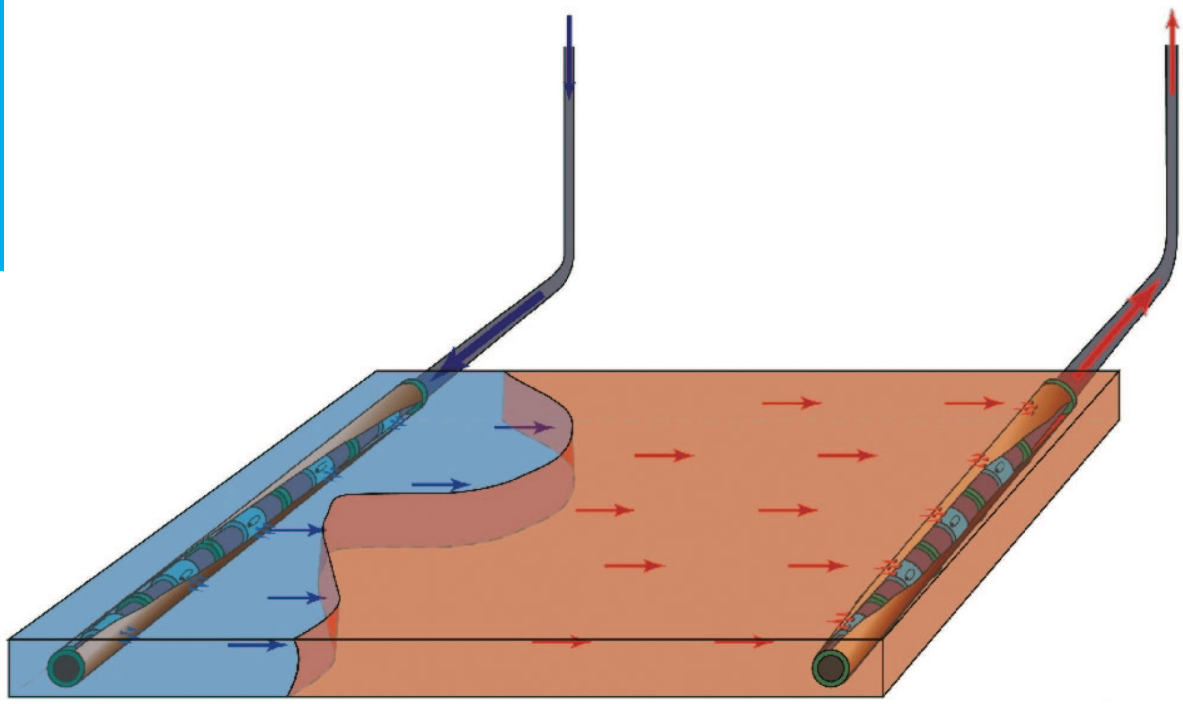
Oil-Reservoir Flooding Optimization using the Simultaneous Method

L.M.C.F. Alblas

Master of Science Thesis



TU Delft Delft University of Technology

Delft Center for Systems and Control

Oil-Reservoir Flooding Optimization using the Simultaneous Method

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

L.M.C.F. Alblas

November 25, 2010

Faculty of Mechanical, Maritime and Materials Engineering · Delft University of Technology

Cover image is adopted from [Brouwer and Jansen, 2004].

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering for acceptance a thesis entitled

Oil-Reservoir Flooding Optimization using the Simultaneous Method

by

L.M.C.F. Alblas

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: November 25, 2010

Supervisor(s):

ir. A.E.M. Huesman

prof.dr.ir. P.M.J. Van den Hof

ir. G.M. van Essen

Reader(s):

ir. A.E.M. Huesman

prof.dr.ir. P.M.J. Van den Hof

ir. G.M. van Essen

prof.dr.ir. A.W. Heemink

prof.dr. S. Weiland

Abstract

Today, oil recovery is often based on reactive control using water flooding. Water is injected to push the oil towards the producing wells until water is being produced, which requires a shutdown of the producing well. This form of production may not yield more than 35% of the oil initially present. A better strategy exists which is based on closed-loop reservoir management (CLRM). CLRM uses a reservoir model which is successively updated and optimized, resulting in a theoretical maximization of the net present value (NPV). The optimization is called flooding optimization.

A flooding optimization problem is an optimal control problem based on a non-linear deterministic reservoir model and solved using the sequential method. This method performs a sequential procedure of integration of every ordinary differential equation (ODE) and optimization using a discrete set of control inputs. The main issue of the sequential method is that it cannot deal with state-constraints directly, resulting in solutions which are in practice infeasible. Furthermore, the optimized control input may suffer from chattering[1]. However, a variance minimization is difficult when using the sequential method. Besides these two issues, the sequential method is a computationally expensive optimization as it performs 30 to 100 ODE integrations. Moreover, implementation of the sequential method may be a laboriously procedure because gradient information has to be pre-programmed manually in the ODE solver.

To overcome the issues of the sequential method, a literature study has been conducted which concluded that the simultaneous method should provide a solution. The simultaneous method is used in the chemical industry to solve the issue of handling state-constraints. Besides, the method can be implemented using algebraic modeling, providing automatic discretization support and symbolic differentiation. Although the simultaneous method may provide a solution to the issues of the sequential method, it is known to have several limitations as well. The first limitation is the large non-linear programming (NLP) problem which is obtained due to full discretization in both space and time of all variables and states. The second limitation is that it may be difficult to obtain an initial guess (IG) for all discrete variables, because the IG has to satisfy all constraints.

[1] Chattering refers to the optimized non-unique solution to the control input, which causes the input-profile to behave irregularly.

It is investigated how the simultaneous method can be applied to a flooding optimization problem, what the limitations are of the current implementation and if the performance is comparable to the results obtained by the sequential method.

Application of the simultaneous method to the flooding optimization problem results in a set of difference equations due to the spatial and time discretization of every partial differential equation (PDE). The states are integrated using implicit Euler. The simultaneous method is implemented using General Algebraic Modeling System (GAMS). The IGs of the implementation and its verification are based on Simple Simulator (SimSim). SimSim is a matrix-oriented forward reservoir simulator. The known limitations of the simultaneous method, which are having a large NLP and being sensitive to IGs, are investigated. The sequential method is compared to the simultaneous method. It is investigated to what extent the simultaneous method handles state-constraints and whether the simultaneous method can be used for multi-objective optimization. The optimization performance of both methods is compared as well. Modular Reservoir Simulator (MoReS), the in-house simulator of the Dutch oil company Shell, has been used to perform the sequential optimizations.

The GAMS model is verified and it is concluded that the simultaneous method can be applied to a flooding optimization problem. GAMS provides automatic discretization support and symbolic differentiation. There are two drawbacks of the current implementation of the simultaneous method. The first is the fact that it is sensitive to IGs, meaning that an incorrectly selected IG will not result in an optimal NPV. This is due to the distinct modeling environments of GAMS and the IG generator, resulting in an initial error. The second drawback is that the reservoir model cannot be scaled up to more than 16 grid blocks. The main issues of the sequential method, which are dealing with state-constraints and having chattering control inputs, are avoided by using the simultaneous method. The dynamics of both models in forward simulations appear to be similar, but distinct modeling environments result in differences in the underlying models. The methods are therefore not directly comparable. Optimization using both methods resulted in different optimal injection and production strategies. GAMS did not start directly with the injection of water where MoReS did. The optimal solution of GAMS results in a lower total production rate with a lower saturation and produces for this reason less water. The optimized NPVs of both methods differ not more than 1%, where GAMS obtains the highest NPV.

For future research it is recommended to discretize the ODEs or PDEs using orthogonal collocation. This includes a function approximation of the states and enables to decrease the number of discretization points, resulting in a smaller NLP. It is also recommended to eliminate the small differences between the IG generator and the simultaneous method by using an equation-oriented modeling environment for the IGs. These improvements will decrease the method's sensitivity to IGs and improve the possibilities for scaling up the model. At last, the comparison of the simultaneous method to the sequential method can be improved to get a better understanding of the convergence properties, computational speed and optimal result.

Table of Contents

Acknowledgements vii

1 Introduction 1

1-1 Background Information . . . 1

1-1-1 Oil Recovery . . . 1

1-1-2 Flooding Optimization . . . 2

1-1-3 Closed-Loop Reservoir Management . . . 2

1-1-4 Non-Linear Optimal Control Methods . . . 3

1-2 Problem Statement . . . 4

1-3 Approach . . . 5

2 Description of the Sequential Method and the Simultaneous Method 7

2-1 Dynamic Non-Linear Optimization . . . 7

2-1-1 Dynamic Optimization . . . 7

2-1-2 Non-Linear Programming . . . 8

2-2 Sequential Method . . . 8

2-2-1 Theory . . . 9

2-2-2 Advantages and Disadvantages . . . 10

2-3 Simultaneous Method . . . 11

2-3-1 Theory . . . 11

2-3-2 Advantages and Disadvantages . . . 14

2-4 Summary and Concluding Remarks . . . 14

3 Application of the Simultaneous Method to a Flooding Optimization Problem 15
3-1 Motivation for using a Two-Phase Two-Dimensional Model . . . 15
3-2 Transformation of the Model PDEs into ODEs . . . 16
3-2-1 PDE Representation . . . 16
3-2-2 Simplifications . . . 17
3-2-3 Simplified PDE Representation . . . 18
3-2-4 ODE Representation . . . 18
3-3 Discretization of the ODEs . . . 18
3-3-1 Spatial Discretization . . . 18
3-3-2 Difference Equation Representation . . . 22
3-3-3 Time Discretization and Integration . . . 22
3-4 Optimization . . . 22
3-4-1 Objective Function . . . 22
3-4-2 Physical Constraints . . . 23
3-4-3 Boundaries . . . 24
3-5 Concluding Remarks . . . 24

4 Implementation of a Flooding Optimization Problem in GAMS 25
4-1 Model Verification . . . 25
4-1-1 The Initial Error in GAMS due to the Initial Guess . . . 26
4-1-2 Comparison of GAMS with the Forward Simulator SimSim . . . 27
4-2 Selection of the Best Performing Solver . . . 29
4-2-1 Description of Different Solvers . . . 29
4-2-2 Comparison of Different Solvers . . . 30
4-3 Scaling . . . 32
4-4 Sensitivity to Initial Guesses . . . 33
4-4-1 Sensitivity Measure to Different Initial Guesses . . . 34
4-4-2 Sensitivity to Ill-Conditoned Initial Guesses . . . 36
4-5 Concluding Remarks . . . 36

5 Comparison of the Simultaneous Method and the Sequential Method 39
5-1 Applying State-Constraints . . . 39
5-1-1 Single Pressure Constraint . . . 39
5-1-2 Reservoir Pressure Constraint . . . 40
5-2 Multi-Objective Optimization . . . 40
5-2-1 Regularization . . . 41
5-2-2 Lexicographic Optimization . . . 42
5-3 Comparison with the Sequential Optimization Method . . . 43
5-3-1 Model Verification using a Forward Simulation . . . 44
5-3-2 Comparison of the Optimization Performance . . . 44
5-4 Concluding Remarks . . . 46

6 Conclusions and Recommendations **49**
6-1 Conclusions . . . 49
6-2 Recommendations . . . 50

A Adjoint Method **53**

B Reservoir Modeling based on the Principles of Flow through Porous Media **55**
B-1 Single-Phase Flow . . . 55
B-1-1 Single-Phase One-Dimensional Flow . . . 55
B-1-2 Single-Phase Two-Dimensional and Three-Dimensional Flow . . . 56

C Non-Linear Properties of the used Reservoir Model **59**
C-1 Types of Non-Linearities . . . 59
C-1-1 Elliptic Equation . . . 60
C-1-2 Parabolic Equation . . . 60
C-1-3 Hyperbolic Equation . . . 60
C-2 Sources of Non-Linearities in the Pressure and Saturation Differential Equations . 60
C-2-1 Pressure Differential Equation . . . 61
C-2-2 Saturation Differential Equation . . . 61

D GAMS code and Matlab code for the 2x3 Reservoir Model **63**
D-1 Edited Files from SimSim . . . 63
D-2 Matlab Code for Communication between SimSim and GAMS . . . 65
D-3 GAMS Code . . . 68

Bibliography **79**

Glossary **83**
List of Acronyms . . . 83
List of Symbols . . . 84

Acknowledgements

I would like to thank my supervisors ir. A.E.M. Huesman, prof.dr.ir. P.M.J. Van den Hof and ir. G.M. van Essen for the data and assistance they provided during the writing of my MSc thesis project. Besides I would like to thank prof.dr.ir. J.D. Jansen and prof.dr.ir. A.W. Heemink for their help regarding reservoir engineering and large-scale optimization respectively. The information provided by prof. L.T. Biegler of Carnegie Mellon University and prof. H.D. Mittelmann of Arizona State University has been valuable on judging the different solvers.

Next, I would like to thank my fellow students of Delft Center for Systems and Control (DCSC); Roel Dobbe, Vincent Gusdorf, Robert Hillen and Riny Vermue. Besides the coffee breaks and workshops we had together it proved to be very valuable to simultaneously work on our thesis projects and to review each others presentations and reports. This provided a fun but also productive learning experience.

Finally, I would like to thank my family and girlfriend who did a great job on reviewing my thesis report.

Lodewijk Alblas

Delft, University of Technology
November 25, 2010

L.M.C.F. Alblas

Chapter 1

Introduction

1-1 Background Information

1-1-1 Oil Recovery

Hydrocarbons (oil) are contained in porous heterogeneous rock hundreds to thousands meters below the surface in subsurface oil reservoirs. These reservoirs typically cover an area of several squared kilometers with a height of tens of meters [Jansen et al., 2008]. When a well is drilled, oil will most likely come out naturally due to the over-pressurized reservoir, a form of production called **primary oil recovery**. The over-pressurized reservoir can only provide oil until the reservoir pressure reaches its hydrostatic pressure, resulting in a recovery factor of 5% to 15% [Van den Hof et al., 2009]. The **recovery factor** is defined as the amount of oil produced divided by the amount of oil initially present. After the primary oil recovery stage is finished, the reservoir still contains oil, which can possibly be acquired using secondary and eventually tertiary techniques.

Secondary oil recovery is not only based on the production of oil, but also on injection of water or gas using injection wells. The injected water or gas pushes the oil towards the production wells. When the injected water or gas is being produced, the producing well is shutdown. This is called **reactive control**, illustrated in Figure 1-1. Water flooding is used for over 50% of the oil production in the U.S.A. [Van den Hof et al., 2009]. Secondary techniques will yield a theoretical recovery factor of 20% to 70% of the total amount of oil [Van den Hof et al., 2009].

Tertiary oil recovery techniques such as steam or polymer injection are used to recover oil which is still trapped in pores after the secondary recovery stage. In theory, it is possible to obtain a recovery factor of 90% when using all three techniques of oil recovery.

In practice, tertiary techniques are expensive and with the current oil price often economically infeasible compared to secondary techniques [Jansen et al., 2008]. The practical recovery factor of secondary recovery will not be more than 35%. This is partly due to the economic feasibility, but mostly due to inefficient deployment of the reservoir because no

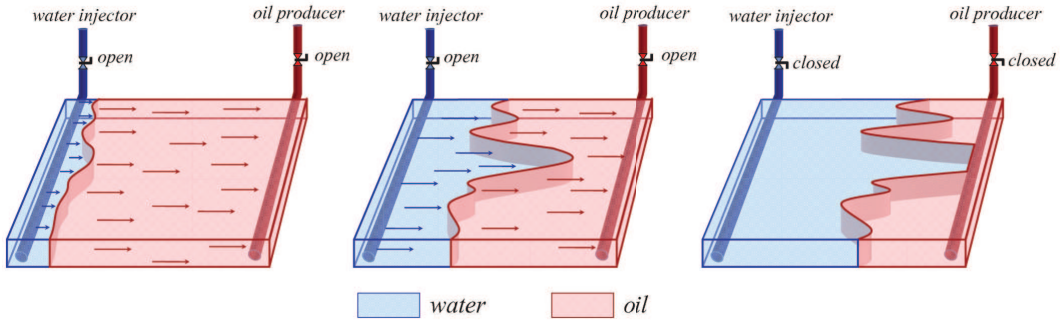


Figure 1-1: Illustration of reactive control using water flooding provided by G.M. van Essen. Water is injected until it is being produced again, resulting in a shutdown of the production well.

knowledge is available on how to use these recovery techniques in an economically efficient way [Van den Hof et al., 2009].

As the recovery factor is lower in practice than in theory, there is still room for improvement of the recovery factor without the cost of any additional production or injection equipment. With this idea flooding optimization has been introduced.

1-1-2 Flooding Optimization

The goal of **flooding optimization** is to maximize the total oil recovery or net present value (NPV) of an oil reservoir model over a finite time horizon. It is an optimal control problem because the underlying reservoir model is dynamic and described by a set of PDEs. In flooding optimization, the controls are often combinations of injection rates, production rates, BHPs and/or valve-settings [Zandvliet et al., 2007]. The bottem-hole pressure (BHP) is the pressure at the opening of the well located in the reservoir.

A **reservoir model** describes the flow of all media within an oil reservoir using an oil phase, a water phase and possibly a gas phase. Such a model is based on the conservation of mass and the conservation of linear momentum. When all three phases are present, a vapor-liquid equilibrium will exist as well [Aziz and Settari, 1979]. Next to these reservoir dynamics, a well model is incorporated which is essential to relate the diffusive behavior of the reservoir pressure with the BHPs and the flow rates of the well [Peaceman, 1983]. The states of the reservoir are described by pressures and saturations, which are volume-percentages of water at a certain point in time in the reservoir. Every well is represented by a BHP and a flow rate, which are related by the states of the model. The physical controlled inputs are injection BHPs and total production rates. The physical observed outputs are injection rates and production BHPs. In a reservoir model, the physical inputs and outputs can be exchanged. Solving a flooding optimization problem requires to discretize the reservoir model in space. This may result in large-scale models with millions of finite volumes or grid blocks [van Essen et al., 2009c].

1-1-3 Closed-Loop Reservoir Management

Flooding optimization is part of a major research area known as 'smart fields' or **closed-loop reservoir management (CLRM)**. CLRM can be interpreted as an adaptive control

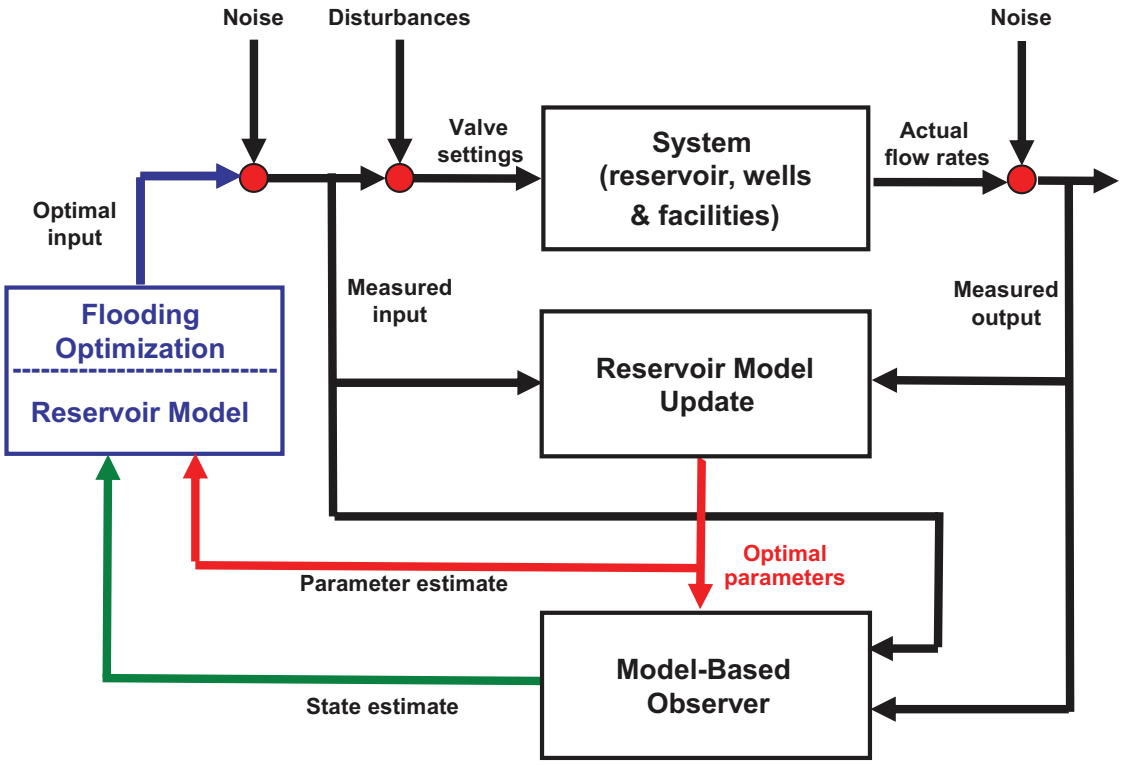


Figure 1-2: Schematic representation of closed-loop reservoir management adapted from [Van den Hof et al., 2009].

technique. Firstly, the reservoir model is defined as a white deterministic physics-based model. Secondly, the model parameters are estimated using heuristic real-life data called data-assimilation. However, when used for academic purpose, synthetic data may be used. Then, thirdly, a flooding optimization determines an optimal control input. After the initialization of the oil production, additional data and knowledge becomes available over time allowing for an improvement of the model. The model will be updated periodically and a flooding optimization will be performed successively. This successive process of CLRM is illustrated in Figure 1-2.

1-1-4 Non-Linear Optimal Control Methods

The behavior of a reservoir is dynamic and highly non-linear; pressure behaves diffusively and saturation diffusive-convectively [Jansen, 2009]. The control problem is therefore a non-linear optimal control problem. Reservoirs have a typical life-time of several decades with a time-delay of multiple years. The optimal control problem is for this reason solved using a reservoir model. Advanced solution techniques are required to solve these types of problems. Three methods can be considered, which are dynamic programming, indirect methods and direct methods.

With **dynamic programming**, all variables and inputs are discretized in time and quantified as a finite set of possible values. This results in a finite set of possible solutions. However, for large optimization problems this method suffers from the *Curse of Dimensionality*, since many

function evaluations are needed to determine the optimal solution. **Indirect methods** first rewrite the optimization problem as a set of first-order necessary conditions for optimality. The indirect method involves intensive work on symbolic manipulation, has problems dealing with inequality constraints and suffers from the difficulty of selecting a suitable initial guess (IG). **Direct methods** transcribe the non-linear optimization problem into a finite time-independent non-linear optimization problem or non-linear programming (NLP) problem using parametrization. Finite refers to a limited number of **decision variables**, which are the variables which may be changed by the NLP solver to obtain the optimum of an objective function.

Direct methods can be classified into two types [Biegler and Grossmann, 2004]. The first direct method is the **sequential method**, in which numerical integration of every differential algebraic equation (DAE) provides state-profiles and gradient information. The integration alternates with an NLP optimization. The second direct method is the **simultaneous method**, in which the underlying model inputs, outputs and states of the optimization problem are fully discretized in time. The states are related through a function approximation of the DAEs, which are applied as supplementary equality constraints. The resulting NLP problem is then solved in one go.

1-2 Problem Statement

The sequential method is used today in industry to solve flooding optimization problems. This method is able to optimize large-scale flooding optimization problems resulting in a theoretical increase in NPV [van Essen et al., 2009c], [Brouwer and Jansen, 2002]. However, the sequential method has also several disadvantages, of which the following are encountered in flooding optimization:

- **State-constraints cannot be handled directly**. The values of the states are obtained by a numerical integration or forward simulation. The states are not incorporated in the finite set of decision variables and thus not included in the NLP problem [Biegler and Grossmann, 2004]. The handling of state-constraints is therefore limited to the control parametrization [Biegler, 2007]. As state-constraints are always present [Sarma et al., 2006], the theoretical increase in NPV may not be feasible when used for practical problems. The handling of state-constraints has still a large scope of further improvement [Kraaijevanger et al., 2007].
- **The optimal control inputs show chattering**[1] **behavior**, which is not accepted as input by production engineers [Jansen et al., 2009]. A solution may be to use lexicographic optimization, which bounds the obtained NPV by adding an inequality constraint and then minimizes the variance. However, the bound on the NPV is dependent on the states and is therefore difficult to implement [van Essen et al., 2010].
- **Repeated numeric integration may become time-consuming for large-scale problems** [Kameswaran and Biegler, 2006]. In the article of [van Essen et al., 2009b],

[1] Chattering refers to the optimized non-unique solution to the control input, which causes the input-profile to behave irregularly.

it can be found one optimization procedure with the sequential method requires about 30 to 100 iterations and thus 30 to 100 integrations of the DAEs. This is a computationally expensive procedure.

- **Programming the gradient information for a sequential flooding optimization is labor-intensive**. The gradient information has to be preprogrammed manually in order to use the sequential optimization method efficiently. This causes a lot of effort as modern reservoir programs may contain up to several million lines of code [Jansen et al., 2009].

Summarizing, the use of the sequential method enables to increase the theoretical NPV. However, in practice, it is still difficult to implement the obtained control input as physical constraints may be violated and chattering is likely to occur. Besides, the sequential method suffers from the expensive cost of multiple integrations of the ordinary differential equation (ODE) and from the difficulty to set up an efficient optimization.

