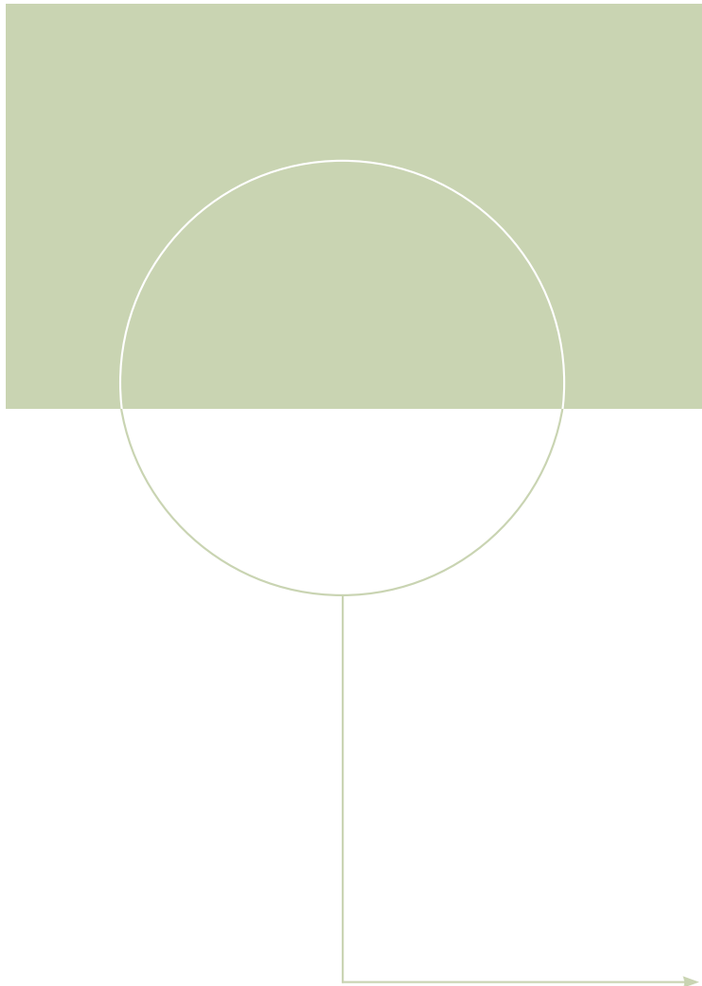


Master's thesis

2019

Subhi Sadigov

NTNU
Norwegian University of
Science and Technology
Faculty of Engineering
Department of Geoscience and Petroleum

Subhi Sadigov

Well Placement Optimization using Open-Source Simulators

A Case Study of a Marginal Field on the NCS

June 2019

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

Well Placement Optimization using Open-Source Simulators

A Case Study of a Marginal Field on the NCS

Subhi Sadigov

Petroleum Engineering
Submission date: June 2019
Supervisor: Carl Fredrik Berg
Co-supervisor: Mathias Bellout

Norwegian University of Science and Technology
Department of Geoscience and Petroleum

Acknowledgment

I would like to thank my supervisor Associate Professor Carl Fredrik Berg and co-supervisor postdoctoral researcher Mathias Bellout for their technical input and guidance throughout this project. I am grateful to A/Prof. Carl Fredrik Berg for all his supervision, patience, and constructive feedback that improved the quality of my work significantly. I also appreciate PhD student Brage Strand Kristoffersen for taking his time to help me through the technical challenges encountered along the way.

Subhi Sadigov

This work is dedicated to my family, whose constant encouragement and support allowed me to be where I am.

" He smiled understandingly – much more than understandingly. It was one of those rare smiles with a quality of eternal reassurance in it, that you may come across four or five times in life. It faced – or seemed to face – the whole external world for an instant, and then concentrated on you with an irresistible prejudice in your favor. It understood you just so far as you wanted to be understood, believed in you as you would like to believe in yourself, and assured you that it had precisely the impression of you that, at your best, you hoped to convey. "

The Great Gatsby, F. Scott Fitzgerald, 1925

Preface

This is a Masters thesis which will result in a Masters degree in Reservoir Engineering and Petrophysics at the Norwegian University of Science and Technology (NTNU). The work was carried out in the spring semester of 2019 in close collaboration with the Petroleum Cybernetics Group at NTNU and OKEA ASA. The topic of research was selected at NTNU with the intention of demonstrating the applicability of open-source simulators in field development optimization projects.

Basic knowledge of reservoir simulation and associated parameters are required to understand the key ideas presented in this document.

Trondheim, 11-06-2019

Subhi Sadigov

Summary

Throughout this work, the applicability of using an open-source approach for optimizing well placement configuration of an oil reservoir was investigated. In petroleum field development, manual tools are frequently employed to optimize reservoir depletion strategies that are tedious and usually inefficient. Therefore, attention to finding a systematic optimization approach for petroleum field development problems in a cost-efficient manner has increased over the past decades. The efficiency of an optimization process can be significantly improved by utilizing automatic optimization procedures. Although the implementation of such procedures is very often associated with costly software licenses, this cost can be eliminated by using software packages with open-source type licenses.

The applicability of an open-source optimization workflow has been demonstrated through the case study described in this thesis. The case study focuses on finding the optimal well placement configuration for a marginal oil field in the Norwegian Continental Shelf, and it consists of two parts. The first part of this work focuses on converting the ECLIPSE model of the reservoir so that it can be run in OPM-Flow, which is a reservoir simulator with an open-source license. The second part of this thesis focuses on solving the well placement optimization problem of this oil field by using FieldOpt, which is a software that applies mathematical programming techniques to ease the optimization procedures applied to field development.

During the simulation model conversion process, a number of approximations and modifications were made to the original simulation deck to develop a work model that can be run in Flow. The impacts of individual modifications were investigated, and the simulation results generated by Flow were validated against a commercial reservoir simulator via a benchmark test. The benchmark test using the work model of the reservoir indicated that results predicted by Flow are in close agreement with the reference simulator results, and simulation run-time comparison showed that the single-core performance of Flow was only slightly slower than the ref-

erence simulator in this test.

In the second part of this thesis, the well placement configuration of the reservoir was optimized. The optimization algorithms used in this part were Particle Swarm Optimization (PSO) and Asynchronous Parallel Pattern Search (APPS). Two different search configurations for each optimization algorithm were explored. The performance of these algorithms was assessed in terms of the total number of function evaluations performed for each optimization run, final incremental objective function (i.e., NPV) improvement yields, and applicability of the placement solutions. Results indicated that for this particular optimization problem, the PSO algorithm yielded slightly higher objective function improvements for single well optimization cases. However, in terms of efficiency, the APPS algorithm converged to optimum solutions in a smaller number of function evaluations. Consequently, the final well placement configuration based on the individual placement solutions improved the objective function by 30.8% over the base case scenario.

Table of Contents

List of Tables

List of Figures

Abbreviations

Mneumonic	Description
NPV	Net Present Value
WOPR / WOPT	Well oil production rate / total
FOPR / FOPT	Field oil production rate / total
WWPR / WWPT	Well water production rate / total
FWPR / FWPT	Field water production rate / total
WLPR / WLPT	Well liquid production rate / total
FLPR / FLPT	Field liquid production rate / total
WBHP	Well bottomhole pressure
WWCT	Well water cut
WOC	Water-oil contact

Chapter 1

Introduction

According to leading energy outlook and analysis firms, the world's consumption of energy is rapidly increasing and this trend will surely continue into the foreseeable future. Although renewable energy sources are the fastest growing source of energy, it is predicted that oil and gas will account for around 50% of the worldwide primary energy consumption by 2040, see figure 1.1.

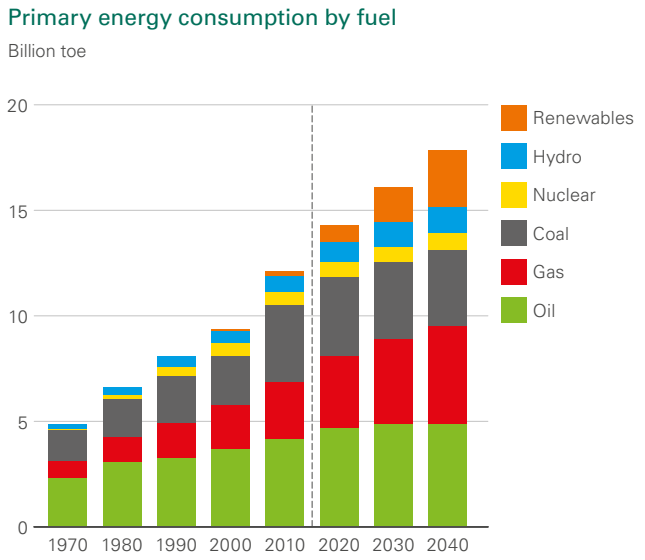


Figure 1.1: Primary energy consumption forecast *(BP Energy Outlook 2019).*

Considering the fact that the fossil fuels are non-renewable energy sources, meeting the required energy demand for oil and gas will be challenging in the future. Although huge reserves of unconventional hydrocarbon resources are available, the excisting recovery mechanisms for these resources are eighter not economically attractive at the current economic environment or politically and environmentally sensitive to meet the ever-increasing energy demand (Muggeridge et al., 2014). Since it is becoming increasingly difficult to find new conventional oilfields, oil and gas companies are focusing on maximizing the hydrocarbon recovery factors (RF) from their existing fields as well as sustaining the economic production rates. A recent study shows that the average RF from mature oilfields around the world is between 20-40 %, which means that more than half of the hydrocarbon volumes are left under the ground (Muggeridge et al., 2014). However, the RF can be improved by choosing the optimal reservoir development strategy for both new discoveries and existing fields that will maximize the oil recovery and minimize the capital expenditure. In traditional reservoir engineering practice, manual tools are commonly employed to come up with optimum reservoir development strategy which might require excessive amounts of trial and error tries before reaching to a decisive reservoir depletion strategy. However, as modern computers are becoming more powerful and less expensive, the manual workflow of finding the optimal solutions to field development problems can be replaced with automatic procedures. This will make the workflow more time efficient as well as less prone to errors if applied properly.

Nowadays, finding a systematic optimization approach for petroleum field development problems in a cost-efficient manner is a popular research topic among researchers, industry leaders, and engineers. The main cost of the automatic optimization approach is very often associated with software licenses. These proprietary software packages, e.g. Petrel and ECLIPSE by Schlumberger, enable reservoir engineers to simulate various field development scenarios and make decisions based on the simulated results. However, this cost can be eliminated by using open-source software suits instead. Therefore, the first part of this thesis consists of converting the ECLIPSE model of the reservoir to be compatible with OPM-Flow, which is an open-source reservoir simulator. This part was carried out as a specialization project in reservoir engineering during the fall semester in 2018.

This work is a case study focusing on the well placement optimization of a marginal oil reservoir in the Norwegian Continental Shelf (NCS) by using FieldOpt, which is a software that applies mathematical programming techniques to search for the optimal solutions in the field development problems. The flexible programming structure of FieldOpt allows it to be adaptable to a wide variety of optimization algorithms, use cases, and methodologies for solving field development and reservoir management problems (Baumann et al., 2019). Throughout this work, different optimization algorithms were employed to find an optimized well placement configuration for a conventional oil field located on the NCS, and performance of these algorithms were compared in terms of the total number of cost function evaluations and the final objective function values obtained. The main purpose of this work is to prove the possibility of using FieldOpt for realistic field development optimization decisions.

Thesis Outline. The structure of the following chapters contained in this thesis are described as below:

Chapter 2: Background. The background topics on reservoir management, field development optimization, and optimization theory are contained in this chapter. A comprehensive description will be provided for each of the optimization algorithms tested in this work. Besides, the software architecture of the FieldOpt software will also be presented in this chapter.

Chapter 3: Methodology. This chapter starts with explaining the key concepts regarding the open-source field development optimization workflow and the benefits that this type of workflow potentially offers to the industry. This will be followed by brief background information on the reservoir and its simulation model used in this case study. The details of the simulation model conversion and validation procedure will also be outlined. Finally, the optimization methodology implemented to optimize the well placement in the reservoir will be presented.

Chapter 4: Results. This chapter consists of two sections. The results of the simulation model conversion and validation of the converted model will be presented in the first section. The second section will be the presentation of the well placement optimization results that were obtained using FieldOpt.

Chapter 5: Discussion. This chapter will expand upon the main results achieved during this work, and try to address the main research questions. Furthermore, the drawbacks and further recommended improvement areas in Flow and FieldOpt will be summarized.

Chapter 6: Conclusions. The end-summary and concluding remarks are contained in this chapter. Recommendations on further work will also be highlighted at the end.

Chapter 2

Background

The work done in this thesis combines the aspects of reservoir management, field development optimization, reservoir simulation, and optimization theory. In this chapter, brief introductions to part of those topics that are relevant to this work are provided.

2.1 Reservoir Management and Field Development Optimization

The reservoir management practice can be defined as maximizing profits from a reservoir by optimizing recovery while minimizing the capital investments and operating expenses with the use of available resources. These resources include technology, human, and financial. Successful reservoir management requires a multidisciplinary team effort and integration of people, tools, data, and technology (Satter et al., 1994). According to Satter, although the reservoir management plan can be implemented at any time during the lifetime of the reservoir, the ideal time to start managing a reservoir is at its discovery. An early implementation of reservoir management leads to better overall project planning, implementation, monitoring, and evaluation as well as saves capital in the long run.

As of today, many different tools are available for reserve estimations and reservoir performance prediction, such as material balance techniques, volumetric estimations, decline curve analysis, and reservoir simulators. Each

of these tools have their own pros and cons depending on the problem and the way these tools are applied. However, reservoir simulators are more commonly employed in reservoir management projects compared to the other alternative tools. This is because simulation models account for the complex flow patterns and changing fluid properties over time, which are crucial parameters that affect the future performance of the reservoir.

Reservoir models are built based upon the data gathered from multiple sources, such as seismic, well logs, well tests, and production history matching from the existing wells. Since it is possible to build different realizations of a reservoir based on the same data, the models should be validated and occasionally updated as new information becomes available about the reservoir. Reservoir simulators are also widely used for solving field development optimization problems, where multiple development scenarios are built and compared based on their future performance. Therefore, building and maintaining a reliable reservoir model is an essential task for reservoir management and development.

2.1.1 Reservoir simulation

A reservoir simulator utilizes a set of partial differential equations to describe the flow of fluids and change of dynamic properties of a reservoir during its production. The reservoir is represented by a numerical model consisting of an array of individual grid cells, where the adjacent grid cells are linked together by the interblock transport equations. The equations that describe fluid flow in porous media are derived by combining the concepts of mass conservation and Darcy's equation. For clarity, the flow equations described in this section entail several simplifications. Details on the flow equations and numerical discretization can be found in Aziz and Settari (1979) or Peaceman (1977).

In the following, the governing equations for fluid flow in the porous media will be presented.

$$\nabla \left[\frac{1}{B_o} u_o \right] + q_o = -\frac{\partial}{\partial t} \left[\frac{1}{B_o} \phi S_o \right] \tag{2.1}$$

Equation 2.1 describes the flow of the oil phase in the system, where B_o is the oil formation volume factor, and u_o is the Darcy velocity of this phase.

Rock porosity and oil saturation are represented by ϕ and S_o, respectively. The source/sink term of the oil phase is represented by q_o.

$$\nabla\left[\frac{1}{B_w}u_w\right] + q_w = -\frac{\partial}{\partial t}\left[\frac{1}{B_w}\phi S_w\right] \tag{2.2}$$

Similarly, Equation 2.2 describes the flow of the water phase in the system, where B_w is the water formation volume factor, and u_w is the Darcy velocity of this phase. The source/sink term and saturation of the water phase are represented by q_w and S_w, respectively.

$$\nabla\left[\frac{R_s}{B_o}u_o + \frac{1}{B_g}u_g\right] + q_g + R_s q_o = -\frac{\partial}{\partial t}\left[\phi\left(\frac{R_s}{B_o}S_o + \frac{1}{B_g}S_g\right)\right] \tag{2.3}$$

Equation 2.3 describes the flow of the gas phase in the system, where B_g is the gas formation volume factor, and u_g is the Darcy velocity of this phase. The source/sink term and saturation of the gas phase are denoted by q_g and S_g, respectively.

