathbf{\Phi} c(\tau) + \bar{\mathbf{f}} \tag{5.32}$$

where $\mathbf{\Phi} = [\Phi_1, \ldots, \Phi_m] \in \mathbb{R}^{\mathrm{n_x} \times \mathrm{m}}$, $c(\tau)$ is the corresponding vector coefficient, and $\bar{\mathbf{f}}$ is the average value of the nonlinear term in the snapshot. The vector $c(\tau)$ is determined by selecting appropriate m rows from the overdetermined $\mathbf{f}(\tau) \simeq \mathbf{\Phi} c(\tau) + \bar{\mathbf{f}}$. The selection is done by a matrix $\mathbf{P} = \left[\mathbf{e}_{\wp_1}, \ldots, \mathbf{e}_{\wp_m}\right] \in \mathbb{R}^{n_x \times m}$, in a way such that $\mathbf{P}^T \mathbf{f}(\tau) = \mathbf{P}^T \left(\mathbf{\Phi} c(\tau) + \bar{\mathbf{f}}\right)$ and after some arrangement the full space nonlinear term is reconstructed as follows

$$\mathbf{f}(\tau) = \left\{\mathbf{\Phi}\left(\mathbf{P}^T \mathbf{\Phi}\right)^{-1} \mathbf{P}^T \left(\mathbf{f}(\tau) - \bar{\mathbf{f}}\right)\right\} + \bar{\mathbf{f}}. \tag{5.33}$$

We now need to construct the $\mathbf{\Phi}$ and $\mathbf{P}$ matrices. $\mathbf{\Phi}$ is selected as the POD basis function, while $\mathbf{P}$ is determined by Algorithm 12. Algorithm 12 is similar to Algorithm 11, except the output from Algorithm 12 is only a set of indices or interpolation points as in Algorithm 11.

Algorithm 12 POD-DEIM

INPUT: $\{\Phi_l\}_{l=1}^m \subset \mathbb{R}^{n_x}$

OUTPUT: $\vec{\wp} = \left[\wp_1, \ldots, \wp_m\right]^T \in \mathbb{R}^m$

1. $\begin{bmatrix} \rho & \wp_1 \end{bmatrix} = \max\{|\Phi_1|\}$
2. $\mathbf{\Phi} = [\Phi_1]$, $\mathbf{P} = \left[\mathbf{e}_{\wp_1}\right]$, $\vec{\wp} = \left[\wp_1\right]$
3. **for** $l = 2$ to m **do**
   Solve $\left(\mathbf{P}^T \mathbf{\Phi}\right)\mathbf{c} = \mathbf{P}^T \Phi_l$ for $\mathbf{c}$
   $\mathbf{r} = \mathbf{u}_l - \mathbf{\Phi}\mathbf{c}$
   $\begin{bmatrix} \rho & \wp_l \end{bmatrix} = \max\{|\mathbf{r}|\}$
   $\mathbf{\Phi} \leftarrow \begin{bmatrix} \mathbf{U} & \Phi_l \end{bmatrix}$, $\mathbf{P} \leftarrow \begin{bmatrix} \mathbf{P} & \mathbf{e}_{\wp_l} \end{bmatrix}$, $\vec{\wp} \leftarrow \begin{bmatrix} \vec{\wp} \\ \wp_l \end{bmatrix}$

end for

We employ the POD-DEIM just for the forward saturation equation, since this is the only equation that contains the nonlinear water cut term. So now the reduced-order equation of water saturation is

$$\hat{\mathbf{s}}^n = \hat{\mathbf{s}}^{n-1} + \triangle t \mathbf{V}_s^T \mathbf{D}_{PV}^{-1} \left(\mathbf{A}\left(\mathbf{v}^n\right) \left\{ \left(\mathbf{\Phi}^T \left(\mathbf{P}^T \mathbf{\Phi} \right)^{-1} \mathbf{P}^T \left(f_w \left(\mathbf{V}_s \hat{\mathbf{s}}^n + \bar{\mathbf{s}} \right) - \bar{f}_w \right) \right) + \bar{f}_w \right\} + \mathbf{q}\left(\mathbf{v}^n\right)_+ \right). \tag{5.34}$$

One may notice there is a nonlinear dependence in the pressure equation (5.6) as well, that is, in the term $\mathbf{B}^n\left(\mathbf{s}^{n-1}\right)$, involving water saturation from the previous time step. Because (5.6) is a linear equation and the POD method has proved to be good for linear terms, we do not apply POD-DEIM for the pressure equations. Nevertheless, this opens an opportunity for future investigation.

5.3 Production Optimization Problem

By injecting water into reservoirs, cumulative oil production may increase. This is cast as the following optimization problem using a reduced-order model

$$\begin{aligned} (\hat{\mathscr{P}}) \quad & \max_{\widetilde{\mathbf{u}} \in \mathbb{R}^{n_{\widetilde{\mathbf{u}}}}} \mathscr{J}\left(\widetilde{\hat{\mathbf{x}}}, \widetilde{\mathbf{u}}\right) \\ \text{subject to}: \quad & \mathbf{F}\left(\widetilde{\hat{x}}, \widetilde{\mathbf{u}}\right) = 0 \\ & g\left(\mathbf{u}^n\right) \geq 0, \ \forall n = 1, \ldots, N \\ & h\left(\widetilde{\hat{\mathbf{x}}}^n, \mathbf{u}^n\right) \geq 0, \ \forall n = 1, \ldots, N \\ & \mathbf{x}^0 \text{ is given.} \end{aligned}$$

$\mathscr{J}\left(\widetilde{\hat{\mathbf{x}}}, \widetilde{\mathbf{u}}\right)$, $g(\mathbf{u}^n)$, $h(\hat{\mathbf{x}}^n, \mathbf{u}^n)$ are assumed $\mathscr{C}^1$. The reduced-order state is represented by $\widetilde{\hat{\mathbf{x}}}$, while the full-space system is in (5.8). The control input and the state constraints are represented by $g : \mathbb{R}^{n_{\widetilde{u}}} \rightarrow \mathbb{R}^{n_g}$ and $h : \mathbb{R}^{n_x \times n_{\widetilde{u}}} \rightarrow \mathbb{R}^{n_h}$, respectively. The objective function is given by $\mathscr{J} : \mathbb{R}^{n_{\widetilde{\hat{\mathbf{x}}}} \times n_{\widetilde{\mathbf{u}}}} \rightarrow \mathbb{R}$ and the state equations are posed as implicit constraints. The state variables and the control inputs are dependent, therefore we are able to perform the optimization in the control input space of $\widetilde{\mathbf{u}}$ instead of in the space of $\left(\widetilde{\hat{\mathbf{x}}}, \widetilde{\mathbf{u}}\right)$. To this end, we denote the objective objective as $\mathscr{J}(\widetilde{\mathbf{u}})$ omitting $\mathscr{J}\left(\widetilde{\hat{\mathbf{x}}}(\widetilde{\mathbf{u}}), \widetilde{\mathbf{u}}\right)$. In this work, we use the recovery factor (RF) (or net present value (NPV) used in the previous chapters) as the objective function

$$\mathscr{J}(\widetilde{\mathbf{u}}) = \frac{\sum_{i=1}^{N_{gb}} \mathbf{D}_{PV_i} s_i^N}{V_{gb}} \times 100\%, \tag{5.35}$$

where $\mathbf{D}_{PV_i}$ is the diagonal element of the pore volume matrix, s_i^N is the water saturation at grid block i at the final time step N, N_{gb} is the number of grid blocks in the

reservoirs, and V_{gb} is the total volume of the reservoir. In other words, the recovery factor represents the percentage of oil that can be produced from the reservoirs. Water flooding can normally give recovery factor somewhere between 20% and 40%, meaning that 20% - 40% of the oil is extracted from the reservoir.

The optimization problem in full-space is described as

$$\begin{aligned} (\mathscr{P}) \quad & \max_{\widetilde{\mathbf{u}} \in \mathbb{R}^{n_{\widetilde{\mathbf{u}}}}} \mathscr{J}(\widetilde{\mathbf{x}}, \widetilde{\mathbf{u}}) \\ \text{subject to}: \quad & \mathbf{F}(\widetilde{\mathbf{x}}, \widetilde{\mathbf{u}}) = 0 \\ & g(\mathbf{u}^n) \geq 0, \ \forall n = 1, \ldots, N \\ & h(\widetilde{\mathbf{x}}^n, \mathbf{u}^n) \geq 0, \ \forall n = 1, \ldots, N \\ & \mathbf{x}^0 \text{ is given.} \end{aligned}$$

Let $\mathscr{U}_{opt}$ and $\hat{\mathscr{U}}_{opt}$ be the solutions of the optimization problem in full-space $\mathscr{P}$ and reduced-space $\hat{\mathscr{P}}$, respectively. In Hinze & Volkwein [2005], the error estimate between $\mathscr{U}_{opt}$ and $\hat{\mathscr{U}}_{opt}$ is

$$\left\| \mathscr{U}_{opt} - \hat{\mathscr{U}}_{opt} \right\| \sim \left\| \mathbf{x}^0 - \mathbf{\Phi} \hat{\mathbf{x}}^0 \right\| + \left\| \widetilde{\mathbf{x}} - \mathbf{\Phi} \widetilde{\hat{x}} \right\| + \| \boldsymbol{\lambda}_{\widetilde{\mathbf{x}}}(\widetilde{\mathbf{x}}(\widetilde{\mathbf{u}})) - \mathbf{\Phi}(\boldsymbol{\lambda}_{\widetilde{x}}(\widetilde{\mathbf{x}}(\widetilde{\mathbf{u}}))) \| + \sqrt{\sum_{i=\ell+1}^{d} \boldsymbol{\lambda}_i}. \tag{5.36}$$

Here Φ is the basis function (eigenvector) obtained from the POD method, $\boldsymbol{\lambda}_{\widetilde{\mathbf{x}}}$ is the Lagrangian multiplier in the adjoint equations (5.11) and (5.12), and the last term contains $\boldsymbol{\lambda}$ from the residual of POD approximation as in (5.21).

The second term of the error estimate (5.36) $\left\| \widetilde{\mathbf{x}} - \mathbf{\Phi} \widetilde{\hat{x}} \right\|$ can be eliminated by taking snapshots of the state equations. Similarly, the term $\| \boldsymbol{\lambda}_{\widetilde{\mathbf{x}}}(\widetilde{\mathbf{x}}(\widetilde{\mathbf{u}})) - \mathbf{\Phi}(\boldsymbol{\lambda}_{\widetilde{x}}(\widetilde{\mathbf{x}}(\widetilde{\mathbf{u}}))) \|$ may vanish by taking snapshots of the adjoint equations. We will proceed with this approach and will explain it in the next section.

5.4 Solution Method

In this section we explain how to use a reduced-order model to solve the optimization problem $\hat{\mathscr{P}}$. Here, we use interchangeably the term low-fidelity model referring to the reduced-order model and the term high-fidelity model for the full-space model.

5.4.1 Trust-region POD

The optimization is performed using a reduced-order model, which is known as surrogate optimization. The principle of surrogate optimization is depicted in Figure 5.2. During the course of optimization many simulation runs are needed therefore by using a reduced-order model the goal is to reduce runtime, while the optimization solution converges to that of the optimization using a high-fidelity model. Furthermore, as

mentioned in the introduction there are alternative ways to construct reduced-order models. Here we will use POD and DEIM described in the previous section.

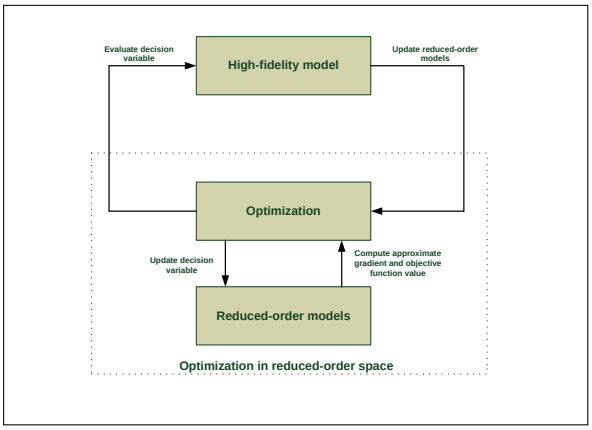

Figure 5.2: Optimization in reduced space (surrogate optimization). Optimization is performed using reduced-order models (ROMs) and the ROMS are updated according to the trust-region rule. This figure is modified after Alexandrov et al. [2001]

To maintain the quality of surrogate models, we apply a trust-region framework. The trust-region framework is used as the globalization strategy in gradient-based optimization Conn et al. [2000]. In a trust-region globalization strategy a quadratic approximation is used to approximate the objective function while in the surrogate optimization trust-region framework, which is called the trust-region POD (TRPOD) method, one builds a POD-based reduced-order model. The method will in principle enlarge its region when good approximations are obtained and reduce the region when the quality of model approximation is poor, or keep the region if the approximation quality is the same as at the previous iteration. The quality of approximation is measured by checking the value of the objective function in the full-space model. Finally, the method will terminate due to some stopping criteria. The details of this method is explained in Algorithm 13 with some remarks below. For sake of clarity, a pictorial sketch of TRPOD from Bergman et al. [2005] is displayed in Figure 5.3.

During the kth iteration the TRPOD method solves the following subproblem

$$\begin{aligned} \max_{\boldsymbol{\delta}\in\mathbb{R}^{n_{\widetilde{\mathbf{u}}}}} \quad & \mathscr{J}^{\mathscr{R}}_{b,k}(\widetilde{\mathbf{u}}+\boldsymbol{\delta}) && \\ \text{s.t.} \quad & \mathbf{F}_k\left(\widetilde{\hat{\mathbf{x}}},\widetilde{\mathbf{u}}+\boldsymbol{\delta}\right) && = 0 \\ & g_k(\mathbf{u}^n+\boldsymbol{\delta}) && \geq 0,\ \forall n=1,\ldots,N \\ & h_k\left(\widetilde{\hat{\mathbf{x}}}^n,\mathbf{u}^n+\boldsymbol{\delta}\right) && \geq 0,\ \forall n=1,\ldots,N \\ & \|\boldsymbol{\delta}\|_\infty \leq \triangle_k && . \end{aligned} \tag{5.37}$$

The optimized variables in the subproblem are the steps $\boldsymbol{\delta}$, where the length, expressed infinite norm, is bounded by a trust-region radius $\triangle_k$. $\mathscr{J}^{\mathscr{R}}_{b,k}$ is the modified objective function using the Lagrangian barrier method (5.39), explained in next subsection, evaluated using the (forward) reduced-order model.

Algorithm 13 Trust-region POD method

Step 0: Initialization Choose $0 < \eta_1 < \eta_2 < 1 \leq \eta_3$, $0 < \gamma_1 \leq \gamma_2 < 1 \leq \gamma_3$, an initial trust-region radius $\triangle_0$, minimum radius $\triangle_{min}$, maximum radius $\triangle_{max}$. Compute snapshots of forward and adjoint equations, $\mathscr{X}_0$ based on the initial control $\tilde{\mathbf{u}}_0$, compute $\mathscr{J}_b(\tilde{\mathbf{u}}_0)$, and set $k = 0$.

Step 1: Definition of POD-based models Compute POD basis functions for forward and adjoint equations and build reduced-order models for both equations.

Step 2: Step calculation Compute step $\boldsymbol{\delta}_k$ by solving the subproblem (5.37)

Step 3: Definition of trust-region ratio Compute new snapshots $\mathscr{X}_{k+}$ based on $\tilde{\mathbf{u}}_k + \boldsymbol{\delta}_k$ and $\mathscr{J}_b(\tilde{\mathbf{u}}_k + \delta_k)$.