In real-life, flooding optimization uses reservoir models with millions of grid blocks. Such large models cannot be solved efficiently by the earlier explained indirect method or dynamic programming method. The mentioned issues with the sequential method may for this reason be solved with an alternative direct method, which is the simultaneous method. This method applies full discretization. Handling (dynamic) constraints turns out to be possible due to the fact that the states are incorporated in the NLP problem [Biegler and Grossmann, 2004]. Therefore, it is likely a lexicographic optimization can be performed as well, which is demonstrated for a small problem in the article of [Huesman et al., 2008]. As the model is solved at once, the simultaneous method does not require multiple integrations of the DAEs, which may avoid computationally expensive intermediate solutions [Biegler, 2007]. The simultaneous method can be implemented using algebraic modeling software. Algebraic modeling allows to use symbolic differentiation and automatic discretization and therefore avoids manual implementation of gradient information.

Although the simultaneous method sounds attractive, two significant disadvantages are known. Firstly, the NLP is large due to the incorporation of discrete states in the NLP problem. As a result efficient algorithms are required [Biegler, 2007]. Secondly, finding suitable values for the IG can be difficult as all the variables and states must be approximated over the complete time-horizon while satisfying all constraints.

Concluding, the simultaneous method may be an interesting alternative to the sequential method, nonetheless, several issues may arise, resulting in the following **problem statement**:

> **To what extent is the simultaneous method able to solve a flooding optimization problem?**

1-3 Approach

The simultaneous method has not been applied yet to a flooding optimization problem. It is for this reason of interest to discover how the simultaneous method can be applied to

a flooding optimization problem, what the limitations are of this implementation and how this method compares to the sequential method. The problem statement is for this reason separated into three parts:

1. How can the simultaneous method be applied to a flooding optimization problem?
2. What are the limitations of the used implementation?
3. How does the simultaneous method compare to the sequential method?

The **scope** of this thesis will be to research two-dimensional small reservoir models with just a water and an oil phase. The dynamic equations will be integrated similarly as done in industry to allow for verification of the model implementation. Furthermore, the simultaneous method will be implemented using algebraic modeling to make use of the advantages of automatic discretization support and symbolic differentiation.

To get a better understanding of the direct methods, both the sequential method and the simultaneous method are explained in detail in Chapter 2. The question on how to apply the simultaneous method to a flooding optimization problem is discussed in Chapter 3, where a flooding optimization problem is described and discretized such that the simultaneous method can be applied. Next, in Chapter 4, the implementation of the simultaneous method is verified and tested to provide a proof of principle. The effects of the known disadvantages of the large NLP and the sensitivity to IGs are investigated. The comparison of the simultaneous method with the sequential method is discussed in Chapter 5. This chapter investigates whether the problems of dealing with state-constraints and having chattering control inputs can be solved with the simultaneous method. Both methods are compared on their optimization performance as well. Chapter 6 concludes the research and gives further recommendations.

Chapter 2

Description of the Sequential Method and the Simultaneous Method

This chapter will explain how the sequential method and the simultaneous method solve a dynamic optimization problem and will summarize the advantages and disadvantages of both methods. The properties of dynamic non-linear optimization are explained first in Section 2-1. The theory, advantages and disadvantages are presented next in Section 2-2 for the sequential method and in Section 2-3 for the simultaneous method. Section 2-4 will summarize the important properties of both methods.

2-1 Dynamic Non-Linear Optimization

Although flooding optimization is both non-linear and dynamic, it is important to distinguish between those two different properties. The basic form of a dynamic optimization problem will be presented first, after which non-linear programming (NLP) will be explained.

2-1-1 Dynamic Optimization

Dynamic optimization aims to optimize an objective function with respect to equality and inequality constraints of which one or more equality constraints are DAEs. The general dynamic optimization problem is described by Eq. (2-1).

$$\begin{aligned} \min_{u(t)} \quad & J(u(t)) \\ \text{subject to} \quad & f(x(t), z(t), u(t), p) = \frac{\delta x(t)}{\delta t}\ ,\ x(t_0) = x_0, \\ & g(x(t), z(t), u(t), p) = 0, \\ & h(x(t), z(t), u(t), p) \geq 0, \end{aligned} \tag{2-1}$$

where $J(u(t))$ is the objective function and $f(x(t), z(t), u(t), p)$ are the dynamic equality constraints. The static equality and static inequality constraints are represented by function $g(x(t), z(t), u(t), p)$ and function $h(x(t), z(t), u(t), p)$ respectively. The variables $x(t), z(t), u(t)$ are defined as the differential, algebraic and control variables. The differential variables represent the states of the system, whereas the algebraic variables represent the time-varying properties of the system. The parameter-vector p contains the time-independent parameters. The time is indicated by t with t_0 the initial point in time.

The sequential method and the simultaneous method can be used to solve the dynamic optimization problem. They transcribe the dynamic problem into an NLP problem, which will be explained next.

2-1-2 Non-Linear Programming

Non-linear programming is the process of optimizing an objective function $J(q)$ using a finite set of decision variables q with respect to equality constraints $g(q)$ and inequality constraints $h(q)$ of which one or more constraints and/or the objective function are non-linear. Decision variables are the variables which may be modified by the NLP solver. The problem can be described as in Eq. (2-2).

$$\begin{aligned} \min_{q} \quad & J(q) \\ \text{subject to} \quad & g(q) = 0, \\ & h(q) \geq 0. \end{aligned} \tag{2-2}$$

The problem of Eq. (2-2) is then solved as illustrated by Figure 2-1. An NLP solver will calculate an optimal q denoted as q^* by using the gradient information of the objective with respect to the decision variables. Such a solver is an algorithm tailored to solving NLP problems, which is explained in detail in [Nocedal and Wright, 1999].

Figure 2-1: Schemetic representation of an NLP solver.

Both the sequential method as well as the simultaneous method will define an NLP from a dynamic optimization problem. The methods differ in their approach, explained in the following sections.

2-2 Sequential Method

The sequential method uses a set of decision variables based on the control inputs which are discretized with respect to time as illustrated in Figure 2-2. A forward simulation is performed by executing a numerical integration of the DAEs. The sensitivity may be computed as well during this integration step. The sensitivity is the derivative of the objective function with

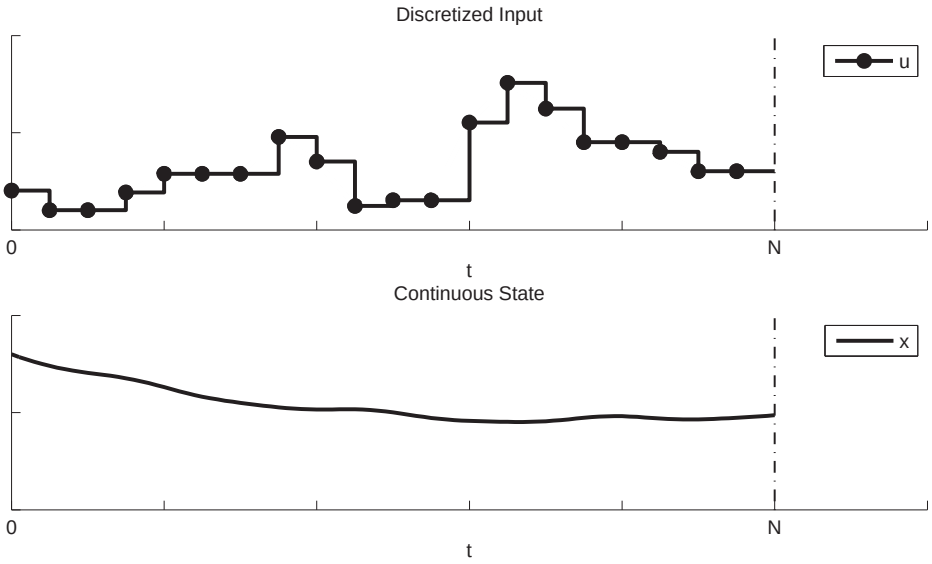


Figure 2-2: Illustration of the discretized inputs used by the sequential method. The discretized inputs are modified by the NLP solver to optimize an objective function. After optimization, an integration of the dynamic states x is performed, resulting in the state-profile. This procedure is repeated sequentially.

respect to the decision variables. Both the objective as well as the sensitivity is used by an NLP solver to compute an optimal set q^*. The forward simulation and the NLP optimization are executed sequentially.

2-2-1 Theory

Referring to Eq. (2-1), the control input $u(t)$ is discretized with respect to time into N time steps as a piecewise constant function:

$$u_n(t) = q_n, \quad t \in [t_n, t_{n+1}], \quad n = \{0, 1, ..., N-1\}, \tag{2-3}$$

where the last control input q_N does not affect the optimization and is therefore left out. The time-independent control inputs q_n, presented by Eq. (2-4), form a set of decision variables q with length $N_u \cdot N_t$ presented by Eq. (2-4). N_u is equal to number of inputs. N_t refers to the number of time-steps in which the input is parametrized.

$$q_n = \{q_0, q_1, ..., q_{N-1}\}. \tag{2-4}$$

The DAEs can be integrated numerically using q^*, resulting in a feasible solution to the dynamic states. The states and sensitivity computed by this integration are used to calculate a new q^*. The sequential process of integration and optimization is repeated until a predefined convergence criterion is met.

Three methods are commonly used to compute the sensitivity [Støren and Hertzberg, 1999]. One is numerical perturbation, which is based on performing a forward simulation for every

perturbed decision variable to obtain the gradient information. This method is computationally expensive. Two other methods use numerical integration, which are the sensitivity method and the adjoint method, explained next.

Gradient Calculation by using Numerical Integration

The gradient of the objective function is defined by the following relation:

$$\left(\frac{dJ}{dq}\right) = \left(\frac{\partial J}{\partial q}\right) + \left(\frac{\partial J}{\partial x}\right)\left(\frac{\partial x}{\partial q}\right), \tag{2-5}$$

where J is the objective function, x are the states and q are the decision variables. Eq. (2-5) is also known as the sensitivity equation. Considering the objective a function of the decision variables will result in a cheap computation for the derivative of $\partial J/\partial q$. The derivatives $\partial J/\partial x$ and $\partial x/\partial q$ are more complex and require a forward integration of the system [Petzold et al., 2006]. The latter calculation can be avoided using the adjoint equation [Kraaijevanger et al., 2007].

The adjoint method uses the sensitivity equation in a more efficient way. The right term of Eq. (2-5) is replaced:

$$\left(\frac{dJ}{dq}\right) = \left(\frac{\partial J}{\partial q}\right) + \lambda\left(\frac{\partial f}{\partial q}\right), \tag{2-6}$$

where f are the dynamic model equations. $\partial f/\partial q$ can be calculated symbolically. The adjoint λ is defined as

$$\lambda \triangleq \left(\frac{\partial J}{\partial x}\right)\left(\frac{\partial f}{\partial x}\right)^{-1}. \tag{2-7}$$

The adjoint has the advantage it does not depend on the number of inputs. λ can be computed backwards, by starting at λ_N. This is further explained in Appendix A. The adjoint equation is currently used by the Modular Reservoir Simulator (MoReS). MoReS is the in-house simulator of Shell, a Dutch oil company.

2-2-2 Advantages and Disadvantages

The advantages (+) and disadvantages (-) of the sequential method are based on the papers of [Binder et al., 2001], [Diehl et al., 2006] and [Biegler and Grossmann, 2004].

+ The relative small-sized NLP problem enables the use of off-the-shelf NLP solvers, which may efficiently limit the numerical effort.
+ Only IGs concerning the initial state-values are needed as the state-values are computed by integration of the DAEs.
+ State-of-the-art DAE solvers can be used enabling to profit from the latest research developments.

- The constraints on the state-profiles and the end-points may be violated. Both are handled by approximation within the limits of the control parametrization.

- Multiple integration of the DAEs can be computationally expensive for large-scale models.

- The sequential method cannot handle unstable systems. Knowledge of the state-profile on initialization cannot be used by the NLP solver. The solution to the DAE may depend nonlinearly on q which makes unstable systems difficult to resolve.

2-3 Simultaneous Method

The simultaneous method uses a set of decision variables based on the control inputs, states and algebraic variables which are discretized with respect to time. This is illustrated for the control inputs and states in Figure 2-3. The dynamic states are related over time by using function approximations of the dynamic states, also known as collocation on finite elements [Biegler, 2007]. After parametrization, the resulting NLP is solved without the need for integrating the DAEs as their function approximations are implemented as equality constraints.

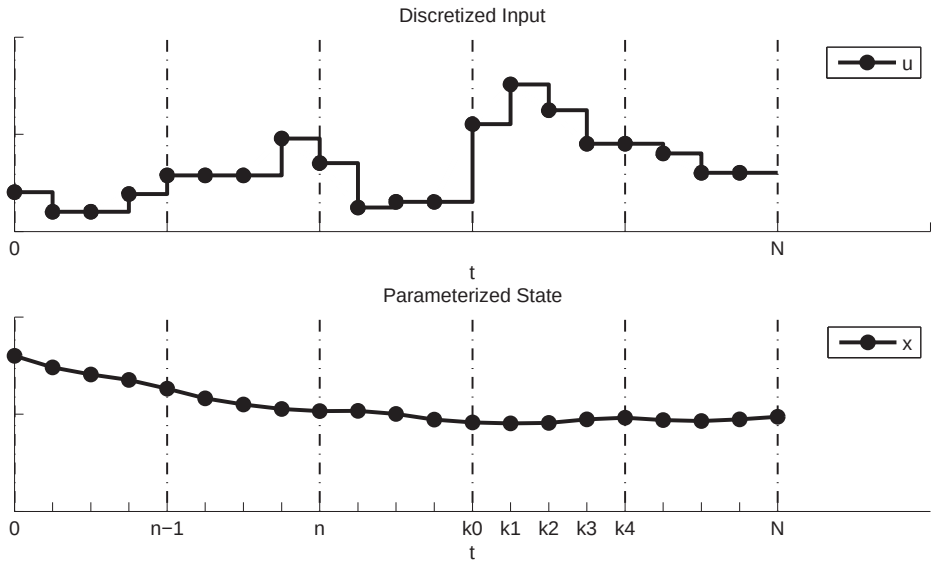


Figure 2-3: Illustration of the in N time-elements and K collocation points discretized inputs and parameterized states used by the simultaneous method. The discretized inputs and parameterized states are modified by the NLP solver to optimize an objective function. After optimization, the state-profiles are directly solved as they are incorporated in the NLP problem.

2-3-1 Theory

The control inputs, variables and dynamic states from Eq. (2-1) are discretized into N time-steps resulting in the finite set of decision variables q:

$$q = \{u_0, u_1, ..., u_{N-1}, z_0, z_1, ..., z_N, x_0, x_1, ..., x_N, p\}. \tag{2-8}$$

The non-dynamic control inputs u_n, variables z_n and parameters p do not require to be related by a function and may therefore be parametrized as piecewise constant. However, the dynamic states have to be approximated by a function at the discrete time-intervals, known as collocation. The most basic form will be presented first.

The dynamic equations of the optimal control problem must be written as first order DAE:

$$\frac{\delta x(t)}{\delta t} = f(x(t), z(t), u(t), p). \tag{2-9}$$

The states will be related by using an implicit Euler integration, which is a well-known method because of its good stability for stiff problems and higher order accuracy according to [Kameswaran and Biegler, 2006]:

$$\begin{aligned} \Delta t_n &= t_{n+1} - t_n, \\ x_{n+1} &= x_n + \Delta t_n f(x_{n+1}, z_{n+1}, u_{n+1}, p), \end{aligned} \tag{2-10}$$

where t is time and n the time-step. The dynamic equations are now replaced by algebraic equations. With the parametrization of $x(t)$ and the discretization of $z(t), u(t)$, the dynamic optimization problem can be written as an NLP as presented in Section 2-1-2, where all constraints are functions of the set of decision variables q.

The function approximation which has been used is still based on the integration of the dynamic equations. More advanced collocation methods are known as orthogonal collocation. With orthogonal collocation, the states are approximated using orthogonal polynomials. This is explained in the following subsection.

Orthogonal Collocation

Orthogonal collocation (also referred to as the pseudospectral method) applies function approximations using collocation points which are chosen to be the roots of orthogonal polynomials [Huntington and Rao, 2008]. The method is explained by using a Lagrange basis representation. The dynamic problem is again fully discretized, but instead of only discretizing the control inputs, states and variables in N time-elements, every time-element is also divided in K collocation points as illustrated in Figure 2-3. This representation applies a piecewise function approximation for every variable or state on every element n through the collocation points k:

$$\begin{aligned} x_n(t) &= \sum_{k=0}^{K} x_{n,k}\phi_{n,k}(t), \\ z_n(t) &= \sum_{k=1}^{K} z_{n,k}\phi_{n,k}(t), \\ u_n(t) &= \sum_{k=1}^{K} u_{n,k}\phi_{n,k}(t), \end{aligned} \tag{2-11}$$

where $z_n(t), u_n(t)$ are a K-th order polynomials and x_n is a $(K+1)$-th order polynomial due to the existence of an initial condition at $k = 0$. This initial condition is the end-point of the previous time-interval and therefore essential to enforce continuity of the dynamic states. The variable $\phi_{n,k}(t)$ is the Lagrange interpolation polynomial

$$\phi_{n,k}(t) = \prod_{p=0,p\neq k}^{K} \frac{t - t_p}{t_k - t_p} = \frac{(t - t_0)}{(t_k - t_0)} \cdots \frac{(t - t_{k-1})}{(t_k - t_{k-1})} \frac{(t - t_{k+1})}{(t_k - t_{k+1})} \cdots \frac{(t - t_K)}{(t_k - t_K)}, \tag{2-12}$$

where the subscript p is used as index for multiplication of all collocation points. The variables and inputs are now approximated by piecewise continuous polynomials and the states are approximated by continuous polynomials. The dynamic states x are not related yet to the dynamic equations, for which a residual equation R_n is introduced. The residual should vanish and is therefore set to zero [Cuthrell and Biegler, 1987].

$$R_n = \sum_{k=0}^{K} x_{n,k} \dot{\phi}_{n,k}(t_n) - f(x_n, z_n, u_n, p) = 0. \tag{2-13}$$

The continuous dynamic function $f(x_n, z_n, u_n, p)$ can be discretized using the implicit Euler relation of Eq. (2-10):

$$f(x_{n+1}, z_{n+1}, u_{n+1}, p) = \frac{x_{n+1} - x_n}{t_{n+1} - t_n}. \tag{2-14}$$

Note the difference of the earlier explained basic collocation method compared with orthogonal collocation. The time-discretization is performed using orthogonal polynomials instead of directly starting with applying the time-discretization.

The residual equation constrains the derivative of the function approximation of the states at the collocation points to the first-order dynamic function. The residual equation R_n is added as an equality constraint to the final NLP problem [Kameswaran and Biegler, 2006], where it replaces the system equations. This results in the following NLP:

$$\begin{aligned} \min_{q} \quad & J(q) \\ \text{subject to} \quad & g(q) \quad = 0, \\ & R_n(q) \quad = 0 \quad \text{where} \quad n = 0, 1, ..., N, \\ & h(q) \quad \geq 0. \end{aligned} \tag{2-15}$$

Instead of performing a DAE integration, the NLP is now tailored to perform a function approximation [Wright, 1970]. The advantage of orthogonal collocation is the fact that a coarser time grid may be selected as the collocation points force the 'in-between' solutions to fit the DAE. To put it even stronger, according to [Huntington and Rao, 2008], the method works the best in terms of precision and robustness when using a global approach where $N = 1$. Global refers to using only a single time-element. This may reduce the size of the NLP drastically.

In general, the number of decision variables for the simultaneous method adds up to $(N_u + N_x + N_z)N_t + N_p$, with N_u, N_x, N_z, N_p the number of inputs, states, algebraic variables and parameters respectively. N_t is equal to the number of time-steps or elements.

2-3-2 Advantages and Disadvantages

The advantages (+) and disadvantages (-) of the simultaneous method are derived from the papers of [Binder et al., 2001], [Diehl et al., 2006], [Biegler and Grossmann, 2004] and [Kameswaran and Biegler, 2006].

+ All constraints are satisfied after successful termination due to complete discretization.
+ Replacing the DAEs by algebraic equations eliminates the need for sequential integration.
+ The system equations are treated as nonlinear constraints which can be violated, but in the end they have to be satisfied. Intermediate solutions which may not exist can therefore be avoided.
+ The simultaneous method perfectly handles unstable systems, because the states can be bounded.

- The relatively large-sized NLP problem requires advanced and tailored NLP solvers.
- initial guess (IG)s are required for all decision variables.
- The model equations are only fulfilled after successful termination of the NLP solver.

2-4 Summary and Concluding Remarks

Both the sequential method and the simultaneous method are explained and the advantages and disadvantages are mentioned. The sequential method has the property of having a small NLP size, but comes at the cost of having difficulties dealing with state-constraints and performing multiple integrations of the DAEs. On the contrary, the simultaneous method can deal with all constraints, but with the cost of having a large NLP size. This is summarized in Table 2-1.

Table 2-1: Summary of the properties of the sequential method and the simultaneous method.

Property	Sequential Method	Simultaneous Method
Relative NLP size	Small	Large
Number of decision variables	$N_u N_t + N_p$	$(N_u + N_x + N_z)N_t + N_p$
Sparse NLP	No	Yes
IGs of the states	Initial values at t_0	Initial values at t_n with $n = \{0, 1, ..., N\}$
Deals with state constraints	Difficult	Yes
DAE Integration	After every NLP solution	Once

Chapter 3

Application of the Simultaneous Method to a Flooding Optimization Problem

This chapter will demonstrate how to apply the simultaneous method to a flooding optimization problem. A two-phase reservoir model is used. The choice for this model will be motivated in Section 3-1, after which the model is presented as a set of PDEs and transformed to continuous ODEs in Section 3-2. This set of continuous ODEs is then discretized in space and in time in Section 3-3 to obtain a set of difference equations. The objective function and constraints of the optimization are described in Section 3-4. Finally Section 3-5 will be used for final remarks on the implementation.

3-1 Motivation for using a Two-Phase Two-Dimensional Model

A reservoir model based on a real-life reservoir has three dimensions and consists of an oil, a water and a gas phase. Oil can be volatile resulting in mass-transfer between these different phases. Every phase is described by its own PDE. In this thesis, a two-phase, two-dimensional model will be used as a compromise between complexity and usability for academic purpose. There are three motivations for using such a model:

- A two-phase model is less complicated compared to a three-phase model. The absence of a compressible gas-phase leaves out an additional mass-balance and allows to neglect the mass-transfer between phases. The non-linear properties of the model as described in Appendix C will not change.
- A two-dimensional model is more straightforward compared to a three-dimensional model. A three-dimensional model is more complex as gravity-effects influence the dynamics. With a two-dimensional model, gravity-effects can be neglected.

- An open-source forward simulator called Simple Simulator (SimSim) is available. SimSim provides two-phase two-dimensional models which can be used for verification. A second advantage is that SimSim can be used as initial guess (IG) generator to obtain values for all in time and in space discretized variables.

Although the two-phase two-dimensional model is simplified due to omitting of the gas-phase and gravitational forces, it is assumed that the fundamental properties of the optimization problem do not change. With the fundamental properties is referred to the non-linear (Appendix C), large-scale, equality and inequality constrained, non-convex and deterministic properties of the optimization problem.

3-2 Transformation of the Model PDEs into ODEs

This section presents the model equations as PDEs and derives two ODEs in continuous time as basis for the discretization step.

3-2-1 PDE Representation

The model used is based on the PDEs as described in [Aziz and Settari, 1979] of which a derivation can be found in Appendix B. The model equations, based on the conservation of both mass and the conservation of momentum through substitution of Darcy's Law (Section B-1-2), are given by Eq. (3-1) and Eq. (3-2) for water and oil denoted by subscripts w and o.

$$\nabla \cdot \left(\alpha \rho_w K \frac{k_{rw}}{\mu_w} \left(\nabla p_w - \rho_w g \nabla d \right) \right) + \rho_w q_w = \alpha \frac{\partial (\phi \rho_w) S_w}{\partial t}, \tag{3-1}$$

$$\nabla \cdot \left(\alpha \rho_o K \frac{k_{ro}}{\mu_o} \left(\nabla p_o - \rho_o g \nabla d \right) \right) + \rho_o q_o = \alpha \frac{\partial (\phi \rho_o) S_o}{\partial t}, \tag{3-2}$$

where ∇ is the difference operator, α the geometry factor in m^3, ρ the fluid density in kg/m^3, K the permeability matrix in m^2, k_r the dimensionless relative permeability, μ the fluid viscosity in $Pa \cdot s$, p the pressure in Pa, g the gravitational force in m^2/s, d the depth within the reservoir in m, q the volumetric flow rate in m^3/s, ϕ the dimensionless porosity of the rock, S the dimensionless fluid saturation and t the time in s. The difference operator ∇ is used to describe the PDEs into terms of Cartesian coordinates:

$$\nabla K = \frac{\partial (k_x)}{\partial x} + \frac{\partial (k_y)}{\partial y}. \tag{3-3}$$

The geometry factor α and the permeability matrix K are defined below as

$$\begin{array}{lll} \alpha(x) & = A \Delta x & (1D) \\ \alpha(x,y) & = H \Delta x \Delta y & (2D) \\ \alpha(x,y,z) & = \Delta x \Delta y \Delta z & (3D), \end{array} \tag{3-4}$$

where A is the area, H the height and

$$K(\vec{x}) = \begin{bmatrix} k_{xx} & k_{xy} & k_{xz} \\ k_{yx} & k_{yy} & k_{yz} \\ k_{zx} & k_{zy} & k_{zz} \end{bmatrix}. \tag{3-5}$$

The vector $\vec{x}$ represents the set with Cartesian coordinates $\{x, y, z\}$.