The Darcy velocity of each fluid flowing through the porous medium is expressed as:

$$u_i = -k\frac{k_{ri}}{\mu_i}\nabla p_i, \quad i = o, w, g \tag{2.4}$$

where i specifies the corresponding fluid phase, k is the absolute (rock) permeability, and ∇p_i is the pressure gradient of phase i. The viscosity and relative permeability of phase i is denoted by μ_i and $k_{ri}(S_i)$, respectively.

In order to solve the equations presented above, three additional equations are required. These equations are:

$$S_o + S_w + S_g = 1 \tag{2.5}$$

$$p_{cow} = p_o - p_w \tag{2.6}$$

$$p_{cog} = p_g - p_o \tag{2.7}$$

where p_{cow} represents the capillary pressure between the oil and water phase, and p_{cog} represents the capillary pressure between the oil and gas phase in the system.

The equations (2.1) to (2.7) are discretized and solved numerically for the saturation and pressure of each phase. Reservoir model sizes for these equations typically range from tens of thousands to several millions of grid blocks for small and large models, respectively. A typical reservoir simulation might require several hundred time steps.

In order to compute the volume of fluids produced and injected at each time step, reservoir models are coupled to well models via the source term q as:

$$q = J(p_R - p_{bh}) \tag{2.8}$$

where p_R is the average reservoir pressure in grid cell, and p_{bh} is the flowing bottom-hole pressure in the well. The constant of proportionality J is referred to as the *productivity index* (PI) for production wells and the *well injectivity index* (WI) for injection wells (Lie, 2014). This well model can be extended further to account for near wellbore formation damage (skin), anisotropic media, multiphase flow, and horizontal wells. Furthermore, there are also models to describe the fluid flow inside the well to compute the surface flow rates and tubing head pressure. See Peaceman et al. (1983) for further details on this topic.

2.1.2 Field development optimization

Optimization project of a petroleum field involves a wide spectrum of engineering disciplines, where decisions have to be made in various field development aspects such as reservoir production, drilling, and facilities operations (Baumann et al., 2019). Since the field level optimization of a reservoir comes down to the well level optimization decisions, a few possible well configuration parameters, e.g. the number, type, location, bottom-hole pressure (BHP) and valve settings, can be identified as potential optimization parameters.

The research on field development optimization is quite extensive, and investigating every aspect of production optimization is out of the scope of this work. However, optimization of well parameters can be narrowed down to three key parameters: well control, well completion design, and well placement configuration. This thesis focuses only on optimization of the well placement in the reservoir.

Well Control Optimization. The goal of the well control optimization is to determine the optimal settings that will maximize the oil and gas production rates and decrease the water production rate of the producer wells, as well as find the optimum injection rates for the injector wells that will maximize the sweep of the reservoir. The main control variables for optimization are the well bottom hole pressure, and the production or injection target rates for the producer and injector wells at specific dates, respectively.

The two main strategies to deal with well control optimization is the reactive and proactive approaches (Ebadi et al., 2006). In the *reactive* strategy, the well controls are based on downhole events, such as a sudden increase in water or gas production rate. On the other hand, with the *proactive* approach, the well control settings are planned long before water or gas breakthrough in the well. This approach relies heavily on the current and forecasted well performance. Therefore, a precise model of the reservoir or the well is required in order to employ this approach.

Well Completion Optimization. The well completion optimization aims to maximize the NPV and minimize unwanted fluid production by finding the optimal settings for inflow control valves (ICVs). The wells that contain such downhole assemblies are often called *smart wells* or *intelligent wells*. This technology allows the control of pressure profiles along the wellbore by restricting or shutting off the flow from the specific compartments in the multi-lateral wells (see figure 2.1). Smart wells are also equipped with downhole sensors to acquire real-time flow rate, downhole temperature, and pressure data for reactive or proactive production control (Arukhe et al., 2017). The surface control unit controls the valve settings of ICVs via eighter hydraulic or electrical control lines.

The main challenge with the use of ICVs is to determine the correct vavle settings to achieve the optimum flow conditions in the well since the number of possible valve configurations is too high for finding the optimum valve setting manually by trial and error method. Therefore, well completion optimization has become a popular research topic in the oil and gas industry since automation of this process will lead to a higher productivity index, better water management, longer well life, and save time at the same time. There are many optimization algorithms available to solve such optimization problems given that a reliable simulation model of the reservoir

and the well exists.

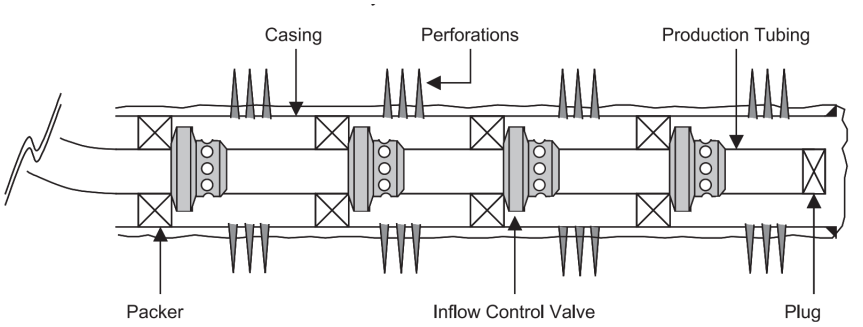


Figure 2.1: The figure illustrates the smart well completion for four separate inflow zones isolated by packers and inflow control valve in each zone. Electric and hydraulic control lines as well as the measurement devices are not shown in the figure (Van der Poel and Jansen, 2004).

Well Placement Optimization. The main idea behind the well placement optimization task is to find the optimal locations for the producer or injector wells to be drilled in the reservoir formation. The optimum placement for each well is usually determined after the total number and type of wells, and their operational settings, such as BHP or injection rates, are specified by the reservoir management team. It is important to have clear limitations on the feasible search area of the new wells, which is usually based on the engineering experience or the specific knowledge about the reservoir, such as faults, thief zones, etc.

Once these limitations are established and translated into the constraints on the problem formulation, the success of the optimization job depends on the constraint-handling capabilities of the optimizer, and the efficiency of the search algorithms being used for the task (Jesmani et al., 2015). The constraints guide the optimizer to search for the optimal solution in the desired region of the formation and avoid locating the well too close to the neighboring wells or to the geological structures that will be challenging to drill through. A wide variety of optimization methodologies have been developed, and have been applied in various engineering disciplines since the 1960s (Jansen et al., 2005). The methodologies that are applied to solve the well placement optimization problems are typically the gradient-free and direct search methods. More detailed information about the methodologies

used in this work will be presented in the next section.

Currently, FieldOpt software (see section 2.3) is capable of solving all three of the field development optimization problems that were mentioned in this section.

2.2 Optimization Theory

In mathematics, an optimization problem consists of finding the input value that maximizes or minimizes the function by systematically searching within an allowed solution space and computing the value of the function. In general, optimization is finding the "best available" values of an objective function given the input domain.

2.2.1 Optimization problem

An optimization problem consists of three components: the objective function, variables, and constraints. Mathematically, optimization problem can be described as (Nocedal and Wright, 2006):

$$\min_{x \in R^n} -f(x) \quad \text{subject to} \begin{cases} c_i(x) = 0, & i \in E \\ c_i(x) \geqslant 0, & i \in I \end{cases} \tag{2.9}$$

Where:

- x is the vector of *variables*, also called as parameters;
- f is the *objective function*, a function of x needs to be maximized or minimized;
- c is the vector of constraints that the *variables* must satisfy. The number of components in c is the number of individual restrictions that are placed on the variables.
- E and I are the set of indices for equality and inequality equations respectively.

In reservoir production optimization problems, a reservoir simulator is used to solve the equations of fluid flow in porous media to compute the objective function associated with a given input variable. *The objective function* in these type of problems is usually Net Present Value (NPV) or the weighted sum of the cumulative fluid productions from the reservoir.

In order to search for the optimal well placement in the specific region of the reservoir, a *reservoir bound constraint* needs to be imposed on the variables. Reservoir bound constraints prevent the movement of the heel and toe of the well outside the specified region. Therefore, optimizing the well placement in order to maximize the NPV can be formulated as below (Bellout, 2014):

$$NPV(x) = \sum_{k=1}^{N_s} \left(\sum_{j=1}^{N_p} p_o q_o^{j,k}(x)\Delta t_k - \sum_{j=1}^{N_p} c_{wp} q_{wp}^{j,k}(x)\Delta t_k - \right. \\ \left. - \sum_{j=1}^{N_i} c_{wi} q_{wi}^{j,k}(x)\Delta t_k \right) \Big/ (1+i)^t \tag{2.10}$$

where N_p and N_i denote total number of producer and injector wells in the system, $\mathrm{q}_o^{j,k}$,$\mathrm{q}_{wp}^{j,k}$ and $q_{wi}^{j,k}$ are flow rates for the produced oil and water, and water injected for the well j at the output interval k respectively, and Δt_k is the length of each of the N_s time steps in the simulation. The weights of each component are the oil price, and cost of water production and injection per barrel, which are represented by p_o, c_{wp} and c_{wi} respectively. The *i* stands for the discount rate (expressed in %), and *t* is the is the total number of the years starting from zero at the first year. The discount rate is the interest rate that is used for calculating the present value of the future cash flow of a project, company, or asset. For simplicity, the discount rate was assumed to be zero in this work.

2.2.2 Optimization algorithms

Once the objective, variables, and constraints for the given problem are identified, an optimization algorithm can be used to find its solution. Depending on the type of optimization problem, a wide variety of algorithms are available to solve the problem. However, it is crucial to pick an algorithm that is suitable for the given problem, since this choice determines if the problem is solved efficiently or not, or even if a solution is obtained at all.

Optimization algorithms require an initial guess of the optimal values of the variables and iteratively generate a set of improved estimates until the convergence conditions are met. The main difference between the optimization algorithms is the strategy they use to move from one estimate to another. For example, some algorithms use the local information from the current point only, while others use the information gathered from other points or previous iterations. An optimization algorithm needs to have the following properties to be considered suitable for real engineering and scientific applications (Nocedal and Wright, 2006):

- **Robustness:** The algorithm should handle the problem properly for any reasonable initial starting points.
- **Efficiency:** It should keep the computation cost low.
- **Accuracy:** It should not be sensitive to the noise in the input data, and identify the solution with precision.

These properties may conflict with each other when employed to solve real optimization cases. Therefore, the users have to find a balance between these three points based on their needs and the problem at hand.

Discrete versus Continuous Optimization. Depending on the type of the variable that the algorithm is searching for, the optimization problems can be categorized as *discrete* optimization problems and *continuous* optimization problems.

In discrete optimization problems, the solution only makes sense if it is an integer value. Solving the problem with real variables and then rounding

them to the nearest integer values does not guarantee that the solution obtained is optimal. As a result, a new mathematical constraint is introduced to the existing constrains in eq. (2.10):

$$x_{ij} \in \mathbf{Z}, \quad \text{for all } i \text{ and } j \tag{2.11}$$

where **Z** is the set of all integers (Nocedal and Wright (2006)). For the general problem of well placement, well locations are considered to be discrete variables due to the fact that wells are assigned to discrete grid blocks in the reservoir simulation model.

The continuous optimization problems, on the other hand, use variables that are chosen from a set of real values in order to calculate the objective function. One typical example of such an optimization problem in the context of this work would be a well control optimization case. Because of this continuity, it is possible to use the objective function information at any point to get information about the function behavior at the neighboring points by using various calculus techniques. The optimization problems in this category can be solved using the gradient-based optimization algorithms, which will be mentioned in the next sections.

Deterministic and Stochastic Optimization. *Deterministic* optimization is the branch of optimization algorithms that incorporate algorithms that rely heavily on linear algebra, i.e. gradient-based, to search for an optimum point in the search space. The results of a deterministic optimization process are replicable because the solution of this method depends on the initial guess point. Therefore, the solution will always converge to the same optimum point if departed from the same starting point (Cavazzuti, 2012).

This branch of optimization algorithms usually require a lower number of objective function evaluations to reach the solution compared to stochastic optimization algorithms. However, these algorithms have a high chance of converging to a local optimum instead of the global optimum.

Stochastic optimization branch, on the other hand, consists of algorithms that solve an optimization problem by including mathematical randomness in their search procedure (Cavazzuti, 2012). Depending on the way the randomness is included, stochastic optimization algorithms can be further

classified into separate families, e.g. Simulated Annealing, Particle Swarm Optimization, Evolutionary Algorithms, Genetic Algorithms, etc. Each of these families contain algorithms that are built to mimic some natural phenomena that are observed in nature. Commonly, the search techniques of these algorithms are population-based, meaning they are imitating the collective behavior of the living organisms, such as flocking or swarming.

The main advantage of stochastic algorithms is that they do not get trapped in a local optimum due to the role of randomness in their search process. In each search step, the initial guess is randomly generated and solved iteratively until the optimum solution is found. Compared to deterministic optimization methods, these algorithms are also less mathematically complicated. However, stochastic methods can be computationally resource-intensive to implement as a large number of objective function evaluations are required for convergence (Sethi, 1983).

Optimization methods can be further classified into *local* and *global* optimization groups. Local optimization algorithms are usually gradient-based and seek for the stationary points of the objective function. However, the stationary points found by the algorithm are not guaranteed to be the global maxima or minima of the function (see figure 2.2). Global optimization algorithms are essentially stochastic optimizers that are non-gradient based, so that they do not get stuck in the local optimum points and rather continue searching for the global optimum.

Regardless of the method used for solving an optimization problem is deterministic or stochastic, some elements are required in order to set up and solve the problem. Especially a starting point in the search domain, and a stopping criterion must be provided. Once the optimization process is run, the algorithms generate a sequence of iterates that terminate when the solution has been reached with sufficient accuracy or when the number of function evaluations exceeds the allowed threshold.

The gradient-based deterministic algorithms search for the feasible solution by using the gradients of the function at the given point in order to determine the search direction at the next iteration. There are two strategies available for searching for the direction of the optimum point: *line-search* approach and *trust region* approach.

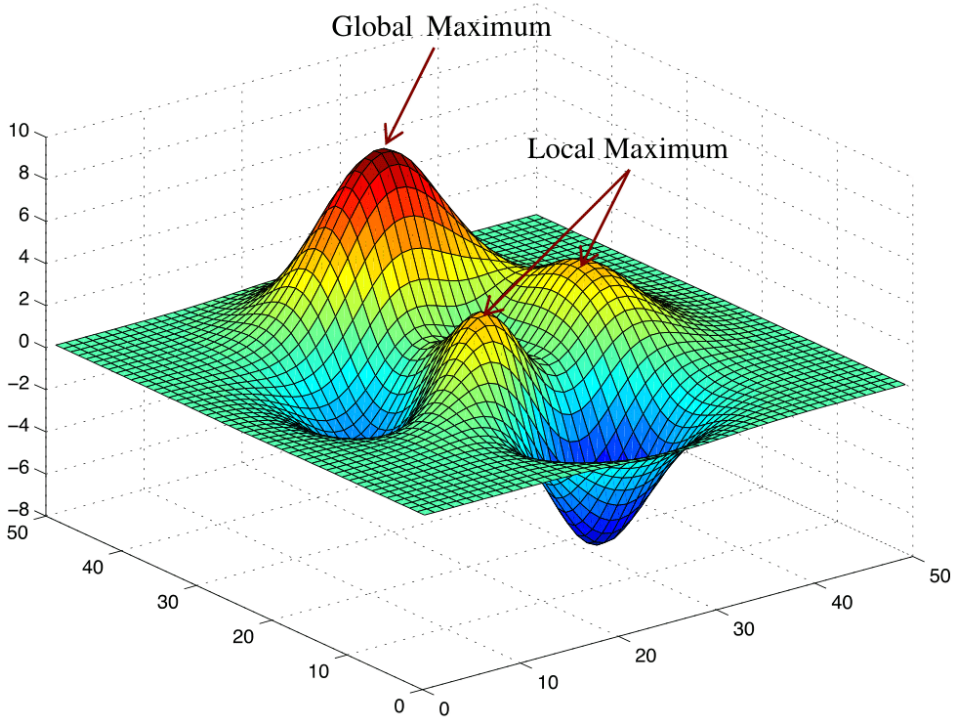


Figure 2.2: An illustration of a function with two local maximums and a global maximum (Jin, 2015).