Define the ratio

$$\rho_k = \frac{ared_k(\delta_k)}{pred_k(\delta_k)} = \frac{\mathscr{J}_b(\tilde{\mathbf{u}}_k + \boldsymbol{\delta}_k) - \mathscr{J}_b(\tilde{\mathbf{u}}_k)}{\mathscr{J}^{\mathscr{R}}_{b,k}(\tilde{\mathbf{u}}_k + \boldsymbol{\delta}_k) - \mathscr{J}^{\mathscr{R}}_{b,k}(\tilde{\mathbf{u}}_k)}.$$

Step 4: Trust-region update:

- If $\rho_k \geq \eta_3$:
  Set $\tilde{\mathbf{u}}_{k+1} = \tilde{\mathbf{u}}_k + \boldsymbol{\delta}_k$,
  $\mathscr{J}_b(\tilde{\mathbf{u}}_{k+1}) = \mathscr{J}_b(\tilde{\mathbf{u}}_k + \boldsymbol{\delta}_k)$, $\mathscr{X}_{k+1} = \mathscr{X}_{k+}$.
  Update trust-region radius $\triangle_{k+1} = \min\left(\triangle_{max}, \gamma_2 \triangle_k\right)$.
  Set $k = k+1$ and go to Step 1.
- If $\eta_2 \leq \rho_k < \eta_3$:
  Set $\tilde{\mathbf{u}}_{k+1} = \tilde{\mathbf{u}}_k + \boldsymbol{\delta}_k$,
  $\mathscr{J}_b(\tilde{\mathbf{u}}_{k+1}) = \mathscr{J}_b(\tilde{\mathbf{u}}_k + \boldsymbol{\delta}_k)$, $\mathscr{X}_{k+1} = \mathscr{X}_{k+}$.
  Update trust-region radius $\triangle_{k+1} = \triangle_k$
  Set $k = k+1$ and go to Step 1.
- If $\eta_1 \leq \rho_k < \eta_2$:
  Set $\tilde{\mathbf{u}}_{k+1} = \tilde{\mathbf{u}}_k + \boldsymbol{\delta}_k$,
  $\mathscr{J}_b(\tilde{\mathbf{u}}_{k+1}) = \mathscr{J}_b(\tilde{\mathbf{u}}_k + \boldsymbol{\delta}_k)$, $\mathscr{X}_{k+1} = \mathscr{X}_{k+}$.
  Update trust-region radius $\triangle_{k+1} = \gamma_2 \triangle_k$.
  Set $k = k+1$ and go to Step 1.
- If $\rho_k < \eta_1$ or $\rho_k = \infty$:
  Set $\tilde{\mathbf{u}}_{k+1} = \tilde{\mathbf{u}}_k$,
  Update trust-region radius $\triangle_{k+1} = \gamma_1 \triangle_k$.
  Set $k = k+1$ and go to Step 2.

Remark 1 The subproblem (5.37) is not the standard quadratic model approximation as in the trust-region globalization strategy. Instead, it is an approximation of the high-fidelity model. In Fahl [2000], an algorithm based on the Cauchy condition is used to solve the subproblem. In this work, we use the KNITRO optimization package Byrd et al. [2006] for finding optimal steps $\boldsymbol{\delta}_k$.

Remark 2 As in Fahl [2000]; Agarwal [2010], to enforce that the optimization using reduced-order models converges to the same solution as the optimization in high-fidelity models, *first-order consistency* conditions are assumed. These condition satisfy

$$\mathscr{J}_b = \mathscr{J}_b^{\mathscr{R}}, \quad \nabla \mathscr{J}_b = \nabla \mathscr{J}_b^{\mathscr{R}}. \tag{5.38}$$

Some scaling techniques are applied in Fahl [2000]; Agarwal [2010] to fulfill the condition. In this work, since we take snapshots of both the forward and adjoint equations, the condition relies on the choice of POD basis functions. It should be noted that the snapshots must be chosen such as to capture the main dynamics.

Remark 3 The stopping criteria applied in full-space optimization are usually the absolute changes objective function or constraints violation as described in Algorithm 14. In the TRPOD method, the stopping criterion is the trust-region radius. If the trust-region radius is less than the minimum trust-region radius $\triangle_{min}$, then the optimization is terminated. Since the objective function in surrogate model can be lower (maximization case) than in the previous iteration, which may yield negative value of ρ_k, we therefore take absolute value of ρ_k. Moreover the value of ρ_k can be infinite due to constraint violation, we hence reduce the trust-region radius.

Remark 4 The bound constraint in the high-fidelity optimization $g(\mathbf{u}^n)$ is adjusted in the low-fidelity optimization due to the infinite norm constraints on the steps $\boldsymbol{\delta}_k$. The optimization in surrogate model may be stopped if the bound constraint in high-fidelity optimization is violated.

Remark 5 The trust-region parameters in the TRPOD method are chosen as follows

$$\eta_1 = 0.02, \eta_2 = 0.5, \eta_3 = 1, \gamma_1 = 0.25, \gamma_2 = 0.5, \gamma_3 = 1.5.$$

The small value of $\eta_1 = 0.02$ means we accept small improvement in the objective function value.

Remark 6 The discussion on convergence and convergence rate of the algorithm can be found in Fahl [2000]; Agarwal [2010].

It should be noted that to speed up the optimization convergence we employ the BFGS method using the first-order gradient from the adjoint method. Alternatively, one may use the SR1 algorithm to approximate the Hessian matrix, which is quite common in the trust-region scheme.

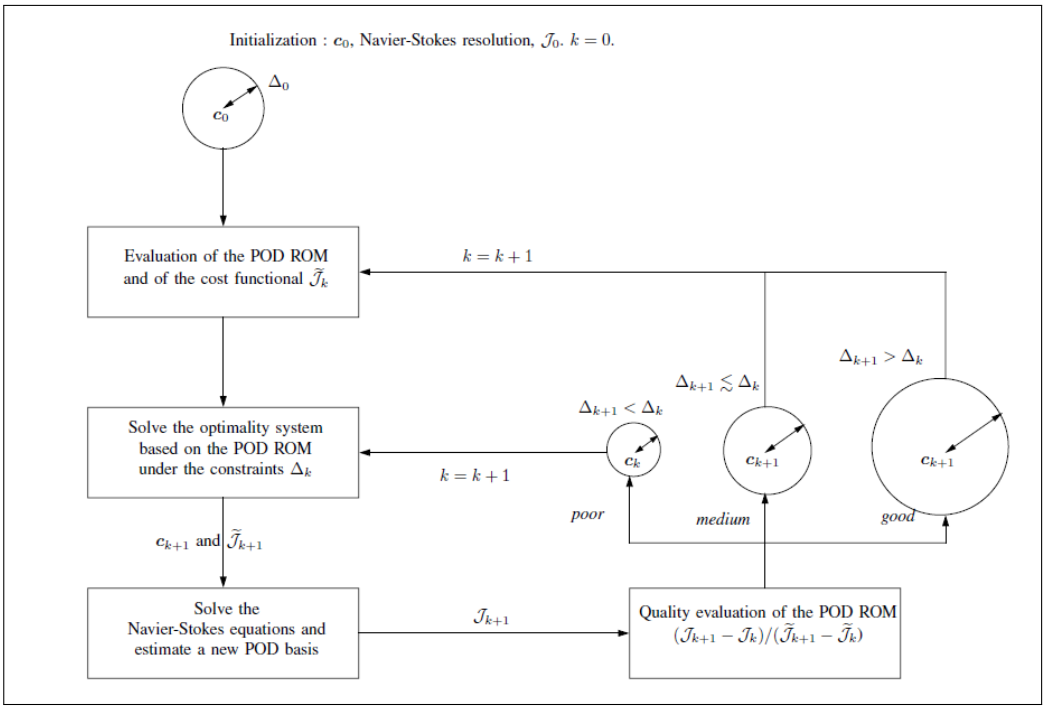

Figure 5.3: Pictorial sketch how the TRPOD method works. This figure is taken from Bergman et al. [2005] without any modification.

5.4.2 Lagrangian barrier methods

Since we also handle state constraints, in this work we employ the Lagrangian barrier method, which requires an augmented objective function

$$\mathscr{J}_b\left(\widetilde{\mathbf{u}}, \boldsymbol{\lambda}, \mu\right) = \mathscr{J}\left(\widetilde{\mathbf{u}}\right) + \mu \sum_{i=1}^{n_h} \lambda_i \log\left(h_i\left(\hat{\mathbf{x}}^n, \mathbf{u}^n\right)\right). \tag{5.39}$$

Here μ is the barrier parameter and λ_i is the componentwise Lagrange multiplier estimates, which are updated during the course of optimization. The Lagrangian barrier method is described in Algorithm 14. The TRPOD method is used in step 1 of the Lagrangian barrier method. This method will terminate either due to (most likely) objective function criterion or constraint violation. We suggest interested readers to refer to Suwartadi et al. [2012b] for further details of algorithm discussion and its uses for production optimization.

Algorithm 14 Lagrangian Barrier method

Step 0: **Initialization**
- Set feasible initial solution of control $\tilde{\mathbf{u}}_0$, step $\boldsymbol{\delta}$, and positive initial multiplier $\boldsymbol{\lambda}_0$
- Set convergence tolerances:
  - gradient tolerance $\omega_* \ll 1$
  - constraint violation $\eta_* \ll 1$
  - objective function changes ϵ_*
- Set positive barrier parameter μ_0 and $\tau < 1$
- Set initial convergence tolerance: ω_0.

Perform iteration: $k = 0, 1, 2, \ldots$

Step 1: **Inner iteration**
- Find $\boldsymbol{\delta}$ such that $\left\| \nabla \mathscr{J}^{\mathscr{R}}_{b,k} \right\| \leq \omega_k$

Step 2: **Test for convergence**
- If $\left\| [h_i(\tilde{\mathbf{x}}_k)]_{i=1}^m \right\| \leq \eta_*$
  or $\left| \mathscr{J}_{k+1}(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}) - \mathscr{J}_k(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}) \right| \leq \epsilon_*$ then **stop**.
- Check $\left\| \bar{\boldsymbol{\lambda}}_{h,k} - \boldsymbol{\lambda}_{h,k} \right\| \leq \frac{\tau_k}{\mu_k}$
- If this holds continue to **Step 3**.
  Otherwise go to **Step 4**.

Step 3: **Update Lagrangian Multipliers**
- $\mu_{k+1} = \mu_k$
- $\omega_{k+1} = \tau_k \omega_k$
- $\boldsymbol{\lambda}_{h,k+1} = \bar{\boldsymbol{\lambda}}_h\left(\tilde{\mathbf{x}}_k, \boldsymbol{\lambda}_{h,k}, \mu_{k+1}\right)$
- Continue to **Step 1**.

Step 4: **Update Barrier parameter**
- $\mu_{k+1} = \tau_k \mu_k$
- $\tau_{k+1} = \mu_k^{0.5}$
- Continue to **Step 1**.

5.5 Case Examples

In this section, we present three case examples. The first case will compare POD and DEIM in building reduced-order models in terms of CPU time and its accuracy. The second example will demonstrate how the TRPOD method works in an optimization case without the presence of nonlinear output constraints. The last case examples will show how the TRPOD and Lagrangian methods handle the nonlinear output constraints in surrogate optimization. Simulations for these case examples were done on a 64-bit Linux box with Intel(R) Xeon(R) CPU @ 3.00GHz. All the SVD computation in the case examples are done by using the SVD function in MATLAB. We use the *economical* option (*SVD('*','econ')*) in order to choose the eigenvectors corresponding to the largest

singular values.

5.5.1 Case 1

The reservoir model in this case is taken from layer 10 of SPE 10th comparative study Christie & Blunt [2001]. The grid consists of 60×220 gridblocks, where the dimension of a grid block is $10\,\text{ft} \times 20\,\text{ft} \times 2\,\text{ft}$. The connate oil saturation and residual water saturation are zero. The porosity, for simplicity, is set homogenously to 0.3, while the permeability is heterogenous as depicted in Figure 5.4. The mobility ratio oil-to-water is set to 5 and initial water saturation is zero. The well configuration is a 5-spot pattern with an injector in the middle and four producers at the corners. The simulation is run for 1200 days and the control inputs are the well rates. We divide the control inputs into 40 intervals, which means we change the well rates every 30 days. The number of controls variables is $40 \times 5 = 200$. Initial injection rate is set to 0.5 PVI. Moreover, the snapshots of forward and adjoint equations are taken from 40 control intervals.

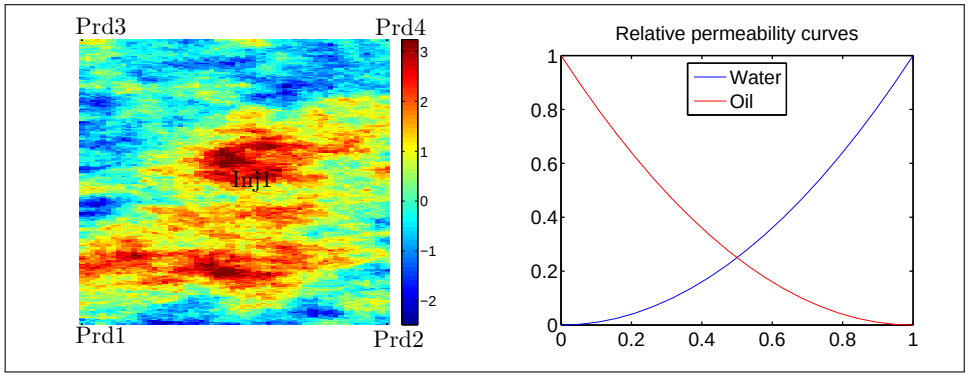

Figure 5.4: The logarithm of permeability field in millidarcy (mD), well location and relative permeability curves. The well locations follow the 5-spot pattern in which 4 producers are placed in the corners and 1 injector in the middle.

In this case example, we compare reduced-order models obtained from the POD and DEIM methods. The reduced-order models are constructed based on the snapshots of the forward and adjoint equations. For the POD method, the snapshots for the forward equations comprise the solution of pressures and water saturation for 40 control steps. While for the DEIM method, we need additional water cut snapshots representing the nonlinear terms, which is also from 40 control steps. In the adjoint equations, since there is no nonlinear term, we apply the POD method. Thus, the snapshot for the adjoint equations will be the solution of the adjoint equations. We will explain the reduced-order models for both types of equations in the following subsection.

Forward Equations

Table 5.1: CPU Time for forward equations using the high-fidelity model

CPU Time (in sec.)		
Pressure Eq.	Saturation Eq.	Total Time
15.6	5.3	20.9

The runtime of the high-fidelity model is described in Table 5.1. To build reduced-order models for the forward equations, we choose an energy level truncation. We vary the value of energy truncation in order to know a good value or dimension of the reduced-order models. Furthermore, we define the error of the reduced-order model by the following equation

$$\mathscr{E} := \frac{\left\| \mathbf{s}^N - \hat{\mathbf{s}}^N \right\|_2}{\left\| \mathbf{s}^N \right\|_2}, \tag{5.40}$$

where $\mathbf{s}^N$ is water saturation at final time step N, and $\hat{\mathbf{s}}^N$ is the reduced-order water saturation at the final time step.

Figures 5.5, 5.6, and 5.7 depict the singular values of the state variables: pressure, saturation, and the nonlinear term, water cut.