3-2-2 Simplifications

The PDEs will be further simplified to comply with the reservoir models used in SimSim.

Assumptions

Simplifications are based on the following assumptions:

- The influence of gravity on a two-dimensional horizontal model can be neglected, leading to $\rho g \nabla d = 0$ [Jansen, 2009].
- It is assumed water and oil are completely miscible, meaning oil and water form a single solution. Capillary pressures are pressures between immiscible fluids and can due to this assumption be neglected. Therefore, it is assumed $p_o = p_w = p$ [Jansen, 2009].
- As theoretical models are considered, it can be assumed the coordinates of the grid are aligned with the layering of the rock. This simplifies K to a diagonal 2×2 matrix, with values k_{xx} and k_{yy} [Peaceman, 1977].

Algebraic Simplifications

Theoretical simplifications are implemented as well:

- The oil saturation is expressed in terms of the water saturation, $S_o = 1 - S_w$. The water saturation S_w is written as s for simplicity.
- The iso-thermal compressibilities $c_l(p)$ and $c_r(p)$ in $1/Pa$ of respectively liquid (either water or oil) and rock represent the relation between density ρ and pressure p (Eq. (3-6)).

$$\begin{aligned} c_l(p) &\triangleq \frac{1}{\rho_l} \frac{\partial \rho_l}{\partial p}\bigg|_{T_R}, \\ c_r(p) &\triangleq \frac{1}{\phi} \frac{\partial \phi}{\partial p}\bigg|_{T_R}, \end{aligned} \tag{3-6}$$

where T_R is the reservoir or reference temperature.

3-2-3 Simplified PDE Representation

Applying the simplifications results in the following model:

$$\nabla \cdot \left(\alpha \rho_w K \frac{k_{rw}}{\mu_w} (\nabla p) \right) + \rho_w q_w = \alpha \phi \rho_w \left(s(c_w + c_r) \frac{\partial p}{\partial t} + \frac{\partial s}{\partial t} \right), \tag{3-7}$$

and

$$\nabla \cdot \left(\alpha \rho_o K \frac{k_{ro}}{\mu_o} (\nabla p) \right) + \rho_o q_o = \alpha \phi \rho_o \left((1-s)(c_o + c_r) \frac{\partial p}{\partial t} - \frac{\partial s}{\partial t} \right). \tag{3-8}$$

3-2-4 ODE Representation

The first-order PDEs of Eq. (3-7) and Eq. (3-8) are written as explicit first-order ODE representation by solving the PDEs for both the terms $\partial p/\partial t$ and $\partial s/\partial t$. Below are the two resulting ODEs.

$$\frac{\partial p}{\partial t} = \frac{(\nabla \cdot \left(\alpha K \frac{k_{rw}}{\mu_w} (\nabla p) \right) + q_w) + (\nabla \cdot \left(\alpha K \frac{k_{ro}}{\mu_o} (\nabla p) \right) + q_o)}{(\alpha \phi)(s(c_w - c_o) + c_r + c_o)}, \tag{3-9}$$

and

$$\begin{aligned} \frac{\partial s}{\partial t} = & \frac{(\nabla \cdot \left(\alpha K \frac{k_{rw}}{\mu_w} (\nabla p) \right) + q_w)((1-s)(c_o + c_r))}{((\alpha \phi)(s(c_w - c_o) + c_r + c_o)} \\ - & \frac{(\nabla \cdot \left(\alpha K \frac{k_{ro}}{\mu_o} (\nabla p) \right) + q_o)(s(c_w + c_r))}{(\alpha \phi)(s(c_w - c_o) + c_r + c_o)}. \end{aligned} \tag{3-10}$$

3-3 Discretization of the ODEs

The continuous ODEs are discretized in space and in time in the following subsections enabling numerical integration. The methods are selected such that they comply with the methods used in SimSim.

3-3-1 Spatial Discretization

Spatial discretization is performed using finite differences, a common used discretization method [Jansen, 2009], explained next.

Finite Difference Discretization

The discretization is explained using the mass-balance for the water-phase equations, but the method applies to the oil-phase equations as well. Firstly, the part of the continuous ODEs containing the difference operator ∇ is elaborated. As earlier stated, K is a 2×2 diagonal matrix with the values k_{xx}, k_{yy}.

$$\nabla \cdot \left(K \frac{k_{rw}}{\mu_w} (\nabla p) \right) = \frac{\partial}{\partial x} \left(k_{xx} \frac{k_{rw}}{\mu_w} \left(\frac{\partial p}{\partial x} \right) \right) + \frac{\partial}{\partial y} \left(k_{yy} \frac{k_{rw}}{\mu_w} \left(\frac{\partial p}{\partial y} \right) \right). \tag{3-11}$$

Secondly, the continuous space is approximated in discrete space as:

$$\begin{aligned} &\frac{\partial}{\partial x} \left(k_{xx} \frac{k_{rw}}{\mu_w} \left(\frac{\partial p}{\partial x} \right) \right) + \frac{\partial}{\partial y} \left(k_{yy} \frac{k_{rw}}{\mu_w} \left(\frac{\partial p}{\partial y} \right) \right) \approx \\ &\frac{\Delta}{\Delta x} \left(k_{xx} \frac{k_{rw}}{\mu_w} \left(\frac{\Delta p}{\Delta x} \right) \right) + \frac{\Delta}{\Delta y} \left(k_{yy} \frac{k_{rw}}{\mu_w} \left(\frac{\Delta p}{\Delta y} \right) \right). \end{aligned} \tag{3-12}$$

Finally, the discretization is done using equally spaced grid-blocks in both the x-direction and the y-direction, using the central difference approximation. The central difference approximation is a method to numerically calculate a discrete derivative:

$$\frac{\Delta f(x)}{\Delta x} = \frac{\left(f(x + \frac{1}{2}\Delta x) - f(x) \right) - \left(f(x) - f(x - \frac{1}{2}\Delta x) \right)}{\Delta x}, \tag{3-13}$$

where $f(x)$ and x are an arbitrary function and an arbitrary variable used for demonstration. Applying the central difference approximation to Eq. (3-12) yields:

$$\begin{aligned} &\frac{\left(\left[k_{xx} \frac{k_{rw}}{\mu_w} \right]_{i+\frac{1}{2},j,n} ([p]_{i+1,j,n} - [p]_{i,j,n}) - \left[k_{xx} \frac{k_{rw}}{\mu_w} \right]_{i-\frac{1}{2},j,n} ([p]_{i,j,n} - [p]_{i-1,j,n}) \right)}{(\Delta x)^2} + \\ &\frac{\left(\left[k_{yy} \frac{k_{rw}}{\mu_w} \right]_{i,j+\frac{1}{2},n} ([p]_{i,j+1,n} - [p]_{i,j,n}) - \left[k_{yy} \frac{k_{rw}}{\mu_w} \right]_{i,j-\frac{1}{2},n} ([p]_{i,j,n} - [p]_{i,j-1,n}) \right)}{(\Delta y)^2}. \end{aligned} \tag{3-14}$$

The subscripts i, j, n refer to the discrete space (i, j) (Figure 3-1) at time-step n.

Implementation of the Discrete Permeabilities

The central difference approximation requires to calculate the permeabilities at the border of two grid blocks. The calculation for both the rock and the relative permeabilities will be explained.

Rock permeabilities are approximated using the harmonic average:

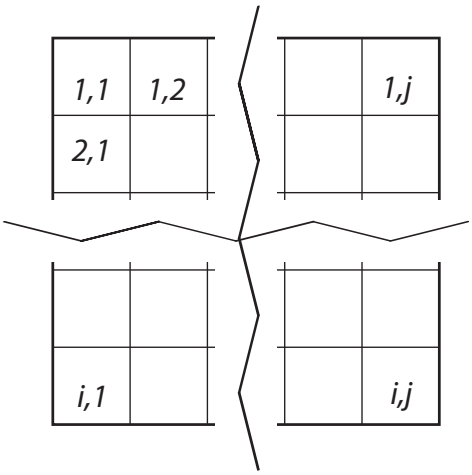


Figure 3-1: Representation of a two dimensional reservoir model with i rows and j columns.

$$\begin{aligned}[k_{xx}]_{i+\frac{1}{2},j} &= 2\frac{[k_{xx}]_{i+1,j}[k_{xx}]_{i,j}}{[k_{xx}]_{i+1,j}+[k_{xx}]_{i,j}},\\ [k_{yy}]_{i,j+\frac{1}{2}} &= 2\frac{[k_{yy}]_{i,j+1}[k_{yy}]_{i,j}}{[k_{yy}]_{i,j}+[k_{yy}]_{i,j+1}}.\end{aligned} \tag{3-15}$$

Relative permeabilities cannot be computed using the harmonic average. The relative permeabilities depend on the saturation which is described by a non-linear hyperbolic PDE, which corresponds to a moving water front through the reservoir (Section C-2-2). The saturation for a certain grid block may change very rapidly in time as the water front moves along this grid block. Using a harmonic average as in Eq. (3-15) for the relative permeabilities is for this reason not correct and will give an incorrect result [Aziz and Settari, 1979, p.153]. The relative permeabilities in a certain grid block will be based on the saturation of the flow into the grid block. This can be implemented by using upstream weighting:

$$[k_{rw}]_{i+\frac{1}{2},j,n} = \begin{cases} [k_{rw}]_{i,j,n} & \text{if } [p]_{i,j,n} \geq [p]_{i+1,j,n} \\ [k_{rw}]_{i+1,j,n} & \text{if } [p]_{i,j,n} < [p]_{i+1,j,n}. \end{cases} \tag{3-16}$$

The relative permeabilities are based on a basic Corey model [Corey, 1954], which is presented in Eq. (3-17) and describes the relative permeability as exponential function of the normalized water saturation $[S_{wn}]_{i,j,n}$.

$$\begin{aligned}[k_{rw}]_{i,j,n} &= k^0_{rw}[S_n]^{n_w}_{i,j,n},\\ [k_{ro}]_{i,j,n} &= k^0_{ro}(1-[S_n]_{i,j,n})^{n_o},\end{aligned} \tag{3-17}$$

where k^0_{rw}, k^0_{ro} are the endpoint permeabilities and n_w, n_o are called the Corey exponents. The normalized water saturation S_{wn} is defined as function of the water saturation s.

$$[S_{wn}]_{i,j,n} \triangleq \frac{[s]_{i,j,n}-S_{wc}}{1-S_{or}-S_{wc}}, \ \textit{with } 0 \geq S_n \geq 1. \tag{3-18}$$

With $S_{wn} \in [0,1]$, this implies $s \in [S_{wc}, 1 - S_{or}]$. S_{wc} is the connate water saturation and S_{or} is the residual oil saturation. The connate water saturation and residual oil saturation determine the minimum and maximum value of the actual water saturation s.

Eq. (3-16) causes the model to be discontinuous. Discontinuous models cannot be optimized using the simultaneous method. The relative permeabilities will be fixed to maintain continuous model equations. The fixation will be based on the pressure-values of the IG to comply with the models as obtained from SimSim.

Combination of the Discrete Equations

The transition from the continuous ODEs into difference equations will be described. The discretization steps are explained using Eq. (3-12) ,Eq. (3-13) and Eq. (3-14). The permeabilities used in the ODEs are described in discrete space by Eq. (3-15), Eq. (3-16), Eq. (3-17) and Eq. (3-18).

Using these equations, the term with the difference operators will be recovered in two stages, using the transmissibility term T and the volumetric flow rate U.

$$\nabla \cdot \left(\alpha K \frac{k_{rw}}{\mu_w} \right) (\nabla p) + q_w = \underbrace{\overbrace{\nabla \cdot \left(\alpha K \frac{k_{rw}}{\mu_w} \right) \left(\frac{1}{\Delta x} + \frac{1}{\Delta y} \right)}^{T_w} \Delta p + q_w}_{U_w} \tag{3-19}$$

First the transmissibility parameters will be explained by defining $[T_{w,1}]_{i,j,n}$, $[T_{w,2}]_{i,j,n}$, $[T_{w,3}]_{i,j,n}$, $[T_{w,4}]_{i,j,n}$. The parameter α is replaced according to Eq. (3-4) with $h\Delta x \Delta y$.

$$\begin{aligned}
[T_{w,1}]_{i,j,n} = [T_w]_{i+\frac{1}{2},j} &= \frac{2h}{\mu_w} \frac{\Delta y}{\Delta x} \frac{[k_{xx}]_{i+1,j}[k_{xx}]_{i,j}}{[k_{xx}]_{i+1,j} + [k_{xx}]_{i,j}} [k_{rw}]_{i+1/2,j,n} \\
[T_{w,2}]_{i,j,n} = [T_w]_{i-\frac{1}{2},j} &= \frac{2h}{\mu_w} \frac{\Delta y}{\Delta x} \frac{[k_{xx}]_{i-1,j}[k_{xx}]_{i,j}}{[k_{xx}]_{i-1,j} + [k_{xx}]_{i,j}} [k_{rw}]_{i-1/2,j,n} \\
[T_{w,3}]_{i,j,n} = [T_w]_{i,j+\frac{1}{2}} &= \frac{2h}{\mu_w} \frac{\Delta x}{\Delta y} \frac{[k_{yy}]_{i,j+1}[k_{yy}]_{i,j}}{[k_{yy}]_{i,j+1} + [k_{yy}]_{i,j}} [k_{rw}]_{i,j+1/2,n} \\
[T_{w,4}]_{i,j,n} = [T_w]_{i,j-\frac{1}{2}} &= \frac{2h}{\mu_w} \frac{\Delta x}{\Delta y} \frac{[k_{yy}]_{i,j-1}[k_{yy}]_{i,j}}{[k_{yy}]_{i,j-1} + [k_{yy}]_{i,j}} [k_{rw}]_{i,j-1/2,n}
\end{aligned} \tag{3-20}$$

The transmissibility terms are multiplied by the pressure differences. This recovers the with the geometry factor multiplied Darcy velocities, which creates the volumetric flow rate U. If a well is enclosed in a grid block, the volumetric flow rate q will be added to the equation of the total volumetric flow rate U.

$$\begin{aligned}
[U_w]_{i,j,n} = \; & [T_{w,1}]_{i,j,n}([p]_{i+1,j,n} - [p]_{i,j,n}) + [T_{w,2}]_{i,j,n}([p]_{i-1,j,n} - [p]_{i,j,n}) \\
+ \; & [T_{w,3}]_{i,j,n}([p]_{i,j+1,n} - [p]_{i,j,n}) + [T_{w,4}]_{i,j,n}([p]_{i,j-1,n} - [p]_{i,j,n}) + [q_w]_{i,j,n}.
\end{aligned} \tag{3-21}$$

Note from Eq. (3-21) the fact that a flow into a grid block has a positive sign, where flow out of a grid block will have a negative sign. Production flow rates are therefore negative.

3-3-2 Difference Equation Representation

With above steps it is possible to write the set of continuous ODEs presented by Eq. (3-9) and Eq. (3-10) as set of difference equations:

$$\left[\frac{\Delta p}{\Delta t}\right]_{i,j,n} = \frac{([U_w]_{i,j,n}) + ([U_o]_{i,j,n})}{(h\Delta x\Delta y)[\phi]_{i,j}([s]_{i,j,n}(c_w - c_o) + c_r + c_o)} \tag{3-22}$$

$$\left[\frac{\Delta s}{\Delta t}\right]_{i,j,n} = \frac{([U_w]_{i,j,n})((1-[s]_{i,j,n})(c_o + c_r)) - ([U_o]_{i,j,n})([s]_{i,j,n}(c_w + c_r))}{(h\Delta x\Delta y)[\phi]_{i,j}([s]_{i,j,n}(c_w - c_o) + c_r + c_o)} \tag{3-23}$$

3-3-3 Time Discretization and Integration

The difference equations represented by Eq. (3-22) and Eq. (3-23) are used to obtain the state-profiles of p and s using the by Eq. (3-24) backward or implicit Euler integration method. This is explained in Section 2-3 and complies to the SimSim models.

$$\begin{aligned} [t]_n &= [t]_{n-1} + \Delta t, \\ [p]_{i,j,n} &= [p]_{i,j,n-1} + \Delta t \left[\frac{\Delta p}{\Delta t}\right]_{i,j,n}, \\ [s]_{i,j,n} &= [s]_{i,j,n-1} + \Delta t \left[\frac{\Delta s}{\Delta t}\right]_{i,j,n}. \end{aligned} \tag{3-24}$$

3-4 Optimization

This section will describe the objective function which needs to be maximized and the physical constraints and the boundary conditions to enforce a feasible solution.

3-4-1 Objective Function

Maximizing the yield of the reservoir is done by setting a profit for produced oil and a loss for injected and produced water, which is used to calculate the net present value (NPV). The objective function φ presented by Eq. (3-25) is based on the function described in [Jansen et al., 2009] and expressed in US dollars (USD).

$$\varphi([q_{w,inj}]_{i,j,n}, [q_{w,prod}]_{i,j,n}, [q_{o,prod}]_{i,j,n}) = \sum_{t=1}^{N_t} \left(\frac{\sum_{i=1}^{N_i}\sum_{j=1}^{N_j} \left([q_{w,inj}]_{i,j,n} \cdot r_{w,inj} + [q_{w,prod}]_{i,j,n} \cdot r_{w,prod} + [q_{o,prod}]_{i,j,n} \cdot r_{o,prod} \right)}{(1+b)^{t_t/\tau}} \Delta t_t \right), \tag{3-25}$$

where N_i, N_j, N_t are the number of rows, columns and time-steps, b is the discount rate, τ is the reference time in seconds, r is the cost or profit of the injected or produced volume in USD$/m^3$ and $q_{w,inj}, q_{w,prod}, q_{o,prod}$ are the flow rates of the injected water, produced water and produced oil in m^3/s. As the objective function now represents the profit, it is important to set $r_{w,inj}$ and $r_{o,prod}$ negative due to the signs of the flow rates.

3-4-2 Physical Constraints

Additional constraints describe the physics within and around the wells. These constraints apply on the grid blocks enclosing a well. The first constraint is a well model which describes the diffusive pressure distribution. The second constraint defines the fractional flow which is applied to every production well to relate the water and oil rate to the total production rate.

Well Model

The well model relates the bottem-hole pressure (BHP) $[p_{well}]_{i,j,n}$, the grid-block pressure (GBP) $[p]_{i,j,n}$ and the total flow rate $[q_t]_{i,j,n}$ for every well. BHPs are pressures at the opening of a well in the reservoir. The relation of a BHP to a grid block pressure is diffusive. The model is based on the assumption of isotropic permeability $[k]_{i,j}$ and rectangular grid blocks [Peaceman, 1983].

$$([p]_{i,j,n} - [p_{well}]_{i,j,n}) \frac{2[k]_{i,j}h\pi}{\ln\left(\frac{0.14\sqrt{\Delta x^2 + \Delta y^2}}{r_{well}}\right)} = -\frac{1}{\frac{[k_{rw}]_{i,j,n}}{\mu_w} + \frac{[k_{ro}]_{i,j,n}}{\mu_o}}[q_t]_{i,j,n}, \tag{3-26}$$

where r_{well} is the radius of the well in meter and

$$[q_t]_{i,j,n} = [q_w]_{i,j,n} + [q_o]_{i,j,n}. \tag{3-27}$$

In this thesis, $[q_t]_{i,j,n}$ will be either defined as the total injection rate q_{inj} ($q_{o,inj} = 0$) or as the total production rate q_{prod}.

Fractional Flow

The total production flow can now be calculated, but the fraction of water and oil are still unknown. This fraction is determined by the fractional flow rate of water $[f_w]_{i,j,n}$ presented in Eq. (3-28).

$$[f_w]_{i,j,n} = \frac{[k_{rw}]_{i,j,n}}{[k_{rw}]_{i,j,n} + [k_{ro}]_{i,j,n}\left(\frac{\mu_w}{\mu_o}\right)}. \tag{3-28}$$

The water flow rate is related to the total flow rate as:

$$[q_w]_{i,j,n} \quad = [f_w]_{i,j,n}[q_t]_{i,j,n}. \tag{3-29}$$

3-4-3 Boundaries

A minimum number of variables needs to be bounded in order to satisfy the physical constraints of the reservoir. This avoids the optimization algorithm to terminate in a non-physical optimum.

Saturations

The normalized water saturation S_{wn} is bounded to $S_{wn} \in [0, 1]$. Neglecting this bound would generate negative water saturations, leading to an infeasible and infinitely high oil production.

Well Pressures

The BHP of the well described by the pressure p_{well} is limited to avoid an infinite pressure difference between the GBP p and the BHP p_{well}, what would result in an infinite flow rate as well.

Flow Rates

To distinguish between an injection or a production well, the flow rates will be limited to be either positive or negative respectively to avoid production wells turning into injection wells and vice versa.

Flow at the Reservoir Boundaries

The reservoir is assumed to be a finite volume with no mass interaction with the surroundings. This closed reservoir is obtained by influencing the transmissibilities at the reservoir boundaries where they are forced to be zero.

3-5 Concluding Remarks

It has been demonstrated how the simultaneous method can be applied to a flooding optimization problem for a simplified reservoir model. Future work might include the use of three-dimensional models and include multiple phases. The steps of obtaining explicit first-order ODEs and applying a time-discretization would still hold for a three-phase model. An additional phase would result in an additional mass-balance and an additional state.

For compliance with SimSim as IG generator, the simulator which will be used for verification and generation of the IGs, two implementation options have been adopted which might not be the best choice in terms of optimization performance. The first is the problem of discontinuous behavior of the relative permeabilities. The directions are now based on the pressure states of the IG, which might result in a bad convergence behavior when the IG is selected less optimal. The second adoption is the use of the implicit Euler method for integration, where orthogonal collocation could be used instead. This would results in a function approximation of the state-profiles, eliminating the need for direct integration as discussed in Section 2-3-1.

Chapter 4

Implementation of a Flooding Optimization Problem in GAMS

The implementation of a flooding optimization problem using the simultaneous method is verified and tested to analyze its limitations. The limitations are explored by scaling up the model and by selecting different values for the initial guess (IG). A flooding optimization problem is implemented in General Algebraic Modeling System (GAMS). GAMS is a software package based on algebraic modeling and is designed for solving optimization problems. It has the advantages of providing automatic discretization support and symbolic differentiation. State-of-the-art solvers can be selected for solving an optimization problem without the need for adjusting the code. Compared to other software packages, solvers are better integrated in GAMS and GAMS can communicate with Matrix Laboratory (MatLab).

This chapter will first compare the GAMS implementation with a Simple Simulator (SimSim) model in order to verify the GAMS implementation in Section 4-1. Next, the optimization performance of four solvers will be described in Section 4-2, because they might give very different results [Wächter and Biegler, 2006], [Poku and Biegler, 2004]. As the simultaneous method is known to be sensitive to IGs and given the fact that the size of the NLP may grow rapidly [Biegler, 2007], both the feasibility of upscaling and the sensitivity to IGs are tested in Section 4-3 and Section 4-4 respectively. The chapter will be concluded in Section 4-5.

4-1 Model Verification

The GAMS model is verified using SimSim. SimSim is being developed by J.D. Jansen since 2004 and used and improved since then. SimSim is based on the simplified PDEs presented in Section 3-2 and is a matrix-oriented forward simulator using implicit Euler integration. It has also been compared to Modular Reservoir Simulator (MoReS), the in-house simulator of Shell. Therefore, it is assumed SimSim provides representative and valid reservoir models which can be used for model verification. The reservoir model used for verification is given in Figure 4-1. The GAMS code is presented in Appendix D.

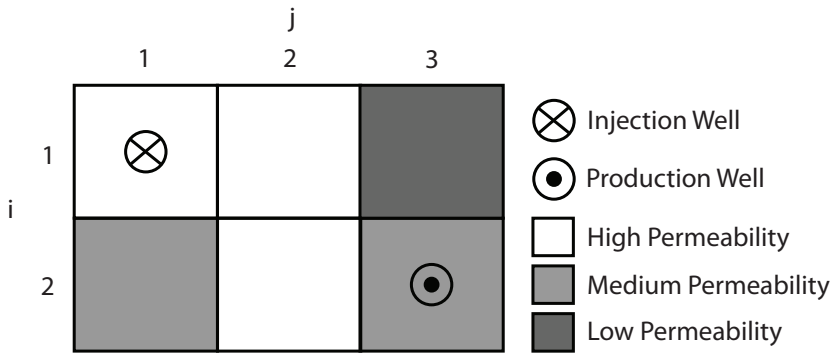


Figure 4-1: Illustration of the reservoir model used for verification. The model consists of high permeability fields ($1e{-}12\ m^2$), medium permeability fields ($1e{-}13\ m^2$) and a low permeability field ($1e{-}14\ m^2$). One injection well is placed in grid block ($(i,j) = (1,1)$) and one production well is placed in grid block ($(i,j) = (2,3)$).