Line Search Methods. For a general optimization problem,

$$\min_{x \in R^n} -f(x), \tag{2.12}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, the optimization algorithm starts the search from point x_0 and generates a sequence of points $x_{(k)}$ in the search space converging to the optimum solution (assuming f is smooth). The steps of a line search method at iteration k can be summarized as:

- determine a direction d_k
- find α_k that minimize $f(x_k + \alpha_k d_k)$, where $\alpha \in \mathbb{R}$
- set $x_{k+1} = x_k + \alpha_k d_k$,

where x_k is a point in search space and d_k is a search direction, also referred to as *descent direction* (Kolda et al., 2003). First, a descent direction vector $d_k \in \mathbb{R}^n$ is calculated at the point x_k, which means it has to be within 90^o of -$\nabla f(\mathrm{x}_k)$ (see figure 2.3):

$$-\nabla f(x_k)^T d_k > 0 \tag{2.13}$$

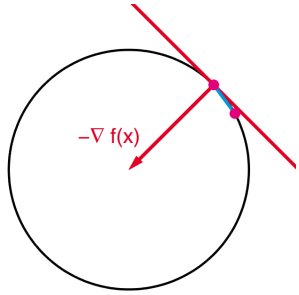
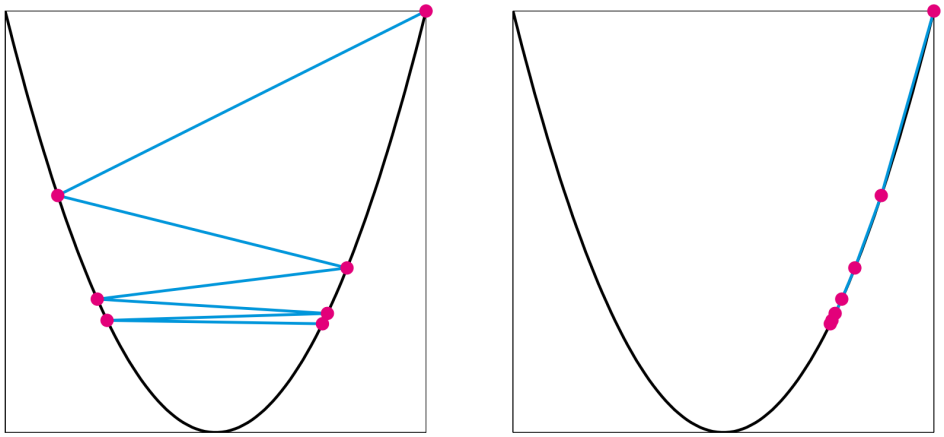


Figure 2.3: The figure illustrates the level curve of a function at x_k, shown in black. The red arrow represents the steepest descent direction from x_k, and the red line represents the tangent plane at the x_k. The descent direction, indicated by the blue line segment, satisfies the descent condition in eq. (2.13) since it is within 90^o of $-\nabla f(x_k)$. However, only a short step along the direction d_k will yield improvement. As a consequence, the iterates may converge prematurely to a point that is not a stationary point as the angle between the d_k and $-\nabla f(x_k)$ approaches 90^o (Kolda et al., 2003).

Second, a step length $\alpha_k > 0$ is calculated so that the objective function value is decreased:

$$f(x_{k+1}) < f(x_k) \tag{2.14}$$

Once the above condition is satisfied, the point x_{k+1} becomes the new iterate, and the first two steps are repeated. However, simple application of these steps does not guarantee that the solution will converge to a stationary point (Gould, 2006). In general, the performance of line search methods depends on the choice of both the step-length α and search direction d_k. This search method, also called the *exact* line search, can fail to find the optimum point of the function if the step-length size is not chosen properly, meaning the step-length size being too long or too short is detrimental to the final result (see figure 2.4).



(a) Steps are too long. (b) Steps are too short.

Figure 2.4: The figure illustrates the possible ways that a line search can fail if poor choices of step-length sizes are made. In figure (a), the step are too long relative to the amount of decrease achieved by one iterate to the next. Therefore, the solution does not converge to an optimal point. On the other hand, in figure (b), the steps sizes are too short relative to the linear rate of decrease of the function so that the search converges prematurely (Kolda et al., 2003).

The poor choices of step lengths can be avoided by imposing acceptance criteria for the step length sizes. The condition

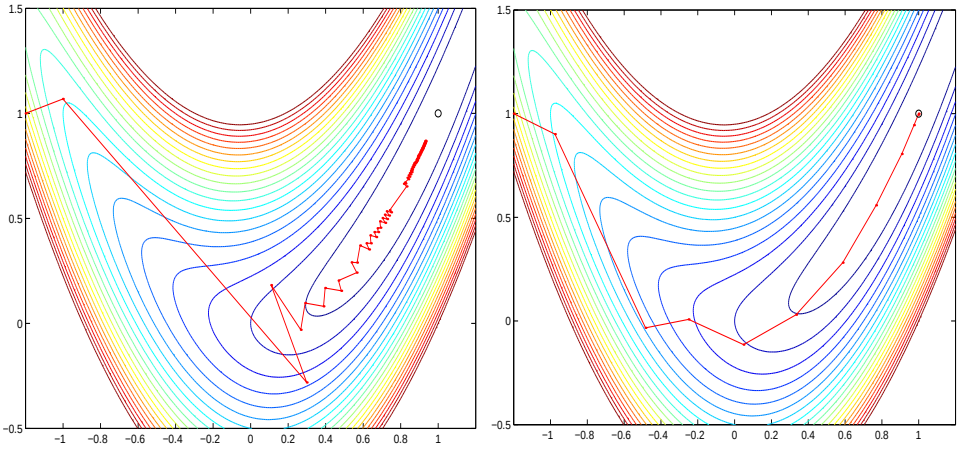
$$f(x_{k+1}) \leq f(x_{k+}) + c_1 \alpha_k \nabla f(x_k)^T d_k \tag{2.15}$$

prevents the steps that are too long via sufficient decrease criteria, and the condition

$$\nabla f(x_k + \alpha_k d_k)^T d_k \geq c_2 \nabla f(x_k)^T d_k \tag{2.16}$$

prevents steps that are too short through a curvature criterion, where $0 < c_1 < c_2 < 1$. The conditions in the eq. (2.15) and eq. (2.16) are referred to as the Armijo-Goldstein-Wolfe conditions (Kolda et al., 2003). Such search methods are referred to as the *inexact* line search methods.

It is preferable to use the *inexact* line search because the steps sizes picked by this method are neighter too long or too short (Cavazzuti, 2012). The modern line search methods, e.g. steepest descent, conjugate gradient, and Quasi-Newton method, aim to pick the initial guess for each step-size so that the convergence to the optimum point is guaranteed. As a result, these methods reach the optimum point of a function in a much less number of objective function evaluations, which makes these methods computationally cheaper.



(a) Method A **(b)** Method B

Figure 2.5: Figure illustrates the iterates generated by two methods while searching for the optimum of the objective function $f(x, y) = 10(y - x^2)^2 + (x - 1)^2$. It can be observed that the Method B performed much better in this case since it converged to the optimum point within significantly less number of objective function evaluations (iterates) compared to Method A (Gould, 2006). Thus, Method B is computationally cheaper than Method A in this case.

Trust Region Methods. Compared to the line search methods, where a descent direction (d_k) and step size (α_k) is picked to reduce $f(x_{k+1})$, the *trust-region* methods pick the overall step s_k to reduce a *model* of $f(x_k + s_k)$, and then choose the next step as ($x_{k+1} = x_k + s_k$), which is a minimizer of the model in this trusted region. In other words, this method defines a region around the current point $f(x_k)$ and constructs a model function whose behavior in this region is assumed to be similar to that of the actual objective function (Nocedal and Wright, 2006). This process allows for the direction and step-length to be chosen simultaneously (see figure 2.6).

The size of the trust region is usually chosen according to the performance of the algorithm at the previous iterations. If the step is unsuccessful, the trust region size is reduced and a new minimizer is defined. On the other hand, if the previous model is reliable in producing successful steps, the size of the trust region is systematically increased so that longer steps can be taken towards the optimum point of the function.

The quadratic model (hence the m^Q) used in the trust-region methods can be formulated as

$$m_k^Q(d) = f_k + \nabla f_k^T d + \frac{1}{2} p^T B_k d \tag{2.17}$$

where $f_k = f(x_k)$, $\nabla f_k = \nabla f(x_k)$, and B_k is a systematic approximation of the local Hessian matrix, which is a square matrix with second-order partial derivatives of a scalar field or scalar-valued function (Nocedal and Wright, 2006).

In practical applications and simulation-based optimization problems, the cost of obtaining the analytical derivatives of the objective function is typically expensive. Therefore, a standard option is to use the finite-difference methods to estimate the objective function derivatives and feed this information to the gradient-based optimization algorithms (e.g. equation 2.17). However, this option brings its own set of problems when dealing with the noise and nonsmoothness in the objective function.

In simulation-based optimization setting, a computer simulation must be run and the output must then be post-processed to evaluate the values of the objective function. In many cases, the computed values may look like the

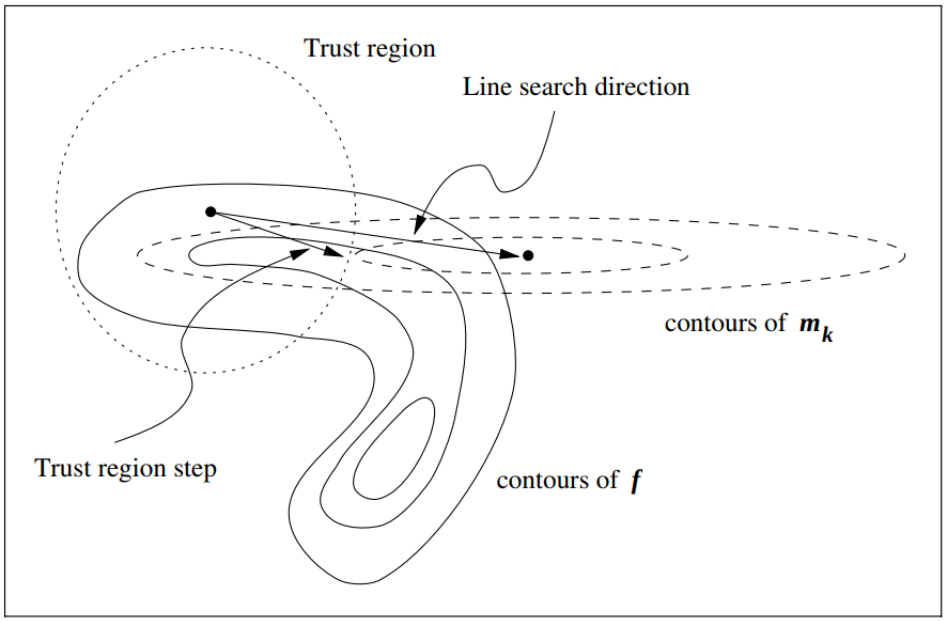


Figure 2.6: The figure illustrates the steps taken by both trust-region method and line search method on a function f of two variables. The model function m_k is based on the function and its derivative information at x_k. The step taken by line search method is based on this model, where it is a minimizer of m_k. It is evident that the step taken by the trust-region method is more efficient in terms reducing f in this case (Nocedal and Wright, 2006).

plot in Figure 2.7. As a result, the function gradients estimated by finite-difference methods can be widely inaccurate to be used in the gradient-based algorithms (Kolda et al., 2003). Although certain mathematical techniques exist to overcome this problem, e.g. adjoint techniques (Sarma et al., 2005), derivative-free optimization methodologies are considered to be a more practical option for well placement optimization problems.

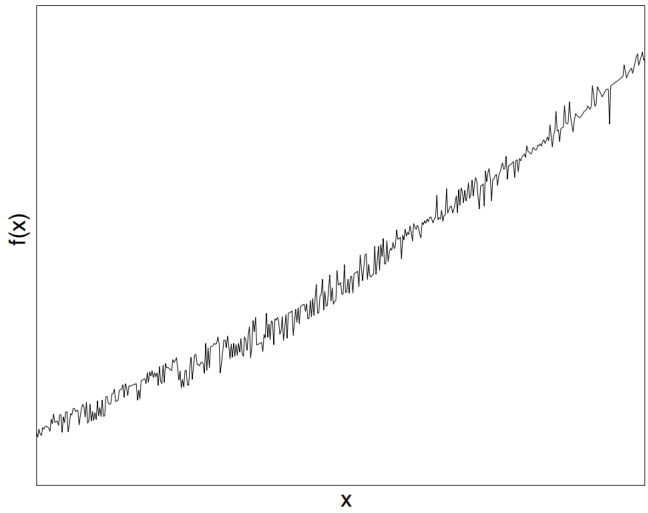


Figure 2.7: The figure illustrates objectives that are computed by solving Navier-Stokes equations with finite element scheme adaptation. The finite-difference approximation of gradients in functions with high frequency, low-amplitude oscillations may not be reliable to be used in gradient-based optimization algorithms (Kolda et al., 2003).

2.2.3 Optimization algorithms used in this work

Due to the deterministic nature of the well placement optimization problems, it possible to deal with a non-smooth objective function with multiple local optimums. Practice shows that the gradient-based optimization algorithms perform poorly compared to the performance of derivative-free optimization algorithms in such problems, since they are more likely to get trapped in a local minima or maxima. Therefore, this work employs two derivative-free and discrete optimization algorithms.

Asynchronous Parallel Pattern Search (APPS). APPS is a deterministic and derivative-free optimization algorithm that dynamically initiates search steps in repose to events instead of cycling through a fixed set of search directions, thus the name *asynchronous* (Kolda and Torczon, 2003). This optimization algorithm mainly concentrates on the parallelization of the search strategy which makes it computationally cost-effective compared to regular parallel pattern search (PPS) algorithms. In this section, an outline of both PPS and APPS is presented as a general background. It is considered that an optimization problem same to the one mentioned in eq. (2.12) is to be solved.

PPS starts the search at $k = 0$ and at the end of iteration $k - 1$, it is assumed that every process knows the best point x^{k-1}, where $f(x^{k-1})$ is the best known value of f. At the step x^{k-1}, the step-length control parameter is defined to be Δ^{k-1}. Each process $i \in \mathcal{P}$, constructs a trial point $x^{k-1} + \Delta^{k-1} d_i$ and initiates the evaluation of $f(x^{k-1} + \Delta^{k-1} d_i)$. After all the evaluations of the f are finalized, the processes communicate these values with each other to determine the new position x_k and Δ_k. This process can be formulated as following:

If there exists $i \in \mathcal{P}$ such that

$$f(x^{k-1} + \Delta^{k-1} d_i) < f(x^{k-1}) \tag{2.18}$$

then $k \in \mathcal{S}$, where $\mathcal{S}$ denotes the set of successful iterations.
Then, the update rules are

$$\mathrm{x}^k = \begin{cases} \mathrm{x}^{k-1} + \Delta^{k-1} d_i, & if \quad k \in \mathcal{S} \\ \mathrm{x}^{k-1}, & otherwise \end{cases} \tag{2.19}$$

$$\Delta^k = \begin{cases} \lambda\Delta^{k-1}, & if \quad k \in \mathcal{S} \\ \theta\Delta^{k-1}, & otherwise \end{cases} \tag{2.20}$$

where λ and θ represent the expansion and contraction factors respectively, and d_i is the direction that produced the largest decrease (Kolda and Torczon, 2003). A pseudo-code of this algorithm is presented in Algorithm [1].

Figure 2.8 depicts the first five iterations of PPS applied to the problem in (2.12). Each of the subfigures from (a) to (f) depicts the position of x_k, the best known point, as a magenta dot and four dark blue dots represent the trial points that are being evaluated at that iteration. It can be observed that the length of the steps taken is reduced near the minimum point as the algorithm missed the opportunity to find the solution at the third iteration (subfigure d) because the step-length was too long. The PSS algorithms are designed to wait for all processes to finish evaluating the function value before moving on to the next iteration. Such synchronized design of PPS results in idling of the processes and prolong the run-time of the algorithms to find the optimum point (see figure 2.9).