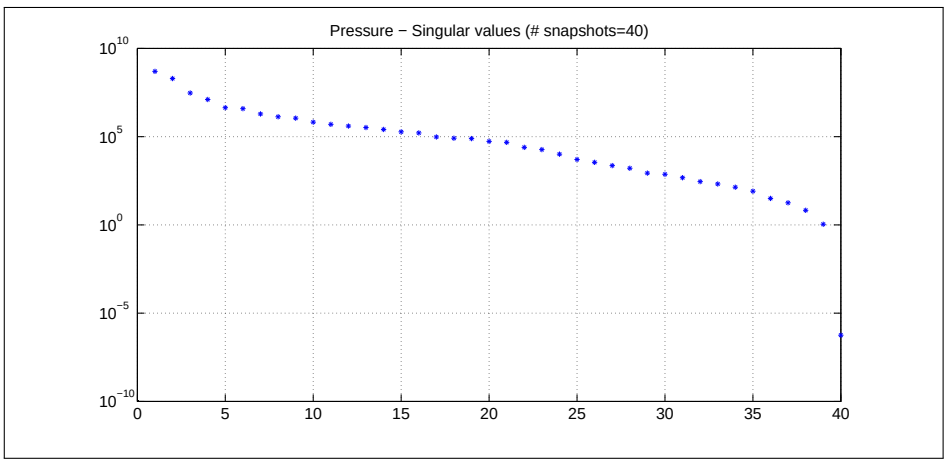

Figure 5.5: Singular values of pressure snapshots.

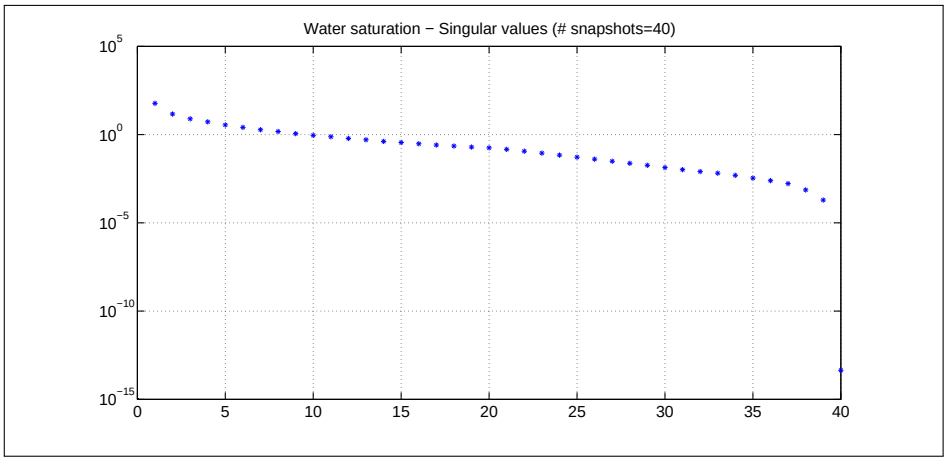

Figure 5.6: Singular values of water saturation snapshots.

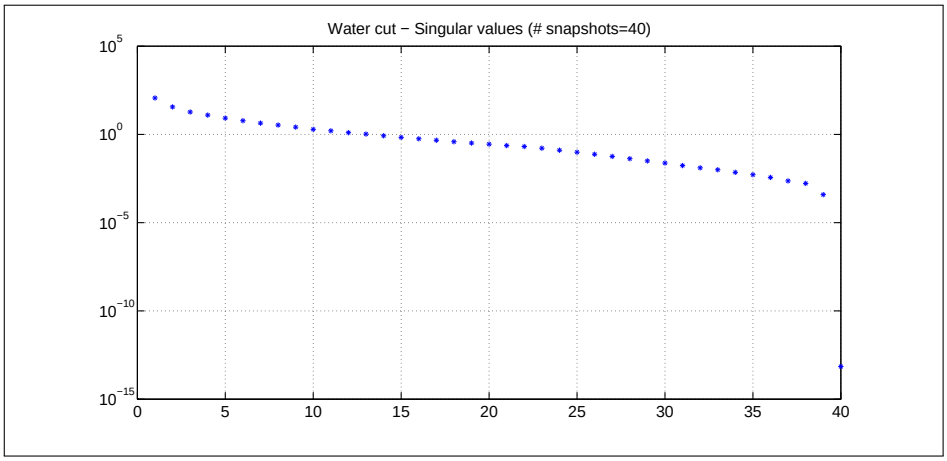

Figure 5.7: Singular values of water cut snapshots.

We then run simulations with a variation of energy truncations and the results are displayed in Tables 5.2 and 5.3. In general the POD method gives significant speedup for the pressure equation compared to that of the high-fidelity model runtime described in Table 5.1. However, only a slight CPU time reduction is obtained for the saturation equation. On the other hand, DEIM gives more speedup for the saturation equation. The approximation errors and the CPU time speedups decrease when the number of basis functions increase.

Table 5.2: POD - variation of energy truncation

Energy Truncation		#Basis Functions		CPU Time (in sec.)			Error Sat.
Pres.	W.Sat.	Pres.	W.Sat.	Pres. Eq.	Sat. Eq.	Total Time	
90%	90%	4	9	4.2	3.3	7.5	0.021
	95%		13		3.6	7.8	0.013
	99%		24		4.6	8.8	0.004
95%	90%	5	9	4.5	4.4	8.9	0.021
	95%		13		4.8	9.3	0.013
	99%		24		5.4	9.9	0.003
99%	90%	11	9	5.1	4.1	9.2	0.021
	95%		13		4.1	9.2	0.013
	99%		24		4.8	9.9	0.003
99.9%	99.9%	24	35	6.9	6.2	13.1	0.000
99.99%	99.99%	35	40	9.6	6.4	16.0	0.000

Table 5.3: DEIM - variation of energy truncation

Energy Truncation			#Basis Functions			CPU Time (in sec.)			Error Sat.
Pres.	W.Sat.	W. Cut	Pres.	W.Sat.	W. Cut	Pres.	Sat.	T. Time	
90%	90%	90%	4	9	9	4.4	2.5	6.9	0.020
	95%	95%		13	13		2.7	7.1	0.010
	99%	99%		24	22		3.8	8.1	0.010
95%	90%	90%	5	9	9	4.6	2.3	6.9	0.020
	95%	95%		13	13		2.7	7.3	0.010
	99%	99%		24	22		4.1	8.7	0.009
99%	90%	90%	11	9	9	5.2	2.6	7.8	0.020
	95%	95%		13	13		2.8	8.0	0.010
	99%	99%		24	22		3.8	9.0	0.010
99.9%	99.9%	99.9%	24	35	34	6.8	5.0	11.8	0.001
99.99%	99.99%	99.99%	35	40	38	9.8	5.3	15.1	0.000

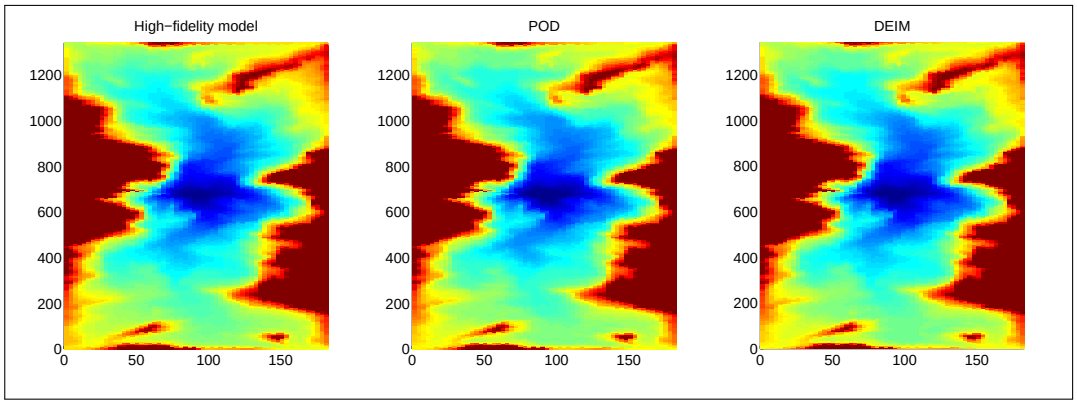

Figure 5.8: Comparison of water saturation at final time for the high-fidelity model and reduced order models; POD and DEIM.

To show the quality of reduced-order model POD and DEIM, we display water saturation at the end time using energy truncation 90% for the pressure, 90% for the water saturation, and 90% for the water cut in Figure 5.8.

Adjoint Equations

Here, we continue to vary the energy truncation of the adjoint equations. The runtime for the full model of the adjoint equations is described in Table 5.4. Furthermore, we plot the singular values of the corresponding pressure and saturation equations in Figure 5.9 and 5.10.

Table 5.4: CPU Time for adjoint equations using the high-fidelity model

Adjoint - CPU Time (in sec.)		
Pressure Eq.	Saturation Eq.	Total Time
15.9	2.1	18.0

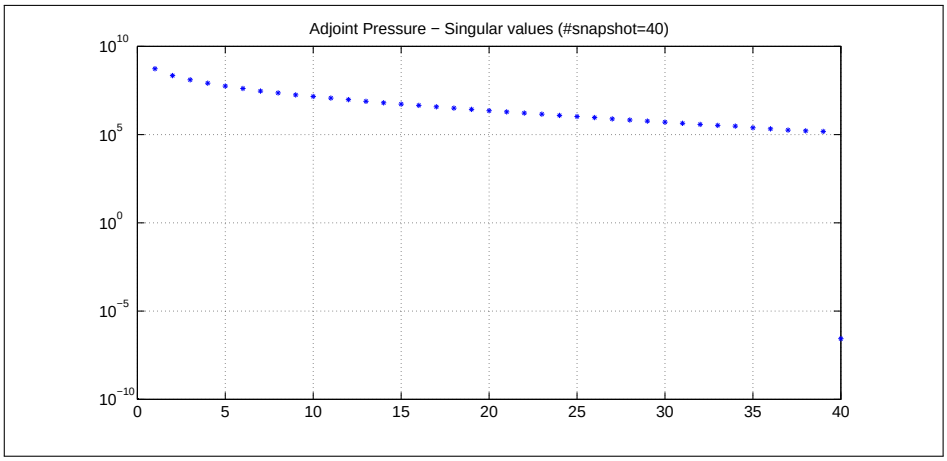

Figure 5.9: Singular values of corresponding adjoint pressure equation snapshots.

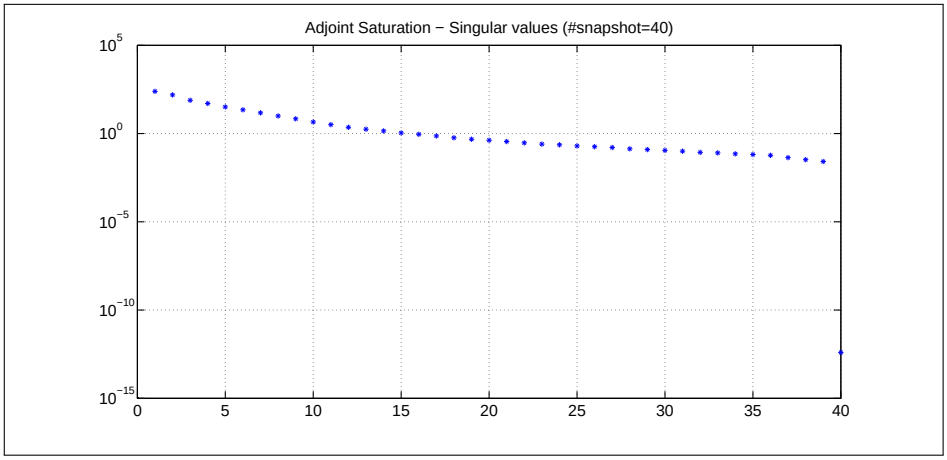

Figure 5.10: Singular values of corresponding adjoint water saturation equation snapshots.

Similarly, we have done some simulations using variation of energy truncations and the results are described in Tables 5.5 and 5.6. We define the error of the adjoint-gradient in the reduced-order model by the following equation

$$\mathscr{E}_{\text{grad}} := \frac{\left\|\mathbf{grad} - \hat{\mathbf{grad}}\right\|_2}{\left\|\mathbf{grad}\right\|_2}, \tag{5.41}$$

which compare the gradient in full-space (**grad**) and in reduced-order ($\hat{\mathbf{grad}}$).

Both the POD and DEIM methods, shown in Tables 5.5 and 5.6, give speedup in runtime compare to the adjoint equations in full-space described in Table 5.4. We run

both POD and DEIM for the adjoint equations since they need forward reduced-order models. Furthermore, the CPU time for corresponding adjoint saturation is comparable to that of full-space runtime. This is because of the sparsity property in the linear adjoint saturation equation. In the full-space equation the adjoint saturation is solved using a sparse linear solver. However, in a reduced-order model we loose the sparsity structure of the adjoint saturation equation. One may get better speedup for the adjoint saturation equation if the reservoir model has a larger number of grid blocks.

Table 5.5: Adjoint POD - variation of energy truncation

Fwd. - E. Trun.		Adj. - E. Trunc.		# Adj. Basis F.		CPU Time (in sec.)			Error
Pres.	W. Sat.	Adj.Pres.	Adj.S_w	Adj.P.	Adj.S_w	Adj.P	Adj.S_w	T. Time	
90%	90%	90%	90%	7	6	6.5	2.1	8.6	0.008
	95%	95%	95%	10	8	8.2	2.2	10.4	0.005
	99%	99%	99%	4	24	15.9	2.4	18.3	0.004
95%	90%	90%	90%	7	6	7.3	2.1	9.4	0.008
	95%	95%	95%	10	8	8.0	2.1	10.1	0.004
	99%	99%	99%	19	13	16.1	2.4	18.5	0.002
99%	90%	90%	90%	7	6	7.1	2.1	9.2	0.007
	95%	95%	95%	10	8	8.7	2.1	10.8	0.003
	99%	99%	99%	19	13	16.1	2.4	18.5	0.001
99.9%	99.9%	99.9%	99.9%	34	26	26.8	3.7	30.5	0.001

Table 5.6: Adjoint DEIM - variation of energy truncation

Fwd. - E. Trunc.			Adj. - E. Trunc.		# Adj. Basis F.		CPU Time (in sec.)			Error
P.	S_w	F_w	Adj.P	Adj.S_w	Adj.P.	Adj.S_w	Adj.P.	Adj.S_w	T.Time	
90%	90%	90%	90%	90%	7	6	6.6	2.2	8.8	0.009
	95%	95%	95%	95%	10	8	8.9	2.3	11.1	0.003
	99%	99%	99%	99%	19	13	16.2	2.4	18.6	0.004
95%	90%	90%	90%	90%	7	6	6.5	2.1	8.6	0.008
	95%	95%	95%	95%	10	8	8.8	2.2	13.0	0.004
	99%	99%	99%	99%	19	13	16.1	2.4	18.5	0.002
99%	90%	90%	90%	90%	7	6	6.8	2.1	8.9	0.008
	95%	95%	95%	95%	10	8	8.8	2.2	13.0	0.004
	99%	99%	99%	99%	19	13	16.2	2.4	18.6	0.001
99.9%	99.9%	99.9%	99.9%	99.9%	34	26	28.3	3.8	32.1	0.001

We also present the quality of the gradient approximation in the reduced-order

models in Figure 5.11 and 5.12, where the truncation is 90% and 90% for the corresponding pressure and saturation, respectively.