The verification is performed in two directions. Firstly, GAMS starts optimizing using flow rates, pressures and saturations obtained from a forward simulation in SimSim. GAMS starts with an initial error, which is a summation of the numerical errors and model differences. Secondly, after termination of the GAMS optimization, the resulting flow rates are used to perform a forward simulation in SimSim. The SimSim simulation is then compared with the values from GAMS for verification.

4-1-1 The Initial Error in GAMS due to the Initial Guess

The cause of the initial error is explained first, after which the origin of the error is discussed.

Cause of the Initial Error in GAMS

Every equation in GAMS is programmed once using spatial and time indexes (e.g. $f_{i,j,n}$), after which GAMS performs an automatic discretization in space and in time. The optimization horizon is 10 years, with a maximum time-step of 30 days, resulting in 123 time-steps. So every equation with spatial indexes i, j and time index n results in $i \cdot j \cdot n = 2 \cdot 3 \cdot 123 = 738$ equations after automatic discretization. In total, the model consists of $13,394$ equations after automatic discretization. The equations are evaluated using the IG from SimSim. However, due to numerical or model errors the left-hand side may not be equal to the right-hand side of these equations. The errors are added up for all $13,394$ equations, resulting in the initial error.

Origin of the Initial Error in GAMS

GAMS initiates with an initial error in the order of $1e{-}1$ due to errors in the implicit Euler equations and the Peaceman well-model. The IG influences the initial error. For this reason, an IG has been selected which does not violate any constraints and which has only a small difference between the injection and production rates. The IG is for this reason based on a constant injection rate of $0.0059\ m^3/s$ and a constant production rate of $-0.0061\ m^3/s$.

It has been investigated which relative errors are the largest. The relative errors are calculated using the absolute error of the equation divided by the order of magnitude of the equation. The highest relative error is selected. This is summarized in Table 4-1.

Equation	Order of Relative Error
Peaceman Well Model	$1e{+}02\%$
Implicit Euler Saturation	$1e{-}03\%$
Implicit Euler Pressure	$1e{-}08\%$

Table 4-1: A presentation of the maximum relative initial errors in GAMS. The maximum relative errors are calculated using the maximum absolute error of the equation divided by the order of magnitude of the equation. The absolute errors are based on the summation of the errors due to differences in the left-hand side and right-hand side of the in GAMS discretized equations. The highest relative error is presented. A relative error in the order of $1e{+}02\%$ for the Peaceman equation implies that one side of the equation may be a multiple of the other side.

A relative error in the order of $1e{+}02\%$ implies that one side of the equation may be a multiple of the other side. Evaluation of the error for every single Peaceman equation reveals the error to be present from time-step $n > 95$. This appears to be the point where water breaks through. The Peaceman well model has been compared to SimSim and both models are identical. All three major errors are reproducible and therefore not due to the limit of computational precision. Increasing the time horizon does not increase the maximum relative error.

4-1-2 Comparison of GAMS with the Forward Simulator SimSim

The results of GAMS are compared to the results of SimSim. The results of SimSim are based on the flow rates of a GAMS optimization. Selecting flow rates as input will result in the BHPs as output. The BHPs have therefore the largest error. The time profiles of the well variables, pressures and saturations are presented in Figure 4-2 , Figure 4-3 and Figure 4-4.

Both models are using an equal time-step discretization of maximum 30 days, which is determined by SimSim. In total, 123 time-steps are used to simulate the forward model for a optimization horizon of 10 years. The fluctuations of BHP are limited to $[+50e5, 0]\ Pa$ and $[0, -100e5]\ Pa$. Although a fluctuation in BHP of $-50e5\ Pa$ may be more realistic, a bound of $-100e5\ Pa$ is selected to allow for an IG based on a production rate of $-0.0061\ m^3/s$ without violation of any boundaries. The boundaries in GAMS are not implemented in SimSim, though they are still satisfied.

Both models behave similar. The error in terms of percentage has been calculated for time-steps $n = 2, 3, ..., N$. The first time-step is left out because the first time-step is not correctly calculated by SimSim. For example, BHPs at time-step $n = 1$ are zero where they should be equal to the reservoir pressure. All the values for the error in terms of percentage are presented in Table 4-2. Although the models behave similar, small differences are present. It is clear well rates are used as simulation input. The highest error is not more than 0.5%. Using a time-horizon of 40 years does not change the maximum error in terms of percentage.

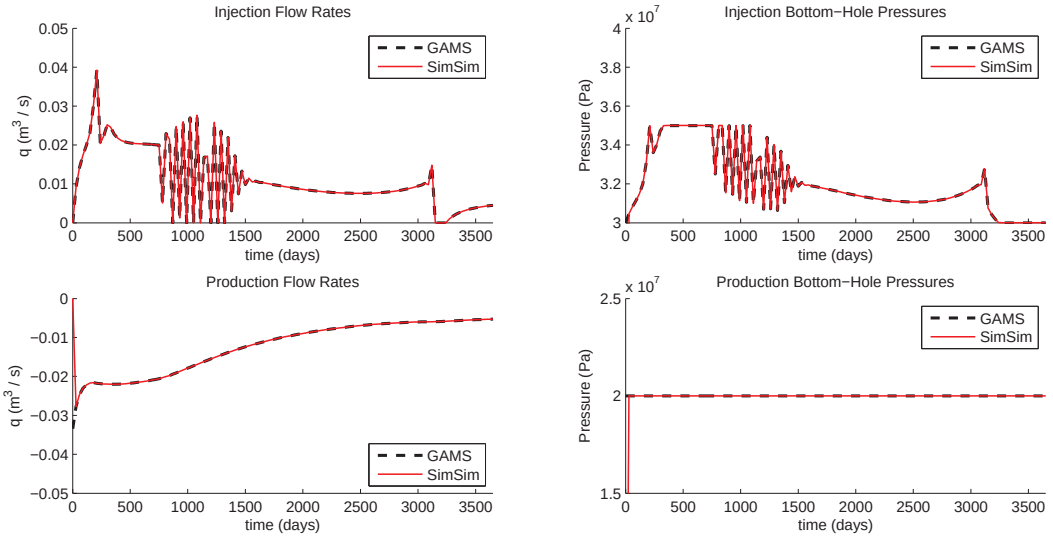


Figure 4-2: Comparison of the well variables of GAMS and SimSim. The flow rates are used as model input where the BHPs are the model output. The plots overlap.

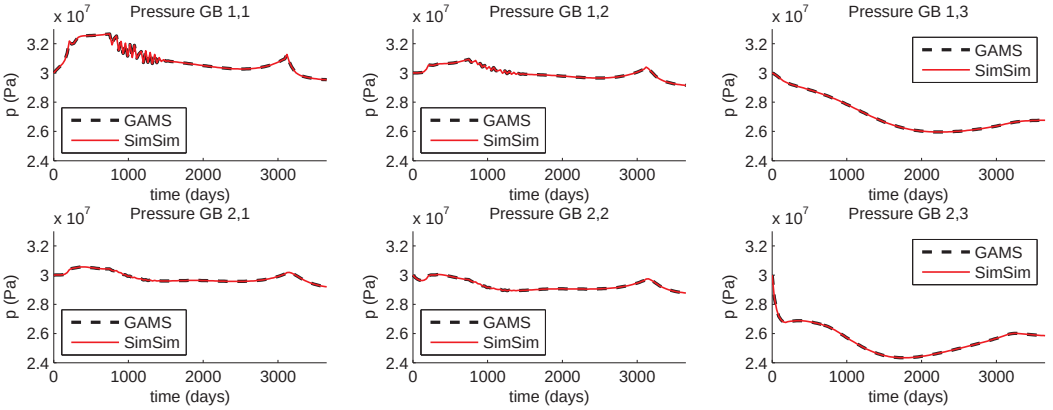


Figure 4-3: Comparison of the pressures of GAMS and SimSim. All six grid blocks of the 2×3 model are plotted. The plots are overlap.

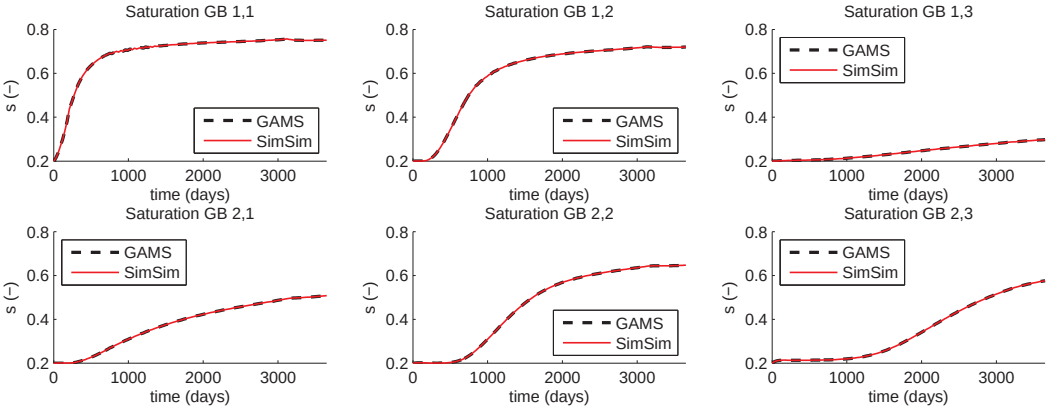


Figure 4-4: Comparison of the saturations of GAMS and SimSim. All six grid blocks of the 2×3 model are plotted. The plots are overlap.

Variable	Error (%)	Variable	Error (%)
$[p]_{1,1,n}$	$1.4e{-}02$	$[s]_{1,1,n}$	$2.1e{-}08$
$[p]_{1,2,n}$	$5.7e{-}03$	$[s]_{1,2,n}$	$4.2e{-}04$
$[p]_{1,3,n}$	$5.5e{-}04$	$[s]_{1,3,n}$	$1.8e{-}03$
$[p]_{2,1,n}$	$4.5e{-}07$	$[s]_{2,1,n}$	$5.0e{-}01$
$[p]_{2,2,n}$	$4.5e{-}07$	$[s]_{2,2,n}$	$6.9e{-}02$
$[p]_{2,3,n}$	$5.1e{-}08$	$[s]_{2,3,n}$	$1.4e{-}02$
$[q_w]_{1,1,n}$	$0.0e{+}00$	$[q_w]_{2,3,n}$	$6.2e{-}02$
$[q_o]_{1,1,n}$	$0.0e{+}00$	$[q_o]_{2,3,n}$	$2.0e{-}07$
$[p_{well}]_{1,1,n}$	$1.4e{-}02$	$[p_{well}]_{2,3,n}$	$7.0e{-}07$

Table 4-2: The maximum error in terms of percentage calculated for all states, inputs and outputs of GAMS and SimSim. The well rates of GAMS are used by SimSim to perform a forward simulation. The highest error is not more than 0.5%.

4-2 Selection of the Best Performing Solver

Solvers may perform very different depending on the optimization problem, as can be found in for example [Wächter and Biegler, 2006] and [Poku and Biegler, 2004]. It is therefore investigated which solvers perform the best when applied to a flooding optimization problem. Multiple solvers have been applied to flooding optimization problems, of which the four most promising solvers are selected. The selected solvers will be described first after which the results will be compared on the obtained NPV, optimization time and behavior.

4-2-1 Description of Different Solvers

Four selected solvers are described. They all exploit the sparsity of the NLP and approximate the second-order derivative information based on the algebraically calculated first-order gradient.

CONOPT 3.0

CONOPT is introduced by [Drud, 1985] and uses a generalized reduced gradient (GRG) algorithm. A GRG algorithm basically takes an iteration-step along its vertices, where the vertices are the boundaries of the search-space, for example equality constraints or active inequality constraints, better known as active sets. Unbounded variables, also called super-basic variables, are then used to minimize the optimization problem. According to the article of [Biegler and Grossmann, 2004], GRG methods are the most poplar among all gradient based NLP solvers.

CONOPT can handle up to 4000 equations [Drud, 1994]. More recent publications are not available, but according to [Mittelmann, 2010] the solver will still be able to handle $1e5$ decision variables.

IPOPT 3.8

IPOPT is initiated by [Wächter, 2002] and uses an interior point (IP) algorithm. An IP algorithm basically adds constraints to the objective function using barrier functions. A barrier function is zero when the constraint is satisfied, but will go to infinity along a smooth path when a constraint is violated. According to [Biegler and Grossmann, 2004], IPOPT can solve problems up to $1e6$ variables on a high-end personal computer.

KNITRO 6.0

KNITRO is initially introduced as an IP solver. In a later stage it has been extended with a second IP solver and with an sequential linear programming (SLP) algorithm using active sets. This SLP algorithm can be interpreted as a GRG algorithm on a linearized problem. This is used to improve the robustness in convergence, where the SLP algorithm is also used effectively to determine active constraints [Byrd et al., 2006]. KNITRO shows to be very effective with many active constraints [Gould et al., 2005].

From [Mittelmann, 2010], and according to [F. Cadoux] of the company Artelys and the KNITRO website[1], it shows that KNITRO should be capable of problems in the order of $1e6$ variables and constraints on a high-end personal computer.

SNOPT 6.0

SNOPT uses a sparse sequential quadratic programming (SQP) algorithm [Gill et al., 2002] with an approximated Hessian. An SQP algorithm adds the constraints to the objective function and uses a second-order Taylor polynomial as minimization function. SNOPT is relatively efficient if the number of degrees of freedom (DOF) is small. Compared to other solvers, SNOPT requires few function evaluations due to the quadratic approximation of the Lagrange function.

According to the articles of [Gill et al., 2002] and [Mittelmann, 2010] SNOPT can handle up to $1e5$ variables.

4-2-2 Comparison of Different Solvers

CONOPT, IPOPT and SNOPT might be able to obtain a local optimum when applied to a flooding optimization problem. KNITRO and all other NLP solvers which come with GAMS were not able to solve a flooding optimization problem, resulting in an infeasible termination. The solvers are compared using the reservoir model presented in Figure 4-1. The IG is based on a constant injection rate of $0.0059\ m^3/s$ and a constant production rate of $-0.0061\ m^3/s$. These settings give typical results which are also obtained using smaller models or different IGs.

The results are presented in Table 4-3, where the solvers are compared on the obtained NPV and the optimization time. SNOPT will only obtain a feasible solution, but does not optimize the problem. IPOPT is almost two times faster than CONOPT, but the NPV found by CONOPT is significantly higher compared to other solvers. KNITRO terminates infeasible. Solvers come

Solver	NPV (USD)	Optimization Time (s)
Initial Guess	3.84e8	not applicable
CONOPT	8.16e8	50.5
IPOPT	5.57e8	27.8
KNITRO	infeasible	> 1800
SNOPT	3.84e8	2.9

Table 4-3: Presentation of four GAMS solvers compared to the IG. The performance is compared on the obtained NPV and the required optimization time. Using CONOPT results in the highest NPV. SNOPT is the fastest, but does not optimize the problem. IPOPT is faster than CONOPT but does not obtain the highest NPV. KNITRO terminates infeasible.

with option files which can be tweaked to improve the performance. This has been done for CONOPT described in Section D-3.

The resulting profiles of the well variables are presented in Figure 4-5. Although CONOPT provides the highest NPV, the output is the most chattering as well. It has been tried to use two solvers successively, which did neither improve the NPV nor the required optimization time nor the output behavior.

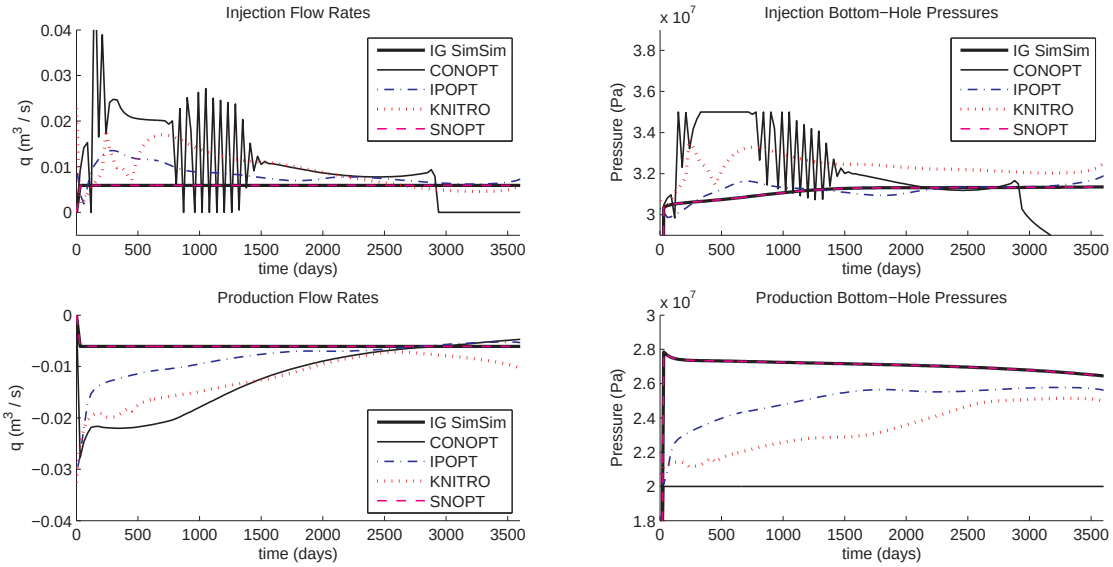


Figure 4-5: Presentation of four GAMS solvers compared to the initial guess. The performance is compared on resulting behavior of the optimal solution. CONOPT gives the most chattering behavior.

[1] http://www.ziena.com/papers/case_performance.pdf

4-3 Scaling

Compared to the sequential method, the simultaneous method generates a very large non-linear programming (NLP) problem due to the spatial and time discretization for all variables and states. This section will be used to demonstrate the size of the NLP problem as well as to discover whether the current implementation can be used for larger models.

A 2×2 model with a homogeneous rock permeability of $1e{-}13$ m^2 and square grid blocks is used initially. The decision for such a model is based on the fact that SimSim cannot deal with smaller models and cannot deal with non-square grid blocks. The model has an injector in the top-left corner and a producer in the bottom-right corner. The rock permeability is the logarithmic average of the model used for verification. The optimizations are initiated with an IG based on a constant injection rate of 0.006 m^3/s and a constant production rate of -0.0065 m^3/s. The upscaling will be done by adding an additional column of two vertical grid blocks in between the wells as illustrated in Figure 4-6.

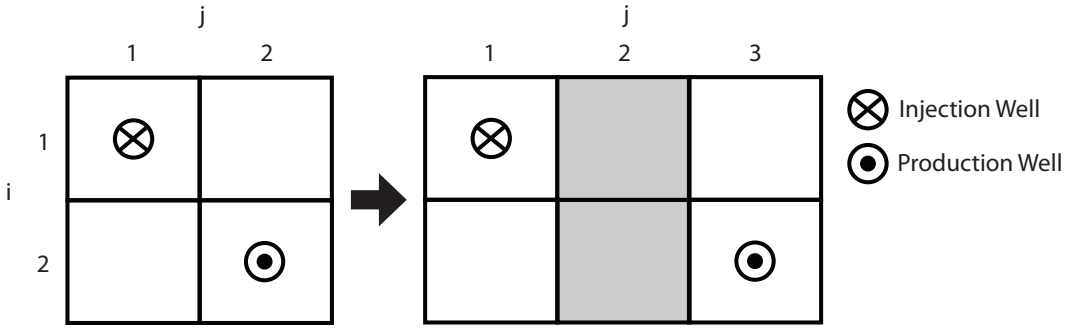


Figure 4-6: Illustration of a first step in upscaling; Grid blocks will be added as a column in between the columns containing the wells. The injection well will be placed in the top-left corner and the production well will be placed in the bottom-right corner.

The upscaling will change the model properties due to the displacement of the wells and the size of the model in terms of cubic meters. Therefore the net present value (NPV) of the different models is not compared. A better approach would be to refine the grid blocks by dividing every grid block in nine smaller grid blocks. This would keep the model properties identical. However, the current implementation of the simultaneous method is not yet able to solve these larger models. For this reason, the upscaling of the model is limited to small problems.

The optimizations are executed on a standard desktop computer with a 2.8 GHz processor and 4 GB of memory running at a 64-bit Windows operating system. 30 test cases have been optimized for different IGs, of which 6 are presented in Table 4-4 to demonstrate the increase in the number of variables and optimization time when scaling up the model. Models of 2×5 grid blocks could only be solved incidentally depending on the time-horizon. Models of 2×6 grid blocks and larger have been tried to optimize as well, but CONOPT was not able to produce feasible solutions.

Comparing cases 3 and 4 of Table 4-4, the optimization horizon quadruples, while the optimization time increases with more than a factor 18. This behavior is similar for cases (1 and 2) and (5 and 6). The optimization horizons are different, because models may become

Case	N_i (-)	N_j (-)	N_t (-)	Horizon (years)	No. Variables (-)	Optimization Time (s)	Initial Error (-)
1	**2**	**2**	**123**	**10**	**9,516**	**3.9**	**0.30**
2	2	2	243	20	18,984	25.9	2.29
3	**2**	**3**	**123**	**10**	**13,777**	**10.3**	**0.73**
4	2	3	487	40	54,545	189.1	6.12
5	**2**	**4**	**123**	**10**	**17,959**	**53.1**	**9.88**
6	2	4	609	50	88,915	2746.5	17.51

Table 4-4: Comparison of six optimizations which differ in model size ($N_i \cdot N_j$) and in the number of time steps (N_t), resulting in a different optimization horizon due to a constant time-step. The models are compared on the number of variables, the required optimization time and the initial error. For every model size, two time-horizons were selected. The horizons differ because some optimizations terminated infeasible. The bold optimizations have identical optimization horizons and can therefore be compared on the effect of upscaling the model in terms of the number of grid blocks. The optimizations are initiated with an IG based on a constant injection rate of $0.006\ m^3/s$ and a constant production rate of $-0.0065\ m^3/s$.

infeasible for specific horizons. Besides an increase in the number of time steps, an increase in the number of model grid blocks is of significant influence on the optimization time as well. Comparing cases 1 and 5, the number of grid blocks doubles while the optimization time increases with a factor 13.6. It can be computed that the number of variables is almost linear to the number of grid blocks multiplied by the number of time steps. On the 30 test-cases, the average number of decision variables is close to 18 variables per grid block per time-step.

It must be noted that the number of decision variables is not a hard number, as auxiliary variables are counted as well. However, the number is a measure of the size of the NLP problem. Note the difference in NLP size compared to the sequential method, where the NLP of cases 1, 3 and 5 would consist of 2 wells · 123 time-steps or 246 variables, independent of the number of grid blocks.

Tests have also been performed to investigate the influence of the selected rock permeability. A model with 2×8 grid blocks and 246 time-steps with a rock permeability of $1e{-}12\ m^2$ has been solved in $702.5\ s$. Decreasing the permeability to values of $1e{-}14\ m^2$ has the opposite effect resulting in less feasible solutions. Again, the number of variables and the number of grid blocks are not a hard indication and dependent on the model parameters, but they demonstrate the problems of upscaling due to use of the simultaneous method.

4-4 Sensitivity to Initial Guesses

An IG provides initial values for all decision variables of an NLP problem. The IG may influence the optimization performance and optimal objective value as it can be difficult to find suitable values for all states for the complete time-horizon ([Cuthrell and Biegler, 1987], [Poku and Biegler, 2004]), as they must satisfy all constraints.

For this reason it is investigated what the impact of different IGs is on the NPV and the optimization time. The model presented in Figure 4-1 is used. The IGs are generated by SimSim and based on continuous injection and production rates. The time horizon is 10 years

divided in 123 time-steps of 30 days. The optimization has injection and production bottemhole pressure (BHP) constraints of $+50e5\ Pa$ and $-100e5\ Pa$. Injection rates are positive and production rates are negative.

4-4-1 Sensitivity Measure to Different Initial Guesses

58 different IGs are tested which are based on flow rates of which the absolute production rate is higher than the injection rate. Or, $q_{inj} + q_{prod} < 0$. The IGs are based on flow rates between $1e{-}1\ m^3/s$ and $1e{-}5\ m^3/s$. For reference, an optimized flow rate will be in the order of $1e{-}2\ m^3/s$. The IGs are compared on NPV, optimization time and other observations as the initial error in GAMS and the initial violation of constraints.

Net Present Value

From the 58 optimizations, 43 terminated locally optimal and 15 terminated infeasible. 22 of the 43 local optimal solutions terminated with an NPV which do not differ more than 1% from the highest obtained NPV. The 20 highest NPVs do not differ more than 0.08%. It has been investigated what the initial NPV has been for the best performing optimizations, and for every optimization the initial NPV was less than the optimized NPV. The smallest initial errors did not give the best optimization performance neither in terms of NPV nor in terms of time. The IGs from 8 of the 22 optimizations with the highest NPV did violate the BHP constraints.