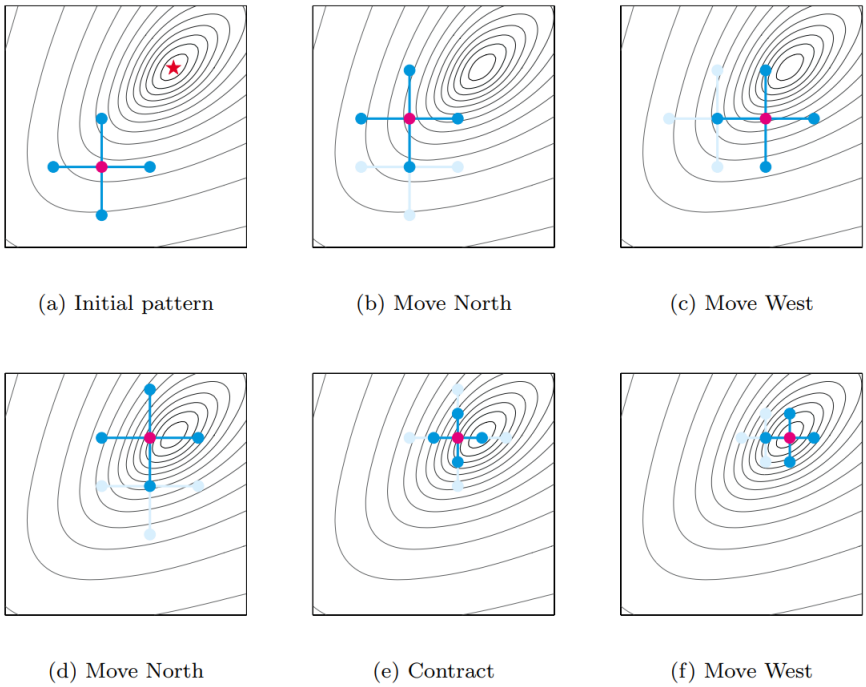
APPS algorithms, on the other hand, are designed to use the idle time of the processes to perform additional function evaluations. The general strategy to achieve this from the perspective of a single process is as following (Kolda and Torczon, 2003):

- Evaluate $f(x_i^{best} + \Delta_i^{best} d_i)$;
- If $f(x_i^{best} + \Delta_i^{best} d_i) < f(x_i^{best})$, communicate this result to the other processes;
- Update local values of x_i^{best} and Δ_i^{best}, based on local calculations and any information that may arrive from other processes;
- Repeat

Such design allows the successful processes to communicate its results to all other processes and move forward without waiting for all evaluations to be finished. As a result, this method cuts down on the run-time of the algorithms and prevents wasting valuable computational resources.

Algorithm 1 Pattern Search Algorithm adapted from (Baumann, 2015)

procedure PATTERNSEARCH($f, x_0, \Delta_{tol}, \Delta_0, \mathcal{D}, \mathbf{x}_d, \mathbf{x}_u, \theta, \lambda$)
  $\mathrm{f}_{best} \leftarrow f$ ▷ Evaluate initial objective value
  $x \leftarrow x_0$ ▷ Set **x** to initial guess value
  $\Delta \leftarrow \Delta_0$ ▷ Set initial step-length value
  while $\Delta > \Delta_{tol}$ **do** ▷ Iterate while step-length is greater than tolerance
    M $\leftarrow (x + \Delta \times d_k)$ **for all** $d_k \in \mathcal{D}$ ▷ Create the list of moves
    for all $\mathbf{x}_k \in$ **M do**
      if $\mathbf{x}_k < \mathbf{x}_d$ **or** $\mathbf{x}_k > \mathbf{x}_u$ **then** ▷ Check if **x** is in the bounds
        Discard $\mathbf{x}_k$ ▷ Discard **x** if it is outside the bounds
      end if
    end for
    $\mathbf{x}_{min} \leftarrow argmin : f(\mathbf{x}_k)$ ▷ Find the best move in iteration
    if $f(\mathbf{x}_{min}) < f_{best}$ **then** ▷ Found a better position
      $x \leftarrow x_{min}$ ▷ Change the best position
      $f_{best} = f(\mathbf{x}_{min})$ ▷ Evaluate new best objective value
    else ▷ If better position is not found
      $\Delta \leftarrow \theta\Delta$ ▷ Reduce the step-length
    end if
  end while
end procedure



(a) Initial pattern (b) Move North (c) Move West

(d) Move North (e) Contract (f) Move West

Figure 2.8: The figure illustrates the parallel pattern search applied to find the minimum of the function $f(x, y)$. In this case, the expansion factor λ is equal to 1 and the contraction factor θ is equal to $\frac{1}{2}$ (Kolda and Torczon, 2003).

Looking at the Figure 2.10, the first eight steps generated by four processes when APPS applied to $f(x, y)$ is illustrated. Unlike PPS, four different best points are used for evaluating trial points at any time step. The *open* circles indicate the points at which the function is being evaluated, and the *filled* circles indicate the best known point. The results from the previous time step, i.e., for $k > 0$, are depicted in the background in gray color. There are three possible outcomes:

- A trial point from the previous iteration becomes the new best point. In this case, the search moves to this point.
- The evaluation at the trial point did improve upon the best-known value at that iteration. However, it was succeeded by an even better point from another process. Such cases are indicated by *starred* circles.

- The value at the trial point did not improve upon the best-known value at that iteration. These cases are indicated by *crossed* circles.

In this example, the first three iterations are successful (i.e., $k \in 1, 2, 3 \subseteq \mathcal{S}$). Therefore, the length of the step for iterations 2 and 3 is expanded since two or more successful steps are taken successively along the same direction.

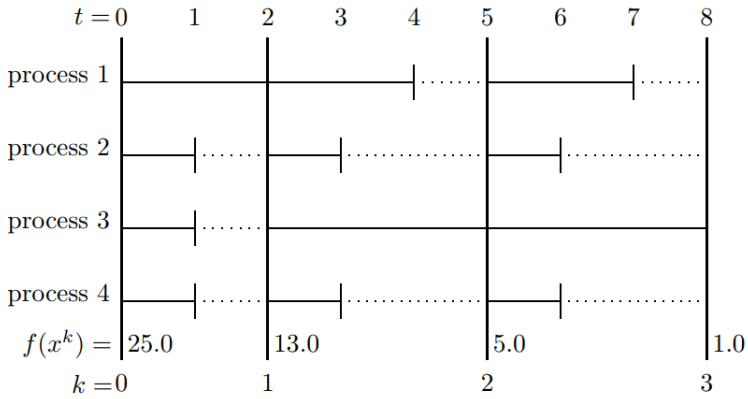


Figure 2.9: An illustration of idling of the processes, represented as dashed lines, for each iteration in a sample case where PPS method is used. Looking at the time $t = 0$, all four processes start evaluating the trial value of the objective function at the initial best point, where $f(x^k) = 25$. The processes 2, 3, and 4 finish evaluating the function value at $t = 1$. However, these processes have to wait (idle) for the process 1 to finalize the calculation before they can move on to the next iteration. At $t = 2$, process 1 finishes the evaluation, and all four of the processes move on to the next best point, where $f(x^{k+1}) = 13$ (Kolda and Torczon, 2003).

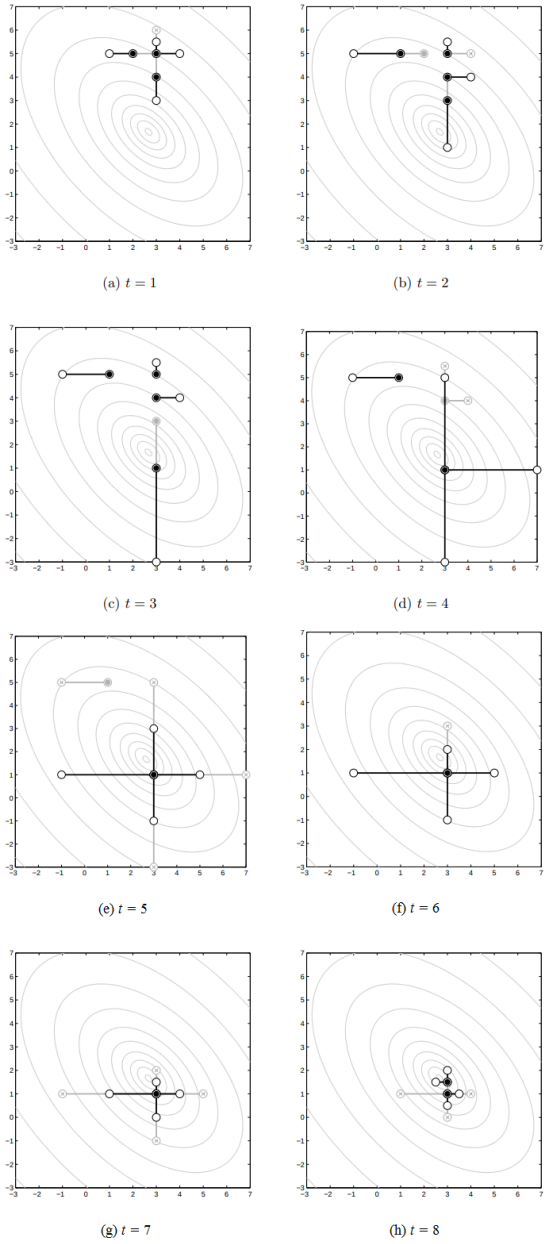


Figure 2.10: Illustration of asynchronous parallel pattern search applied to find the minimum a function $f(x, y)$. Unlike PPS, this method allows the processes to evaluate a new trial point as soon as they finish a function evaluation. Each process evaluates a trial point, communicates the results to all other processes, and moves on to the next known best location in the search space (Kolda and Torczon, 2003).

Particle Swarm Optimization (PSO). The PSO is a rather simple and powerful stochastic algorithm that was first developed to be used in artificial intelligence applications (Eberhart and Kennedy, 1995). However, it was observed that this algorithm can be applied to solve optimization problems as well. The algorithm, which the pseudo-code is presented in Algorithm[2], solves the problem by generating candidate solutions, also known as the *particles*, in the search space and moves them around according to mathematical rules using the particle's position and velocity. In order to solve the problem in the equation [2.12], the PSO algorithm can be formulated as (Kennedy and Mendes, 2002)

$$\begin{aligned}\mathbf{v}_i(k+1) = \mathbf{v}_i(k) + c_1\rho_1(k)(\mathbf{p}_{l,i}(k) - \mathbf{x}_i(k)) + \\ + c_2\rho_2(k)(\mathbf{p}_{g,i}(k) - \mathbf{x}_i(k))\end{aligned} \tag{2.21}$$

$$\mathbf{x}_i(k+1) = \mathbf{x}_i(k) + \mathbf{v}_i(k+1) \tag{2.22}$$

where $\mathbf{x}_i$ is the location of the *i*th particle of the *k*th generation ($i \in \{1, .., n_p\}$ and $k \in \mathbb{N}$), $\mathbf{v}_i$ is the velocity of the particle, $\mathbf{p}_{l,i}(k)$ is the best location of the *i*th particle over all generations, and $\mathbf{p}_{g,i}(k)$ is the location of the best particle in the neighborhood of the *i*th particle over all generations. The coefficients c_1 and c_2 stand for cognitive and social acceleration factors respectively.

The PSO algorithm takes the following steps in order to find the optimum point of the search space (Sharaf and Elgammal, 2018):

1. Generate the initial swarm by positioning the particles in the search space ($\mathcal{S}$) in a random order, where each particle has a random velocity.

2. Evaluate the function value of each particle in the swarm, also known as the *fitness* of each particle.

3. For each particle: $if\ p_{l,i}(k+1) > p_{l,i}(k)$, then set $p_{l,i}(k+1)$ as the new best location value and update the location of the particle.

4. Compare $p_{l,i}(k+1)$ of all the particles in the swarm and update $p_{g,i}(k+1)$ as the new best global location of the swarm.

5. Change the velocity (accelerate) of each particle towards its $p_{l,i}(k+1)$ and $p_{g,i}(k+1)$, both weighted by a random coefficient.

6. Repeat from step 2 until the convergence to the optimum is reached, where $p_{l,i}(k)$ of all the particles in the swarm is equal to $p_{g,i}(k)$.

Algorithm 2 Pseudo-code of the PSO adapted from (Brownlee, 2011)

procedure PSO($\mathcal{S}$, $N_{particles}$, k_{max}, Δ_{tol}) ▷ Define the search space, number of particles in swarm, maximum number of iterations, and deviation tolerance
  $Population \leftarrow \emptyset$
  for ($i = 1$ **to** $N_{particles}$) ▷ Initialize the swarm
    $P_{velocity} \leftarrow randomVelocity(i)$ ▷ Calculate a random velocity
    $P_{position} \leftarrow randomPosition(i)$ ▷ Calculate a random position
    $P_{p\_best} \leftarrow P_{position}$ ▷ Set the particle positions
    if $f(P_{p\_best}) \leq f(P_{g\_best})$ ▷ Check the fitness of each particle
      $P_{g\_best} \leftarrow P_{p\_best}$ ▷ Set the initial global best position
    end
    $Population \leftarrow$ (P$_{velocity}$ & $P_{position}$) ▷ Populate the swarm
  end
  $k = 0$
  while $\Delta > \Delta_{tol}$ **and** $k < k_{max}$ **do** ▷ Set termination conditions of the loop
    for ($P \in Population$)
      $P_{velocity} \leftarrow UpdateVelocity$(P$_{velocity}$, P_{g_best}, P_{p_best})
      $P_{position} \leftarrow UpdatePosition$(P$_{position}$, $P_{velocity}$)
      if ($f(P_{position}) \leq f(P_{p\_best})$) ▷ Check the fitness of each particle
        $P_{p\_best} \leftarrow P_{position}$
        if $f(P_{p\_best}) \leq f(P_{g\_best})$) ▷ Check the fitness of the swarm
          $P_{g\_best} \leftarrow P_{p\_best}$
        end
      end
    end
  $k \leftarrow +1$ ▷ Update the iteration count
  $\Delta \leftarrow \frac{P^{k}_{g\_best} - P^{k-1}_{g\_best}}{P^{k-1}_{g\_best}}$ ▷ Update the relative difference
  end
  Return:(P_{g_best})

The performance of the PSO algorithm mainly depends on the selected parameters for the search of the global optimum location of the search space. In the basic configuration of the PSO algorithm, all particles are allowed to communicate with all other particles. In such setting, the topology of the swarm is global, where all particles in the swarm share the same global best position. Using global topology as a particle communication structure might lead the swarm to get trapped in a local optimum instead of the global optimum.

One of the commonly used topology types in practice is *von Neuman* topology configuration, where each particle communicates with its four neighbors: neighbors above, below, and on each side on a two-dimensional lattice (Kennedy and Mendes, 2002). The other important parameter to choose is the number of particles in the swarm. Too many particles will lead to an excessive amount of objective function evaluations making the process computationally expensive, while on the other hand, swarm with a few particles might not converge to the optimum at all. A rule of thumb is to choose the population size directly proportional to the number variables in the objective function.

2.3 FieldOpt

As it was mentioned in section 2.1, the field development optimization workflow involves building multiple potential development scenarios, simulating these scenarios, and comparing the simulation results based on their future performance. Traditionally, these tasks are executed manually, which usually requires a high engineering effort, is less productive, and often prone to errors. FieldOpt is a software designed to automate and increase the productivity of the petroleum field development workflow. It is currently being developed by NTNU Petroleum Cybernetics Group (PCG) aimed for MSc. and Ph.D. students to conduct research. The FieldOpt is an open source platform, and the source code is available in GitHub:
https://github.com/PetroleumCyberneticsGroup/FieldOpt

This programming framework automates the manual workflow of comparing the performance of different field development scenarios, which is a repetitive process in nature, by integrating mathematical optimization pro-

cedures with reservoir simulation. FieldOpt is implemented in the C++ programming language. It has a modular architecture, i.e. it is composed of separate components, also referred to as *modules*, that are connected.

The main advantage of such software architecture is that any module in the system can be replaced or new modules can be added without affecting the rest of the system. FieldOpt was developed by keeping these quality attributes in mind: *modifiability, performance, scalability,* and *testability* (Baumann, 2015).

Modifiability dictates how easy it is to modify the system. A common strategy to improve the modifiability of a system is to reduce the size of modules, increase cohesion in the modules, and reduce the dependency of modules from all other modules.