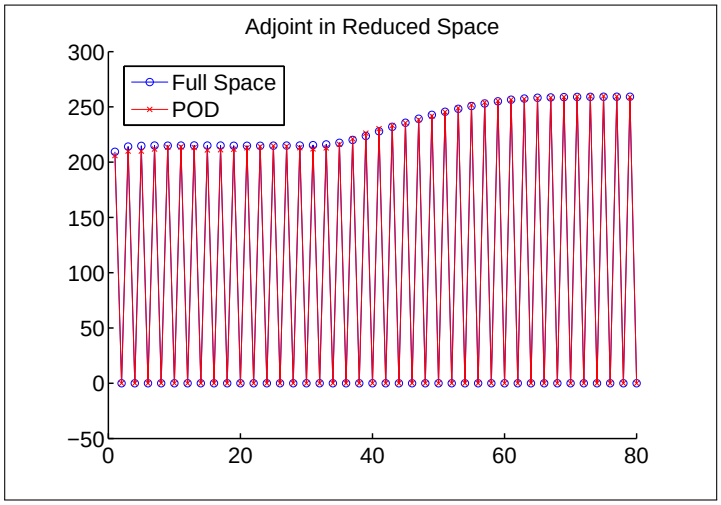

Figure 5.11: Comparison of adjoint-gradient in full-space and POD.

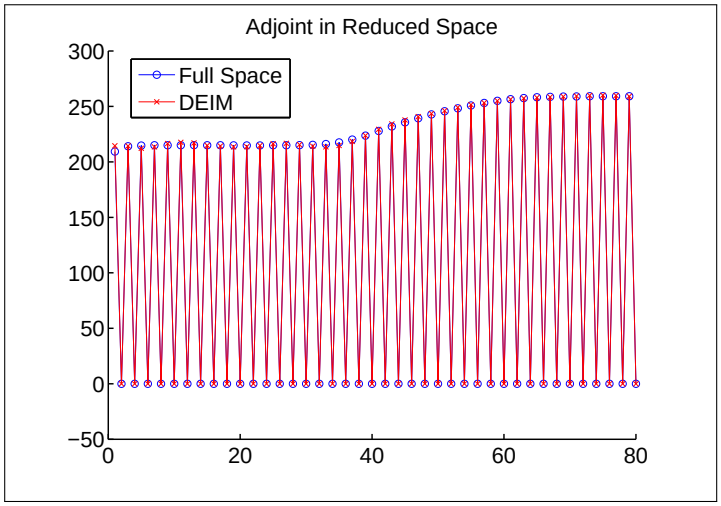

Figure 5.12: Comparison of adjoint-gradient in full-space and DEIM.

Effect of Perturbations

In order to know the robustness of basis functions, we first build a reduced order model using DEIM with 90% energy truncation for pressure, saturation, and water cut. We then change well rate at producer wells around 5%, 10%, and 20% in the sense that

we perturb the initial well rates when the basis functions are constructed. As seen in Figures 5.13, 5.14, and 5.15 below, the reduced-order model is good enough to approximate the high-fidelity model. The relative error saturation is the water saturation difference between high-fidelity model and reduced-order model divided by saturation in high-fidelity model.

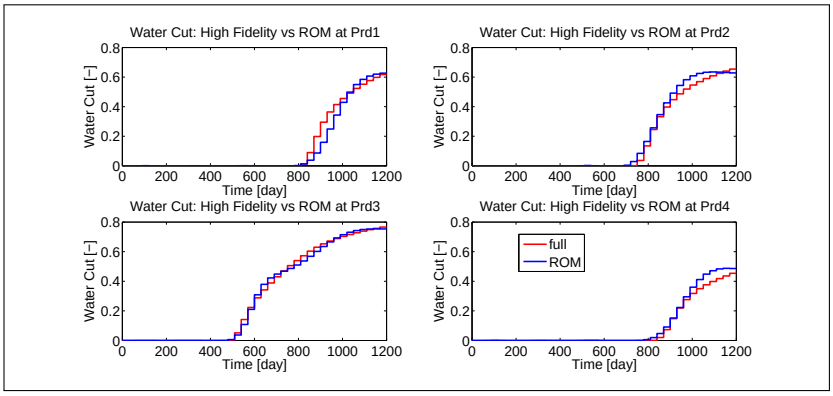

Figure 5.13: 5% variation of producers well rates with relative error saturation approximation is 0.021.

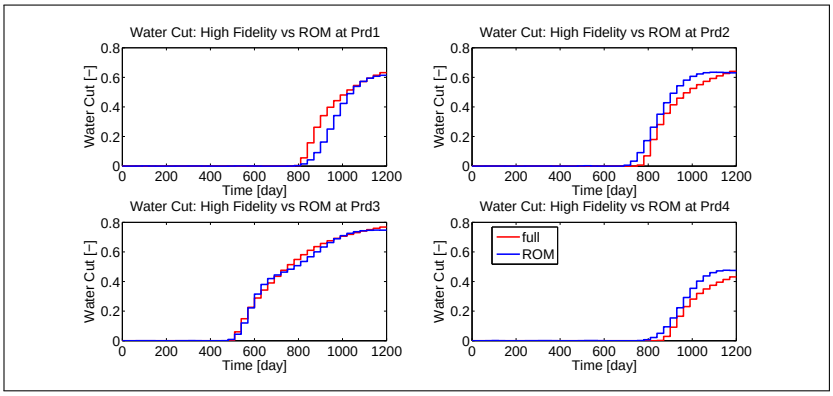

Figure 5.14: 10% variation of producers well rates with relative error saturation approximation is 0.029.

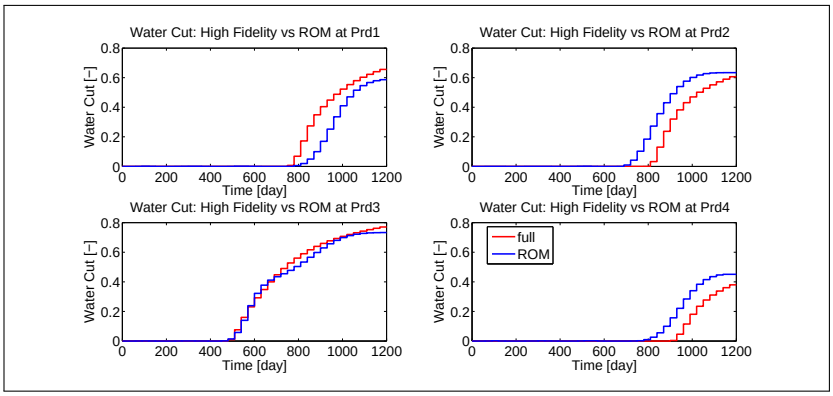

Figure 5.15: 20% variation of producers well rates with relative error saturation approximation is 0.052.

5.5.2 Case 2

In this case we set up a surrogate optimization without any output constraints. The goal is to show the performance of TRPOD method compared to the optimization using a high-fidelity model. Since there is no nonlinear output constraint, the constraints appear only on the control, that is, bound constraints and an equality constraint due to the incompressible flow (the total injector rate must be equal the total producer rate). The objective function in this case is net present value (NPV) with oil price $80\frac{\$}{m^3}$, water separation cost $19\frac{\$}{m^3}$, and water injection cost $1\frac{\$}{m^3}$. It should noted there is no augmented objective function in this case. Furthermore, we continue using the reservoir setting described in case 1 with initial injection rate is 0.4 PVI for 1200 simulation days (40 control intervals). The control inputs are well rates at producer and injector wells. The reduced-order model is built using DEIM due to its faster CPU time than POD.

To fully capture the dynamics of reservoir, we extend the simulation a bit further until 1800 days ensuring a water cut value 0.80 reaches all producer wells (based on the price setting). This is performed when building the initial basis functions but not during the basis functions update within the TRPOD strategy. We use energy level 99% for the forward pressure, 95% for saturation equation, and 95% for the water cut. For the adjoint equations we use 95% energy level for the corresponding pressure and saturation equations. Using this energy truncation, an initial forward reduced-order model consists of 12, 16, and 15 basis functions for the pressure, saturation, and water cut, respectively. The interpolation points are shown in Figure 5.16. The adjoint reduced-order model has 13 and 8 basis functions for the corresponding pressure and saturation equations, respectively.

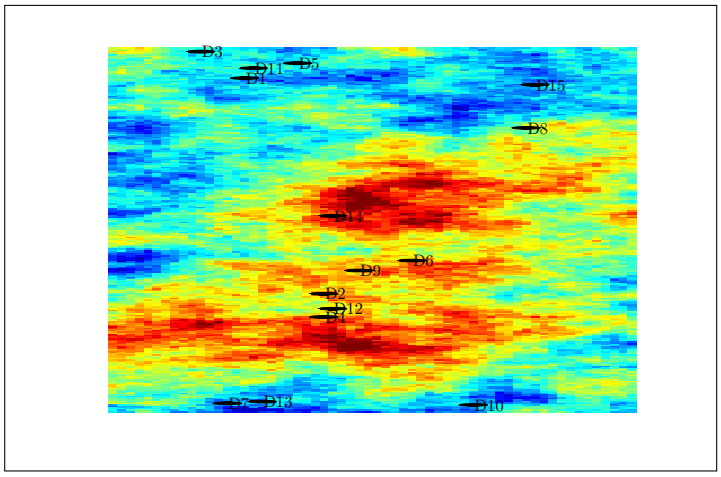

Figure 5.16: Interpolation points (represented with D) for the nonlinear water cut term are located at grid blocks: 12076, 4285, 13031, 3445, 12622, 5495, 314, 10308, 5129, 282, 12437, 3746, 378, 7106, and 11869.

We then run optimization using the reduced-order models. To evaluate the optimization, we also run the optimization using the high-fidelity model. The stopping criteria are the absolute gradient tolerance 10^{-8} and absolute step length 10^{-8}. These stopping criteria apply both for reduced and full-space model optimizations. The surrogate optimization is run with an initial trust-region radius, $\triangle_0$, set to $0.03 \times \frac{V_{gb}}{1200\,\text{days}}$, the maximum trust-region radius, $\triangle_{max}$, is $0.03 \times \frac{V_{gb}}{1200\,\text{days}}$. This maximum trust-region radius represents the bound in which the reduced order model is robust enough to the control input perturbation. Moreover, the minimum trust-region radius $\triangle_{min}$, is $10^{-3} \times \frac{V_{gb}}{1200\,\text{days}}$. The minimum trust-region ratio ρ_{min} is set 0.001. The bound constraints on the control input are set between 0 and $0.6 \times \frac{V_{gb}}{1200\,\text{days}}$.

After running the optimization, the evolution of the objective function is depicted in Figure 5.17. Note that the number of iterations in the full-space optimization represents the number of inner iteration while the number of iterations in the reduced-space optimization denotes the number of outer iteration, involved in TRPOD method. Table 5.7 describes the runtime and obtained objective function values. The details of the optimization using the reduced-order model are described in Table 5.8, where $\hat{\mathcal{J}}$ is the objective function in reduced-order model and $\mathcal{J}$ is the objective function evaluated in high-fidelity model. The surrogate optimization terminates due to the minimum trust-region radius.

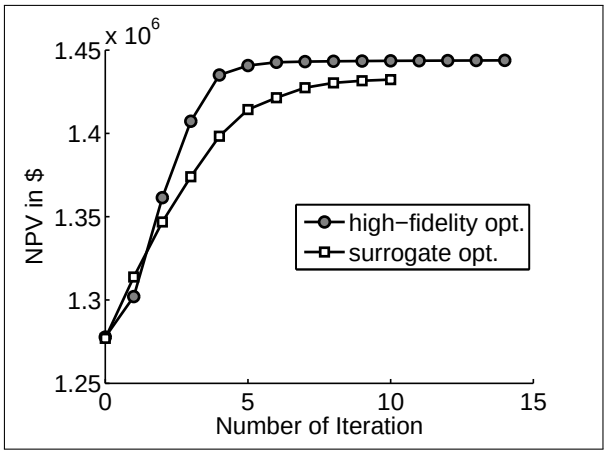

Figure 5.17: Evolution of the objection functions using the initial injection 0.4 PVI.

Table 5.7: Comparison of optimization in full-space and reduced-space using initial injection of 0.4 PVI.

Comparison	Full-model	POD-DEIM
NPV (in $)	1.44e+06	1.43e+06
CPU Time (in seconds)	5715	1270

Table 5.8: Iteration in surrogate optimization using initial injection of 0.4 PVI.

k	ρ	$\hat{\mathscr{J}}$	$\mathscr{J}$
0	-	1.27e+06	1.27e+06
1	1.16e+01	1.28e+06	1.32e+06
2	1.07e+00	1.31e+06	1.35e+06
3	9.28e-01	1.34e+06	1.37e+06
4	8.09e-01	1.37e+06	1.39e+06
5	2.37e-01	1.44e+06	1.41e+06
6	2.34e-01	1.41e+06	1.42e+06
7	5.67e-01	1.42e+06	1.43e+06
8	3.44e-01	1.43e+06	1.43e+06
9	3.88e-01	1.43e+06	1.43e+06
10	3.28e-01	1.43e+06	1.43e+06

In Table 5.8 above, the trust-region ratio ρ determines when the basis functions must be updated as well as when the radius ratio will be enlarged or shrunk. The trust-region parameter settings are described in Section 5.4. The basis functions and the

trust-region region will not be updated and enlarged if the trust-region ratio ρ has value less than 0.02. The trust-region radius will be enlarged only if the trust-region ratio is larger than 1. Otherwise, it will be kept or reduced. Hence, starting from the fifth iteration, seen in Table 5.8, the trust-region radius is shrunk. In the seventh iteration, the trust-region radius is kept as the same previous iteration and is decreased afterwards. Consequently, the optimization terminates due to the minimum trust-region radius criterion. Figure 5.18 shows the comparison of optimization solution in full-space and surrogate optimization. Here, we denote the injector rate with positive sign and producer rate with negative sign.

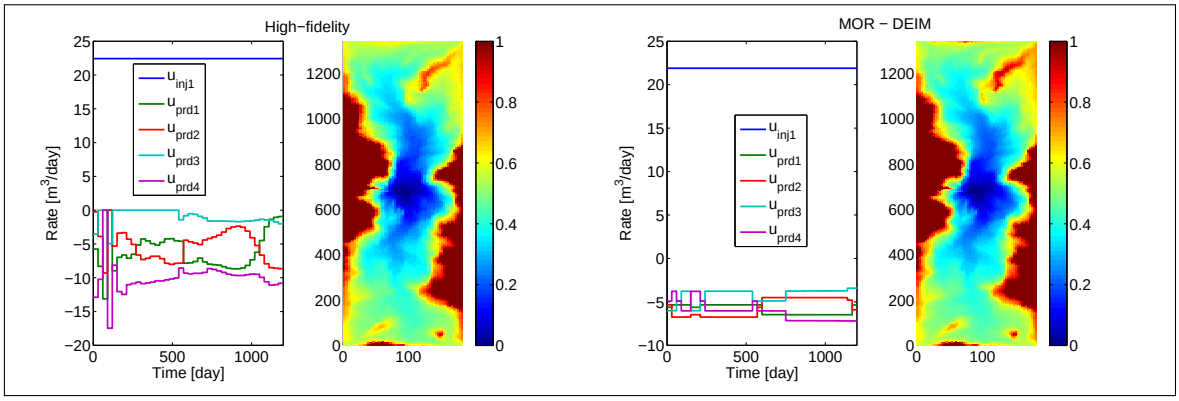

Figure 5.18: Optimization solutions in full-space and reduced-space models and water saturation at final time.