The absolute differences between the injection and production flow rates from the 22 best performing optimizations in terms of NPV where less then $0.005\ m^3/s$. The production rate was never more than twice the injection rate. However, the injection and production rates did vary between values from $1e{-}2\ m^3/s$ to $1e{-}5\ m^3/s$.

Optimization Time

The optimization times varied between several seconds to minutes. From the 10 fastest optimizations which terminated locally optimal, only 3 were also one of the ten highest obtained NPVs. The 10 best performing optimizations in terms of time all violated the BHP constraints, where some BHPs were even zero over a period of multiple time-steps.

Observations

Several IGs are presented in a plot to observe whether the best IGs resulting in the highest NPVs do have similar dynamics compared to the optimized well variables. In terms of the resulting NPV of the optimization, the best 7 (optimal), the worst 5 (feasible) and 4 infeasible IGs are plotted in Figure 4-7 and Figure 4-8. This is done for the grid block with the injection well, because this grid block is the first to show changes in the states. The pressures of the IGs resulting in the highest NPV all show an initial increase to a value above $300e5Pa$, however, it is not a guarantee for an optimal solution as also infeasible IGs may have such a profile. For the saturations the profile of the optimal IGs do not follow a clear path.

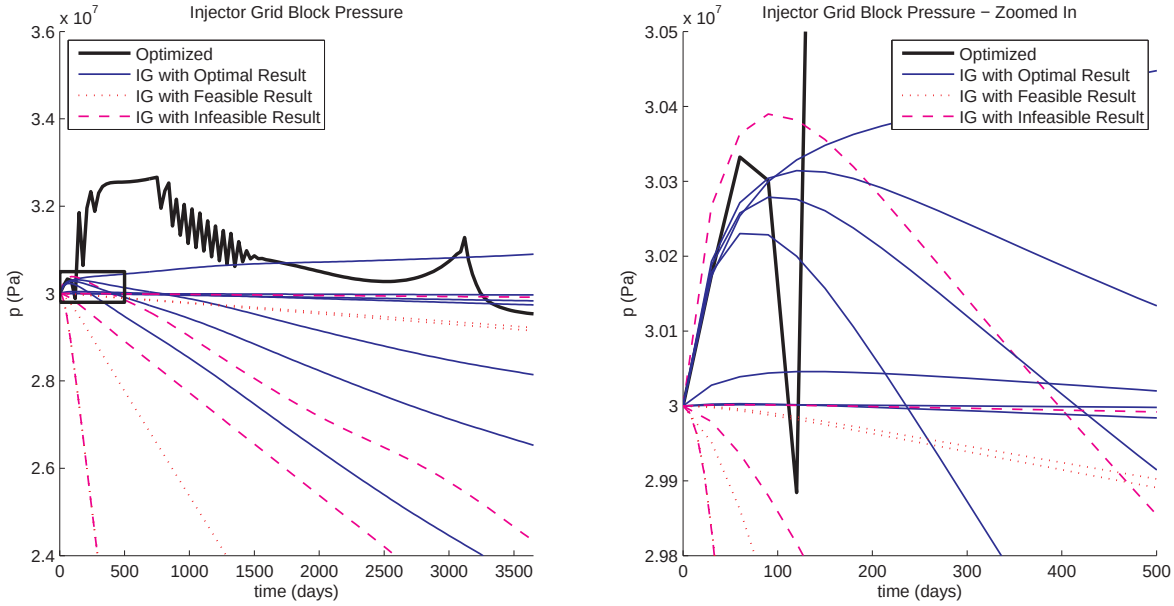


Figure 4-7: In terms of the resulting NPV after optimization, the best 7 (optimal), the worst 5 (feasible) and 4 infeasible IGs are plotted. The left graph shows the pressure for the full optimization horizon, where the right graph is zoomed in on the first 500 days, indicated in the left graph with a box. There is no clear distinction between the optimal, feasible and infeasible IGs.

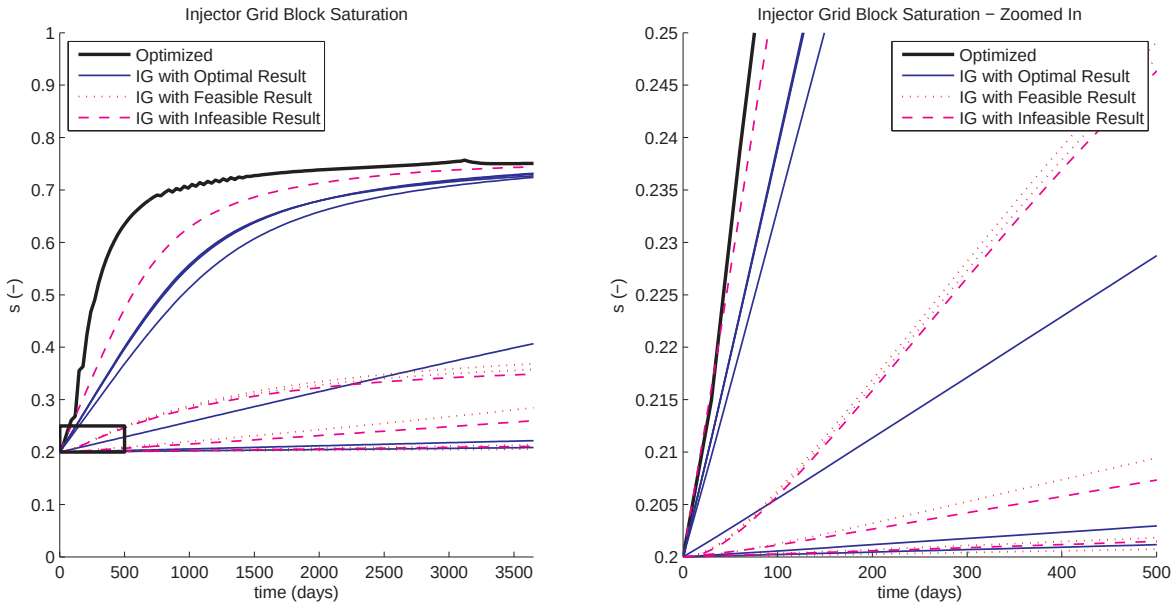


Figure 4-8: In terms of the resulting NPV after optimization, the best 7 (optimal), the worst 5 (feasible) and 4 infeasible IGs are plotted. The left graph shows the saturation for the full optimization horizon, where the right graph is zoomed in on the first 500 days, indicated in the left graph with a box. There is no clear distinction between the optimal, feasible and infeasible IGs.

4-4-2 Sensitivity to Ill-Conditoned Initial Guesses

Other IGs have also been tried, based on the following criteria:

1. Contra-intuitive flow rates; $q_{inj} + q_{prod} > 0$, creating a lower production rate than injection rate.
2. No activity; $q_{inj} = q_{prod} = 0$.
3. Elimination of flow rates in the pressure ordinary differential equation (ODE) (Eq. (3-9)) is known to cause singularities; choosing $q_w + q_o = 0$ can be achieved by choosing $q_{inj} + q_{prod} = 0$ as the production of water is initially very small relative to the oil production.
4. No IG; so all variables are set to zero initially.

Selecting IGs based on contra-intuitive flow rates or zero flow rates resulted in an infeasible termination of the optimization. No IG lead to an infeasible solution as well. However, choosing the IGs such that the flow rates cancel out did provide comparable results as where the IGs were based on higher production rates than injection rates.

As indicated in Section 3-5, the relative permeabilities are fixed to enable continuous modeling. The influence of optimal solutions has been investigated and the changed flow directions are less than 2%. The change in the transmissibility term may be significant, as this terms is directly dependent on the relative permeability, which can change from grid block to grid block up to a factor 25.

4-5 Concluding Remarks

The GAMS model has been verified with SimSim and four solvers have been compared. Analysis showed that the implementation of the current simultaneous method terminates infeasible when scaled up and that it is sensitive to IGs.

GAMS supports automatic discretization and calculates gradient information symbolically. A relative large initial error is caused by the equations of the Peaceman well model. Nonetheless the well models used in both equations are similar. Comparing the GAMS results in SimSim shows the error to be maximum 0.5%. Every optimization run produces an identical error. This indicates that the error is not due to computational limitations, but due to implementation differences. The differences are caused by distinct modeling environments, where SimSim is matrix-oriented and GAMS is equation-oriented.

The behavior of the optimal state-profiles can be physically interpreted. As much as possible oil is pushed out of the reservoir, after which the injection rate is relaxed in order to let water flow into the less permeable grid blocks. In the end, more water is injected to enable to push out the last oil. The hyperbolic behavior of the saturations is encountered as explained in Section C-2-2.

CONOPT gives the best performance in terms of NPV. Although CONOPT is not the fastest solver, the NPV is significantly higher compared to IPOPT and SNOPT. Other solvers terminate

infeasible, this points to either an ill-conditioned optimization problem or indicates that the other solvers cannot be applied to this type of optimization problem. An ill-conditioned problem is a problem which does not have a unique optimal solution or which is discontinuous. In this case a non-unique or weak solution exists.

Upscaling of the model is limited to 16 grid blocks or 88,915 decision variables. Increasing the number of variables will enhance the chance of the optimization to terminate infeasible. The computer is not running out of memory, indicating the infeasibilities are likely to be due to the fact the optimization problem is ill-conditioned, the IG generator is not good enough or that CONOPT cannot deal with larger models. As described in Section 4-2, the simultaneous method has been applied to problems in the order of $1e6$ variables. This indicates there should still be room for improvement.

IGs based on constant injection and production rates showed to have significant influence on the outcome of the simultaneous method. Only 22 of the 58 optimizations obtained its maximum within a bound of 1% of the highest NPV. The high sensitivity can be either caused by large model differences or it can be due to a weak solution, because the 20 highest NPVs differ less than 0.08%

To emphasize the relevance of the next chapter, it is observed that a chattering injection flow rate influences the pressure of the injection grid block. However, the chattering signal vanishes over space and completely disappears in the producing grid block. The pressure distribution is diffusive as described in Section C-2-1. Together with the fact none of the NPVs are identical, this indicates the existence of a weak solution.

Chapter 5

Comparison of the Simultaneous Method and the Sequential Method

This chapter will compare the simultaneous method with the sequential method with respect to dealing with state-constraints in Section 5-1 and multi-objective optimization in Section 5-2. Section 5-3 compares both optimizations of the sequential method and the simultaneous method on the obtained optimal well-variable profiles. The sequential optimization will be performed using Modular Reservoir Simulator (MoReS), the in-house reservoir simulator of Shell. Finally, Section 5-4 will end the chapter with concluding remarks.

5-1 Applying State-Constraints

One of the main issues of the currently used sequential method is the limited ability to deal with state-constraints [Kraaijevanger et al., 2007]. The simultaneous method should provide an advantage, which will be tested. The reservoir model has two types of states, pressures and saturations. The pressure constraints are always present in practice [Sarma et al., 2006]. Saturations are already bounded as explained in Section 3-4-3 for implementation reasons. For this reason, only pressure constraints are applied. The constraints are applied on a single grid block and on the whole reservoir.

5-1-1 Single Pressure Constraint

A single pressure constraint is applied on grid block $(i, j) = (1, 2)$. As can be seen in Figure 4-4, most of the water flows through this grid block due to the high permeability. The pressure constraint is set to $[p]_{1,2,n} \in [299e5, 301e5]$. The initial guess (IG) is based on a constant injection rate of 0.0059 m^3/s and a constant production rate of -0.0061 m^3/s.

The simultaneous method is able to deal with the state-constraint. The net present value (NPV) changes from $8.155e8$ USD to $8.134e8$ USD, which is a change of less than -0.27%. It

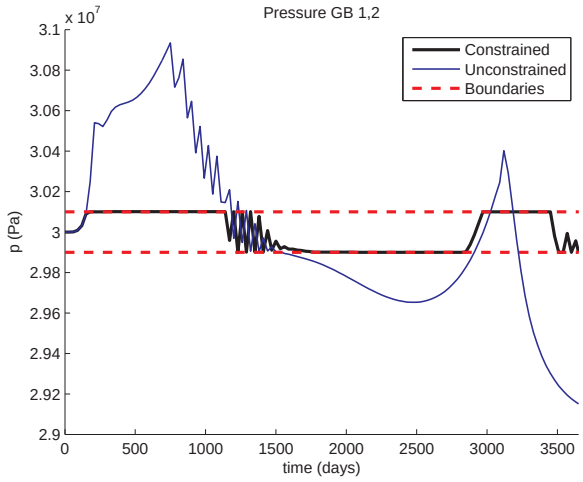


Figure 5-1: Pressure in grid block $(i, j) = (1, 2)$ for a constrained ($[p]_{1,2,n} \in [299e5, 301e5]$) and unconstrained optimization.

has also been investigated and confirmed that the simultaneous method is able to deal with state-constraints at single points in time.

5-1-2 Reservoir Pressure Constraint

The reservoir pressure is constrained to $[p]_{i,j,n} \in [295e5, 305e5]$ as presented in Figure 5-2. The IG have been adjusted and multiplied by $1e{-}1$ to avoid having an infeasible IG violating the pressure constraints. The IG is for this reason based on a constant injection rate of $0.00059\ m^3/s$ and a constant production rate of $-0.00061\ m^3/s$.

From Figure 5-2 it can be observed all the constraints are satisfied. The optimum however is affected and equal to $2.132e8$ USD. The chattering disappears.

5-2 Multi-Objective Optimization

A multi-objective optimization is difficult to implement in the sequential method as it requires a combination of input, output and state constraints. This has been encountered by [van Essen et al., 2010], in which a multi-objective optimization shows to be computationally expensive. The chattering behavior of the solution demonstrated in Figure 4-2 and Figure 4-3 is also mentioned in [Jansen et al., 2009]. Both [van Essen et al., 2010] and [Jansen et al., 2009] recognize the existence of a weak solution. An indication for a weak solutions is described in Section 4-4, where different values for the NPV are found. However, the highest 20 NPVs do not differ more than 0.08%. The existence of a weak solution indicates an under-constrained problem. The remaining degrees of freedom (DOF) can for example be used to minimize the variance of the solution. This can be done in two ways. The first method is to optimize the problem by adding the variance as cost to the objective function, called regularization. The second method is called lexicographic optimization and bounds the optimal NPV after which a variance minimization is performed. The variance of the inputs

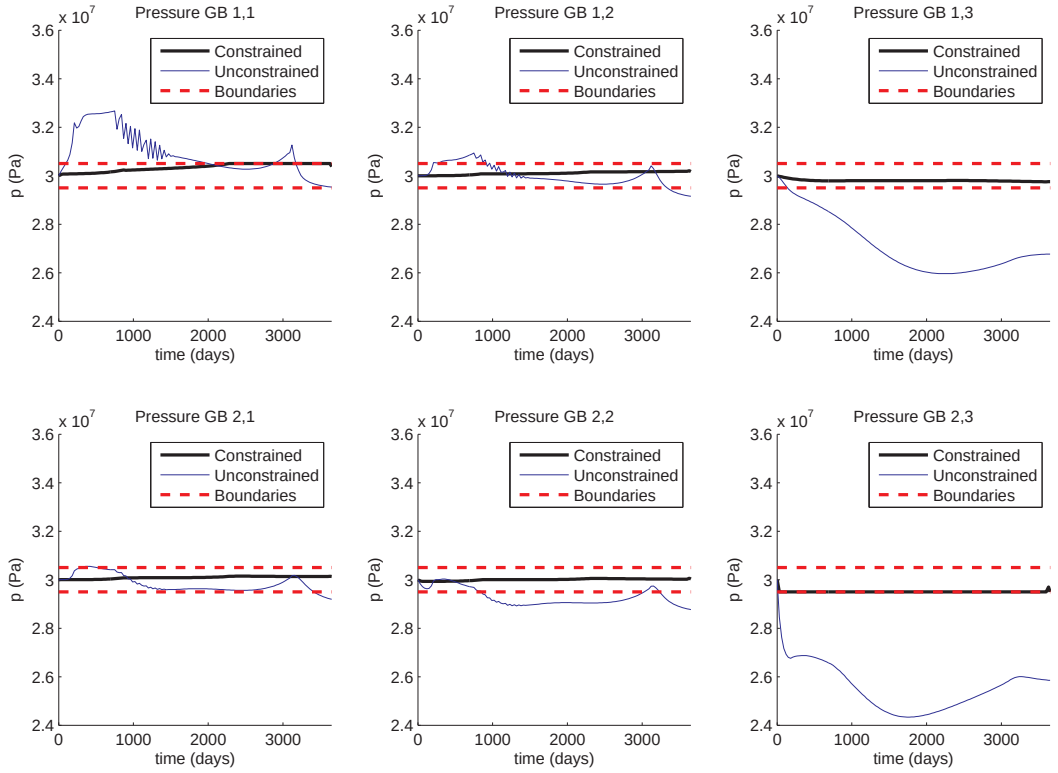


Figure 5-2: The reservoir model pressures for a constrained ($[p]_{i,j,n} \in [295e5, 305e5]$) and unconstrained optimization.

is described using a quadratic convex function often used in model predictive control (MPC) [Qin and Badgwell, 2003]. This is done using the Euclidean norm with a finite time-horizon, presented by Eq. (5-1).

$$\mathcal{J}_{var} = -\sum_{n=1}^{N_t-1} \left[[w_w]_n \left([q_w]_{i,j,n} - [q_w]_{i,j,n+1} \right)^2 + [w_o]_n \left([q_o]_{i,j,n} - [q_o]_{i,j,n+1} \right)^2 \right], \tag{5-1}$$

where $\mathcal{J}_{var}$ is the additional objective representing the variance of the weighted flow rates with the weightings $[w_w]_n, [w_o]_n$. The term above is a sum of convex functions and therefore convex as well. The term has to be negative since a maximization is performed instead of a minimization.

5-2-1 Regularization

The regularized optimization is executed by adding the term of Eq. (5-1) to the NPV presented by Eq. (3-25). Eleven regularized optimizations are compared to a standard or non-regularized optimization.

For the 11 best performing IGs in terms of NPV obtained in Section 4-4, the final NPV showed to be affected less than 0.06%. For the weighting values are selected such that the smoothening is in the order $1e{-}2$ of the NPV, putting the weighting on optimization of the NPV. This

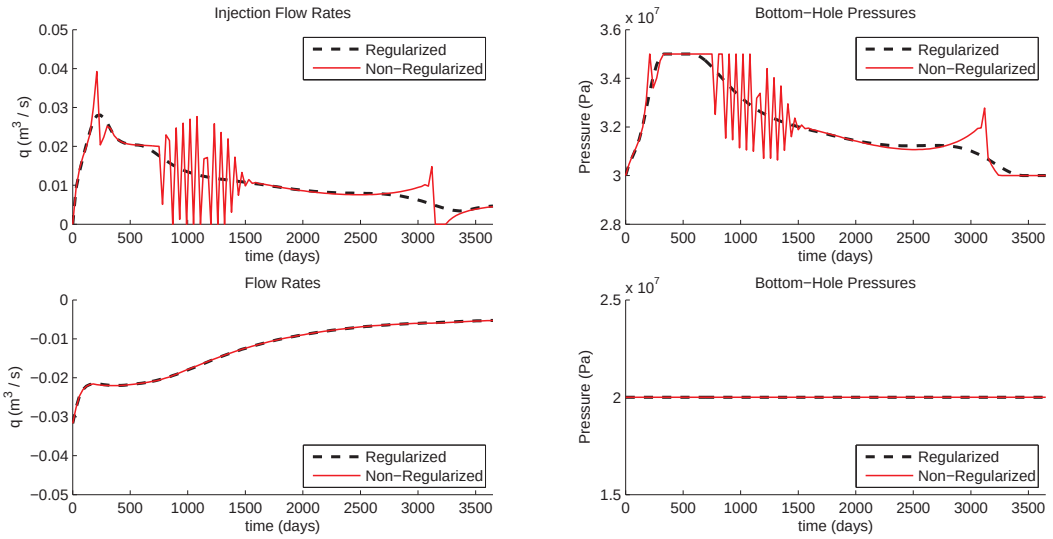


Figure 5-3: Regularized optimization compared to a non-regularized optimization. The well variables of the regularized optimization are smooth with a loss in NPV of less than 0.06%.

resulted in $[w_w]_n = [w_o]_n = 1e10$. Tuning the weighting may be a difficult procedure as the weighting influences the value of the variance. The difference in optimization time has also been compared and is summarized in Table 5-1. The optimization time of the regularized optimization compared to non-regularized optimization may be significant. Chattering is eliminated as shown in Figure 5-3. The optimal solution becomes non-singular and therefore unique or non-weak. This is also described in the paper of [Jacobson et al., 1970].

	Change in Optimization Time
Median	$+19\%$
Mean	$+23\%$
Minimum	-45%
Maximum	$+125\%$

Table 5-1: Time performance of a regularized optimization compared to a non-regularized optimization based on 11 optimizations. A regularized optimization can improve the optimization time, but may take on average 23% more time.

It has also been tried to increase the weighting of the smoothening function drastically, after which the influence would slowly diminish. However, this resulted in an NPV of $< 50\%$ of the non-regularized NPV. The values of the NPV for the regularized optimizations were 8 out of 11 times identical where they were all different for non-regularized optimizations. The variance decreases with a factor in the order of $1e2$.

5-2-2 Lexicographic Optimization

The lexicographic optimization method first optimizes the NPV, after which the NPV is bounded and a variance minimization is performed. This method is a more convenient way to

minimize variance, as the influence of the variance minimization can be controlled very well by bounding the NPV.

Different lower-bounds have been implemented on the NPV. The bounds on the NPV were always satisfied during the 11 test-cases. The results of lexicographic optimization are very similar to regularized optimization as presented in Figure 5-3.

To compare the lexicographic method with the regularized method, the lower bound is set to $\varphi_{min} = \varphi - 5e5$ USD. This results in a change in the value of the objective function of $< 0.06\%$ which has also been encountered in the regularized optimization. The results in optimization time are presented in Table 5-2.

	Change in Optimization Time
Median	+52%
Mean	+66%
Minimum	+32%
Maximum	+152%

Table 5-2: Time performance of a lexicographic optimization compared to a basic optimization based on 11 optimizations. A lexicographic optimization may take on average 66% more time.

The optimization takes on average 66% more time which is significant and more compared to a regularized optimization, where the average additional time is 23%.

5-3 Comparison with the Sequential Optimization Method

This section will compare the simultaneous method with the sequential method. MoReS, the in-house simulator of Shell, is used for the sequential optimizations. Two optimizations will be performed, of which Case 1 has only maximum injection and production constraints on the bottem-hole pressure (BHP), where the optimization of Case 2 also has constraints on the flow rates, presented in Table 5-3.

	Case 1	Case 2
$q_{inj,IG}$	$0.0059\ m^3/s$	$0.01\ m^3/s$
$q_{prod,IG}$	$-0.0061\ m^3/s$	$-0.012\ m^3/s$
$q_{inj,max}$	n.a.	$0.01\ m^3/s$
$q_{prod,min}$	n.a.	$-0.012\ m^3/s$
$p_{inj,max}$	$350e5\ Pa$	$350e5\ Pa$
$p_{prod,min}$	$200e5\ Pa$	$200e5\ Pa$

Table 5-3: IGs and constraints for two test-cases which will be used to compare the simultaneous method with the sequential method.

In the first subsection, a comparison will be made between the models of MoReS and Simple Simulator (SimSim) by comparing the IGs used for optimizations in MoReS and General Algebraic Modeling System (GAMS). The results of these optimizations are presented in the second subsection.

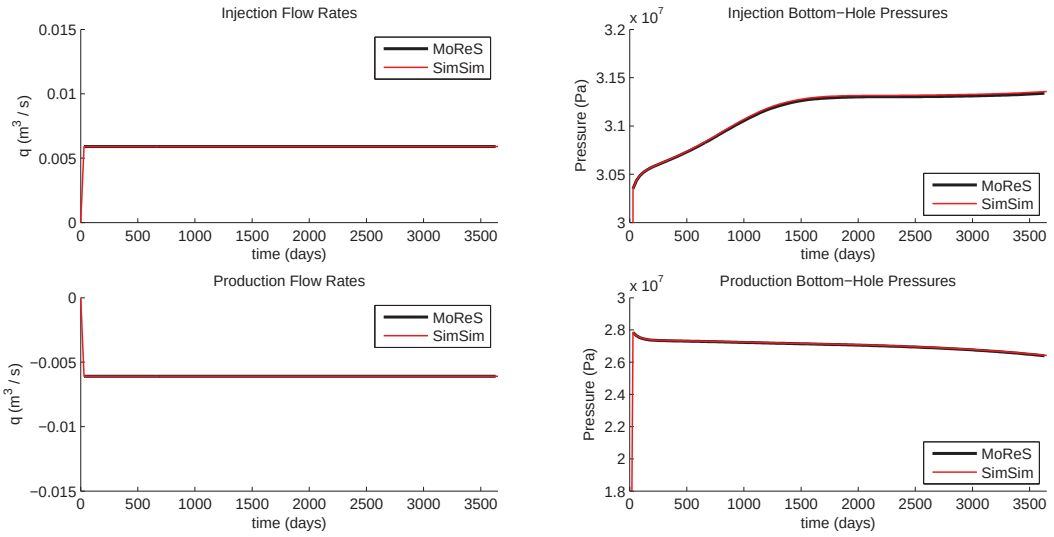


Figure 5-4: IG of Case 1 for the comparison of MoReS with SimSim. Except for an initial error both plots overlap.