Performance indicates how stable is the software under a particular workload. The performance of FieldOpt largely depends on the third-party applications, such as a reservoir simulator. Therefore, efficient execution of reservoir simulation can enhance the performance of FieldOpt.

Scalability is the property of a system to handle the additional workload by adding resources to the system. Enabling the software to exploit a larger amount of computational resources, such as allowing it to use multiple cores, can increase the scalability of the system significantly.

Testability indicates how easy it is to find faults in the system in a given test context. A common strategy to improve the testability of the system is to reduce the complexity of the modules so that each module can be tested for faults independently from the other modules. The testability and the modifiability of the system are inherently connected to one another, as reducing the complexity of module usually results in a reduction of its size.

Overall, all these characteristics mentioned above make FieldOpt easily adaptable to a wide variety of petroleum field optimization problems, testing new field development techniques, and prototyping new optimization algorithms. In general, FieldOpt operates by taking in a driver file and outputs a solution file and log files (see figure 2.11). It is composed of four primary modules:

- **Model**
- **Runner**
- **Optimization**
- **Simulation**

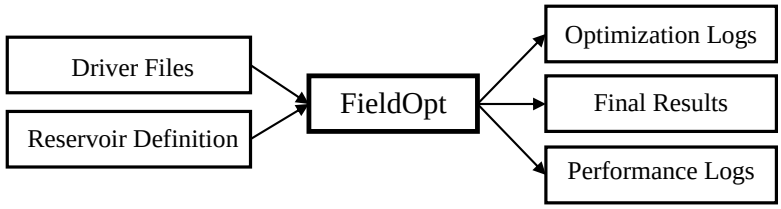


Figure 2.11: How FieldOpt operates from users perspective. Figure adapted from (Baumann, 2015)

The ***Model*** module contains all variable properties of the field development plan, such as well placements, completion configuration, production constraints, and all other properties required by the simulator. The *Model* encompasses the *Well objects*, where each *Well object* further contains Trajectory, a list of Control objects, and some other objects that needed to describe a functional well (see figure 2.12). The properties contained in this module are communicated to other modules through the *Case objects*.

The purpose of the ***Runner*** module is to drive the optimization loop of the FieldOpt (see figure 2.13). It initializes FieldOpt by reading the runtime settings provided through the input files. Once the run is initialized, the runner starts the optimization loop, evaluates the results, and checks if the termination conditions are met. It also contains the Bookkeeper class, where optimization algorithms submit Case objects to it, as well as check if a Case object has already been submitted to the bookkeeper. The Runner has the capability to execute a serial optimization loop, where simulations are run in sequential order, or a parallel optimization loop, where simulations are performed concurrently.

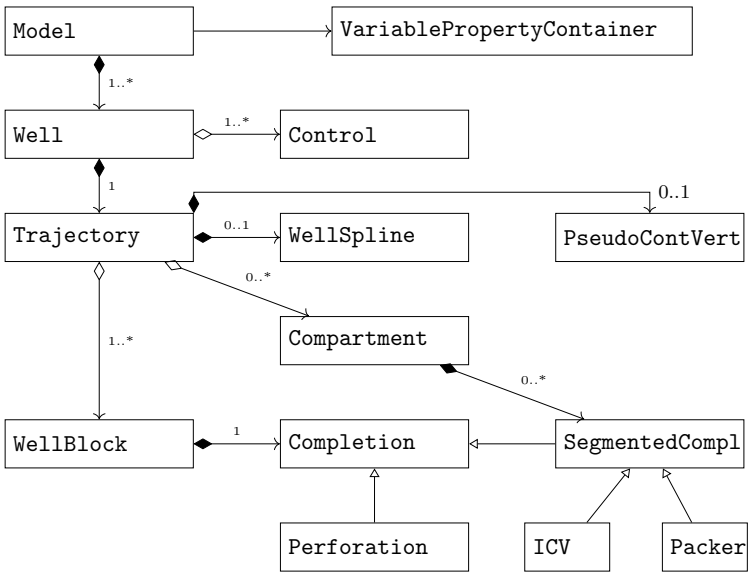


Figure 2.12: The figure illustrates the class diagram for the Model module (Baumann et al., 2019).

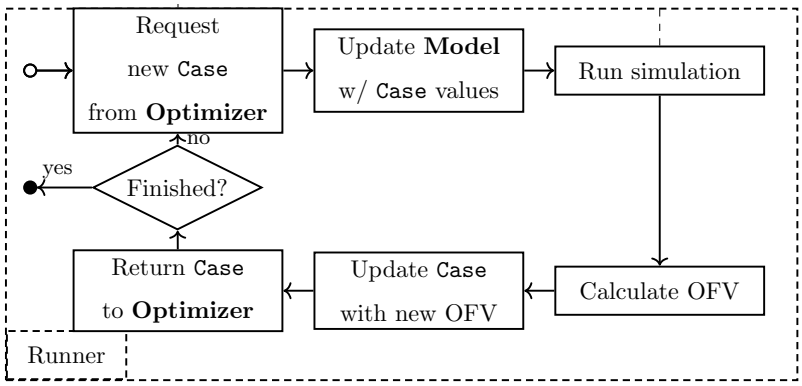


Figure 2.13: The figure illustrates the serial optimization loop of the Runner module (Baumann et al., 2019).

The ***Optimization*** module handles the algorithm implementation, search space constraints, objective function evaluations, and the Case class in FieldOpt. The Case class contains a full set of variable values of a specific perturbation of the model, including the post-processed simulation results, such as the objective function value. Each variable in this class is assigned with a universally unique identifier (UUID). Looking at the figure[2.14], it can be observed that the Optimizer class is the parent class to all optimizers, and has *ConstraintHandler* and *CaseHandler* as its supporting classes. These classes provide the functionality to all of the optimization algorithms in the Optimizer class, such as case logging, evaluation of new candidate cases, and application of constraints. Such a module structure eases the process of adding new optimization algorithms and constraints to FieldOpt since no changes outside of the Optimization module is required.

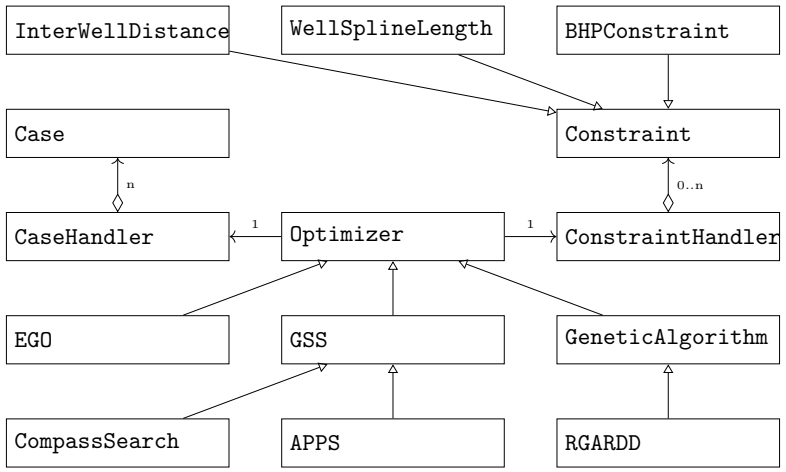


Figure 2.14: The figure illustrates the class diagram of the Optimization module (Baumann et al., 2019).

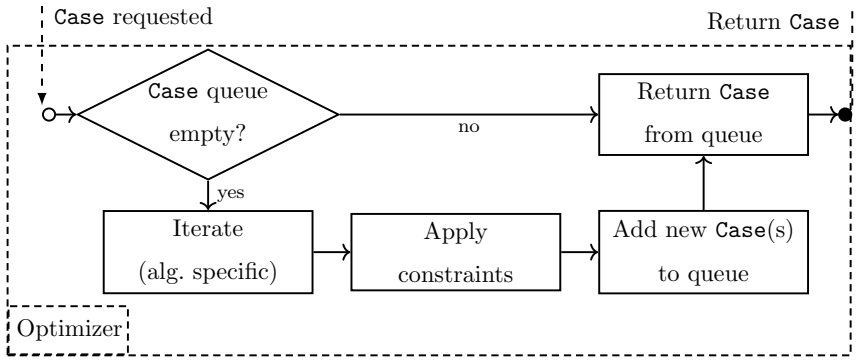


Figure 2.15: The figure shows the simplified version of the optimization loop of the Optimizer module (Baumann et al., 2019).

The function of the ***Simulation*** module is to launch the simulations and read the results. The code contained in this module launches reservoir simulations by executing shell scripts in the child processes. When asked by the Runner module to evaluate the current state of the Model, the well information is taken from the Model to write a simulator input for the well blocks, controls, and completions. Once the reservoir simulation is ready to be run, a shell script is executed to start the simulation. Finally, when the simulation is finalized, the results are read from the disk and communicated to other modules for further processing to evaluate the objective function.

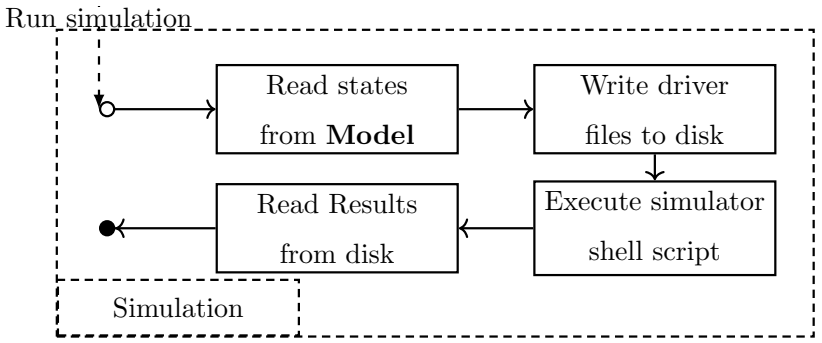


Figure 2.16: The figure illustrates the processes performed by the Simulation module (Baumann et al., 2019).

Currently, the reservoir simulators supported by the FieldOpt software are:

- **ECLIPSE 100/300.** Supported features: well control, well placement, and well completions optimization.
- **Flow.** Supported features: well control, and well placement optimization.
- **AD-GPRS.** Supported features: well control, and well placement optimization.
- **INTERSECT.** Supported features: well control, and well completions optimization.

Driver Configuration File. FieldOpt is configured through a driver file, where this file should include the input and output directories, the data configuration and grid file of the reservoir simulation model, as well as specification of which reservoir simulator to use for the run. The FieldOpt driver file contains this information in separate objects; namely the Global, Optimizer, Simulator, and Model objects. This file is designed to be in a JavaScript Object Notation (JSON) format. Aforementioned file format is easy to read and write by humans as well as easy to parse and write it by machines because the information is well organized and stored in a particular structure.

Chapter 3

Methodology

Oil and gas companies rely on advanced subsurface software to develop and manage their assets successfully. Currently, there are many software packages capable of performing such tasks. However, some software suits are more popular and frequently used throughout the industry. Schlumberger's reservoir simulator ECLIPSE (Eclipse) would be a fitting example for such subsurface software packages. Eclipse is a reservoir simulator that is well known and widely used by reservoir engineers because of its technical capabilities, continuous support and development, and ease of use. However, Eclipse is a proprietary (also known as closed-source licensed) software, meaning the end-user can only buy the right to use the software for a certain amount of time but not the software itself. In this type of software distribution, the source code of the software is claimed to be a trade secret by the manufacturer, and the end-user is not allowed to access the inner workings of the program (Kristoffersen, 2017). An alternative to the closed-source software environment is the open-source environment.

Unlike the closed-source environment, software manufacturers that operate in the open-source environment allow their products to be freely used, modified, and shared by the end-users. Such an environment allows free access to the inner workings of the software so that the end-users can fix bugs, make modifications to improve their workflow, and contribute to the future development of the used product. An example of an open-source licensed reservoir simulation software package is the OPM-Flow (Flow) reservoir simulator. This simulator was made and is continuously developed by the

Open Porous Media (OPM) Initiative, which is a organization that coordinates collaborative software development, maintains and distributes open-source software for petroleum field development applications (OPM, 2017).

As it was mentioned in the introduction chapter, the cost of automatic optimization techniques in the oil and gas industry is mainly associated with the subsurface software license fees, such as reservoir simulator license fees. However, using an open-source licensed software not only reduces the cost of the automatic optimization projects, but also provides programming flexibility so that the end-user can custom fit the software to their specific needs to improve the project workflow. For example, Schlumberger charges extra for the Eclipse license fee which enables running simulations in parallel (ensemble) in multi-core systems compared to the one-core license, where reservoir simulations can only be run in series. On the other hand, Flow is equipped with Message-Passing Interface (MPI) that enables two or more processors to communicate. This feature allows Flow to run simulations in parallel if more than one core is available for its disposal and such a setup can reduce the computational time significantly (Kristoffersen, 2017). Therefore, running ensembles in Flow with a multi-core setup instead of in Eclipse will reduce cost in terms of license fees and computational time for the end-user. These points inspired the first part of this thesis, where the Eclipse model of the conventional oil reservoir was converted to run in Flow.

The software tools that were utilized for generating and post-processing the results in this work are presented in Table 3.1. It is worth mentioning that except for Eclipse, these software tools have open-source licenses.

Table 3.1: Tools used in this work.

Software		Purpose
Flow	(v.2018.10)	Reservoir simulations
FieldOpt	(v.1.0.0)	Optimization
ResInsight	(v.2019.04)	Visualization & result post-processing
Eclipse	(v.2016.2)	Reservoir simulations

The first part of this chapter includes general information about the reservoir and its current development plan, as well as the methodology used for

converting the simulation model. The next part of this chapter focuses on the optimization methodology utilized to find the optimum well placement in this reservoir. The results and interpretations of both of these sections will be presented in the next chapter.

3.1 Field Introduction

The oil field that was used in this case study is located in the NCS. The exact location, depth, and name of the reservoir are regarded as confidential information by the owner of the field. Therefore, from now on, the field where the reservoir is located will be referred to as "the Field" in this report. The Field was discovered in the first decade of 2000 by four exploration wells (see figure 3.1), and as of today, commercial oil production has not yet started from this reservoir. Thus, no production history data is available except for the production tests performed on appraisal wells. The simulation model of this reservoir was built based on the information gathered from the exploration wells, such as well logs and pressure transient tests. The base case development plan is to drain the reservoir via four horizontal oil production wells, and two water injection wells. Moreover, this plan assumes that there will be no pressure support from the aquifer during production, and because of this, the purpose of water injection is to both provide pressure support and increase the sweep efficiency in the reservoir.

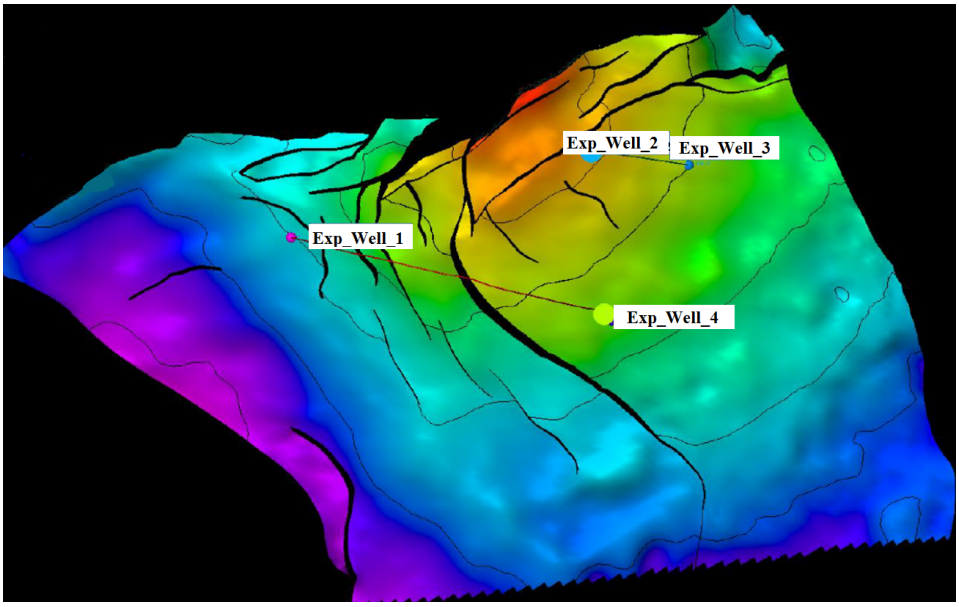


Figure 3.1: The figure illustrates the top of the reservoir based on the depth geomodel. The faults are visible as the black polygons. The map also depicts the location of the four exploration wells, marked as **Exp_Well_1, 2, 3,** and **4**, drilled into the reservoir. *Figure adapted from Ross Offshore AS.*

Reservoir structure. The Field is composed of four sedimentary formations laying on top of each other sequentially. These formations will be referred to as Formation One, Two, Three, and Four throughout this document. Formation One is the upper formation and Formation Four is the lower formation (see figure 3.2).