Next, we run another optimization with initial injection rate of 0.5 PVI using the same parameter values (initial trust-region radius). This initial rate is closer to the optimization solution in a high-fidelity model. The objective function evolutions are described in Figure 5.19. The details of the optimization in the reduced-space model can be seen in Table 5.10 and CPU time speedup is described in Table 5.9. The optimization in high-fidelity model terminates due to the step length tolerance, while in surrogate optimization it stops because it hits the upper bound constraint, that is, $0.6 \times \frac{V_{gb}}{1200\,\text{days}}$.

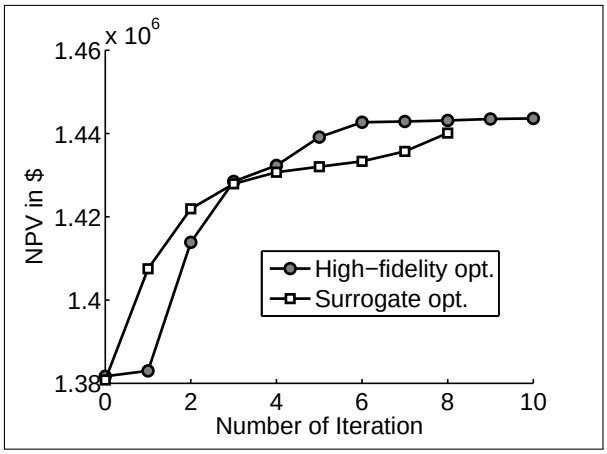

Figure 5.19: Evolution of the objection functions using the initial injection of 0.5 PVI.

Table 5.9: Comparison of optimization in full-space and reduced-space using initial injection of 0.5 PVI.

Comparison	Full-model	POD-DEIM
NPV ($)	1.44e+06	1.44e+06
CPU Time (in seconds)	26681	1269

Table 5.10: Iteration in surrogate optimization using initial injection of 0.5 PVI.

k	ρ	$\hat{\mathscr{J}}$	$\mathscr{J}$
0	-	1.38e+06	1.38e+06
1	1.06e+00	1.36e+06	1.41e+06
2	2.16e-01	1.42e+06	1.42e+06
3	4.75e-01	1.41e+06	1.43e+06
4	1.31e-01	1.43e+06	1.43e+06
5	7.09e-01	1.43e+06	1.43e+06
6	1.45e+00	1.43e+06	1.43e+06
7	1.03e+00	1.43e+06	1.44e+06
8	3.69e-01	1.42e+06	1.44e+06

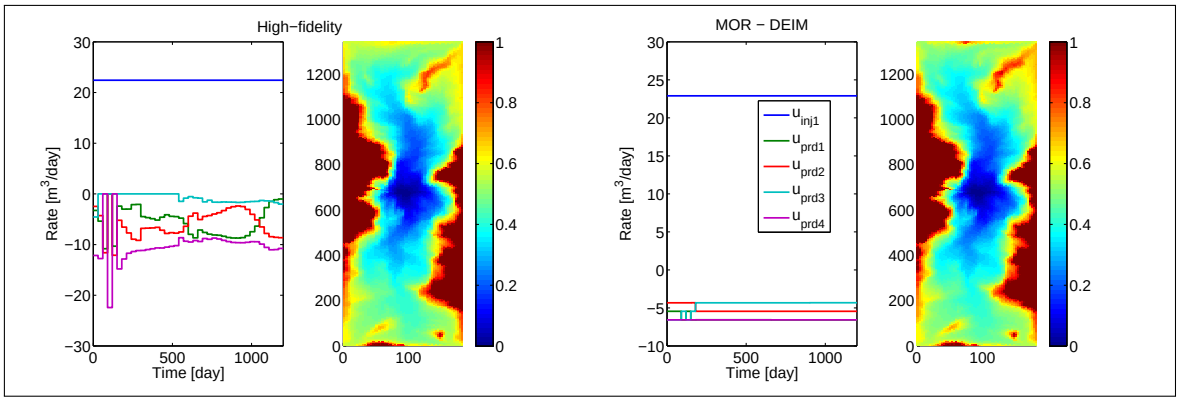

Figure 5.20: Optimization solutions in full-space and reduced-space models and water saturation at final time.

It turns out that the surrogate optimization reaches a different local maximum than optimization with high-fidelity model. However, the runtime is still quite cheap. The optimization in high-fidelity seems to converge in the same local maxima but with the cost of higher CPU time. The speedup factor in this case can be up to 20 times.

5.5.3 Case 3

This case is a continuation of the previous case, which uses the same reservoir, and well setting, however, with the inclusion of output constraints and a more accurate reduced-order model with an increased energy truncation for saturation and water cut to 99%. The objective function is now recovery factor (RF), described in (5.35). In this case we constrain the water fractional flow (water cut), which is function of water saturation, at the producer wells. We limit the water cut for the producer wells at the final time to $f_{w,max}$, which is set to 0.80. To this end, the augmented objective function is

$$\mathscr{J}_b\left(\tilde{\mathbf{u}},\boldsymbol{\lambda},\mu\right)=\mathscr{J}\left(\tilde{\mathbf{u}}\right)+\mu\sum_{i=1}^{4}\lambda_i\log\left(f_{w,max}-f^N_{w,prod_i}\right). \tag{5.42}$$

The parameter settings in this case are: $\boldsymbol{\lambda}_0=\begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix}^T$, $\tau=0.1$, $\mu_0=10^4$, $\omega_0=10^{-6}$, absolute maximum water cut tolerance $\eta_*=10^{-6}$, and absolute objective function changes $\epsilon_*=10^{-4}$ percent of recovery factor. We choose an active set algorithm in KNITRO to the handle control input constraints $g(\tilde{\mathbf{u}})$. The initial trust-region radius, $\triangle_0$, and the maximum trust-region radius, $\triangle_{max}$, are set equally to $0.01\times\frac{V_{gb}}{1200\,\mathrm{day}}$, the minimum trust-region radius $\triangle_{min}$, is $10^{-4}\times\frac{V_{gb}}{1200\,\mathrm{day}}$, and minimum trust-region ratio ρ_{min} is 0.001. The bound constraints on the control input are set

between 0 and $0.8 \times \frac{V_{gb}}{1200\,\text{days}}$. We run two optimizations with different initial injector settings.

Initial injection rate 0.5 PVI

We start the optimization with an initial injector rate 0.5 PVI. The optimization with the high-fidelity model stops due to the objective function change criterion. Furthermore, Table 5.11 describes the results and constraint violations are shown in Figure 5.21. The comparison of the objective function evolution is displayed in Figure 5.22. The infinite objective function value in the reduced-order space indicates that the output constraint is violated.

Table 5.11: Optimization results. The water injected is measured in pore volume injected (PVI)

Comparison	Full Model	POD-DEIM
Recovery factor (%)	48.41	47.96
CPU time (in sec.)	12054	8809
Total Water Injected (in PVI)	0.77	0.75

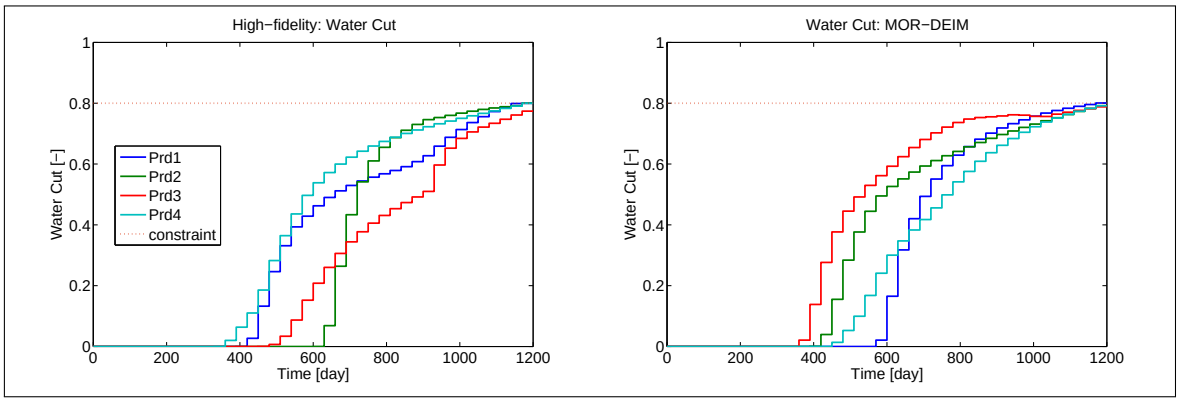

Figure 5.21: The state constraints satisfaction, i.e., the water-cut at final time step.

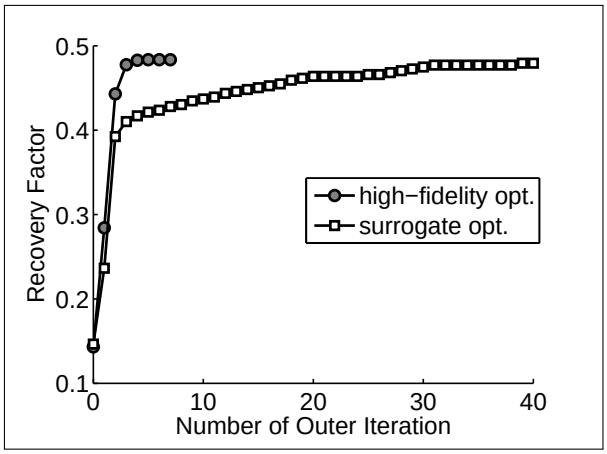

Figure 5.22: Comparison of the objective function evolution in full-space and reduced-space. The objective value from reduced-space in the figure is the objective function evaluation using the high-fidelity model given the solution of surrogate optimization.

The optimization in reduced-space terminates because of the minimum trust-region radius. POD-DEIM in this case gives a speedup more than 1.3 times. The output constraints are active only for $Prd1$ in the surrogate model while other production wells are closely becoming active.

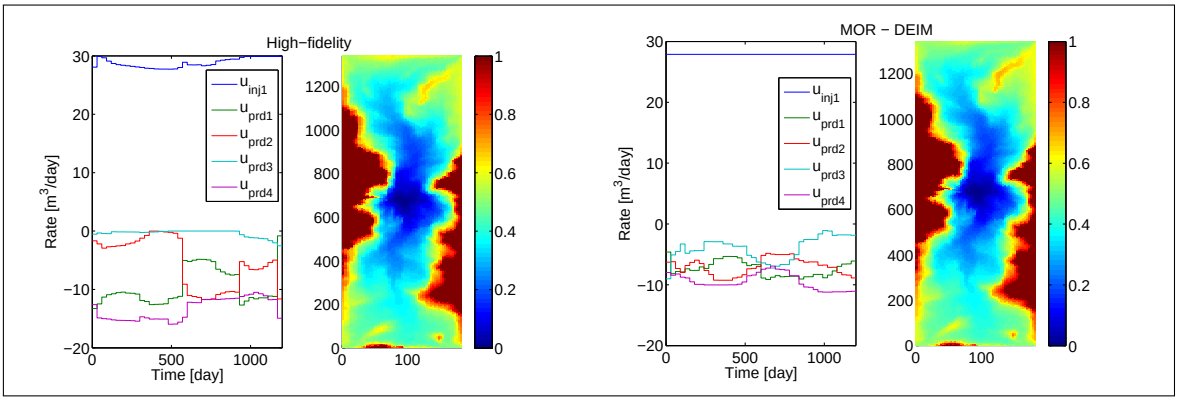

Figure 5.23: Optimization solutions in full-space and reduced-space models and water saturation at final time.

In Figure 5.23 the optimization solutions using high-fidelity and reduced-order model are shown. The producer rates are shown in negative sign while the injector rate is in positive sign. Again, it is clear the surrogate optimization resulted in a local maximum. The figure also shows water saturation at final time step. As seen, the water saturation is slightly different around $Prd4$ area.

Initial injection rate 0.4 PVI

We continue the optimization using reduced-order model with different initial solution. Here, we set initial injection rate to 0.4 PVI. The results are shown in Table 5.12, constraints satisfaction in Figure 5.24. Similarly, the optimization terminates due to the minimum trust-region radius and $Prd1$ is active, while the other production wells are almost active. The evolution of the objective function is shown in Figure 5.25, and control input solution is in Figure 5.26.

Table 5.12: Optimization results. The water injected is measured in pore volume injected (PVI)

Comparison	Full Model	POD-DEIM
Recovery factor (%)	48.35	47.88
CPU time (in sec.)	13615	11181
Total Water Injected (in PVI)	0.77	0.74

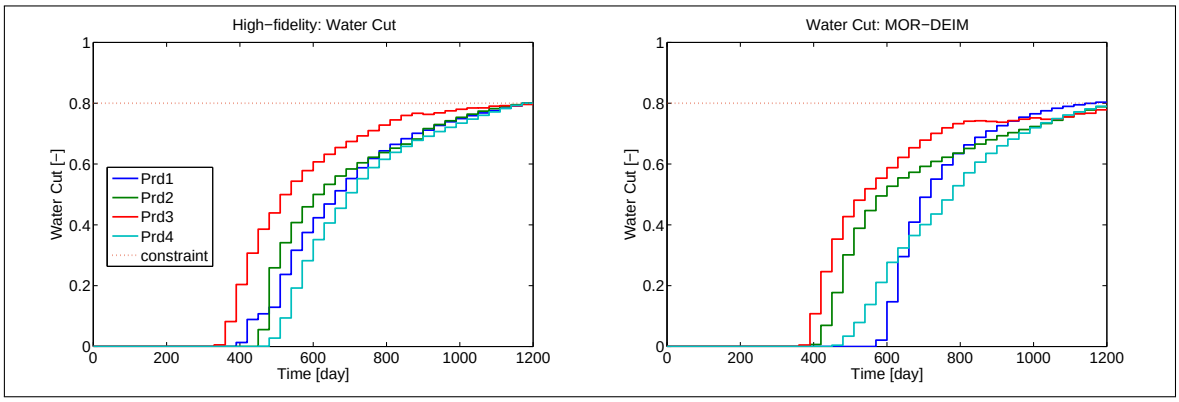

Figure 5.24: The state constraints satisfaction, i.e., the water-cut at final time step.

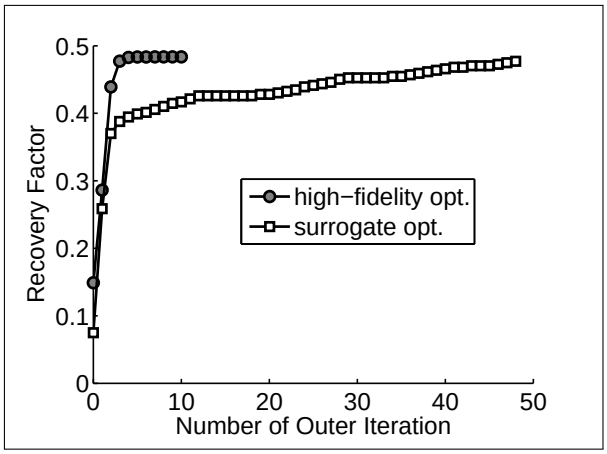

Figure 5.25: Comparison of the objective function evolution in full-space and reduced-space. The objective value from reduced-space in the figure is the objective function evaluation using the high-fidelity model given the solution of surrogate optimization.