5-3-1 Model Verification using a Forward Simulation

For both test-cases, the IGs are plotted in Figure 5-4 and Figure 5-5. The time-steps and integration methods are identical. For case 1, the models behave comparable, however, for case 2 errors develop from the moment the constraints become boundaries of the variables. As the SimSim model behaves identical to GAMS, the comparison of GAMS and MoReS may for this reason not be valid. However, as the dynamics of both models are comparable, it is still interesting to see what the different results will be of both optimization methods.

5-3-2 Comparison of the Optimization Performance

The optimal injection and production strategies of both MoReS and GAMS are presented in Figure 5-6 for Case 1. GAMS generates a chattering injection flow rate. For this reason, the injection is smoothened using regularization to allow for a clearer distinction between both optimization methods. Although the models differ, it is clear the injection strategy of GAMS does not directly fully inject water while MoReS does. Comparing the NPVs, GAMS wins slightly with less than 1% due to the fact it delays the water-front by waiting with the injection of water. This results in a lower saturation in the producing grid block. However, the increase in NPV may not be due to the use of the simultaneous method as the model dynamics are not completely similar.

Studying Case 2 in Figure 5-7 shows comparable results as in Case 1. The water injection in GAMS is delayed as well, which is compensated with a lower production. The producing grid block of the GAMS model has a delayed increase in the value of the saturation compared to the MoReS optimization. The delay observed in the saturation at the producer grid block is similar to the delay found in the IG in Figure 5-5. The better performance of GAMS can therefore not be presumed to be due to the different optimization methods. The solution of GAMS to the injection flow rate is a bang-bang solution, which is a result of the rate constraints. The

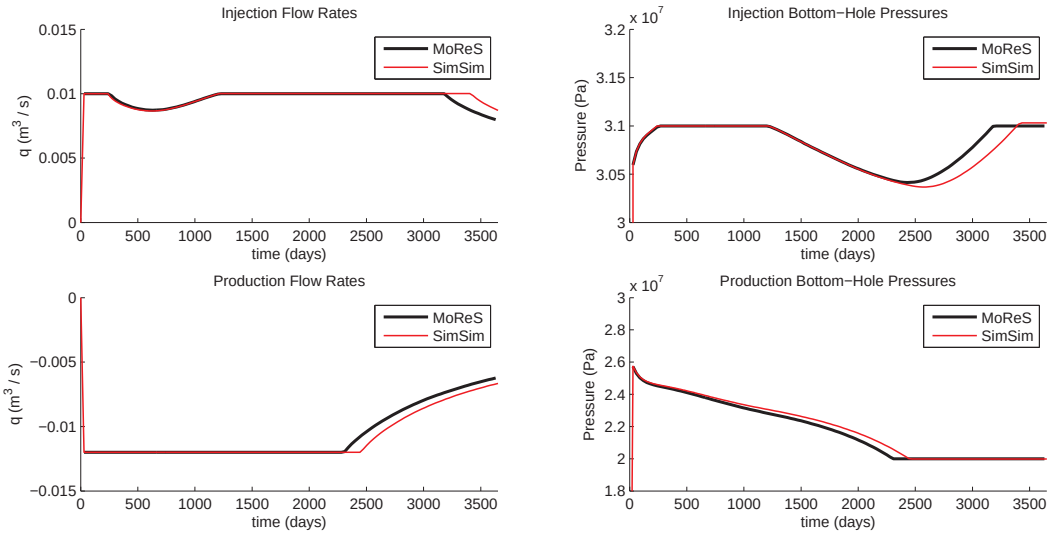


Figure 5-5: IG of Case 2 for the comparison of MoReS with SimSim. Activation of the constraints results in different profiles of the well variables for MoReS and SimSim. The dynamics of both models are comparable.

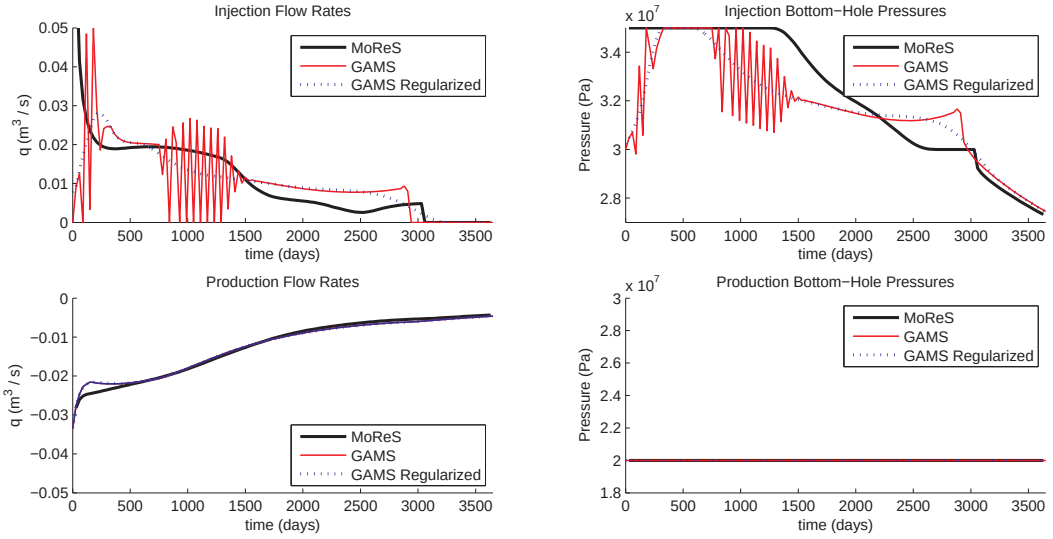


Figure 5-6: Optimal solutions of Case 1 obtained by both MoReS and GAMS. The solver CONOPT generates a chattering profile, which has been smoothened using regularization for comparison purposes. Both MoReS and GAMS obtain similar optima, but with different dynamics.

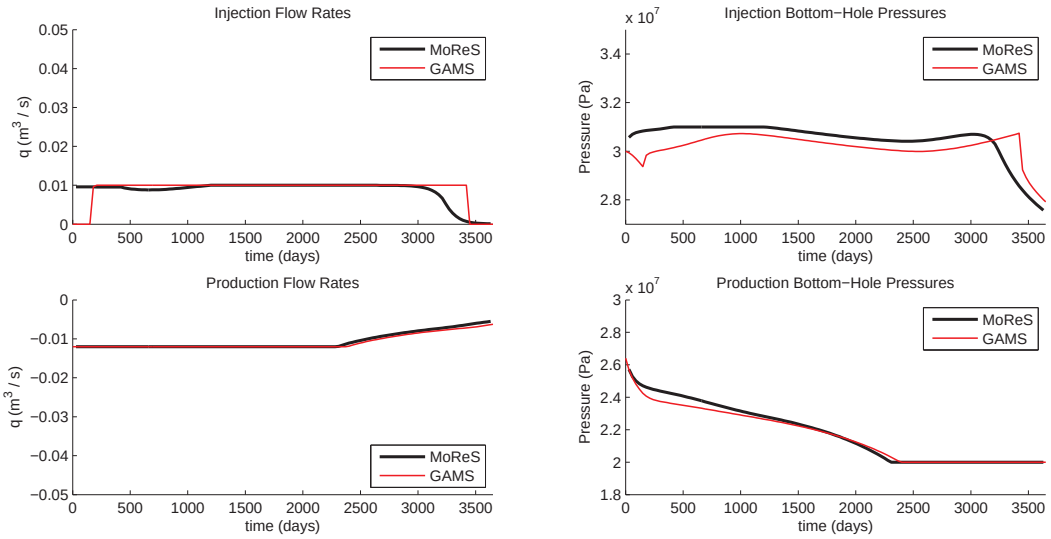


Figure 5-7: Optimal solutions of Case 2 obtained by both MoReS and GAMS. Both MoReS and GAMS obtain similar optima, but with different dynamics.

pressure constraint in GAMS for the injection BHP is not triggered, where the constraint of the sequential method is. Using the optimal variables from GAMS as IG for MoReS and vice versa does not result in an optimal solution directly, emphasizing the differences in the models.

An interesting remark on the computational time; MoReS takes about 3 minutes for an optimization with 30 integrations on a special server where GAMS needs less then 1 minute on a desktop computer. The speed of GAMS is remarkable. However, as observed in Section 4-3, the optimization time of GAMS quickly increases when used for larger models.

It has been tried to eliminate the differences in both optimizations by choosing different IGs for the GAMS optimization, including the optimal result of MoReS. This gave worse or identical results as which have been presented.

5-4 Concluding Remarks

The issues of dealing with state-constraints and performing a variance minimization when using the sequential method are solved by using the simultaneous method. A comparison of the simultaneous method with the sequential method has been made, but the underlying models are different.

A pressure constraint has been implemented which could not be controlled directly by the injector or producer. Such a constraint is almost impossible to satisfy using the sequential method. Although the constraint became active, it has not been violated. Also, pressure constraints were applied to the whole reservoir, which were all satisfied as well.

Multi-objective optimization with both regularization and lexicographic optimization is possible and results in smooth variable profiles. Regularization may be faster than lexicographic optimization. Regularized optimization has resulted 8 out of 11 times in identical optima and eliminates the weak optimum. Regularization has a disadvantage compared to lexicographic

optimization, because tuning of the weighting of the variance is required. The weighting influences the value of the variance and is therefore difficult to control. This is not a problem when using lexicographic optimization, in which the influence of the minimization can be controlled perfectly as the NPV can be bounded. This is a computationally very expensive optimization when performed with the sequential method.

The simultaneous method is compared to the sequential method as used by MoReS, the in-house simulator of Shell. Although the dynamics are comparable, the models are not completely identical. The resulting NPVs of both models differ less than 1%. The difference can be either due to model differences or due to the fact that the simultaneous method keeps the saturations in the production well low for a longer period, resulting in less water production. Physically this may also be logical, as waiting with injection of water will give time to the lower permeable grid blocks to release their oil. Well constraints were satisfied in both optimizations, which may be due to the fact the reservoir model used for comparison is not complex enough and the states are relatively easy to be controlled indirectly. Using the optimal data of one method as IG to the other method did not result in an instant optimality, emphasizing the differences in the models. GAMS requires less than 1 minute to solve a model with 2×3 grid blocks and 123 time-steps where MoReS requires about 3 minutes to execute the optimization, this is due to multiple integrations of the ordinary differential equation (ODE)s in the sequential optimization method.

The simultaneous method and the sequential method are compared, but due to differences in the underlying reservoir models, not completely comparable. Although the dynamics showed to be similar, it cannot be concluded which method is better in terms of NPV. The simultaneous method is able to deal with state-constraints and allows for using multi-objective optimization. The method is for this reason more practical for real-life situations.

Chapter 6

Conclusions and Recommendations

6-1 Conclusions

In this research, it has been investigated to what extent the simultaneous method is able to solve a flooding optimization problem. The problem is divided into three parts. Firstly, it has been demonstrated how the simultaneous method can be applied to a flooding optimization problem. Secondly, the current implementation has been verified and it has been researched what its limitations are. Thirdly, the simultaneous method has been compared to the sequential method. The main conclusions are given below.

1. The simultaneous method can be applied to a flooding optimization problem by applying a discretization in both space and time. All PDEs are transcribed to first-order difference equations. Integration of the difference equations is based on in industry used implicit Euler integration. he difference equations are implemented in General Algebraic Modeling System (GAMS). GAMS supports automatic discretization and symbolic differentiation which improves the time needed for model implementation as well as the computational time.

2. The GAMS model has been verified to Simple Simulator (SimSim), which is a forward reservoir simulator. The difference between both models is not more than 0.5%. This is due to distinct modeling environments. GAMS is equation-oriented and SimSim matrix-oriented.

   The current implementation of a flooding optimization problem in GAMS has solved problems based on reservoir models of up to 16 grid blocks and 88,915 decision variables. This number gives an indication of the current possibilities. However, it is not a hard measure as it depends on the model parameters. The current implementation of the simultaneous method is very sensitive to IGs. From 58 reasonable IGs, only 22 converged to an NPV which did not differ more than 1% from the highest obtained NPV. 15 optimizations terminated infeasible.

3. Compared to the sequential method, the simultaneous method is able to deal with state-constraints and allows for using multi-objective optimization to eliminate chattering. The simultaneous method is for this reason more practical for real-life situations compared to the sequential method. Multi-objective optimization can be achieved relatively easy by using regularization or lexicographic optimization. The regularization method may be difficult to tune, but has the advantage that it results in a non-singular optimization problem. Regularization increases optimization time with 23% on average compared to a non-regularized optimization, where lexicographic optimization takes 66% longer on average. The lexicographic optimization allows to control the influence of the variance minimization on the NPV perfectly.

   The simultaneous method has been compared to the sequential method in terms of optimization performance as well. The underlying models of both methods are not completely identical. During an optimization executed via the simultaneous method, GAMS initially holds off any water injection, which results in lower saturation values. The simultaneous method produces therefore less water compared to the sequential method, resulting in a slightly higher net present value (NPV). The NPVs obtained by both methods differ less than 1%, but this may not be due the different optimization methods as the models are not completely identical. MoReS has been used to perform the sequential optimizations. For a model with 6 grid blocks and 123 time-steps, MoReS takes about 3 minutes for an optimization with 30 integrations on a special server where GAMS needs less then 1 minute on a desktop computer.

On the one hand it is demonstrated that drawbacks of the sequential method can be avoided by using the simultaneous method. On the other hand, solving a flooding optimization problem with the current implementation of the simultaneous method may introduce new issues. Recommendations are given in the next section which are likely to improve the performance of the current implementation.

6-2 Recommendations

The current implementation of the simultaneous method is still limited to small reservoir models and sensitive to IGs. This may be improved by subsequent research, which should include the following recommendations.

- The application of the simultaneous method to a flooding optimization problem can be improved on two points. The first is the use of orthogonal collocation. This method applies a function approximation on the model ODEs or even on the model PDEs. The approximated functions are fitted to the ODEs or PDEs at discrete points in space and/or in time. This drastically reduces the NLP size and generally improves the precision of the solution and the required optimization time. The improvement can be used for upscaling to larger reservoir models. The second point of improvement is the fact that the relative permeabilities are based on the IG and not on the GAMS model. The static relative permeabilities can be made time-varying, which will give a more realistic outcome and may improve the optimization performance as well.

- The implementation of a flooding optimization problem in GAMS can be improved by using a better initial guess (IG). Firstly, GAMS starts with an initial error and therefore as an infeasibility problem, instead of an optimization problem. This is caused by distinct modeling environments. GAMS is equation-oriented, where the sequential method and the IG generator are implemented in a matrix-oriented environment. Using an IG from an equation-oriented environment would eliminate the initial error and allow GAMS to start as an optimization problem instead of an infeasibility problem. Secondly, the IGs are based on constant flow-rates, but the optimal flow-rates are time-varying. It may for this reason be better to obtain IGs which are based on more realistic varying flow-rates. Both improvements will reduce the sensitivity to IGs.
- The comparison of the sequential method with the simultaneous method can be improved. Both optimization methods are not compared in depth due to the distinct modeling environments. An improved comparison will give a better understanding of convergence properties, computational speed and optimal result of both methods.

The simultaneous method is an improvement to the currently used sequential method in terms of dealing with state-constraints and performing multi-objective optimizations. However, real-life problems cannot be solved yet due to the fact real-life reservoir models are based on millions of grid blocks.

For this reason, the current implementation of the simultaneous method must be improved before it is possible to compare both optimization methods. The first step is to eliminate the initial error by using an equation-oriented IG generator. The second step is to implement orthogonal collocation which enables to reduce the number of decision variables. At last, it is important to have identical reservoir models to compare the performance of both optimization methods.

With these improvements, reservoir models which are currently used in industry may not be solved yet. In literature, problems have been solved with millions of decision variables. This allows to have 1 decision variable per grid block, where the current implementation has around 18 decision variables per grid block per time-step. This indicates a big gap between the possibilities of the current implementation and the requirements for real-life flooding optimization problems.

Although real-life reservoir models may not be solved today, it is likely that ongoing developments may result in a future possibility of using the simultaneous method to solve flooding optimization problems. The first development is that it is being investigated to reduce the resolution of reservoir models. High resolution models do not add value due to high uncertainties in the model data. The second ongoing development is research to more efficient solver algorithms. Better solvers allow for better convergence properties. The third development is that computers are improving, enabling to solve larger models.

Summarizing, the current implementation is not able to deal with real-life reservoir models. However, in the future the method is likely to be able to be applied to these models. It is therefore of interest to continue this research because the simultaneous method is able to obtain feasible practical solutions, which is difficult with the currently used sequential method.

Appendix A

Adjoint Method

The adjoint or backward sensitivity method is the most efficient method for calculating the sensitivity equations and currently used in flooding optimization. The method is independent of the number of decision variables and about as expensive as the forward simulation of the model [Sarma et al., 2005]. The adjoint equations are obtained analytically, which is a labor-intensive process for large models due to the large number of equations and symbolic work involved to obtain Equation (A-5). The method is explained according to [Sarma et al., 2005], [van Essen et al., 2009a] and [Brouwer and Jansen, 2004].

The adjoint method makes use of the augmented objective function $\hat{\varphi}_A$, which is the accumulation of the objective function with the system equations multiplied by a Lagrange multiplier. The system equations are adjoined to the objective function as follows:

$$\hat{\varphi}_A(x, q, \lambda) = \sum_{i=0}^{N-1} \left[L_i(x_{i+1}, x_i, q_i, \lambda_{i+1}) \right], \tag{A-1}$$

where x represents both the differential and algebraic states for simplicity, q the decision variables, λ a Lagrange multiplier and L_i the auxiliary function:

$$L_i(x_{i+1}, x_i, q_i, \lambda_{i+1}) = \varphi_i(q_i) + \lambda_{i+1}^T f_i(x_{i+1}, x_i, q_i). \tag{A-2}$$

In order to achieve optimality, the derivative of the augmented objective function with respect to the states x, the decision variables q and the Lagrange multiplier λ must be equal to zero for all separate four terms of Equation (A-3).

$$\begin{aligned}\delta\hat{\varphi}_A = 0 = & \sum_{i=1}^{N-1} \left[\left(\frac{\partial L_{i-1}}{\partial x_i} + \frac{\partial L_i}{\partial x_i}\right)\delta x_i\right] \\ & + \left(\frac{\partial L_{N-1}}{\partial x_N}\right)\delta x_N \\ & + \sum_{i=0}^{N-1} \left[\left(\frac{\partial L_i}{\partial q_i}\right)\delta q_i\right] \\ & + \sum_{i=0}^{N-1} \left[\left(\frac{\partial L_i}{\partial \lambda_{i+1}}\right)\delta \lambda_{i+1}\right].\end{aligned} \tag{A-3}$$

The latter term is equal to zero by definition, because $\partial L_i/\partial\lambda_{i+1} = f_i = 0$. The first and second terms are:

$$\begin{aligned}\left(\frac{\partial L_{i-1}}{\partial x_i} + \frac{\partial L_i}{\partial x_i}\right)\delta x_i &= \lambda_i^T \frac{\partial f(i-1)}{\partial x_i} + \frac{\partial \hat{\varphi}_i}{\partial x_i} + \lambda_{i+1}^T \frac{\partial f_i}{\partial x_i} = 0, \\ \left(\frac{\partial L_{N-1}}{\partial x_N}\right)\delta x_N &= \lambda_N^T \frac{\partial f_{N-1}}{\partial x_N} + \frac{\partial \hat{\varphi}_N}{\partial x_N} = 0.\end{aligned} \tag{A-4}$$

The equations can be solved for λ_i. The next equation is known as the adjoint model:

$$\begin{aligned}\lambda_i^T &= -\left[\frac{\partial \hat{\varphi}_i}{\partial x_i} + \lambda_{i+1}^T \frac{\partial f_i}{\partial x_i}\right]\left(\frac{\partial f_{i-1}}{\partial x_i}\right)^{-1}, \\ \lambda_N^T &= -\left[\frac{\partial \hat{\varphi}_N}{\partial x_N}\right]\left(\frac{\partial f_{N-1}}{\partial x_N}\right)^{-1}.\end{aligned} \tag{A-5}$$

The Lagrange multipliers are solved backward, starting with the latter term λ_N. The multipliers are calculated by using the stored states of a previous forward simulation of the reservoir model by the DAE solver. Notice the independence of the decision variables q. The above steps reduce Equation (A-3) to:

$$\begin{aligned}\delta\hat{\varphi}_A &= \sum_{i=0}^{N-1}\left[\left(\frac{\partial L_i}{\partial q_i}\right)\delta q_i\right], \\ &= \sum_{i=0}^{N-1}\left[\left(\frac{\partial \hat{\varphi}_i}{\partial q_i} + \lambda_{i+1}^T \frac{\partial f_i}{\partial q_i}\right)\delta q_i\right].\end{aligned} \tag{A-6}$$

Appendix B

Reservoir Modeling based on the Principles of Flow through Porous Media

A reservoir model describes the flow of all media within an oil reservoir. This appendix will describe the derivation of a white-box model. The model is based on the conservation of mass, conservation of linear momentum using Darcy's Law and possibly also a vapor-liquid equilibrium. Firstly, the basic equations for a single-phase flow are derived. Next, a basic multi-phase multi-dimensional model is described.

B-1 Single-Phase Flow

Models describing single-phase flow are relatively easy to obtain. This section will briefly explain the model by describing one-dimensional flow, after which a model for two-dimensional and three-dimensional flow will be presented. All the media described are assumed to be compressible and the processes are assumed to be iso-thermal.

B-1-1 Single-Phase One-Dimensional Flow

A fluid is considered to flow horizontal in a one-dimensional direction (Figure B-1).

The equations are obtained from [Peaceman, 1977], [Aziz and Settari, 1979] and [Jansen, 2009]:

$$\underbrace{(A\rho v_x)_x}_{\text{In}} - \underbrace{(A\rho v_x)_{x+\Delta x}}_{\text{Out}} + \underbrace{qA\Delta x}_{\text{Source}} = \underbrace{A\frac{\partial(\phi\rho)}{\partial t}\Delta x}_{\text{Accumulation}} \quad , \tag{B-1}$$

where $A(x)$ is the cross-sectional area and constant in time, $\rho(t,x)$ is the fluid density, $v_x(t,x)$ is the Darcy Velocity in the x-direction (Section B-1-2), $q(t,x)$ is the flow rate of unit volume

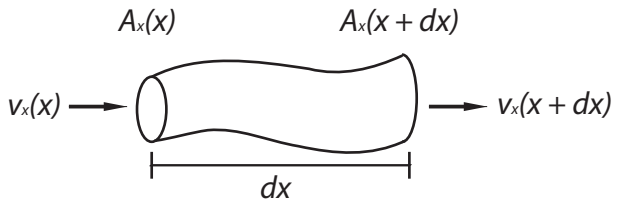


Figure B-1: Single-Phase One-dimensional Flow

per unit time, $\phi(t, x)$ is the porosity of the medium and t is the time. The flow rate $q(t, x)$ will be negative for a production well.

By firstly dividing Eq. (B-1) by Δx and secondly taking the limit $\Delta x \to 0$, Eq. (B-1) can be simplified:

$$-\frac{\partial(A\rho v_x)}{\partial x} + Aq = A\frac{\partial(\phi\rho)}{\partial t}. \tag{B-2}$$

B-1-2 Single-Phase Two-Dimensional and Three-Dimensional Flow

We can write Eq. (B-1) for two or three-dimensional flow as:

$$-\nabla \cdot (\alpha\rho\vec{v}) + \alpha q = \alpha\frac{\partial(\phi\rho)}{\partial t}, \tag{B-3}$$

where ∇ is the difference operator, where v_z is only present in the three-dimensional case;

$$\nabla \cdot (\vec{v}) = \frac{\partial(v_x)}{\partial x} + \frac{\partial(v_y)}{\partial y} + \frac{\partial(v_z)}{\partial z}, \tag{B-4}$$

and with α as the geometry-factor defined in [Peaceman, 1977] to allow for downscaling of the dimensions;

$$\begin{array}{lll} 1D: & \alpha(x) & = A(x) \\ 2D: & \alpha(x, y) & = H(x, y) \\ 3D: & \alpha(x, y, z) & \equiv 1. \end{array} \tag{B-5}$$

Darcy's Law

Darcy's Law describes the flow through a porous medium as a function of permeability, viscosity and pressure difference. From this law, the Darcy velocity vector $\vec{v}$ is introduced as:

$$\vec{v}_i = -\frac{\vec{K}k_{ri}}{\mu_i}(\nabla p_i - \rho_i g \nabla d). \tag{B-6}$$

where μ is the fluid viscosity, $\vec{K}(\vec{x})$ is the rock permeability tensor, g is the gravitational acceleration, $d(\vec{x})$ is the depth and subscript i refers to the water, oil or gas phase. The vector $\vec{x}$ represents (x), (x, y) or (x, y, z) all depending on the flow dimensions.

$$\vec{K}(\vec{x}) = \begin{bmatrix} k_{xx} & k_{xy} & k_{xz} \\ k_{yx} & k_{yy} & k_{yz} \\ k_{zx} & k_{zy} & k_{zz} \end{bmatrix}. \tag{B-7}$$

The tensor $\vec{K}(\vec{x})$ is diagonal if the coordinates can be aligned with the geological layering. If $k_{xx} = k_{yy} = k_{zz}$ the medium is called isotropic.