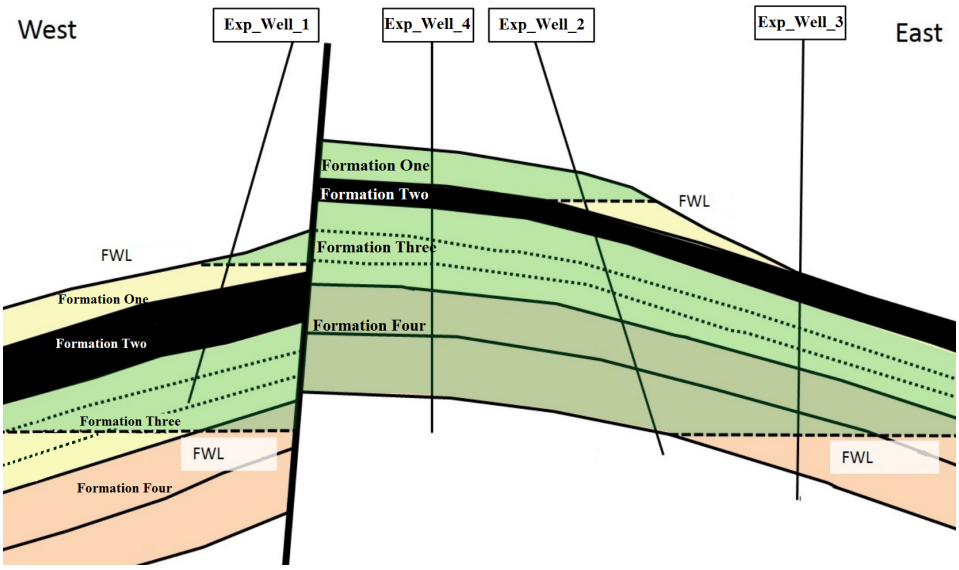


Figure 3.2: This figure depicts the cross-section of the Field looking from south to north. Formation Two (marked as the black layer) is the coal layer that separates the Formation One from the other two formations at the bottom. The figure also illustrates the central fault located in the middle of the Field. This fault is a sealing-fault, and it divides the Field into Western and Eastern segments. *Figure adapted from Ross Offshore AS.*

Each of these formations has distinct reservoir characteristics, where these characteristics significantly affect the fluid dynamics in the reservoir. Hence, having a clear description of the rock formation in any hydrocarbon bearing reservoir is a crucial ingredient for designing a successful development plan for a petroleum field. The description of each of the formations constituting the Field are as following:

- ***Formation One.*** The depositional environment of this formation is interpreted to be shallow marine where it varies from distributary channels to lower shoreface. The deposits are comprised of light brown, poorly sorted, fine to coarse-grained, and slightly silty sandstones. The flow channels commonly consist of fine sediments. The formation thickness varies from approximately 35 meters in the western segment to 28 meters in the eastern segment. The average Net-to-Gross (NTG) is around 75%.

- ***Formation Two.*** This formation consists of a massive black coal layer in the upper part and contains some interbedded layers of sand,

with NTG ratio of 3%, in the lower part. Therefore, Formation Two acts as a sealing layer between the Formation One and Formation Three. The depositional environment of the coal section is interpreted to be a coastal swamp or a bay setting. The thickness of this formation varies from approximately 55 meters in the western segment to 14 meters in the eastern segment.

- ***Formation Three.*** This formation comprises a thick sandstone layer, which is subdivided into three subzones based on the well logs and stratigraphic picking. The bottom part represents a vertically stacked channel system deposited in a coastal swamp or a bay setting. The middle part also consists of a sandstone layer, which is interpreted to be deposited in a terrestrial fluvial setting. The upper part consists of fluvial deposits influenced by marine input suggesting a depositional environment to be a coast. Therefore, the top and middle parts of this formation has a channeled structure. The thickness of the Formation Three is relatively constant, and it is approximately 20 meters thick throughout the reservoir. The average NTG ratio of the whole formation is estimated to be around 40%.

- ***Formation Four.*** This formation consists of 2 broad systems based on their interpreted depositional environments: a fluvial distributary system and an Aeolian dune belts. The fluvial system deposits are typically comprised of light brown colored, poor to moderately sorted, fine to medium grained, and silty sandstones. The Aeolian system, on the hand, is composed of light brown colored, moderately sorted, very fine grained, and cross-stratified sandstone layer. The approximate thickness of the Formation Four in the eastern segment is around 30 meters. The average NTG ratio is around 60% for the whole formation.

Furthermore, the Field is limited by faults on three sides; north, west, and east. Pressure transient analysis indicates that these faults are sealing, which means that there might not be additional aquifer pressure support across them during the production. On the southern side, there is no fault limiting the model, and an aquifer could act as pressure support from this side. However, the base case field development scenario does not include aquifer support since the presence of an aquifer is highly uncertain. Therefore, the only way to maintain the reservoir pressure in the Field is through

the water injection.

Another critical parameter to consider while designing a field development plan is the pressure communication between the formations. The presence of pressure communication between the two rock structures indicates that fluids contained in these structures can flow from one to another while having no pressure communication means that fluids can not flow between the formations. This parameter plays a central role when designing the pressure support and injection strategy of the field.

The well tests carried out in the exploration wells drilled in the Field indicated that there is a pressure communication between the Formation Three and Formation Four. However, no pressure communication was observed between the Formation One and lower formations. In theory, this means that the reservoir pressure in the Formations Three and Four can be supported by one common injector instead of having two injection wells for each of these formations. Considering the cost of drilling and injecting fluids on an offshore platform, the former option will save a significant amount of capital in the project. On the other hand, Formation One requires a dedicated water injector for pressure support. Another important geological feature to note is that there is no pressure communication between the western and eastern segments of the Field, since the central fault is a sealing fault.

Reservoir fluids. The Field formations contain undersaturated oil, meaning there is no free gas in the reservoir at the initial conditions. The oil has a relatively low viscosity of around 1.4 cP at reservoir conditions. Such a low oil viscosity is especially advantageous in water flooding projects. The low mobility ratio prevents the injected water from rushing past the oil in the reservoir because both of the fluids have similar mobilities in this case. In the case of the Field, it is likely that the displacement of oil by the injected water will be effective in terms of sweep efficiency, and it can be expected that water breakthrough will not occur at the early stages of the production phase.

Eclipse Model of the Field. The original simulation model of the Field was built in Petrel (Schlumberger), which is a software that serves as a base platform to communicate multi-disciplinary subsurface information for vi-

sualization and simulation, and it was intended to be run in Eclipse 100. This model consists of 1100736 $(112 \times 117 \times 86)$ grid cells, where only 188051 of these grid cells are active. The fluid model applied to replicate the fluid behavior in the reservoir was the black-oil model, i.e., a model where the composition of the fluids in the system is assumed to be constant. The base case scenario is designed with the production time frame of 25 years, starting in early 2015. In this work, only one geological realization of the Field is employed, and the reservoir model used in this work is the facies based model.

Initial drainage strategy. The initial drainage plan includes four oil producer wells and two water injector wells, where all of these six wells are located eighter in the Formation Three or Formation Four (see figure 3.3). The specific drilling schedule has not been taken into account in this time frame. Therefore, all of the wells in the model get online on the first day of the simulation.

An important point worth mentioning is the well control schedule of the base case development plan. In general, all wells in a reservoir simulation model are operated by a set of scheduled controls. A basic control schedule of a well should specify the preferred phase flowing in the well (e.g., oil, gas, water), its planned state of operation (e.g., open or closed), and if the well flow is primarily controlled by the rate (e.g., oil, gas, water, or liquid rate) or pressure (e.g., well bottom hole pressure). The simulator can switch between the controls of a well during the simulation if the limiting targets (e.g., minimum bottom hole pressure or maximum water production rate), when specified, have been reached (Bellout, 2014).

A more advanced well production control schedule can include gas lift injection parameters, electric submersible pump settings, well group constraints, and field production targets and limits. The field liquid production and injection rate constraints usually originate from the designed fluid handling capacity of the production facilities, such as the separator unit capacity or the storage and transportation unit capacities.

In the base case scenario, the producer wells in the Field are specified to be operated with minimum well bottom hole pressure (BHP) of 60 bars, target liquid production rate of 2000 m^3/day at the surface conditions, and

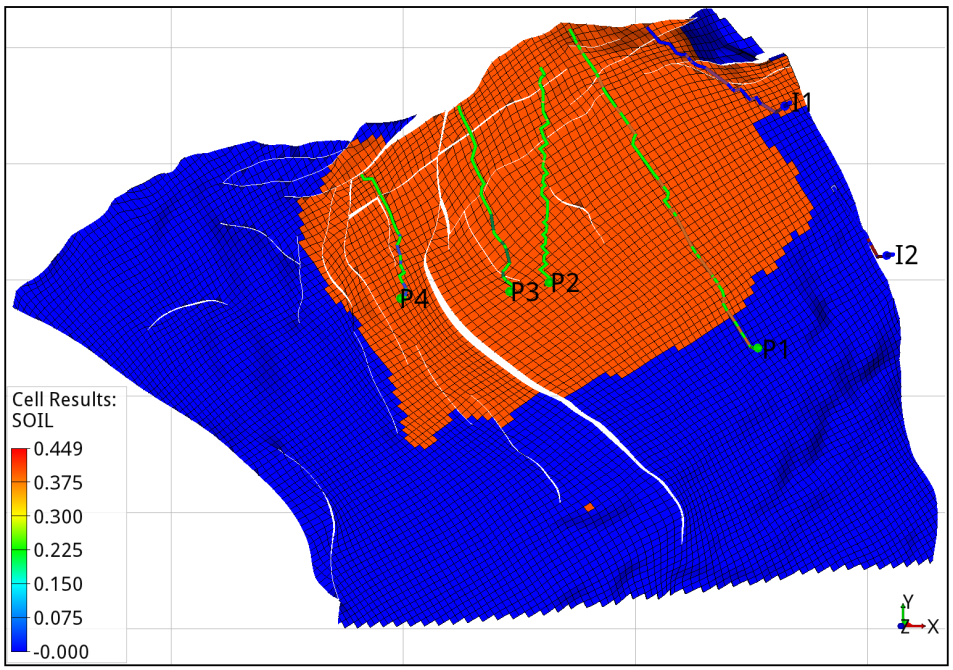


Figure 3.3: The initial saturations of oil (red) and water (blue) in the Field model (K-slice: 40-44). The figure also shows the base case well placement configuration in the reservoir. Notice that the wells are not in the same horizontal plane (viewed from above), and the well trajectories were converted to splines by FieldOpt.

to have gas lift as an artificial lift mechanism. The water injectors, on the other hand, are specified to be operated with water injection rate of 1500 m^3/d at the surface conditions, and the maximum well BHP being 500 bars. This means that the injection wells will inject 1500 m^3 of water per day as long as the well BHP is below 500 bars. If the injector well BHP reaches 500 bars, the simulator will reduce the injection rate accordingly to comply with the well BHP constraint.

In addition, a well group constraint is also imposed on the injection wells, where these injectors should maintain the voidage replacement ratio (VRR) of one at the reservoir conditions. The VRR is the ratio of the total volume of fluids injected into the reservoir to the total volume of fluids withdrawn from the reservoir during the production phase (Clark et al., 2003). Therefore, keeping the VRR close to unity will help to maintain the reservoir pressure throughout the production life of the field, and ultimately increase

the hydrocarbon recovery. In the base case drainage plan, the simulator will automatically regulate the water injection rates of both of the injector wells so that the total water injection volume is equal to the total volume of fluids produced by the four producer wells at the reservoir conditions.

3.1.1 Model conversion and validation

A work model that can be run in Flow was developed based on the Eclipse model of the Field that was provided by the operator company. This work model was used in all of the optimization cases launched by the FieldOpt software. As it was mentioned at the beginning of this chapter, using an open-source reservoir simulator can provide a more efficient workflow compared to the industry-standard reservoir simulator. This section describes the original simulation model of the Field as well as introduces the Flow reservoir simulator and outlines the conversion procedure followed for the development of the work model.

During the conversion phase, the main effort was spent on obtaining a sufficient overlap in the production curves predicted by the original model and the work model. This part of the work was carried out as a specialization project in reservoir engineering during the fall semester in 2018. The results and validation of the work model will be presented in Chapter 4. The final results of this process will be presented and discussed in the next chapter (see Chapter 4.1.1).

OPM-Flow reservoir simulator. Flow is a reservoir simulator that does not require a license to use it. Also, no additional license is required for running reservoir simulations in parallel. This means that the optimization algorithms implemented in the FieldOpt software can launch multiple reservoir simulation runs without any license requirement. However, the parallel implementation of the simulation runs are still limited by a number of other factors, such as the number of processors available, server load, and the constraints on read-and-write access to the disk on the server hardware (Bellout, 2014). In this work, OPM-Flow version 2018.10 was used.

Conversion method. All data needed by both Eclipse and Flow to execute a simulation should be collected in an input data file, which is a ASCII

file. These reservoir simulators identify different data items by keywords, which are usually followed by its associated data. Eclipse's input data file consists of eight data sections, where each keyword belongs to only one of these data sections. If a required keyword is missing in the input file or placed in a data section other than it belongs, the simulator will raise an error and end the simulation run. For more detailed information about the input syntax of Eclipse and Flow reservoir simulators, the reader is encouraged to refer to the user manual of these simulators (Schlumberger, 2014; OPM, 2018).

Although the syntax is usually unique for each simulator, the general set-up of input files for most of the reservoir simulators is very similar. The input data syntax required for Flow is very similar to the input data syntax required for Eclipse 100, where the input file for both of the simulators consists of the same data sections. Even though this makes converting the input data deck from one simulator to another much easier, Flows current keyword library is not as extensive as the ECLIPSE 100's keyword library. The keywords that are not recognized by Flow needs to be replaced with a supported keyword since the unrecognized keywords in the input deck may prevent initialization of the simulation run.

The work model that was converted from the original reservoir model of the Field required several approximations in terms of model parameters and simulator functions. These approximations include modifications to the saturation functions (e.g., relative permeability tables), well group control handling, and artificial lift mechanisms (e.g., gas lift injection). In general, the model conversion and validation process was focused on finding how these approximations affect the simulation outputs, and how to minimize the difference between the production curves predicted by the work model and the original model.

The model conversion process started with reducing the original input data deck to the bare minimum data that is required for initialization of the simulation to run in Flow by removing all of the extra functionalities (e.g., lift gas injection, well group controls) and include files (e.g., fault coordinates). The main idea behind this approach was to start with a very simple input deck that generates the same production curves when run in both of the simulators. Step-by-step, as the previously removed data items were added

back to the input deck, the data items that are not recognized or do not work as intended in Flow reservoir simulator were identified. At each step, the original Eclipse model was compared to different work model configurations for a number of production quantities, such as the field reservoir pressure (FPR), well BHPs, field production rates and totals (FFPR and FFPT), and well rate curves.

The rest of this subsection will present the details regarding the modifications and approximations made to the original Eclipse reservoir model that were required for the conversion to the work model in Flow. Each paragraph will start with a short introduction about the data section, which the modified keyword item belongs to in the input data file, and followed by the specifics of how these keywords were tuned.

RUNSPEC section. This section includes the description of the simulation run, such as the units, grid size, table sizes, number of wells in the model, and fluid phases included in the system. The following items in this section where modified for model conversion:

- *Petrel Options (PETOPTS).* This keyword indicates that the geometry files (exported from Petrel) contain transmissibility and pore volume data, where these parameters are usually provided separately from the geometry files. Typically, this option is used in Petrel when modeling a reservoir with a complex rock structure. This feature is designed to reduce the error in the simulation calculations in Eclipse due to the complex geometry of the grid structure, such as faults and pinchouts (Schlumberger, 2014). However, this leads geometry files to be exported in *GSG* file format, where Flow cannot read the data from such files. The original Eclipse model of the Field includes this item because the reservoir structure contains a significant amount of faults. Therefore, the PETOPTS option was deactivated and a new Eclipse model of the Field, with grid geometry file in GRDECL format, was exported from Petrel.