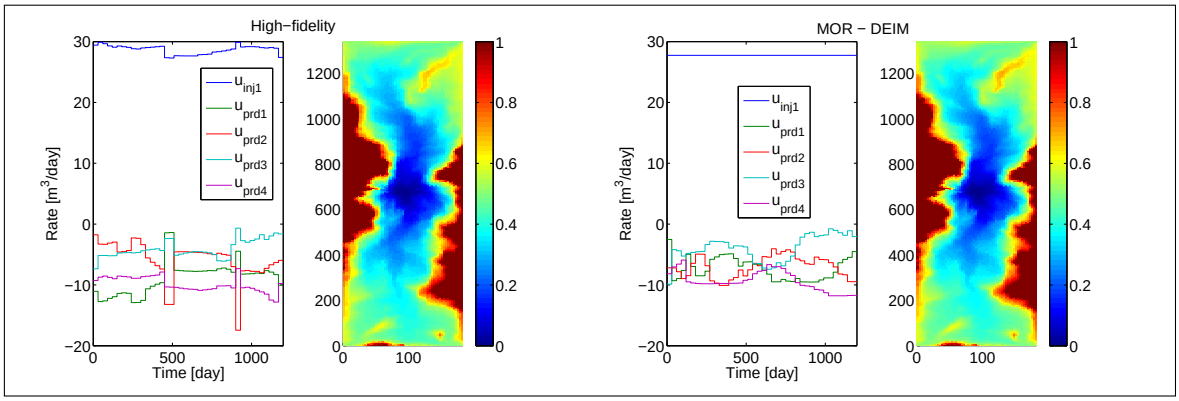

Figure 5.26: Optimization solutions in full-space and reduced-space models and water saturation at final time.

5.5.4 Case 4

This case originates from the Norne comparative study Rwechungura et al. [2010] with a simplified model. The reservoir is depicted in Figure 5.27 and there are 6 wells. Initial water saturation and pressures at each grid block are set to 0.2 and 40 bar, respectively. The mobility ratio between water and oil is 1 to 5. The end points of connate water saturation and oil saturation are both set to 0.2. The relative permeability curves are displayed in Figure 5.27. The simulation is run for 500 days and divided into 50 control

intervals. Thus, in total the controls consist of 300 variables. The controls in this case are well rates. Similar to the previous cases, we deal with equality constraints due to incompressible flow, that is, total injectors rate must equal total producers rate. In addition, we set bound constraint on the injectors, which is set lower than 2 PVI. The initial water injection rate are 0.25 and 0.30 of total pore volume using constant rates. The number of snapshots for building the reduced-order models is the same as the number control intervals, which is 50 snapshots.

In this case we constrain the total water production at the final control interval to 5×10^{-3} of the pore volume of the reservoir and define this constraint as $Q_{w,max}$. Hence, in this case the augmented objective function is

$$\mathscr{J}_b\left(\tilde{\mathbf{u}}, \lambda, \mu\right) = \mathscr{J}\left(\tilde{\mathbf{u}}\right) + \mu\lambda \log\left(Q_{w,max} - \sum_{i=1}^{n_h=4} Q_{w,i}^N\right). \tag{5.43}$$

Here the output constraint is just a scalar, that is, the total water production of each producer well at final control interval.

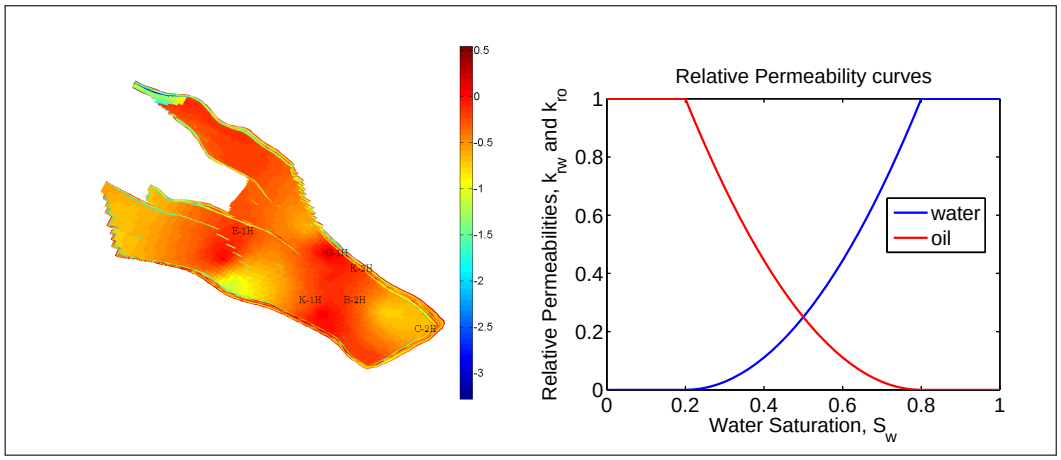

Figure 5.27: Norne field, a 3D reservoir, with 6 wells: 4 producers (E-1H, K-1H, B-2H, K-2H) and 2 injectors (C-1H and C-2H). Permeability field is plotted in millidarcy (mD). The right hand figure shows relative permeabilities

The reduced-order models are constructed using 90% energy truncation for the pressure and saturation equations, and 90% for the non-linear water cut term. This results in 4, 7, 7 number of basis functions for the pressure, saturation and water cut, respectively. For the adjoint equations, we use a 95% energy level both for the corresponding pressure and saturation equations. To this end, the reduced-order adjoint equations have dimension 16 and 4, respectively. The CPU time comparison of the full-model and reduced-order models are described in Table 5.13. The POD-DEIM results are consistently faster than the vanilla POD. Significant speedup is obtained for

the forward equations, but the corresponding saturation equation is again similar to the first case. Due to sparsity property in the full-space adjoint equation and the dense matrix in the reduced-order models, the speedup of corresponding saturation adjoint equations is not that significant. To this end, we use the reduced-order model of POD-DEIM for surrogate optimization.

The parameter settings in this case are: $\lambda = 1$, $\tau = 0.1$, $\mu = 10^7$, $\omega_0 = 10^{-3}$, and absolute total water production tolerance $\eta_* = 10^{-4}$. We choose an active set algorithm in KNITRO to handle control input constraints $g(\tilde{\mathbf{u}})$. The maximum trust-region radius, $\triangle_{max}$, is $0.1 \times \frac{V_{gb}}{500\,\text{day}}$, and minimum trust-region radius $\triangle_{min}$, is $0.001 \times \frac{V_{gb}}{500\,\text{day}}$ and initial trust-region radius, which are $\triangle_0 = 0.1 \times \frac{V_{gb}}{500\,days}$.

Table 5.13: Comparison of CPU time of forward and adjoint equations

CPU Time (in sec.)	Full Model	POD	POD-DEIM
Forward equations			
Pressure Eq.	154	32	29
Saturation Eq.	51	19	11
Total Time Forward Eqs.	205	51	40
Adjoint equations			
Pressure Eq.	215	92	86
Saturation Eq.	18	15	15
Total Time Adjoint Eqs.	233	107	101

Initial injection rate 0.25 PVI

We firstly run the optimization with initial injection 0.25 PVI. The results are summarized in Table 5.14, the constraint satisfaction in Figure 5.28, and comparison of optimized control inputs in Figure 5.29. The optimization terminated due to the minimum trust-region radius.

Table 5.14: Optimization results with initial injection rate 0.25 PVI. The water injected is measured in pore volume injected (PVI)

Comparison	Full model	POD-DEIM
Recovery factor (%)	37.52	37.61
CPU time (in hours)	9.20	2.32
Total Water Injected (in PVI)	0.31	0.32

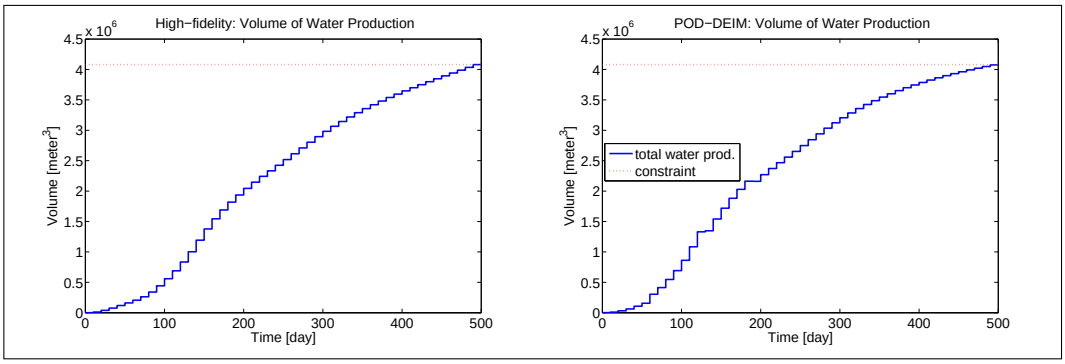

Figure 5.28: The output constraints satisfaction, i.e., the total volume of water production at the final time step.

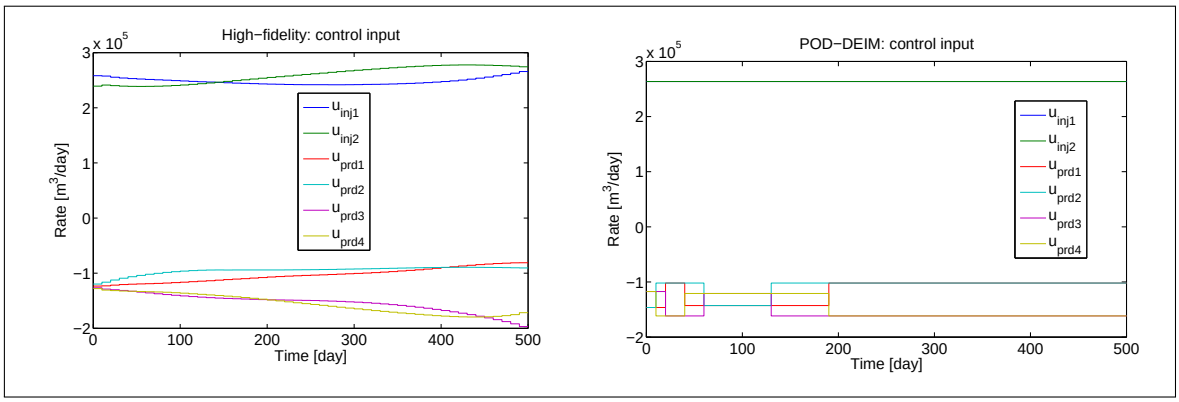

Figure 5.29: Optimization solutions in full-space and reduced-space models.

Initial injection rate 0.3 PVI

We then run optimization with initial injection rate 0.3 PVI. The results are described in Table 5.15, Figure 5.30, and Figure 5.31. The optimization in this case terminates due to the minimum trust-region radius.

Table 5.15: Optimization results with initial injection rate 0.3 PVI. The water injected is measured in pore volume injected (PVI)

Comparison	Full model	POD-DEIM
Recovery factor (%)	37.29	37.28
CPU time (in hours)	7.30	3.57
Total Water Injected (in PVI)	0.32	0.32

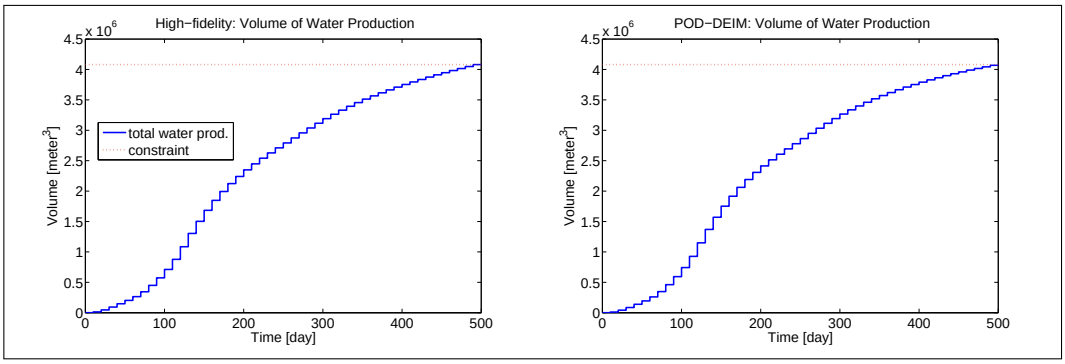

Figure 5.30: The output constraints satisfaction, i.e., the total volume of water production at the final time step.

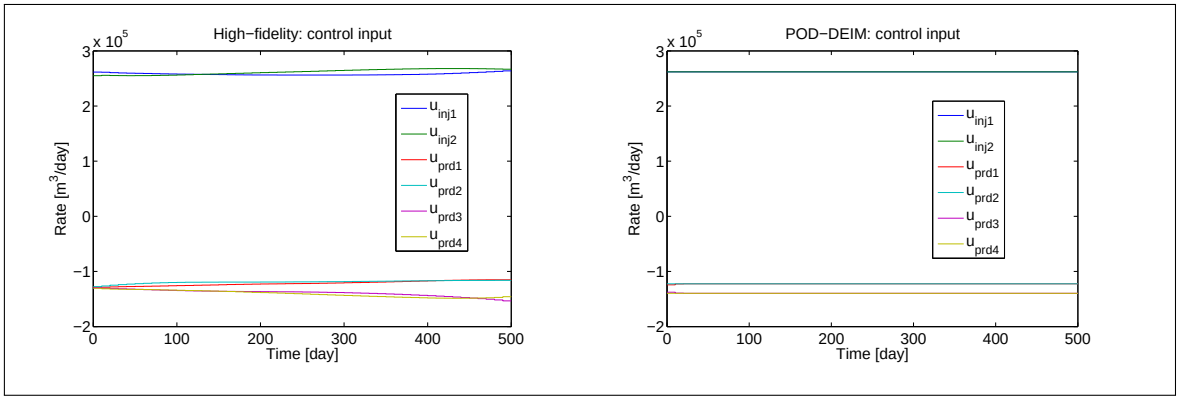

Figure 5.31: Optimization solutions in full-space and reduced-space models.

5.5.5 Discussion

We have presented four case examples. The first case example shows the quality of a reduced-order model given a variation of energy truncation. It turns out that the POD method is more CPU intensive than the POD-DEIM simulation run. Reducing the energy truncation will reduce CPU time at the expense of accuracy. The case examples shown use a 2D reservoir consisting of 13200 grid blocks. The POD method results in significant speedup for the pressure equation. However, for the saturation equation the CPU time in the reduced-order model is marginally faster than the high-fidelity model. In the first case it is shown that DEIM can improve slightly the CPU time of saturation equation. Using larger number of grid blocks may give further speedup. Furthermore, in the saturation adjoint equation we loose the sparsity property. Consequently, the sparse linear solver, which is used to solve the adjoint equation, consumes

more CPU time.

In the second case, we demonstrate the performance of the TRPOD method. Based on the first case, we continue the optimization only with the DEIM reduced-order model to the next cases because the method give more speedup than POD. Using two different initial controls (injector rate), the TRPOD method is trapped into local maxima with comparable objective function (NPV) values. This implies that the choice of initial control and initial trust-region radius are important considerations. The speedups in this case are significant, between 5 and 20 times for the two different initial controls. Another finding from Sachs [2009], shown in Figure 5.32, seems to be similar to what the second case implies. The reduced-order model may have a smaller '*region of attraction*' than the high-fidelity model.

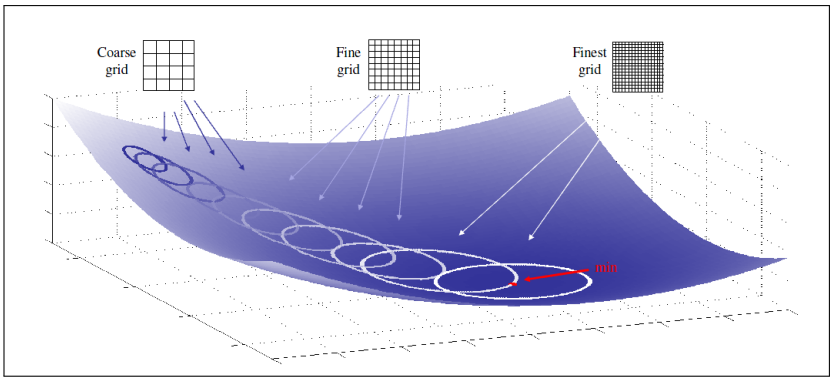

Figure 5.32: A correlation between different scale of grid (coarse/fine) can yield different local maxima. This figure is taken from Sachs [2009] without any modification.