A combination of Eq. (B-3) and Eq. (B-6) gives:

$$\nabla \cdot \left(\alpha \rho_i K \frac{k_{ri}}{\mu_i} \left(\nabla p_i - \rho_i g \nabla d \right) \right) + \rho_i q_i = \alpha \frac{\partial (\phi \rho_i) S_i}{\partial t}, \tag{B-8}$$

Appendix C

Non-Linear Properties of the used Reservoir Model

Studying the non-linearities of the reservoir may provide useful information to increase the optimization performance. In general, three types of PDEs exist, discussed in Section C-1. The non-linearities of the differential equations for pressure and saturation will be evaluated in Section C-2.

C-1 Types of Non-Linearities

Three types of non-linearities are known [Farlow, 1993]. The types are explained using a general second-order continuously differentiable PDE ($L(u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial t})$) with variable $u(x,t)$ [Courant et al., 1962]:

$$L(u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial t}) = A\frac{\partial^2 u}{\partial x^2} + B\frac{\partial^2 u}{\partial x \partial t} + C\frac{\partial^2 u}{\partial t^2} \tag{C-1}$$

The non-linearities are in this case caused by the second-order differentials. Eq. (C-1) is an analogy with the following quadratic equation [Peaceman, 1977]:

$$L(x,t) = Ax^2 + Bxt + Ct^2 = 0, \tag{C-2}$$

or

$$A(x/t)^2 + B(x/t) + C = 0. \tag{C-3}$$

The root to this equation is obtained by:

$$(x/t) = \frac{B \pm \sqrt{B^2 - 4AC}}{2A}. \tag{C-4}$$

The three types of behavior depend on the value of the discriminant $B^2 - 4AC$.

C-1-1 Elliptic Equation

Elliptic equations occur when $B2 - 4AC < 0$, which means there will be no real solution to Eq. (C-4). An example is the Laplace equation, where B is zero;

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0. \tag{C-5}$$

C-1-2 Parabolic Equation

When $B^2 - 4AC = 0$, there is only one real solution to Eq. (C-4). Considering the case where B and C are zero, this will result in a diffusive behavior in space as time propagates. An example is the Fourier equation of diffusion;

$$\frac{\partial^2 u}{\partial x^2} = \frac{\partial u}{\partial t}. \tag{C-6}$$

C-1-3 Hyperbolic Equation

A hyperbolic equation occurs when $B2 - 4AC > 0$ and Eq. (C-4) has two real solutions. An example is the wave equation:

$$\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial t^2} = 0. \tag{C-7}$$

However, it may be noticed more simplistic hyperbolic PDEs do exist as well. Such as the single-wave equation, which is comparable to the equation above with the second-order terms replaced with first-order terms.

C-2 Sources of Non-Linearities in the Pressure and Saturation Differential Equations

The reservoir model equations are non-linear. To get a better understanding on the type of non-linearities in the reservoir model, the simplified differential equations as given in Eq. (3-7) and Eq. (3-8) are further examined.

C-2-1 Pressure Differential Equation

Eq. (3-9) can be written as:

$$\nabla \cdot \left(\alpha \left(\vec{K} \frac{k_{rw}}{\mu_w} + \vec{K} \frac{k_{ro}}{\mu_o} \right) (\nabla p) \right) + q_w + q_o = \alpha \phi \left((s(c_w - c_o) + c_o + c_r) \frac{\partial p}{\partial t} \right). \tag{C-8}$$

When evaluating the differential equation for the pressure as given in Eq. (C-8), the equation can be further simplified assuming all variables except for the pressure as constant:

$$\nabla \cdot (\kappa_1 (\nabla p)) = \kappa_2 \frac{\partial p}{\partial t}. \tag{C-9}$$

Eq. (C-9) is a parabolic PDE as the discriminant is zero. However, currently, the capillary pressures are neglected. When they are present, though most of the time very small, the pressure differential equation will be elliptic [Peaceman, 1977]. This elliptic behavior is then caused by the second-order derivatives of the pressure with respect to time. The diffusive or near-elliptic behavior is plausible as the pressure is known to behave diffusively.

C-2-2 Saturation Differential Equation

The differential equation for the saturation is computed differently. Again assume one of the mass-balances;

$$\nabla \cdot \left(\alpha \vec{K} \frac{k_{rw}}{\mu_w} (\nabla p) \right) + q_w = \alpha \phi \left(s(c_w + c_r) \frac{\partial p}{\partial t} + \frac{\partial s}{\partial t} \right), \tag{C-10}$$

The Darcy velocity is used to eliminate the pressure on the left-hand side while the compressibilities for rock and liquid as given in Eq. (3-6) are used to eliminate the pressure differential on the right-hand side, resulting in:

$$\begin{aligned} \nabla \cdot (\alpha \vec{v}_w) + q_w = &\quad \alpha \phi \left(s \left(\frac{1}{\rho_w} \frac{\partial \rho_w}{\partial p} + \frac{1}{\phi} \frac{\partial \phi}{\partial p} \right) \frac{\partial p}{\partial t} + \frac{\partial s}{\partial t} \right) \\ = &\quad \alpha \phi \left(s \left(\frac{1}{\rho_w} \frac{\partial \rho_w}{\partial t} + \frac{1}{\phi} \frac{\partial \phi}{\partial t} \right) + \frac{\partial s}{\partial t} \right). \end{aligned} \tag{C-11}$$

In order to evaluate the differential equation only for the change in saturation, the porosity ϕ and fluid density ρ_w are assumed to be constant. Next, as the water velocity is unknown, it will be replaced by the total velocity $\vec{v}_t$ multiplied by the fractional flow of water f_w. The saturation differential equation is then:

$$\nabla \cdot (\alpha f_w \vec{v}_t) + q_w = \alpha \phi \left(\frac{\partial s}{\partial t} \right) \tag{C-12}$$

As the fractional flow is a function of the saturation, we can write [Peaceman, 1977]:

$$\nabla f_w = \frac{\delta f_w}{\delta s} \nabla s. \tag{C-13}$$

When looking explicitly at the flow within the reservoir it is possible to neglect the well flow rates. Next, implementation of Eq. (C-13) results in:

$$\frac{\delta f_w}{\delta s}\frac{\delta s}{\delta x}\nabla\left(\alpha\vec{v}_t\right) = \alpha\phi\left(\frac{\partial s}{\partial t}\right) - f_w q_t. \tag{C-14}$$

Above equation shows the nature of the saturation differential equation to be first-order hyperbolic when neglecting other influences as capillary pressures and assuming incompressible flow [Aziz and Settari, 1979]. When including the capillary pressures, the behavior may turn to be elliptic when they dominate, which is mostly not the case [Peaceman, 1977]. The hyperbolic nature of the equation results in a convective behavior of the saturation, describing the movement of the water front through the reservoir. The behavior of the saturation equation is highly non-linear [Aziz and Settari, 1979] compared to the pressure equation.

Appendix D

GAMS code and Matlab code for the 2x3 Reservoir Model

The model used for verification is based on a 2×3 finite difference model. The model consists of one injector and one producer and has a heterogeneous permeability field as shown in Figure 4-1.

Firstly, the files of the IG generator SimSim which needed to be edited will be given. These are edited to enable import data from GAMS and export data to GAMS. will be given as which are used to import and run GAMS data. Next, the two files used for data transfer between SimSim and GAMS and vice versa will be presented. As last, the GAMS code as used for model verification is given.

D-1 Edited Files from SimSim

Several m-files have been edited in order to simulate the GAMS data in SimSim, based on the flow-rates.

simsim_inp_six_blocks.m

```
77 % % Well locations and constraints:
78 W_p = [];
79 W_q = [1,  0.0059, inf;  % injector with prescribed rate
80        6, -0.0061, 0  ]; % producer with prescribed rate
81 r_w = 4.5 * 0.0254; % well bore radius, m
```

```
132 if intyp == 4
133
```

```
134     load ../RM_06_Output.mat
135
136     kk = 1;
137
138     injection  = q_w(1,1,kk);
139     production = q_w(2,3,kk) + q_o(2,3,kk);
140
141     p_well_inj  = p_well(1,1,kk);
142     p_well_prod = p_well(2,3,kk);
143
144     % Rate Contrained
145     W_p = [];
146     W_q = [1,  injection,  inf;  % injector with prescribed rate and
           no pressure constraint
147            6,  production, 0  ]; % producer with prescribed rate and
                  no pressure constraint
148
149
150 end
```

simsim_forward.m

```
29 if intyp == 4
30
31     load ../RM_06_Output.mat
32     kk = 1;
33     Delta_t = dt(kk);
34
35 end
```

```
50     case 4 % implicit Euler with Newton iteration (variable time step)
51
52         [E,M,stats,t_x,t_y,X,Y] =       simsim_Eul_Newton_GAMS(comp,
               con,C,Delta_t,geo,intpar,...
53              intyp,I_q,j_con,k,L_uq,mu,phi,rel,solpar,t_c,well,x_0);
```

simsim_Eul_Newton_GAMS.m

In the file simsim_forward.m is referred to simsim_Eul_Newton_GAMS.m, this file is the edited version of simsim_Eul_Newton.m file with replacing the code at the given line-numbers.

```
60 load ../RM_06_Output.mat
61
62 while kk < length(q_w(1,1,:))
```

```
63    % Determine time step:
64    if kk > 1
65        % Determine time step:
66        Delta_t = dt(kk); % computes new time step
67        if t_kk + Delta_t > t_end
68            Delta_t = t_end - t_kk;
69        end
70    end
71
72     % write info to screen every kk_info steps
73    kk_info = 5;
74    if round (kk/kk_info) == kk/kk_info
75        kk
76        t_d = t_kk/(3600*24)
77        Delta_t_d = Delta_t/(3600*24)
78    end
79
80    injection  = q_w(1,1,kk+1);
81    production = q_w(2,3,kk+1) + q_o(2,3,kk+1);
82
83    % Rate Contrained
84    well.W_p = [];
85    well.W_q = [1,  injection,  inf;  % injector with prescribed rate
86                6,  production, 0  ]; % producer with prescribed rate
87
88    Delta_x = 500; % grid block length, m
89    Delta_y = 500; % grid block width, m
90    n_x = 3; % number of grid blocks in x-direction, -
91    well.W_xy = simsim_aux_well(Delta_x,Delta_y,n_x,well.W_p,well.W_q);
```

D-2 Matlab Code for Communication between SimSim and GAMS

Data is transferred from SimSim to GAMS for initiating the initial guesses after running RM_06_ObtainInitialGuesses.m. Then, after running GAMS, data is transferred back to MatLab by RM_06_ReadGAMSOutput.m.

RM_06_ObtainInitialGuesses.m

```
1  %% PARAMETERS
2
3  i = model.geo.n_y;
4  j = model.geo.n_x;
5  t = length(results.t_y);
6
7  label_i = cell(1,i);
8  label_j = cell(1,j);
9  label_t = cell(1,t);
10
```

```
for ii = 1:i
    label_i(ii) = { strcat('I' , num2str(ii)) };
end
for jj = 1:j
    label_j(jj) = { strcat('J' , num2str(jj)) };
end
for tt = 1:t
    label_t(tt) = { strcat('T' , num2str(tt)) };
end

dt = diff(results.t_y)
dt(t) = dt(t-1);

%% PRODUCTION & INJECTION VARIABLES

q_w = zeros(i,j,t);
q_w(1,1,1:end) = results.Y(5,:); q_w(2,3,1:end) = results.Y(6,:);
q_o = zeros(i,j,t);
q_o(1,1,1:end) = results.Y(3,:); q_o(2,3,1:end) = results.Y(4,:);

p_well = zeros(i,j,t);
p_well(1,1,1:end) = results.Y(1,:);
p_well(2,3,1:end) = results.Y(2,:);

%% STATE VARIABLES
p = zeros(i,j,t);
s = zeros(i,j,t);

for ii = 1:i
    for jj = 1:j
        p(ii,jj,:) = results.X( j*(ii-1) + jj ,:);
        s(ii,jj,:) = results.X( j*i+ j*(ii-1) + jj ,:);

    end
end

%%%%%%%%%% FOR CHECK
dpdt = zeros(i,j,t-1);
dsdt = zeros(i,j,t-1);

for ii = 1:i
    for jj = 1:j
        dpdt(ii,jj,:) = diff(p(ii,jj,:));
        dsdt(ii,jj,:) = diff(s(ii,jj,:));

    end
end

for tt = 1:t-1
    dpdt(:,:,tt) = dpdt(:,:,tt) / dt(tt);
    dsdt(:,:,tt) = dsdt(:,:,tt) / dt(tt);
end

```

```
64 %% COUNTER VARIABLE
65 time = results.t_x;
66
67 %% WELL VARIABLES
68
69 L_q_w.name = 'set_q_w';
70 L_q_w.val = q_w;
71 L_q_w.labels = {label_i label_j label_t};
72
73 L_q_o.name = 'set_q_o';
74 L_q_o.val = q_o;
75 L_q_o.labels = {label_i label_j label_t};
76
77 L_p_well.name = 'set_p_well';
78 L_p_well.val = p_well;
79 L_p_well.labels = {label_i label_j label_t};
80
81 %% STATE VARIABLES
82 L_p.name = 'set_p';
83 L_p.val = p;
84 L_p.labels = {label_i label_j label_t};
85
86 L_s.name = 'set_s';
87 L_s.val = s;
88 L_s.labels = {label_i label_j label_t};
89
90 %% TIMER VARIABLE
91 L_dt.name = 'set_dt';
92 L_dt.val = dt;
93 L_dt.labels = {label_t};
94
95 L_time.name = 'set_time';
96 L_time.val = time;
97 L_time.labels = {label_t};
98
99 gams('', L_q_w,L_q_o,L_p,L_s,L_p_well,L_time,L_dt);
```

RM_06_ReadGAMSOutput.m

```
1 %% read output
2 [q_w,q_o,p_well,p,s,time,dt] = gams();
3
4 q_w = q_w.val;
5 q_o = q_o.val;
6 p_well = p_well.val;
7 p = p.val;
8 s = s.val;
9 time = time.val;
10 dt = dt.val;
11
```

```
file_name_out = 'Output.mat';
save(file_name_out,'q_w','q_o','p_well','p','s','time','dt');
```

D-3 GAMS Code

The GAMS model is given below. The relative permeabilities are calculated by using the input-date from SimSim. The if-signs used in GAMS are dollar signs, however, due to problems with LaTeX changed to double dollar signs.

GAMS.gms

```
*** GAMS code of the Reservoir Model by Lodewijk Alblas

SETS
*****************************************************
*** SETS is used to define the spce and time grid

i         row index of model cells                    / I1 *   I2 /
j         column index of model cells                 / J1 *   J3 /
t         time index                                  / T1 * T123 /

;

SCALARS
*****************************************************

dx       grid size x (m)                             /500/
dy       grid size y (m)                             /500/
h        grid size height (m)                        / 20/
co       compressibility (1:Pa) - oil                /  1.000E-8/
cw       compressibility (1:Pa) - water              /  1.000E-8/
cr       compressibility (1:Pa) - rock               /  1.000E-8/

no       Corey exponent (-) - oil                    /  2/
nw       Corey exponent (-) - water                  /  2/

Sor      Residual Oil Saturation (-)                 /  0.2/
Swc      Connate Water Saturation (-)                /  0.2/

muo      viscosity (Pa s) - oil                      /  0.500E-3/
muw      viscosity (Pa s) - water                   /   1.000E-3/
rhoo     density (kg : m^3) - oil                    /850/
rhow     density (kg : m^3) - water                 /1000/

kro0     end point relative permeability (-) - oil       /0.9/
krw0     end point relative permeability (-) - water     /0.6/

rwell    well bore radius for all wells (m)        /0.1143/
```

```
pi      pi                                              /3.141592653589793/

rwinj   cost of injecting water (dollar per m^3)        /  10 /
rwprod  cost of producing water (dollar per m^3)        /  30 /
roprod  profit of producing oil (dollar per m^3)        / 300 /
b       discount rate of money (0 - 1) (-)              /   0.10 /
tau     reference time for discount rate 1 yr (s) / 31536000 /

setvarphi
;

PARAMETER WellType(i,j);
*****************************************************
*** (1 = injection | 0 = no well | -1 = production)

WellType(i,j) = 0;
WellType('I1','J1') =   1;
WellType('I2','J3') =  -1;

PARAMETERS
k_xx(i,j)               grid permeabilities x-direction (m^2)
k_yy(i,j)               grid permeabilities y-direction (m^2)
phi(i,j)                porosity vector (-)
J_con(i,j)              Well-term constants
dt(t)                   time-step

set_q_w(i,j,t)          Set initial guess water flow using Matlab
set_q_o(i,j,t)          Set initial guess oil flow using Matlab
set_p_well(i,j,t)       Set initial guess well pressure using Matlab
set_p(i,j,t)            Set initial guess pressures using Matlab
set_s(i,j,t)            Set initial guess saturations using Matlab
set_time(t)             Set time using Matlab
set_dt(t)               Set timesteps using Matlab

;

TABLE k(i,j)             grid permeabilities  (m^2)

         J1       J2       J3
I1    1e-12    1e-12    1e-14
I2    1e-13    1e-12    1e-13
;

kxx(i,j)                        = k(i,j);
kyy(i,j)                        = k(i,j);
phi(i,j)                         =    0.3;


Jcon(i,j)$$(WellType(i,j) ne 0)  =
          (2*pi*k(i,j)*h) / (log(0.14* sqrt(dx*dx + dy*dy) / rwell ) );
dt(t) = 1;