- *Constrained Pressure Residual Solver (CPR).* The default linear equation solvers may fail to converge in complex reservoir simulation models. Such cases might lead to a significant numerical error in the outcome and longer simulation run-time. Eclipse uses the CPR linear solver to reduce the run time and improve the performance of

the simulation when the linear equations fail to converge frequently. Although Flow also has this capability, the CPR solver feature on the version 2018.10 is implemented at the experimental level, and has an adverse effect on the run-time of the simulations. Therefore, this option was disabled in the work model.

GRID section. This section contains the basic geometry of the simulation grid and the rock properties, such as the porosity, permeability, net-to-gross ratios in each grid cell. The simulator uses this information to calculate the grid block pore volumes, grid center depths, and inter-block transmissibilities of the model. The items that were modified during the model conversion are the following:

- *Fault Threshold Pressure (THPRESFT).* In Eclipse reservoir models, it is possible to set threshold pressure for a fault, where the flow occurring between adjacent grid blocks on each side of the fault is prevented until the pressure difference in these grids exceeds the specified threshold pressure value. Some of the faults in the original Eclipse model of the Field have threshold pressure value of zero, which is equal to the default value used for all faults in Eclipse models if no threshold pressure value is specified. This feature is not supported in Flow. Therefore, it was removed from the work model of the Field.

PROPS section. This section of the input data file contains the pressure and saturation dependent properties of the reservoir fluids and rocks, such as the relative permeability tables, fluids compressibility tables, and rock compressibility tables. The following items where modified in this section:

- *Oil Properties (PVTO tables).* This table contains the properties of the oil phase (with dissolved gas in it) in the simulation model. It is composed of four columns, where each column contains the dissolved gas-oil ratio, bubble point pressure, oil formation volume factor, and saturated oil viscosity. The input syntax of both of the simulators require that the data entries in the second and third columns of this table to be entered in increasing order. However, some of the data entries in the original Eclipse model of the Field violates this rule. In such cases, Eclipse issues a warning message about this problem in the simulation run-log and continues the simulation. Flow, on the

other, issues an error message and stops the simulation due to violation of the input syntax. This issue was fixed by re-arranging data entries in increasing order in the work model so that Flow accepts the input.

- *End-point Scaling (SCALECRS).* This keyword is used for end-point scaling of the relative permeability curves of the fluids in the system. Relative permeability curves are used for predicting the mobility of each phase in the system during reservoir drainage. End-point scaling is an option that allows the use of the same normalized relative permeability curve to a set of different regions and rock types in the reservoir while still honoring the variations in the rock properties, such as connate water saturation and maximum values of relative permeabilities to flowing phases (Schlumberger, 2014). The connate water saturation in the original Eclipse model is scaled to be 60%, meaning that the water phase is immobile in the grid cells with less 60% water saturation. Although Flow does support end-point scaling of the relative permeability curves, it does not work as intended because the simulation does not initialize in Flow when *SCALECRS* parameter is set to 60%. Therefore, the end-points of the relative permeability curves in the converted work model was scaled manually to match them to the scaled relative permeability curves in the original model.

SCHEDULE section. This section specifies the operations of the wells in the simulated run, such as the well completion data, production and injection controls as well as the times at which output reports are required (Schlumberger, 2014). The following items were modified in this section:

- *Well Group Controls.* Group constraints can be used for imposing an upper limit for production or injection parameters on a selected group of wells to manage the overall production of a field. For example, if an upper limit for group gas production rate is exceeded, the simulator will reduce the fluid production from the wells with high gas-oil ratio (GOR) to keep the group gas production rate within the predefined limits. Similarly, group controls can also be imposed on injector wells. In the original Eclipse model of the Field, both of the injection wells are under group control to maintain VRR equal to one, where the simulator sets the injection rates automatically so that total

water volume injected is equal to the total volume of fluids produced from the field at the reservoir conditions. However, this feature does not work as intended in Flow because of a bug in the software algorithm. Therefore, well group controls were disabled in the converted work model.

- *Gas Lift (AQL).* Artificial gas lift is an advanced production technique, where gas is injected into the production well to reduce the density of the fluid column. As a result, a lower pressure differential between reservoir and surface is needed to bring the fluid with less density to the surface. In the original Eclipse model of the Field, all four of the wells include artificial gas lift. Although Flow also supports this feature, it does not work as intended since the well calculations do not converge when the gas lift is activated. Hence, the gas lift option was deactivated for all four of the production wells in the work of the Field.

- *Periodic Testing of Closed Wells (WTEST).* Periodic testing of closed wells can be applied to wells that are closed due to an economic limit during the simulation. The producer wells in the simulation model of the Field have 95% water cut as their upper economic limit. This means the simulator will automatically shut-in a producer well if it's water cut is higher than 0.95. Such limits can be specified via *WECON* keyword for each well individually in Eclipse. If the *WTEST* keyword is included in the SCHEDULE section of a simulation deck, the simulator will test closed wells if they are capable to flow under the current operating conditions at each time-step (Schlumberger, 2014). If a tested well is capable to flow, then that well will be put back to production. Even though Flow supports this feature, it does not work as intended. Thus, the periodic testing of closed wells option was deactivated in the work model of the field.

3.2 Optimization Work

This section introduces the well placement optimization methodology implemented to find improved locations for the four production wells in the Field. Detailed information about the objective function, function parameters, constraints, and implemented optimization algorithms are presented.

3.2.1 Optimization problem

The main problem addressed in this work is to find the optimum locations for the producer wells so that NPV of the Field is maximum. Therefore, the variables in this problem are the four horizontal well locations subject to reservoir boundary constraints. The problem formulation can be presented as

$$\min_{x_p \in R^n} -NPV(x_p) \quad \text{subject to} \quad \Big\{ c_{rb}(x_p) \leqslant 0 \tag{3.1}$$

where x_p and n denote the well placement coordinates and number of objective function variables, respectively. The non-linear constraint on the reservoir bounds, $c_{rb}(x_p)$, defines the region where optimization algorithm should search for the optimum coordinates of the well.

Objective function. The objective function used in this work was the weighted sum of the cumulative fluid production volumes from the reservoir. As it was mentioned in section 2.2.1, the discount rate of future cash flow was assumed to be zero for simplicity. Therefore, the equation (2.10) can be simplified as

$$NPV(x_p) = \sum_{k=1}^{N_s} \left(\sum_{j=1}^{N_p} p_o q_o^{j,k}(x_p) \Delta t_k - \sum_{j=1}^{N_p} c_{wp} q_{wp}^{j,k}(x_p) \Delta t_k - \right. \\ \left. - \sum_{j=1}^{N_i} c_{wi} q_{wi}^{j,k}(x_p) \Delta t_k \right) \tag{3.2}$$

where the Table 3.2 presents the economic parameters used in this equation.

Table 3.2: This table contains the economic parameters used for calculating the objective function. These parameters were adapted from Volkov and Bellout (2017).

Parameter	Value
p_o - Oil price	60 $/bbl
c_{wp} - Water treatment cost	5 $/bbl
c_{wi} - Water injection cost	3 $/bbl

Objective function variables. The objective function variables in this work are the heel and toe coordinates of the well, for which its placement is to be optimized. For each well, the case operator needs to determine six coordinates (three for the heel and three for the toe) since the wells are highly deviated (see figure 3.4). Therefore, the optimization problem where only one horizontal well location needs to be optimized yields $n = 6$ well placement variables ($n = 6 \cdot N_w$ with N_w being the number of wells).

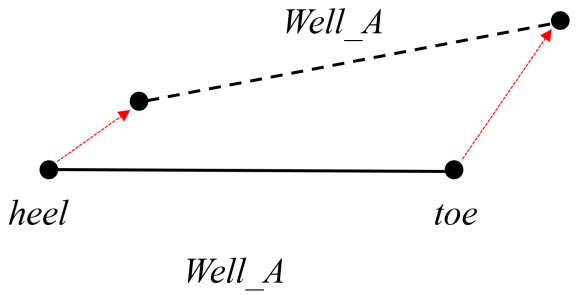


Figure 3.4: The figure illustrates an example of a well movement in the solution space (looking from above) when its placement is optimized. The solid line represents the initial placement of well A. The red arrow denotes the movement direction of the heel and toe of the well towards their new positions determined by the algorithm. The dashed line represents the new position of well A, where the optimization algorithm will evaluate the objective function for this case, and decide whether the new location of well A is better or not.

Reservoir-bound constraints. The reservoir bound constraints are based on not allowing the heel and toe of the well to move outside the predefined region in the reservoir, which contains the initial coordinates of the well. In a case where the heel and toe coordinates lie outside the constrained region, the operator projects hell and toe coordinates onto their respective bounds

(Bellout, 2014). These constraints guide the search for the new well coordinates that are more realistic with respect to the geological features of the reservoir, such as avoiding placing a well too close to a fault. In this work, only the rectangular-shaped reservoir bound constraints were imposed on the well placement variables (see figure 3.5). The bounding region was defined by a range of (i,j,k) indices, that created a box-shaped boundary in (x,y,z) space.

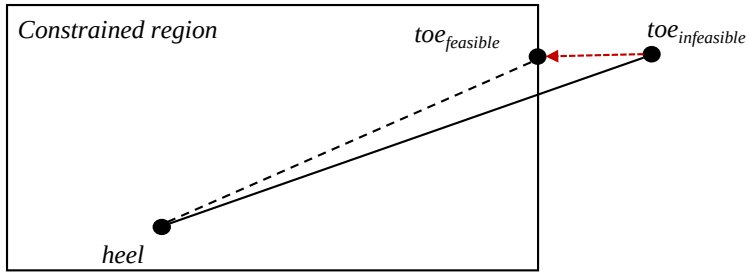


Figure 3.5: The figure illustrates an example of a projection of infeasible well toe coordinate onto the constrained bound area (looking from above). Figure adapted from (Bellout, 2014).

Optimization algorithms. In this work, the optimization algorithms implemented to solve the well placement optimization were the APPS and PSO algorithms (both were introduced in section 2.2.3).

APPS. The APPS algorithm requires the following control parameters to initialize the search for an optimum well location: initial coordinates for both hell and toe of the well, an initial step-length, a convergence criterion, a contraction factor, an expansion factor, and a maximum number evaluations allowed before convergence. The control parameters used for APPS algorithm are presented in Table 3.3.

In the simulation model of the Field, the scale ratio between the z-axis and the x and y-axes is approximately about $1:100$. This makes choosing a suitable initial step-length (and consequently, the convergence criterion) more challenging because there is a high probability that the well coordinates would be outside of the feasible region. Therefore, the *AutoStep* option, which is implemented in FieldOpt's Optimization module, was used.

With *AutoStepInitScale* set to 0.75, the initial step-length is calculated as 0.75 times the distance between the minimum and maximum values defined by the bound constraints along each axis. The same principle applies for defining the minimum step-length for convergence criterion with *AutoStepConvScale* setting.

Table 3.3: This table presents the APPS algorithm's control parameter specifications applied for the well placement optimization problems in this work. Both initial step-length and convergence criterion were calculated automatically based on the bound region dimensions. After every successful iteration, the step length was kept the same, i.e. the expansion factor was set to 1. On the other hand, the step-length was reduced by half if the iteration did not find a better position, and the algorithm converged to a solution once the step-length was 0.01 times the bound region dimensions. The algorithm was allowed to execute a maximum of 1000 case evaluations, and the search automatically terminated if the algorithm did not find the optimal well placement coordinates within these evaluations.

Parameter	**Value**
$AutoStepConvScale$	0.01
$AutoStepInitScale$	0.75
$AutoStepLengths$	true
$Contraction\ Factor$	0.5
$Expansion\ Factor$	1
$Maximum\ Evaluations$	1000
$Maximum\ QueueSize$	2

In general, choosing proper control parameters for an optimization algorithm is not always a straightforward task, and even can be very challenging in some problems, e.g. simulation based optimization problems. Although there are commonly used rules of thumb for some algorithms, the control parameters that are picked based on these rules do not guarantee an efficient performance of an optimization algorithm. Therefore, a sensitivity analysis needs to be performed to determine the suitable control parameters of an optimization algorithm for each optimization problem.

In this work, the initial step-length parameter was tested to assess the performance of the APPS algorithm in the well placement optimization problem of the Field. The main reason for testing this parameter was to determine the relationship between the initial step-length parameter and the number of objective function evaluations executed by the algorithm before

converging to a solution. It was expected that a longer initial step-length parameter would yield a faster convergence, i.e. lower number of function evaluations executed, as well as the algorithm would have a lower probability of getting trapped in a local optimum. Therefore, three different values for *AutoStepInitScale* were tested for each optimization case: 0.25, 0.50, and 0.75.

PSO. The PSO algorithm requires the following control parameters to initialize the optimization of a well location: number of particles in the swarm, the social and cognitive learning factors, and a maximum number of swarm generations. The control parameters used for the PSO algorithm in this work are presented in Table 3.4. Due to the significant difference in scale between the z-axis and the x and y-axes in the reservoir model, the velocity of each particle was scaled down by 25% ($Velocity\ Scale$ setting) at each iteration to reduce the probability of well coordinates being outside the feasible region. The cognitive and social acceleration factors were chosen such that $c_1 + c_2 = 4$, based on Shi et al.'s paper (Shi et al., 2001). These factors weigh the stochastic acceleration terms that pull each particle toward P_{p_best} and P_{g_best} coordinates in the bound region. Consequently, low values for these factors allow particles to travel far from the target regions before converging to the best position, while high values result in an abrupt movement toward, or past, the best position. Hence, both acceleration factors were chosen to be equal to 2 in this work.

One of the critical control parameters to decide in the PSO algorithm is the number of particles in a swarm since this parameter influences how many objective function evaluations will be executed during the search process. A general rule of thumb is to choose the number of particles directly proportional to the number of variables in the objective function, i.e., a function with a higher number of variables require a swarm with higher amount of particles for the efficient performance of the PSO algorithm. Therefore, a well placement problem of a horizontal well (six coordinate variables) would require around six particles in the swarm, and a swarm with a higher number of particles is needed if more than one horizontal well placement is optimized by the PSO algorithm.

Table 3.4: This table presents the control parameters specified for the PSO algorithm in this work. The algorithm generates a swarm with six particles to search the bound space. The velocity of each particle was multiplied with 0.25 to prevent the abrupt movement of the particles. A maximum of 138 iterations were allowed for each search.

Parameter	Value
$Max\ Generations$	138
$Cognitive\ Factor:\ c_1$	2
$Social\ Factor:\ c_2$	2
$Swarm\ Size:\ N_{particles}$	6
$Velocity\ Scale$	0.25

In this work, a sensitivity to the swarm size parameter (number of particles) was tested to asses the performance of the PSO algorithm. The main reason for testing this parameter was to determine the influence of this parameter on the total number of objective function evaluations and the final objective function value obtained. Therefore, two different values for swarm size ($N_{particles}$) parameter were tested for each optimization case: 6 and 12.

3.2.2 FieldOpt configuration

This section gives a brief overview of the optimization workflow that was used to optimize the well placement of the Field and describes how the various features of FieldOpt were employed in this work. In addition, relevant parts of the FieldOpt driver file are also presented as code snippets throughout this section.

Optimization workflow. Upon a more detailed analysis of the production results from the work model of the Field, it was observed that the field and well oil production rates peak very early during the production time frame, and no substantial production plateau is observed (see Appendix B). This means that a large part of the oil recovery is predicted to occur during the early stages of production. Therefore, considering this fact and the limitations of computational resources available for this work, the placement of the wells were optimized only for the first 300 days.

FieldOpt is capable of optimizing placement of any number of wells in a single optimization run. As it was mentioned in the previous sections, placement optimization of a single deviated well adds 6 variables to the

well placement optimization problem. Consequently, the number of optimization variables to be solved increases as more wells are optimized simultaneously. However, the relationship between the computational cost and the number of optimization variables is not linear. That is, the cost of computation increases exponentially as the number of variables gets larger in an optimization problem.