In the third case, we introduce output constraints in reduced-space optimization. The output constraints are water cut at the producer wells. This represents a multidimensional output constraint problem. The TRPOD combined with Lagrangian barrier does not achieve the same solution as the high-fidelity optimization. We have tried to use two different initial controls. The water cut value at one of the producer wells is active, while the other wells are almost active. The speedup in this case is slightly faster than the high-fidelity optimization.

In the fourth case, we apply TRPOD and the Lagrangian barrier method to constrain total water production. This case is a one-dimensional output constraint problem. The results from this case show that our proposed method is able to make the optimization in reduced-space converge to a better local maxima than that of high-fidelity optimization. Moreover, the constraint is active and give speed up factor more than two times.

Apart from the results above, both the TRPOD and Lagrangian barrier methods rely on some parameter settings. In the Lagrangian barrier method we need to supply

suitable values of initial μ and its stopping criteria, and the TRPOD method needs still more parameters, which are the minimum trust-region radius $\triangle_{min}$, its initial value $\triangle_0$, and its maximum value $\triangle_{max}$. These parameter values do obviously have an impact on the optimization results. To find good values for them, it would be interesting to use a derivative-free optimization method, see e.g. Audet & Orban [2006], rather than to use a heuristic approach.

Another point worthwhile to note is the fact that gradient-based optimization is sensitive to initial guess values. We have therefore run some optimizations with different initial controls. The results consistently show that the surrogate optimization have substantial lower CPU time while honoring the nonlinear output constraints.

5.6 Conclusion

The use of the TRPOD method in two case examples has been presented in this paper. Two kinds of model order reduction techniques, the POD and POD-DEIM methods, have been explained. Because of the nonlinear nature of oil reservoirs, particularly the water saturation equation, the POD method may result in a slight speedup in terms of CPU runtime. To get more CPU time speedup, we use POD-DEIM that is consistently faster for the forward equations. In the 2D reservoir example, the sparse linear solver seems to be efficient to solve the linear equation of the adjoint systems. Hence, the corresponding adjoint-saturation equation in the reduced-order model cannot be faster. Surrogate optimization using the POD-DEIM reduced-order model have shown to give considerable speedup and may also give a comparable objective function value. In addition, result from surrogate optimization can be used as an initial guess to an optimization algorithm using a high-fidelity model.

The Lagrangian barrier method is sensitive to the choice of algorithm parameters, such as the barrier multiplier. In addition, the TRPOD method also requires suitable choices of parameter values. The choice of these parameters will affect the optimization solution.

The state equations in this work are solved using a sequential method, that is, implicit-pressure and implicit-saturation solvers. Here, the fluxes are solved explicitly and the pressure are computed afterwards. The POD-DEIM is then applied to the implicit water saturation equation. In commercial reservoir simulators, a fully-implicit method, that is, implicit solutions for both pressure and saturation are commonly used. We foresee the use of POD-DEIM in fully-implicit reservoir simulator, where it may give a better speedup and reasonable approximation of the high-fidelity than those using the vanilla POD method.

Chapter 6

Concluding Remarks and Recommendations for Further Work

The goal of this thesis is to use concepts from system and control in dealing with the increasing challenges in reservoir engineering problems. One of the problems which has been discussed intensely in this thesis is production optimization. Chapter 1 has given an overview of the reservoir engineering problems. Two methods and computation tools from system and controls have been used in this thesis. First, the adjoint method from optimal control theory which enables efficient gradient computation of the objective function independent of the number of decision variables. Second, model order reduction (MOR) techniques for approximating the oil reservoir models and the resulting reduced-order models are adapted and utilized, giving a substantial CPU time speedup while maintaining a reasonable quality of approximation.

6.1 Concluding Remarks

The primary topic has been nonlinear output constraint handling. Chapter 3 explains the proposed method, i.e., the Lagrangian barrier method and further demonstrates the applicability of the method in three case examples. Two alternative methods, SLQP and the pure barrier method, are compared to the Lagrangian barrier method. Initial control input values given to the three methods must be feasible ones. In all case examples, the Lagrangian barrier method consumes less CPU time than the other methods. Hence, the method preserves the efficiency of the adjoint method. The case examples represent one-dimensional and multidimensional output constraints. In the multidimensional cases, the Lagrangian multiplier estimates can be used as information on which constraints are active. The Lagrangian barrier is shown to provide accurate handling of active constraints without violating the nonlinear output constraints. The method maintains the feasibility of the given initial control inputs and yields a solution

on the boundary of the feasible domain. On the other hand, the Lagrangian barrier method requires good initial values of barrier parameters. The stopping criteria is also an important consideration. In all case examples, the optimization terminates due to the preset tolerance on objective function changes.

First-order gradient optimization (steepest descent) is slow to converge. In Chapter 4 we have developed a Hessian-times-vector product using the adjoint method in Steihaug's conjugate gradient method, known as the Truncated Newton (TN) method. We compare our proposed method to the BFGS method. It was shown in two case examples that the TN method improves the convergence rate. However, the computational cost for each iteration is increased due to the overhead in runtime for computing the Hessian-times-vector product. Instead of solving one adjoint equation and one forward equation as in the first-order adjoint-gradient, the Hessian-times-vector procedure requires two corresponding adjoint equations and two corresponding state equations. Therefore, the choice of the number of inner iterations in the conjugate gradient algorithm is an important input. We have fixed this value to 5 in all case examples. Fine tuning of this number may result in fewer Hessian-times-vector function calls and can potentially reduce the CPU time. The case examples use well-rate and BHP-control. For well-rate control, the BFGS method gives comparable results to the TN method. Although it uses more iterations to converge, BFGS still gives favorable CPU time. Based on this, we use well-rate control along with the BFGS method in the case examples in Chapters 3 and 5. The well-rate control case seems to have the same local optima given variations of the initial control inputs. On the contrary, in the BHP control case each initial control input may result in different local optima. On average, the objective function value for the TN method is better than the BFGS. The optimization in all case examples terminate mostly due to the absolute gradient stopping criteria and a few times because of the step length tolerance.

Chapter 5 also addresses nonlinear output constraints and shows the use of reduced-order models in order to speedup the optimization runtime. In addition to the stopping criteria in the Lagrangian barrier method, the size of the minimum trust-region radius is also used as a stopping criterion. Four case examples are presented to test the quality of POD and DEIM, and the performance of surrogate optimization with and without the presence of nonlinear output constraints. In out test-cases we found that the POD method for the non-linear water saturation equation can be almost comparable to the high-fidelity simulation run. DEIM mitigates this by building another basis function for the nonlinear water cut term. The speedup in the case examples can be up to a factor of 20.

6.2 Further work

It is an obvious need to further demonstrate the proposed methods in this thesis to real oil reservoirs.

Nonlinear Output Constraints Handling

The manifestation of output constraints are shown as water cut constraints, total water production, and in hierarchial optimization. The nonlinear output problem can also appear in well placement optimization as in Zhang et al. [2010]. It is thus interesting to test our proposed method for the well placement optimization problem.

Second-order Adjoint Information for Production Optimization

To reduce the CPU time of the Hessian-times-vector algorithm, it would be beneficial to apply model order reduction techniques. Using different kinds of preconditioner matrices for the Hessian might also reduce the CPU time. Furthermore, the Lanczos-CG method may lead to better results than the Steihaug-CG method.

The use of conjugate gradients with Hessian-times-vector supplied by the adjoint method can be beneficial in parameter estimation (history matching) optimization. This may be particularly interesting for the iterative ensemble Kalman filter (IEnKF) [Li & Reynolds, 2009] in which an optimization algorithm is used for better handling of nonlinearities in oil reservoirs rather than the standard EnKF.

Surrogate Optimization of Oil Reservoir Water Flooding

In this work the choice of POD basis functions is paramount. Due to the limited operating range, the POD basis functions are updated with a trust-region strategy during the course of optimization. However, there exist plenty variants of POD methods to extend the operating range of the POD model. One example is Burkardt et al. [2006] which may give a better approximation since it includes an extrapolation strategy. Furthermore, the TRPOD method depends heavily on some important parameters, namely, the initial and maximum trust-region radius. Methods like adaptive TRPOD Sachs [2009] can be applied to partially overcome dependency of these parameters.

References

Aanonsen, S. I., Naevdal, G., Oliver, D. S., Reynolds, A. C., & Valles, B. (2009). The Ensemble Kalman Filter in Reservoir Engineering–a Review. *SPE Journal*, 14:3, 393–412.

Aarnes, J. E., Gimse, T., & Lie, K. A. (2007). *An introduction to the numerics of flow in porous media using Matlab*, (pp. 265–306). Springer Verlag.

Agarwal, A. (2010). *Advanced Strategies for Optimal Design and Operation of Pressure Swing Adsorption Processes*. PhD thesis, Carnegie Mellon University.

Agarwal, A. & Biegler, L. T. (2011). A trust-region framework for constrained optimization using reduced order modeling. *Optimization and Engineering*, online first. dx.doi.org/10.1007/s11081-011-9164-0.

Alexandrov, N. M., Lewis, R. M., Gumbert, C. R., Green, L. L., & Newman, P. A. (2001). Approximation and model management in aerodynamic optimization with variable-fidelity models. *Journal of Aircraft*, 38:6, 1093 – 1101.

Asheim, H. A. (1986). Optimal control of water drive. SPE 15978. *SPE Journal*.

Audet, C. & Orban, D. (2006). Finding optimal algorithmic parameters using derivative-free optimization. *SIAM Journal of Optimization*, 17, 642–664.

Aziz, K. & Settari, A. (1979). *Petroleum Reservoir Simulation*. Applied Science Publisher.

Barrault, M., Maday, Y., Nguyen, N. C., & Patera, A. T. (2004). An 'empirical interpolation' method: application to efficient reduced basis discretization of partial differential equations. *Comptes Rendus Mathematique*, 339:9, 667–672.

Becerra, V. M. (2004). Solving optimal control problems with state constraints using

nonlinear programming and simulation tools. *IEEE Trans. On Education*, 43, No.3, 377 – 384.

Bergman, M., Cordier, L., & Brancher, J. P. (2005). Control of the cylinder wake in the laminar regime by trust-region methods and POD reduced order models. In *Proceeding of IEEE CDC 2005. Seville, Spain. pp. 524-529*.

Biegler, L. T. (2010). *Nonlinear Programming. Concepts, Algorithms, and Applications to Chemical Processes*. MOS-SIAM Series on Optimization. SIAM.

Bloss, K. F., Biegler, L. T., & Schiesser, W. E. (1999). Dynamics process optimization through adjoint formulations and constraint aggregation. *Ind. Eng. Chem. Res*, 38:2, 421 – 432.

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

Brouwer, D. R. & Jansen, J. D. (2004). Dynamic optimization of waterflooding with smart wells using optimal control theory. *SPE Journal*, 9:4, 391 – 402.

Bryd, R. H., Gould, N. I. M., Nocedal, J., & Waltz, R. A. (2004). An algorithm for nonlinear optimization using linear programming and equality constrained subproblems. *Math. Program., Ser. B*, 100, 27–48.

Bryson, A. & Ho, Y. (1975). *Applied Optimal Control*. Hemisphere.

Burkardt, J., Gunzburger, M., & Lee, H. C. (2006). Centroidal voronoi tessellation-based reduced-order modeling of complex systems. *SIAM Journal of Scientific Computing*, 28:2, 459–484.

Byrd, R. H., Nocedal, J., & Waltz, R. A. (2006). Knitro: An integrated package for nonlinear optimization. *Large-Scale Nonlinear Optimization*, 83, 35 – 59.

Cardoso, M. A. (2009). *Development and Application of Reduced-Order Modeling Procedures for Reservoir Simulation*. PhD thesis, Stanford University.

Cardoso, M. A. & Durlofsky, L. J. (2010a). Linearized reduced-order models for subsurface flow simulation. *Journal of Computational Physics*, 229:3, 681 – 700.

Cardoso, M. A. & Durlofsky, L. J. (2010b). Use of reduced-order modeling procedures for production optimization. *SPE Journal*, 15:2, 426 – 435.

Cardoso, M. A., Durlofsky, L. J., & Sarma, P. (2009). Development and application of reduced-order modeling procedures for subsurface flow simulation. *International Journal for Numerical Methods in Engineering*, 77:9, 1322 – 1350.

Chaturantabut, S. & Sorensen, D. C. (2010). Nonlinear model order reduction via discrete empirical interpolation. *SIAM J. Sci. Comp.*, 32:5, 2737–2764.

Chaturantabut, S. & Sorensen, D. C. (2011). Application of POD and DEIM on dimension reduction of non-linear miscible viscous fingering in porous media. *Mathematical and Computer Modelling of Dynamical Systems*, 17:4, 337–353.

Chen, C. (2011). *Adjoint-gradient-based production optimization with Augmented lagrangian method.* PhD thesis, The University of Tulsa.

Chen, C., Li, G., & Reynolds, A. (2011). Robust Constrained Optimization of Short and Long-term NPV for Closed-Loop Reservoir Management. SPE 141314. In *Proceeding SPE RSS, Houston, TX, U.S.A, 21–23 February*.

Chen, Y., Oliver, D. S., & Zhang, D. (2008). Efficient ensemble-based closed loop production optimization. SPE-112873. In *Proceeding of SPE/DOE Improved Oil Recovery Symposium,Tulsa, Oklahoma, U.S.A, 19-23 April*.

Christie, M. A. & Blunt, M. J. (2001). Tenth SPE comparative solution project: A comparative of upscaling technique. *SPE Reservoir Eval.Eng*, 4, 308 – 317.

Conn, A. R., Gould, N. I. M., & Toint, P. L. (1997). A Globally Convergent Lagrangian Barrier Algorithm for Optimization with General Inequality Constraints and Simple Bounds. *Mathematics of Computation*, 66:217, 216 – 288.

Conn, A. R., Gould, N. I. M., & Toint, P. L. (2000). *Trust-Region Methods.* SIAM.

Conn, A. R., Scheinberg, K., & Vicente, L. N. (2009). *Introduction to Derivative-Free Optimization.* MPS-SIAM Book Series on Optimization. SIAM.

Davis, T. A. (2006). *Direct Methods for Sparse Linear Systems.* SIAM.

Dembo, R. S., Eisenstat, S. C., & Steihaug, T. (1982). Inexact Newton methods. *SIAM J. Numer. Anal*, 19, 400 – 408.

Evensen, G. (2009). *Data Assimilation: The Ensemble Kalman Filter*. Springer.

Fahl, M. (2000). *Trust-region Methods for Flow Control based on Reduced Order Modelling*. PhD thesis, Trier University.

Fiacco, A. V. & McCormick, G. P. (1968). *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. John Wiley.

Forouzanfar, F., Li, G., & Reynolds, A. C. (2010). A two-stage well placement optimization method based on adjoint gradient. SPE 135304. In *Proceeding of SPE Annual Technical Conference and Exhibition, 19-22 September 2010, Florence, Italy*.

Forsgren, A., Gill, P. E., & Wright, M. H. (2002). Interior methods for nonlinear optimization. *SIAM Review*, 44, No.4, 525 – 597.