```

```
VARIABLES
*****************************************************

*** WELL VARIABLES
qt(i,j,t)
qw(i,j,t)
qo(i,j,t)
pwell(i,j,t)            Well pressure

*** STATE VARIABLES
p(i,j,t)                Pressure (Pa)
s(i,j,t)                Saturation (-)

dpdt(i,j,t)             Pressure Differential Equation (Pa : s)
dsdt(i,j,t)             Saturation Differential Equation (- : s)

*** RELATIVE PERMEABILITIES
Swn(i,j,t)              Normalized Water Saturation
krw(i,j,t)              Relative Permeability Water
kro(i,j,t)              Relative Permeability Oil
fw(i,j,t)               Fractional Flow Water

*** TRANSMISSIBILITY VARIABLES
Tw1(i,j,t)              Transmissibility of water to grid-block i+1
Tw2(i,j,t)              Transmissibility of water to grid-block i-1
Tw3(i,j,t)              Transmissibility of water to grid-block j+1
Tw4(i,j,t)              Transmissibility of water to grid-block j-1
To1(i,j,t)              Transmissibility of oil   to grid-block i+1
To2(i,j,t)              Transmissibility of oil   to grid-block i-1
To3(i,j,t)              Transmissibility of oil   to grid-block j+1
To4(i,j,t)              Transmissibility of oil   to grid-block j-1

*** TOTAL FLOW RATES
Uw(i,j,t)               Total water flow rate into a grid-block
Uo(i,j,t)               Total oil flow rate into a grid-block

*** OBJECTIVE VARIABLE(S)
varphi                   Optimization Variable

*** TIME VARIABLE
time(t)                  Cumulative time

;

*****************************************************
*** BOUNDARIES
*****************************************************
qw.up(i,j,t)$$(WellType(i,j) eq -1) = 0;
qw.lo(i,j,t)$$(WellType(i,j) eq  1) = 0;
qo.up(i,j,t)$$(WellType(i,j) eq -1) = 0;
qo.fx(i,j,t)$$(WellType(i,j) ne -1) = 0;
```

```
144 pwell.lo(i,j,t)$$(WellType(i,j) eq -1) = 200e5;
145 pwell.up(i,j,t)$$(WellType(i,j) eq -1) = 300e5;
146
147 pwell.lo(i,j,t)$$(WellType(i,j) eq  1) = 300e5;
148 pwell.up(i,j,t)$$(WellType(i,j) eq  1) = 350e5;
149
150 Swn.lo(i,j,t) = 0;
151 Swn.up(i,j,t) = 1;
152
153
154 *** INITIAL VALUES
155 p.fx(i,j,'T1') = 300e5;
156 s.fx(i,j,'T1') = 0.2;
157 time.fx('T1') = 0;
158
159 ****************************************************
160 *** INITUAL GUESSES FROM MATLAB
161 ****************************************************
162 $$if exist matdata.gms $$include matdata.gms
163
164 p.l(i,j,t) = setp(i,j,t);
165 s.l(i,j,t) = sets(i,j,t);
166
167 qw.l(i,j,t) = setqw(i,j,t);
168 qo.l(i,j,t) = setqo(i,j,t);
169
170 pwell.l(i,j,t) = setpwell(i,j,t);
171
172 time.l(t) = settime(t);
173 dt(t) = setdt(t);
174
175
176 DISPLAY setqw,qw.l,setqo,qo.l,setp,p.l,sets,s.l,setpwell,pwell.l;
177 DISPLAY settime,time.l,setdt,dt,Jcon;
178
179 *** Bound saturations to avoid undefined initial guesses
180 s.l(i,j,t)$$(s.l(i,j,t) < 0.2) = 0.2;
181 s.l(i,j,t)$$(s.l(i,j,t) > 0.8) = 0.8;
182
183 Swn.l(i,j,t) = (s.l(i,j,t)-Swc)/(1-Sor-Swc);
184 Swn.l(i,j,t)$$(Swn.l(i,j,t) < 0) = 0;
185 Swn.l(i,j,t)$$(Swn.l(i,j,t) > 1) = 1;
186
187 krw.l(i,j,t) = krw0 * (  Swn.l(i,j,t))**nw;
188 kro.l(i,j,t) = kro0 * (1-Swn.l(i,j,t))**no;
189
190
191 qt.l(i,j,t) = qw.l(i,j,t) + qo.l(i,j,t);
192 fw.l(i,j,t)$$(WellType(i,j) eq -1) = krw.l(i,j,t) / (krw.l(i,j,t) +
       kro.l(i,j,t) * (muw/muo));
193
194
195 *** Transmissibilities including upstream weighting of the pressures
```

```
196 Tw1.l(i,j,t) = ( (2*h/muw) * (dy/dx) * (kxx(i+1,j)*kxx(i,j))
197                  / (kxx(i+1,j)+kxx(i,j))
198                  * (  krw.l(i,j,t)$$(setp(i,j,t)  ge setp(i+1,j,t))
199                     + krw.l(i+1,j,t)$$(setp(i,j,t) < setp(i+1,j,t)) )
200               )$$(ord(i) ne card(i));
201
202 Tw2.l(i,j,t) = ( (2*h/muw) * (dy/dx) * (kxx(i-1,j)*kxx(i,j))
203                  / (kxx(i-1,j)+kxx(i,j))
204                  * (  krw.l(i,j,t)$$(setp(i,j,t)  ge setp(i-1,j,t))
205                     + krw.l(i-1,j,t)$$(setp(i,j,t) < setp(i-1,j,t)) )
206               )$$(ord(i) ne      1 );
207
208 Tw3.l(i,j,t) = ( (2*h/muw) * (dx/dy) * (kyy(i,j+1)*kyy(i,j))
209                  / (kyy(i,j+1)+kyy(i,j))
210                  * (  krw.l(i,j,t)$$(setp(i,j,t)  ge setp(i,j+1,t))
211                     + krw.l(i,j+1,t)$$(setp(i,j,t) < setp(i,j+1,t)) )
212               )$$(ord(j) ne card(j));
213
214 Tw4.l(i,j,t) = ( (2*h/muw) * (dx/dy) * (kyy(i,j-1)*kyy(i,j))
215                  / (kyy(i,j-1)+kyy(i,j))
216                  * (  krw.l(i,j,t)$$(setp(i,j,t)  ge setp(i,j-1,t))
217                     + krw.l(i,j-1,t)$$(setp(i,j,t) < setp(i,j-1,t)) )
218               )$$(ord(j) ne      1 );
219
220 To1.l(i,j,t) = ( (2*h/muo) * (dy/dx) * (kxx(i+1,j)*kxx(i,j))
221                  / (kxx(i+1,j)+kxx(i,j))
222                  * (  kro.l(i,j,t)$$(setp(i,j,t)  ge setp(i+1,j,t))
223                     + kro.l(i+1,j,t)$$(setp(i,j,t) < setp(i+1,j,t)) )
224               )$$(ord(i) ne card(i));
225
226 To2.l(i,j,t) = ( (2*h/muo) * (dy/dx) * (kxx(i-1,j)*kxx(i,j))
227                  / (kxx(i-1,j)+kxx(i,j))
228                  * (  kro.l(i,j,t)$$(setp(i,j,t)  ge setp(i-1,j,t))
229                     + kro.l(i-1,j,t)$$(setp(i,j,t) < setp(i-1,j,t)) )
230               )$$(ord(i) ne      1 );
231
232 To3.l(i,j,t) = ( (2*h/muo) * (dx/dy) * (kyy(i,j+1)*kyy(i,j))
233                  / (kyy(i,j+1)+kyy(i,j))
234                  * (  kro.l(i,j,t)$$(setp(i,j,t)  ge setp(i,j+1,t))
235                     + kro.l(i,j+1,t)$$(setp(i,j,t) < setp(i,j+1,t)) )
236               )$$(ord(j) ne card(j));
237 To4.l(i,j,t) = ( (2*h/muo) * (dx/dy) * (kyy(i,j-1)*kyy(i,j))
238                  / (kyy(i,j-1)+kyy(i,j))
239                  * (  kro.l(i,j,t)$$(setp(i,j,t)  ge setp(i,j-1,t))
240                     + kro.l(i,j-1,t)$$(setp(i,j,t) < setp(i,j-1,t)) )
241               )$$(ord(j) ne      1 );
242
243
244 *** Total water and oil flows
245 Uw.l(i,j,t) =
246    ( Tw1.l(i,j,t) * (p.l(i+1,j,t)-p.l(i,j,t)))$$(ord(i) ne card(i)  )
247  + ( Tw2.l(i,j,t) * (p.l(i-1,j,t)-p.l(i,j,t)))$$(ord(i) ne      1   )
248  + ( Tw3.l(i,j,t) * (p.l(i,j+1,t)-p.l(i,j,t)))$$(ord(j) ne card(j)  )
```

```
 + ( Tw4.l(i,j,t) * (p.l(i,j-1,t)-p.l(i,j,t)))$$(ord(j) ne      1   )
 + ( qw.l(i,j,t)                             )$$(WellType(i,j) ne  0);

Uo.l(i,j,t) =
   ( To1.l(i,j,t) * (p.l(i+1,j,t)-p.l(i,j,t)))$$(ord(i) ne card(i)  )
 + ( To2.l(i,j,t) * (p.l(i-1,j,t)-p.l(i,j,t)))$$(ord(i) ne      1   )
 + ( To3.l(i,j,t) * (p.l(i,j+1,t)-p.l(i,j,t)))$$(ord(j) ne card(j)  )
 + ( To4.l(i,j,t) * (p.l(i,j-1,t)-p.l(i,j,t)))$$(ord(j) ne      1   )
 + ( qo.l(i,j,t)                             )$$(WellType(i,j) eq -1);

*** Initialize ODE
dpdt.l(i,j,t) =
   (Uw.l(i,j,t)                          + Uo.l(i,j,t)                  )
 / (h*dx*dy * phi(i,j) * (s.l(i,j,t)*(cw-co) + cr + co));
dsdt.l(i,j,t) =
   (Uw.l(i,j,t)*(1-s.l(i,j,t))*(co+cr)-Uo.l(i,j,t)*s.l(i,j,t)*(cw+cr))
 / (h*dx*dy * phi(i,j) * (s.l(i,j,t)*(cw-co) + cr + co));


varphi.l   = - sum( (i,j,t)$$(WellType(i,j) eq  1),
                    qw.l(i,j,t)*rwinj *dt(t)/(1+b)**(time.l(t)/tau))
             + sum( (i,j,t)$$(WellType(i,j) eq -1),
                    qw.l(i,j,t)*rwprod*dt(t)/(1+b)**(time.l(t)/tau))
             - sum( (i,j,t)$$(WellType(i,j) eq -1),
                    qo.l(i,j,t)*roprod*dt(t)/(1+b)**(time.l(t)/tau));

EQUATIONS
*****************************************************

*** Well Reate
TotalWellRate(i,j,t)
FractionalFlowWater(i,j,t)
RelationWaterTotalRate(i,j,t)

*** RELATIVE PERMEABILITIES
SNormalized(i,j,t)
RelPermw(i,j,t)
RelPermo(i,j,t)

*** TRANSMISSIBILITY FUNCTIONS
TotalRateWater(i,j,t)
TotalRateOil(i,j,t)
EQTw1(i,j,t)
EQTw2(i,j,t)
EQTw3(i,j,t)
EQTw4(i,j,t)
EQTo1(i,j,t)
EQTo2(i,j,t)
EQTo3(i,j,t)
EQTo4(i,j,t)

*** STATE EQUATIONS
ODEp(i,j,t)
```

```
ODEs(i,j,t)

ImplicitEulerp(i,j,t)
ImplicitEulers(i,j,t)

*** WELL MODELS
Peaceman(i,j,t)

*** TIME COUNTER
Counter(t)

*** OBJECTIVE FUNCTION(S)
ObjectiveFunction

;
SNormalized(i,j,t)..         Swn(i,j,t) =e= (s(i,j,t)-Swc)/(1-Sor-Swc);
RelPermw(i,j,t)..            krw(i,j,t) =e= krw0 * (  Swn(i,j,t))**nw;
RelPermo(i,j,t)..            kro(i,j,t) =e= kro0 * (1-Swn(i,j,t))**no;

TotalWellRate(i,j,t)$$(WellType(i,j) ne 0)..
     qt(i,j,t) =e= qw(i,j,t) + qo(i,j,t);
FractionalFlowWater(i,j,t)$$(WellType(i,j) eq -1)..
     fw(i,j,t) =e= krw(i,j,t) / (krw(i,j,t) + kro(i,j,t) * (muw/muo));
RelationWaterTotalRate(i,j,t)$$(WellType(i,j) eq -1)..
     qw(i,j,t) =e= fw(i,j,t) * qt(i,j,t);


EQTw1(i,j,t)..
  Tw1(i,j,t)=e= ( (2*h/muw)*(dy/dx)*(kxx(i+1,j)*kxx(i,j))
                  / (kxx(i+1,j)+kxx(i,j))
                  * (  krw(i,j,t)$$(setp(i,j,t)  ge setp(i+1,j,t))
                     + krw(i+1,j,t)$$(setp(i,j,t) < setp(i+1,j,t)) )
          )$$(ord(i) ne card(i));

EQTw2(i,j,t)..
  Tw2(i,j,t) =e= ( (2*h/muw) * (dy/dx) * (kxx(i-1,j)*kxx(i,j))
                   / (kxx(i-1,j)+kxx(i,j))
                   * (  krw(i,j,t)$$(setp(i,j,t)  ge setp(i-1,j,t))
                      + krw(i-1,j,t)$$(setp(i,j,t) < setp(i-1,j,t)) )
                 )$$(ord(i) ne      1 );
EQTw3(i,j,t)..
  Tw3(i,j,t) =e= ( (2*h/muw) * (dx/dy) * (kyy(i,j+1)*kyy(i,j))
                   / (kyy(i,j+1)+kyy(i,j))
                   * (  krw(i,j,t)$$(setp(i,j,t)  ge setp(i,j+1,t))
                      + krw(i,j+1,t)$$(setp(i,j,t) < setp(i,j+1,t)) )
                 )$$(ord(j) ne card(j));
EQTw4(i,j,t)..
  Tw4(i,j,t) =e= ( (2*h/muw) * (dx/dy) * (kyy(i,j-1)*kyy(i,j))
                   / (kyy(i,j-1)+kyy(i,j))
                   * (  krw(i,j,t)$$(setp(i,j,t)  ge setp(i,j-1,t))
                   + krw(i,j-1,t)$$(setp(i,j,t) < setp(i,j-1,t)) )
                 )$$(ord(j) ne      1 );
EQTo1(i,j,t)..
```

```
  To1(i,j,t) =e= ( (2*h/muo) * (dy/dx) * (kxx(i+1,j)*kxx(i,j))
                   / (kxx(i+1,j)+kxx(i,j))
                   * (  kro(i,j,t)$$(setp(i,j,t)  ge setp(i+1,j,t))
                      + kro(i+1,j,t)$$(setp(i,j,t) < setp(i+1,j,t)) )
                 )$$(ord(i) ne card(i));
EQTo2(i,j,t)..
 To2(i,j,t) =e= ( (2*h/muo) * (dy/dx) * (kxx(i-1,j)*kxx(i,j))
                  / (kxx(i-1,j)+kxx(i,j))
                  * (  kro(i,j,t)$$(setp(i,j,t)  ge setp(i-1,j,t))
                     + kro(i-1,j,t)$$(setp(i,j,t) < setp(i-1,j,t)) )
                )$$(ord(i) ne      1 );
EQTo3(i,j,t)..
 To3(i,j,t) =e= ( (2*h/muo) * (dx/dy) * (kyy(i,j+1)*kyy(i,j))
                  / (kyy(i,j+1)+kyy(i,j))
                  * (  kro(i,j,t)$$(setp(i,j,t)  ge setp(i,j+1,t))
                     + kro(i,j+1,t)$$(setp(i,j,t) < setp(i,j+1,t)) )
                )$$(ord(j) ne card(j));
EQTo4(i,j,t)..
  To4(i,j,t) =e= ( (2*h/muo) * (dx/dy) * (kyy(i,j-1)*kyy(i,j))
                   / (kyy(i,j-1)+kyy(i,j))
                   * (  kro(i,j,t)$$(setp(i,j,t)  ge setp(i,j-1,t))
                      + kro(i,j-1,t)$$(setp(i,j,t) < setp(i,j-1,t)) )
                 )$$(ord(j) ne      1 );


TotalRateWater(i,j,t)..
 Uw(i,j,t) =e= (Tw1(i,j,t)*(p(i+1,j,t)-p(i,j,t)))$$(ord(i) ne card(i))
             + (Tw2(i,j,t)*(p(i-1,j,t)-p(i,j,t)))$$(ord(i) ne      1 )
             + (Tw3(i,j,t)*(p(i,j+1,t)-p(i,j,t)))$$(ord(j) ne card(j))
             + (Tw4(i,j,t)*(p(i,j-1,t)-p(i,j,t)))$$(ord(j) ne      1 )
             + (qw(i,j,t))$$(WellType(i,j) ne 0 );

TotalRateOil(i,j,t)..
 Uo(i,j,t) =e= (To1(i,j,t)*(p(i+1,j,t)-p(i,j,t)))$$(ord(i) ne card(i))
             + (To2(i,j,t)*(p(i-1,j,t)-p(i,j,t)))$$(ord(i) ne      1 )
             + (To3(i,j,t)*(p(i,j+1,t)-p(i,j,t)))$$(ord(j) ne card(j))
             + (To4(i,j,t)*(p(i,j-1,t)-p(i,j,t)))$$(ord(j) ne      1 )
             + (qo(i,j,t))$$(WellType(i,j) eq -1);


ODEp(i,j,t).. dpdt(i,j,t) =e=
           (Uw(i,j,t)                      +Uo(i,j,t)                  )
           / ( h*dx*dy * phi(i,j) * (s(i,j,t)*(cw-co) + cr + co) );
ODEs(i,j,t).. dsdt(i,j,t) =e=
           (Uw(i,j,t)*(1-s(i,j,t))*(co+cr)-Uo(i,j,t)*s(i,j,t)*(cw+cr))
           / ( h*dx*dy * phi(i,j) * (s(i,j,t)*(cw-co) + cr + co) );



ImplicitEulerp(i,j,t)$$(ord(t) ne card(t))..
                      p(i,j,t+1) =e= p(i,j,t) + dt(t) * dpdt(i,j,t+1);
ImplicitEulers(i,j,t)$$(ord(t) ne card(t))..
                      s(i,j,t+1) =e= s(i,j,t) + dt(t) * dsdt(i,j,t+1);
```

```
408
409
410 Peaceman(i,j,t)$$(WellType(i,j) ne 0)..
411                   (p(i,j,t) - pwell(i,j,t)) * Jcon(i,j) =e=
412                   -(1/ (kro(i,j,t)/muo + krw(i,j,t)/muw)) * qt(i,j,t);
413
414
415 Counter(t)$$(ord(t) ne 1)..        time(t)  =e= time(t-1) + dt(t-1);
416
417 ObjectiveFunction..
418  varphi =e= - sum( (i,j,t)$$(WellType(i,j) eq  1) ,
419                    qw(i,j,t)* rwinj*dt(t) / (1+b)**(time(t)/tau) )
420             + sum( (i,j,t)$$(WellType(i,j) eq -1) ,
421                    qw(i,j,t)*rwprod*dt(t) / (1+b)**(time(t)/tau) )
422             - sum( (i,j,t)$$(WellType(i,j) eq -1) ,
423                    qo(i,j,t)*roprod*dt(t) / (1+b)**(time(t)/tau) );
424
425
426
427 ******************************************************
428 *** SOLVE THE MODEL
429 ******************************************************
430
431 MODEL reservoirmodel /all/;
432 reservoirmodel.scaleopt = 1;
433 reservoirmodel.optfile = 1;
434 *** Limit number of written out equations.
435 Option limrow = 1000;
436 reservoirmodel.reslim = 10000 ;
437
438
439 Option NLP = CONOPT;
440 SOLVE reservoirmodel using nlp maximizing varphi;
441
442 DISPLAY  varphi.l ;
443
444 ******************************************************
445 *** OUTPUT TO MATLAB
446 ******************************************************
447
448
449 $$libinclude matout qw.l i j t
450 $$libinclude matout qo.l i j t
451 $$libinclude matout pwell.l i j t
452 $$libinclude matout p.l i j t
453 $$libinclude matout s.l i j t
454 $$libinclude matout time.l t
455 $$libinclude matout dt t
```

CONOPT.opt

The option file for CONOPT used to enable scaling down of the model equations with a factor of maximum $1e{-}7$. Upscaling is standard incorporated in the algorithm with a maximum of $1e7$.

```
1 RTMINS=1E-7
```

Bibliography

[Aziz and Settari, 1979] Aziz, K. and Settari, A. (1979). *Petroleum Reservoir Simulation*. ISBN 0-85334-787-5. Elsevier Applied Science Publishers.

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

[Biegler and Grossmann, 2004] Biegler, L. T. and Grossmann, I. E. (2004). Retrospective on optimization. *Computers & Chemical Engineering*, 28:1169–1192.

[Binder et al., 2001] Binder, T., Blank, L., Bock, H. G., Bulirsch, R., Dahmen, W., Diehl, M., Kronseder, T., Marquardt, W., Schil Joder, J., and von Stryk, O. (2001). Introduction to model based optimization of chemical processes on moving horizons. *Online Optimization of Large Scale Systems*, 1:295–339.

[Brouwer and Jansen, 2002] Brouwer, D. R. and Jansen, J. D. (2002). Dynamic optimization of water flooding with smart wells using optimal control theory. In *European Petroleum Conference, Aberdeen, U.K., 28-29 Januray 2002*.

[Brouwer and Jansen, 2004] Brouwer, D. R. and Jansen, J. D. (2004). Dynamic optimization of waterflooding with smart wells using optimal control theory. *SPE Journal*, 9:391–402.

[Byrd et al., 2006] Byrd, R. H., Nocedal, J., and Waltz, R. A. (2006). Knitro: An integrated package for nonlinear optimization. In *Large-Scale Nonlinear Optimization*, volume 83 of *Nonconvex Optimization and Its Applications*, pages 35–59. Springer US.

[Corey, 1954] Corey, A. T. (1954). The interrelation between gas and oil relative permeabilities. *Producers Monthly*, 19:38–41.

[Courant et al., 1962] Courant, R., Hilbert, D., and Teichmann, T. (1962). Methods of mathematical physics. *Physics Today*, 15:62–63.

[Cuthrell and Biegler, 1987] Cuthrell, J. E. and Biegler, L. T. (1987). On the optimization of differential-algebraic process systems. *AIChE Journal*, 33:1257–1270.

[Diehl et al., 2006] Diehl, M., Bock, H. G., Diedam, H., and Wieber, P. B. (2006). *Fast Motions in Biomechanics and Robotics*. Springer Berlin / Heidelberg.

[Drud, 1985] Drud, A. J. (1985). Conopt: A grg code for large sparse dynamic nonlinear optimization problems. *Mathematical Programming*, 31:153–191.

[Drud, 1994] Drud, A. J. (1994). Conopt - a large-scale grg code. *ORSA Journal on Computing*, 6:207–216.

[Farlow, 1993] Farlow, S. J. (1993). *Partial Differential Equations for Scientists and Engineers*. Dover Publications.

[Gill et al., 2002] Gill, P. E., Murray, W., and Saunders, M. A. (2002). Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM Journal on Optimization*, 12:979–1006.

[Gould et al., 2005] Gould, N., Orban, D., and Toint, P. (2005). Numerical methods for large-scale nonlinear optimization. *Acta Numerica*, 14:299–361.

[Huesman et al., 2008] Huesman, A. E. M., Bosgra, O. H., , and Van den Hof, P. M. J. (2008). Integrating mpc and rto in the process industry by economic dynamic lexicographic optimization; an open-loop exploration. In *AIChE Annual Meeting, Philadelphia, Pennsylvania, U.S.A., 16-21 November 2008*.

[Huntington and Rao, 2008] Huntington, G. and Rao, A. (2008). Comparison of global and local collocation methods for optimal control. *Journal of Guidance, Control, and Dynamics*, 31:432–436.

[Jacobson et al., 1970] Jacobson, D., Gershwin, S., and Lele, M. (1970). Computation of optimal singular controls. *IEEE Transactions on Automatic Control*, 15:67–73.

[Jansen, 2009] Jansen, J. D. (2009). Lecture notes; systems theory for reservoir management. Version 5e, September 2009.

[Jansen et al., 2008] Jansen, J. D., Bosgra, O. H., and Van den Hof, P. M. J. (2008). Model-based control of multiphase flow in subsurface oil reservoirs. *Journal of Process Control*, 18:846–855.

[Jansen et al., 2009] Jansen, J. D., Douma, S. D., Brouwer, D. R., Van den Hof, P. M. J., Bosgra, O. H., and Heemink, A. W. (2009). Closed loop reservoir management. In *SPE Reservoir Simulation Symposium, The Woodlands, Texas, U.S.A., 2-4 February 2009*.

[Kameswaran and Biegler, 2006] Kameswaran, S. and Biegler, L. T. (2006). Simultaneous dynamic optimization strategies: Recent advances and challenges. *Computers & Chemical Engineering*, 30:1560–1575.

[Kraaijevanger et al., 2007] Kraaijevanger, J. F. B. M., Egberts, P. J. P., Valstar, J. R., and Buurman, H. W. (2007). Optimal waterflood design using the adjoint method. In *SPE Reservoir Simulation Symposium, Houston, Texas, U.S.A., 26-28 February 2007*.

[Mittelmann, 2010] Mittelmann, H. D. (2010). Ampl-nlp benchmark of 15 november 2010 . http://plato.asu.edu/ftp/ampl-nlp.html.

[Nocedal and Wright, 1999] Nocedal, J. and Wright, S. J. (1999). *Numerical Optimization*. ISBN 0-387-98793-2. Springer.

[Peaceman, 1977] Peaceman, D. W. (1977). *Fundamentals of Numerical Reservoir Simulation*. ISBN 0-444-41578-5. Elsevier Scientific Publishing Company.

[Peaceman, 1983] Peaceman, D. W. (1983). Interpretation of well-block pressures in numerical reservoir simulation with nonsquare grid blocks and anisotropic permeability. *SPE Journal*, 23:531–543.

[Petzold et al., 2006] Petzold, L., Li, S., Cao, Y., and Serban, R. (2006). Sensitivity analysis of differential-algebraic equations and partial differential equations. *Computers & Chemical Engineering*, 30:1553–1559.

[Poku and Biegler, 2004] Poku, M. Y. B. and Biegler, L. T. (2004). Nonlinear optimization with many degrees of freedom in process engineering. *Industrial and Engineering Chemistry Research*, 43:6803–6812.

[Qin and Badgwell, 2003] Qin, S. J. and Badgwell, T. A. (2003). A survey of industrial model predictive control technology. *Control Engineering Practice*, 11:733–764.

[Sarma et al., 2005] Sarma, P., Aziz, K., and Durlofsky, L. J. (2005). Implementation of adjoint solution for optimal control of smart wells. In *SPE Reservoir Simulation Symposium, Houston, Texas, U.S.A., 31 January - 2 February 2005*.

[Sarma et al., 2006] Sarma, P., Chen, W. H., Durlofsky, L. J., and Aziz, K. (2006). Production optimization with adjoint models under nonlinear control-state path inequality constraints. In *Intelligent Energy Conference and Exhibition, Amsterdam, The Netherlands, 11-13 April 2006*.

[Støren and Hertzberg, 1999] Støren, S. and Hertzberg, T. (1999). Obtaining sensitivity information in dynamic optimization problems solved by the sequential approach. *Computers & Chemical Engineering*, 23:807–819.

[Van den Hof et al., 2009] Van den Hof, P. M. J., Jansen, J. D., van Essen, G. M., and Bosgra, O. H. (2009). Model-based control and optimization of large scale physical systems - challenges in reservoir engineering. In *Chinese Control and Decision Conference, Guilin, China, 17-19 June 2009*.

[van Essen et al., 2009a] van Essen, G. M., Jansen, J. D., Brouwer, D. R., Douma, S. G., Rollett, K. I., and Harris, D. P. (2009a). Optimization of smart wells in the st. joseph field. In *Asia Pacific Oil and Gas Conference and Exhibition, Jakarta, Indonesia, 4-6 August 2009*.

[van Essen et al., 2010] van Essen, G. M., Jansen, J. D., and Van den Hof, P. M. J. (2010). Lexicographic optimization of multiple economic objectives in oil production from petroleum reservoirs. In *American Control Conference, Baltimore, Maryland, U.S.A., 30 June - 2 July 2010*.

[van Essen et al., 2009b] van Essen, G. M., Van den Hof, P. M. J., and Jansen, J. D. (2009b). Hierarchical long term and short term production optimization. In *SPE Annual Technical Conference and Exhibition, New Orleans, Louisiana, U.S.A., 4-7 October 2009*.

[van Essen et al., 2009c] van Essen, G. M., Zandvliet, M. J., Van den Hof, P. M. J., Bosgra, O. H., and Jansen, J. D. (2009c). Robust waterflooding optimization of multiple geological scenarios. *SPE Journal*, 14:202–210.

[Wächter, 2002] Wächter, A. (2002). *An Interior Point Algorithm for Large-Scale Nonlinear Optimization with Applications in Process Engineering*. PhD thesis, Carnegie Mellon University.

[Wächter and Biegler, 2006] Wächter, A. and Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106:25–57.

[Wright, 1970] Wright, K. (1970). Some relations between implicit runge-kutta, collocation and lanczos methods, and their stability properties. *BIT Numerical Mathematics*, 10:217–227.

[Zandvliet et al., 2007] Zandvliet, M. J., Bosgra, O. H., Jansen, J. D., Van den Hof, P. M. J., and Kraaijevanger, J. F. B. M. (2007). Bang-bang control and singular arcs in reservoir flooding. *Journal of Petroleum Science and Engineering*, 58:186–200.

Glossary

List of Acronyms

BHP	bottem-hole pressure
CLRM	closed-loop reservoir management
CONOPT	constrained optimization
DAE	differential algebraic equation
DCSC	Delft Center for Systems and Control
DOF	degrees of freedom
GAMS	General Algebraic Modeling System
GBP	grid-block pressure
GRG	generalized reduced gradient
IG	initial guess
IP	interior point
IPOPT	Interior Point Optimizer
KNITRO	Nonlinear Interior-point Trust-Region Optimizer (the K is silent)
MatLab	Matrix Laboratory
MoReS	Modular Reservoir Simulator
MPC	model predictive control
NLP	non-linear programming
NPV	net present value
ODE	ordinary differential equation
PDE	partial differential equation
SimSim	Simple Simulator
SLP	sequential linear programming
SNOPT	sparse nonlinear optimizer
SQP	sequential quadratic programming

List of Symbols

Dynamic Optimization

J	Objective Function
f	Dynamic Equality Constraint
g	Static Equality Constraint
h	Static Inequality Constraint
x	Differential Variable
z	Algebraic Variable
u	Control Variable
p	Time-independent Parameter
q	Finite set with Decision Variables
λ	Adjoint Equation
t	Time
ϕ	Lagrange Interpolation Polynomial
R	Residual Equation
N_u, N_x, N_z	Number of Inputs, States, Algebraic Variables
N_p, N_t	Number of Parameters and Time-Steps

Reservoir Modeling and Flooding Optimization

α	Geometry Factor
A	Area
b	Discount Value
c_o, c_w, c_r	Oil, Water, Rock Compressibility
d	Depth
φ	Objective Function representing the Net Present Value
ϕ	Porosity
f	Fractional Flow Rate
g	Gravitational Velocity
H	Height
$\mathcal{J}_{var}$	Objective Function for Minimization of the Variance
k	Permeability
k_r	Relative Permeability
$\vec{\mathbf{K}}$	Rock Permeability Tensor
μ	Fluid Viscosity
N_i, N_j	Number of Rows and Columns
N_t	Number Time-Steps
p	Pressure

q	Flow Rate
r	Cost
ρ	Fluid Density
s	Water Saturation
S	Saturation
S_{or}	Residual Oil Saturation
S_{wc}	Connate Water Saturation
S_{wn}	Normalized Water Saturation
t	Time
τ	Reference Time
T	Transmissibility
T_R	Reference Temperature
U	Total Volumetric Flow Rate
w	Weight of the Variance

Subscripts

i, j, n	Index referring to Grid Block (i, j) and Time-Step n
o, w	Oil or Water Phase
$prod, inj$	Production or Injection Well