Foss, B. (2011). Process control in conventional oil and gas production - Challenges and opportunities Control Engineering. *To appear in Control Engineering Practice*.

Foss, B. & Jensen, J. P. (2011). Performance analysis for closed-loop reservoir management. *SPE Journal*, 16:1, 183–190.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc.

Grepl, M. A., Maday, Y., Nguyen, N. C., & Patera, A. T. (2007). Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equations. *ESAIM: Mathematical Modelling and Numerical Analysis*, 41:3, 575–605.

Griewank, A. & Korzec, M. (2005). Approximating Jacobians by the TR2 formula. *Proc. Appl. Math. Mech.*, 5, 791–792.

Griewank, A. & Walther, A. (2008). *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. SIAM.

Gu, Y. & Oliver, D. S. (2007). An Iterative Ensemble Kalman Filter for Multiphase Fluid

Flow Data Assimilation. *SPE Journal*, 12:4, 438–446.

Gunzburger, M. D. (2003). *Perspective in Flow Control and Optimization*. SIAM.

Hadamard, J. (1902). Sur les problems aux derives partielles et leaur signification physique. *Princeton University Bulletin*, (pp. 49–52).

Handels, M. & Zandvliet, M. J. (2007). Adjoint-based well-placement optimization under production constraints. SPE 105797. In *Proceeding of SPE RSS, Houston, TX, U.S.A, 26–28 February*.

Hargraves, C. & Paris, S. (1987). Direct trajectory optimization using nonlinear programming and collocation. *Journal Guidance Cont. Dyn.*, 10:4, 338 – 342.

Hasan, A., Foss, B., & Sagatun, S. (2011). Flow control of fluids through porous media. *Applied Mathematics and Computation*.

Heijn, T., Markovinovic, R., & Jansen, J. D. (2004). Generation of low-order reservoir models using system-theoretical concepts. *SPE Journal*, 9:2, 202–218.

Heinkenschloss, M. (2008). *Numerical Solution of Implicitly Constrained Optimization Problems*. Technical report, Dept. of Computational and App. Math., Rice University.

Hinze, M., Pinnau, R., Ulbrich, M., & Ulbrich, S. (2009). *Optimization with PDE Constraints*. Springer.

Hinze, M. & Volkwein, S. (2005). Proper orthogonal decomposition surrogate models for nonlinear dynamical systems: error estimates and suboptimal control. In *P.Benner, V.Mehrmann, D.C. Sorensen (eds.), Reduction of Large-Scale Systems, Lecture Notes in Computational Science and Engineering, Vol 45*. Springer, Berlin.

Hooke, R. & Jeeves, T. A. (1961). "Direct Search" solution of numerical and statistical problems. *J. ACM*, 8:2, 212 – 229.

Horst, R., Pardalos, P. M., & Thoai, N. V. (2000). *Introduction to Global Optimization*. Springer.

Ito, K. & Kunisch, K. (2000). Newton's method for a class of weakly singular optimal

control problems. *SIAM Journals Optimization*, 10:3, 896 – 916.

Ito, K. & Kunisch, K. (2008). *Lagrange Multiplier Approach to Variational Problems and Applications*. SIAM.

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

Jansen, J. D., Brouwer, D. R., Naevdal, G., & van Kruijsdijk, C. P. J. W. (2005). Closed-loop reservoir management. *First Break*, 23, 43–48.

Jansen, J. D., Douma, S. D., Brouwer, D. R., Van den Hof, P. M. J., Bosgra, O. H., & Heemink, A. W. (2009a). Closed loop reservoir management. SPE-119098. In *Proceeding SPE Reservoir Simulation Symposium, 2-4 February The Woodlands, Texas*.

Jansen, J. D., van Doren, J. F. M., Heidary-Fyrozjaee, M., & Yortsos, Y. C. (2009b). *Front controllability in two-phase porous media flow*, (pp. 203–219). Springer Verlag.

Jittorntrum, K. & Osborne, M. R. (1980). A modified barrier function method with improved rate of convergence for degenerate problems. *J. Austral. Math. Soc, Series B*, 21, 305 – 329.

Kang, W. & Xu, L. (2009). A quantitative measure of observability and controllability. In *Proceeding of Joint 48th IEEE CDC and 28th Chinese Control Conference, Shanghail, P.R. China, December 16-18*.

Kang, W. & Xu, L. (2010). Analyzing control systems by using dynamic optimization. In *Proceeding of 8th IFAC Symposium on Nonlinear Control Systems, University of Bologna Italy, September 1-3*.

Kim, J., Bates, D. G., & Postlethwaite, I. (2006). Nonlinear robust performance analysis using complex-step gradient approximation. *Automatica*, 42, 177–182.

Kraaijevanger, J. F. B. M., Egberts, P. J. P., Valstar, J. R., & Buurman, H. W. (2007). Optimal waterflood design using the adjoint method. SPE 105764. In *SPE Reservoir Simulation Symposium, 26-28 February, Houston, Texas, U.S.A.*

Kreisselmeier, G. & Steinhauser, R. (1979). Systematic control design by optimizing a

vector performance index. In *Proceeding of IFAC Symposium on CADS, Zurich*.

Krogstad, S. (2011). A sparse basis POD for model reduction of multiphase compressible flow. SPE-141973. In *Proceeding of the SPE 2011 Reservoir Simulation Symposium*.

Krogstad, S., Hauge, V. L., & Gulbransen, A. F. (2011). Adjoint multiscale mixed finite elements. *SPE Journal*, 16:1, 162 – 171.

Leemhuis, A. P., Belfroid, S. P. C., & Alberts, G. J. N. (2007). Gas coning control for smart wells. SPE 110317. In *Prooceeding of the 2007 Annual Technical Conference and Exhibition, November 11-14*.

Leemhuis, A. P., Nennie, E. D., Belfroid, S. P. C., Alberts, G. J. N., Peters, E., & Joosten, G. J. P. (2008). Gas coning control for smart wells using a dynamic coupled well-reservoir simulator. SPE 112234. In *Proceeding of the 2008 SPE Intelligent Energy Conference and Exhibition, Amsterdam, The Netherlands, 25-27 February*.

Li, G. & Reynolds, A. C. (2009). Iterative Ensemble Kalman Filters for Data Assimilation. *SPE Journal*, 14:3, 496–505.

Li, Y., Navon, M., Courtier, P., & Gauthier, P. (1993). Variational data assimilation with a semi-lagrangian semi-implicit global shallow-water equation model and its adjoint. *Monthly Weather Review*, June, 1759–1769.

Lie, K.-A., Krogstad, S., Ligaarden, I. S., Natvig, L. R., Nilsen, H. M., & Skafltestad, B. (2011). Open-source MATLAB implementation of consistent discretisations on complex grids. *Computational Geosciences*, online first.

Markovinovic, R. (2009). *System-Theoretical Model Reduction for Reservoir Simulation and Optimization*. PhD thesis, TU. Delft.

Markovinovic, R., Geurtsen, E. L., Heijn, T., & Jansen, J. D. (2002a). Generation of low-order reservoir models using POD, empirical gramians and subspace identification. In *Proc. 8th European conference on the mathematics of oil recovery (ECMOR). Freiberg, Germany, pp. E31-1-E31-10*.

Markovinovic, R., Geurtsen, E. L., & Jansen, J. D. (2002b). Subspace identification of

low-order reservoir models. In *Proceeding of IV international conference on computational methods in water resources, Delft, The Netherlands. pp. 281-288.*

Markovinovic, R. & Jansen, J. D. (2006). Accelerating iterative solution methods using reduced-order models as solution predictors. *International Journal for Numerical Methods in Engineering*, 68:5, 525 – 541.

Mehra, R. & Davis, R. (1972). A generalized gradient method for optimal control problems with inequality constraints and singular arch. *IEEE Transactions on Automatic Control*, 17, 69 – 79.

Montleau, P. d., Cominelli, A., Neylon, K., Rowan, D., Pallister, I., Tesaker, O., & Nygard, I. (2006). Production optimization under constraints using adjoint gradient. In *Proceedings of ECMOR X-10th European Conference on the Mathematics of Oil Recovery. Number A041, Amsterdam, The Netherlands.*

Naevdal, G., Mannseth, T., & Vefring, E. H. (2002). Near-well reservoir monitoring through ensemble kalman filter. SPE 75235. In *Proceeding of SPE/DOE Improved Oil Recovery Symposium, 13-17 April, Tulsa, Oklahoma.*

Nocedal, J. & Wright, S. J. (2006). *Numerical Optimization.* Springer.

Oliver, D. S. & Chen, Y. (2010). Recent progress on reservoir history matching: a review. *Computational Geosciences*, 15:1, 185–221.

Peaceman, D. (1983). Interpretation of well-block pressures in numerical reservoir simulation with nonsquare grid blocks and anisotropic permeability. *SPE Journal*, 23:3, 531–543.

Peters, L., Arts, R. J., Brouwer, G. K., Geel, C. R., Cullick, S., Lorentzen, R. J., Chen, Y., Dunlop, K. N. B., Vossepoel, F. C., Xu, R., Sarma, P., Alhuthali, A. H., & Reynolds, A. C. (2010). Results of the brugge benchmark study for flooding optimization and history matching. *SPE Reservoir Evaluation & Engineering*, 13, Number 3, 391–405.

Raffard, R. L., Amonlirdviman, K., Axelrod, J. D., & Tomlin, C. J. (2008). An adjoint-based parameter identification algorithm applied to planar cell polarity signaling. *IEEE Transaction of Automatic Control, Special Issue on System Biology*, 53, 109 – 121.

Raffard, R. L. & Tomlin, C. J. (2005). Second order adjoint-based optimization of ordinary and partial differential equations with application to air traffic flow. In *Proceedings of 2005 American Control Conference. pp. 798-803.*

Robinson, T. D. (2007). *Surrogate-Based Optimization using Multifidelity Models with Variable Parameterization.* PhD thesis, Massachusetts Institute of Technology.

Rommelse, J. R. (2009). *Data Assimilation in Reservoir Management.* PhD thesis, TU Delft.

Rwechungura, R., Suwartadi, E., Dadashpour, M., Kleppe, J., & Foss, B. (2010). The norne field case -a unique comparative case study. SPE 127538. In *Proceeding of SPE Intelligent Energy Conference and Exhibition, Utrecht, The Netherlands.*

Sachs, E. W. (2009). Adaptive trust region POD algorithms.

Saputelli, L., Nikolaou, M., & Economides, M. J. (2005). Self-learning reservoir management. *SPE Reservoir Evaluation & Engineering*, 8:6, 534–547.

Saputelli, L., Nikolaou, M., & Economides, M. J. (2006). Real-time reservoir management: A multiscale adaptive optimization and control approach. *Computational Geosciences*, 10, 61–96.

Sarma, P., Chen, W. H., Durlofsky, L. J., & Aziz, K. (2008). Production optimization with adjoint models under nonlinear control-state path inequality constraints. *SPE Reservoir Evaluation & Engineering*, 11:2, 326–339.

Sarma, P., Durlofsky, L. J., Aziz, K., & Chen, W. H. (2007). A new approach to automatic history matching using kernel PCA. SPE 106176. In *Proceeding of SPE Reservoir Simulation Symposium, 26-28 February, Houston, Texas, U.S.A.*

Schlumberger (2009). Eclipse - technical description. In *Eclipse - Simulation Software Manuals 2009.2*. Schumberger.

Steihaug, T. (1983). The conjugate gradient method and trust regions in large scale optimization. *SIAM Journal on Numerical Analysis*, 20:3, 626–637.

Su, H.-J. & Oliver, D. S. (2010). Smart well production optimization using an ensemble-

based method. *SPE Reservoir Evaluation & Engineering*, 13:6, 884–892.

Suwartadi, E., Krogstad, S., & Foss, B. (2009). On adjoint state constraints of adjoint optimization in oil reservoir waterflooding. SPE 125557. In *Proceeding of SPE Reservoir Simulation and Characteristic Conference, Abu Dhabi*.

Suwartadi, E., Krogstad, S., & Foss, B. (2010a). A Lagrangian-Barrier Function for Adjoint State Constraints Optimization of Oil Reservoir Water Flooding. In *Proceeding of IEEE Conference on Decision and Control 2010, Atlanta, Georgia, USA*.

Suwartadi, E., Krogstad, S., & Foss, B. (2010b). Nonlinear output constraints handling for production optimization of oil reservoirs. In *Proceeding of European Conference on Mathematical Oil Recovery 2010 (ECMOR XII), Oxford, UK*.

Suwartadi, E., Krogstad, S., & Foss, B. (2010c). Second-Order Adjoint-Based Control for Multiphase Flow in Subsurface Oil Reservoirs. In *Proceeding of IEEE Conference on Decision and Control 2010, Atlanta, Georgia, USA*.

Suwartadi, E., Krogstad, S., & Foss, B. (2012a). Adjoint-based Surrogate Optimization of Oil Reservoirs Water Flooding. *Submitted to Journal of Computational Physics*.

Suwartadi, E., Krogstad, S., & Foss, B. (2012b). Nonlinear output constraints handling for production optimization of oil reservoirs. *Computational Geosciences*, 16:2, 499–517.

Troltzsch, F. (2010). *Optimal Control of Partial Differential Equations: Theory, Methods and Applications*. American Mathematical Society.

van Doren, J. F. M., Markovinovic, R., & Jansen, J. D. (2006). Reduced-order optimal control of water flooding using proper orthogonal decomposition. *Computational Geosciences*, 10:1, 137 – 158.

van Doren J. F. M. (2010). *Model Structure Analysis for Model-based Operation of Petroleum Reservoirs*. PhD thesis, TU Delft.

van Essen, G. M., Van den Hof, P. M. J., & Jansen, J. D. (2010). Hierarchical long-term and short-term production optimization. SPE 124332. *SPE Journal*, 16(1), 191 – 199.

Virnovsky, G. A. (1991). Waterflooding strategy design using optimal control theory. In *Proceeding of 6th. European IOR-Symp. Stavanger, Norway.*

Volkwein, S. (2003). Model reduction using proper orthogonal decomposition. In *Lecture Note. Institute of Mathematics and Scientific Computing*. University of Graz.

Walther, A. & Griewank, A. (2004). Advantages of binomial checkpointing for memory-reduced adjoint calculations. In *Proceeding of ENUMATH 2003, Prague, pp. 834 - 843, Springer.*

Wen, Z., Durlofsky, L. J., & van Roy, B. (2011). Use of approximate dynamic programming for production optimization. SPE 141677. In *Proceeding of the SPE 2011 Reservoir Simulation Symposium.*

Wu, Z. (1999). *Conditioning Geostatistical Models to Two-Phase Flow.* PhD thesis, University of Tulsa.

Zakirov, I. S., Aanonsen, S. I., Zakirov, E. S., & Palatnik, B. M. (1996). Optimizing reservoir performance by automatic allocation of well rates. In *Proceeding of 5th. ECMOR, Leoben, Austria.*

Zandvliet, M. J., Handels, M., van Essen, G. M., Brouwer, D. R., & Jansen, J. D. (2008a). Adjoint-based well-placement optimization under production constraints. *SPE Journal*, 13:4, 392–399.

Zandvliet, M. J., Van Doren, J. F. M., Bosgra, O. H., Jansen, J. D., & Van den Hof, P. M. J. (2008b). Controllability, observability, and identifiability. *Computational Geosciences*, 12, 605–622.

Zhang, K., Li, G., Reynolds, A. C., Yao, J., & Zhang, L. (2010). Optimal well placement using an adjoint gradient. *Journal of Petroleum Science and Engineering*, Volume 73, Issues 3-4, 220–226.