Table 3.5: Specifications of the computer used in this work.

Component	Specification
Operating System	Ubuntu 16.04 LTS 64-bit
CPU	Intel® Xeon® CPU E5520 @ 2.27GHz x4
Memory	24GiB System Memory @ 1066 MHz
Storage	500GB Hitachi HDP72505

The specifications of the computer used in this work are presented in Table 3.5. Considering the limited computational power of this computer, the placement of only one well was optimized in each optimization case to keep the computational requirement of the optimization problem feasible. The results of each optimization case are presented and analyzed in the next chapter (see section 4.2).

Case configuration example in FieldOpt. This section briefly outlines the overall architecture of a driver file used in FieldOpt along with the description of placement optimization case setup of the well *PROD_1*.

In FieldOpt, each well is parameterized as a straight line between two points in space. Therefore, the (x,y,z) coordinates of heel and toe of the well are required to initialize an optimization run. The section of the driver file, where initial well spline coordinates and well control parameters are specified, is shown in Listing 3.1. In this optimization case, the optimal location of a single highly deviated oil producer that operates with BHP constraint of 60 bars with no liquid target rate is to be searched.

Listing 3.1: Specification of the initial placement of the producer *PROD_1*. The heel and toe coordinates are anonymized for confidentiality reasons (relative distances are preserved).

```
"Wells": [
    {
        "Name": "PROD_1",
        "Group": "G1",
        "Type": "Producer",
        "DefinitionType": "WellSpline",
        "PreferredPhase": "Oil",
        "WellModel": "Projection",
        "WellboreRadius": 0.1905,
        "SplinePoints": {
          "Heel": {
              "x": 100000.00,
              "y": 1000000.00,
              "z": 1000.00,
              "IsVariable": true
          },
          "Toe": {
              "x": 98724.38,
              "y": 1002071.07,
              "z": 870.48,
              "IsVariable": true
          }
        },
        "Controls": [
            {
                "TimeStep": 0,
                "State": "Open",
                "Mode": "BHP",
                "BHP": 60.0,
                "IsVariable": false
            }
        ]
    }]
```

The algorithm searches for an optimum placement solution within the bounding region defined by the user. It is defined by a range of (i,j,k) indices which contain the initial coordinates of the well. Listing 3.2 shows the relevant part of the driver file where the bounding region is defined for well *PROD_1*. This well may be positioned anywhere inside of the box marked with solid black lines in Figure 3.6.

Listing 3.2: Specification of the bounding region of the producer *PROD_1*.

```
"Constraints": [
    {
        "Wells": ["PROD_1"],
        "Type": "ReservoirBoundary",
        "BoxImin": 86,
        "BoxImax": 102,
        "BoxJmin": 29,
        "BoxJmax": 62,
        "BoxKmin": 10,
        "BoxKmax": 44
      }
]
```

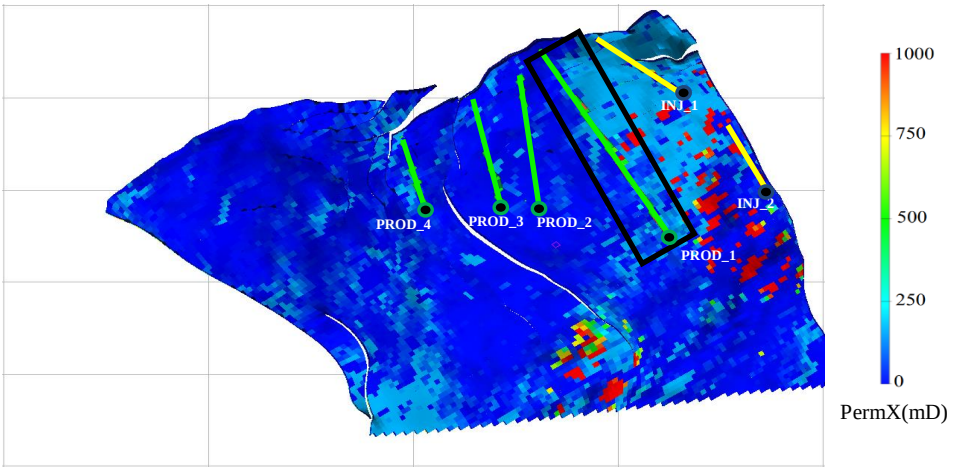


Figure 3.6: The figure shows the permeability map of Formation Three in the reservoir. The trajectories of the production and injection wells are represented as *green* and *yellow* lines, respectively. The reservoir bound constraint is illustrated by the *black box* (2D projection).

The rest of the driver file contains the NPV parameters, optimization algorithm configurations, and specification of which reservoir simulator to be used for the optimization run (see listing 3.3). In this example, the optimization problem is to maximize the NPV subject to the reservoir bound constraints specified in the *Constraints* section of the driver file. The PSO

algorithm with 12 particles will iterate the well location coordinates to find the optimal solution within 138 generations. A complete JSON driver file is presented in Appendix A.

Listing 3.3: Specification of the NPV parameters, algorithm setup, and simulator selection *PROD_1*.

```
    "Global": {
        "Name": "OPM_WORK",
        "BookkeeperTolerance": 1e-3
    },
    "Optimizer": {
        "Type": "PSO",
        "Mode": "Maximize",
        "Parameters": {
            "MaxGenerations": 138,
            "PSO-LearningFactor1": 2,
            "PSO-LearningFactor2": 2,
            "PSO-SwarmSize": 12,
            "PSO-VelocityScale": 0.25
        },
        "Objective": {
            "Type": "WeightedSum",
            "UsePenaltyFunction": false,
            "WeightedSumComponents": [
                {
                    "COMMENT": "Coefficient: 60 $/barrel *
                        6.2898 barrel/sm3 = 377.389"
                    "Coefficient": 377.389,
                    "Property": "CumulativeOilProduction",
                    "TimeStep": -1,
                    "IsWellProp": false
                },
                {
                    "COMMENT": "Coefficient: -3 $/barrel *
                        6.2898 barrel/sm3 = -18.869"
                    "Coefficient": -18.869,
                    "Property": "CumulativeWaterInjection",
                    "TimeStep": -1,
                    "IsWellProp": false
                },
                {
                    "COMMENT": "Coefficient: -5 $/barrel *
                        6.2898 barrel/sm3 = -31.449"
```

```
                    "Coefficient": -31.449,
                    "Property": "CumulativeWaterProduction"
                        ,
                    "TimeStep": -1,
                    "IsWellProp": false
                }

            ]
        },
    "Simulator": {
        "Type": "Flow",
        "FluidModel": "BlackOil",
        "ExecutionScript": "bash_flow.sh",
        "ScheduleFile": "OPM_WORK_SCH.INC"
    },
    "Model": {
"ControlTimes": [
            0, 25, 50, 75, 100, 125, 150, 175, 200, 225, 25
                0, 275, 300],
        "Reservoir": {
            "Type": "Flow"
        }]
```

Chapter 4

Results

Similar to the previous chapter, this chapter also consists of two sections. The results of the simulation model conversion and the validation of the work model will be presented in the first section of this chapter. As it was mentioned in Chapter 3, Eclipse is often referred to as an industry standard reservoir simulator due to its technical capabilities and continuous support by its developer. Hence, the simulation results from Flow will be considered as valid only if the deviation is insignificant when compared to the output results from Eclipse. The second part of this chapter will be the presentation of the well placement optimization results that were obtained using FieldOpt. The performance of the optimization algorithms implemented in this work will also be investigated throughout this part.

4.1 Model Conversion and Validation

Throughout this section, a systematic framework was used to execute a consistent comparison between the results that were obtained from Flow and Eclipse reservoir simulators. In this framework, the relative change and the deviation between the results are quantified as shown in the equations (4.1) and (4.2) respectively.

$$\text{Relative change } (x, x_{ref}) = \frac{\textit{Actual change}}{x_{ref}} = \frac{x - x_{ref}}{x_{ref}} \tag{4.1}$$

$$\text{Deviation (\%)} = \frac{x - x_{ref}}{x_{ref}} \times 100\% \tag{4.2}$$

where, in both of the equations, x and x_{ref} stand for the test variable value and reference variable value respectively. The extent of the deviation from the reference was divided into three categories for a constructive analysis:

- $< 5\%$ - The deviation is small.
- 5-10% - The deviation is moderate.
- $> 10\%$ - The deviation is large.

All reservoir simulation results are only a mathematical representation of reservoir dynamics that occur in real space and time. It is difficult to compare simulation results to the real values that they are representing due to the inherent uncertainty of the mathematical models. Although there is no evidence on which reservoir simulator has a better description of what happens in a reservoir during its production lifetime, Eclipse has been used as a reliable reservoir simulator in a wide variety of reservoir management projects around the world for decades. Thus, Eclipse was used as the reference simulator throughout this work.

4.1.1 Model comparison results

The comparison and validation process of the work model of the Field was performed in multiple stages. The results will be presented in the following order: First, the impacts of individual modifications (mentioned in section 3.1.1) on the simulation results were investigated. Second, the combined impact of all modifications on the simulation results was investigated. Lastly, the simulation results generated by Flow and Eclipse were compared. Although each simulation run of the Field model generates an extensive amount of output data, comparing each one of these prediction parameters would be time inefficient and unnecessary for validation of the work model. Hence, only the prediction results on field-level data (e.g., field pressure, total fluid production, and injection volumes) and well-level data (e.g., well BHPs, fluid production, and injection rates) which are indicative enough to presume that the rest of the output data yield similar

comparison results will be presented.

4.1.2 Impacts of individual modifications

The impact of each modified item was investigated by applying only one modification to the original Eclipse model, and running this modified simulation deck in Eclipse. The production quantities obtained from this model were compared to production quantities predicted from the original model. The results for the impacts of each modified item in the original input deck is presented in the same order as in section 3.1.1.

Petrel Options: The *ECL_PETOPTS_disabled* case. Deactivating the *PETOPTS* option and exporting a new with geometry file in *GRDECL* format resulted in a higher pressure depletion in the reservoir. Looking at the average field pressure (FPR) profiles in Figure 4.1, it can be observed that the deviation between the modified and reference case results is around 17% at the end of the simulation run, which is a large deviation.

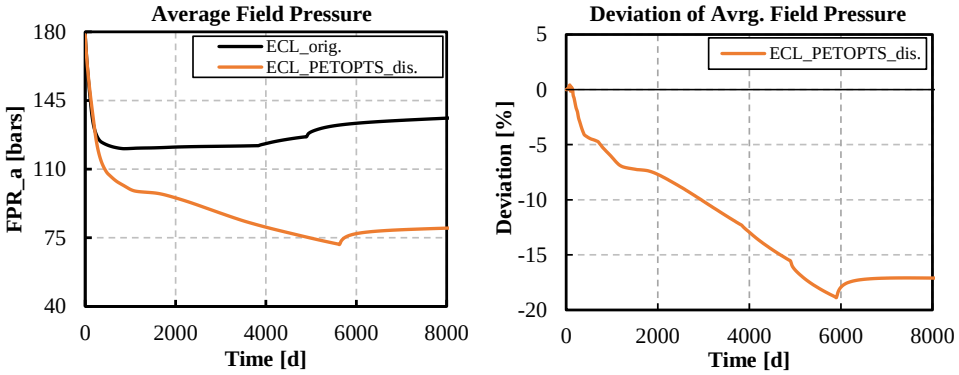


Figure 4.1: The plots show the average field pressure profiles and deviation of the average field pressure of the modified case from the reference. The suffix "a" on the title of y-axis signifies that the pressure values have been normalized for confidentiality reasons.

The cumulative fluid production and injection quantities predicted by the modified case also deviate considerably from the reference case. Table 4.1 shows that the total oil and water production volumes deviate from the reference volumes by around 36% and 17%, respectively. The pressure maps

shown in Figure 4.2 indicate a non-uniform pressure depletion of the reservoir in the original Eclipse model of the Field, where significant pressure differences along the faults are observed.

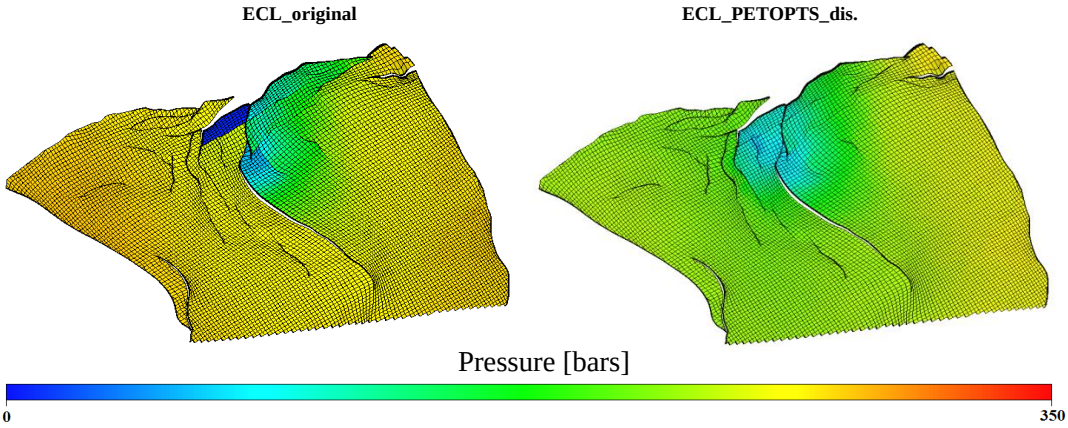


Figure 4.2: Shown in the figure are the pressure maps of Formation Four (K-slice: 35-44) predicted by the reference case (left) and modified case (right) at the end of the field production (25 years). The scale of the pressure color bar was modified (while keeping the relative pressure differences the same) for confidentiality reasons. It can be observed the reference case predicts huge pressure differences (up to around 300 bars) along the faults in some parts of the reservoir.

CPR Solver: The *ECL_CPR_disabled* case. Disabling the *CPR* solver for the simulation calculations in the reservoir model of the Field did cause an observable deviation in the prediction results. Looking at the average field pressure curves predicted by the reference case and modified case in Figure 4.3, it can be observed that the maximum deviation between the predicted results is around 1.6%, which is small. Similarly, Table 4.1 indicates that the maximum deviation in cumulative fluid production and injection volumes predicted by the modified case is less than 2%.

This modification did have a significantly adverse effect on the run-time of the simulation in the modified case compared to the reference case (see Table 4.1). Upon further investigation, it was revealed that the number of linear equation calculations performed by the simulator was around 3.6 times higher in the modified case in comparison to the reference case. This was the case because the linear equations fail to converge more frequently when

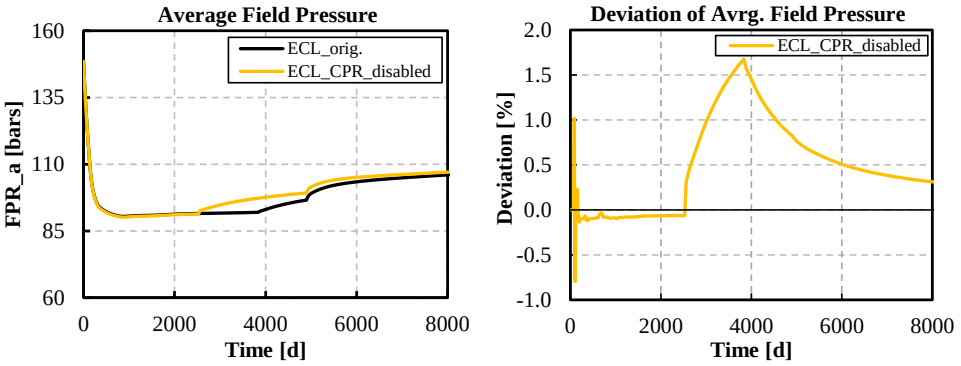


Figure 4.3: The plots show the average field pressure profiles and deviation of the average field pressure of the modified case from the reference. The suffix "a" on the title of y-axis signifies that the pressure values have been normalized for confidentiality reasons.

the *CPR* solver feature is disabled in the original Eclipse model. Therefore, in the modified case, the simulator had to reduce the time-step size (see figure 4.4) every time a linear equation failed to converge in order to minimize the truncation error in the simulation calculations.

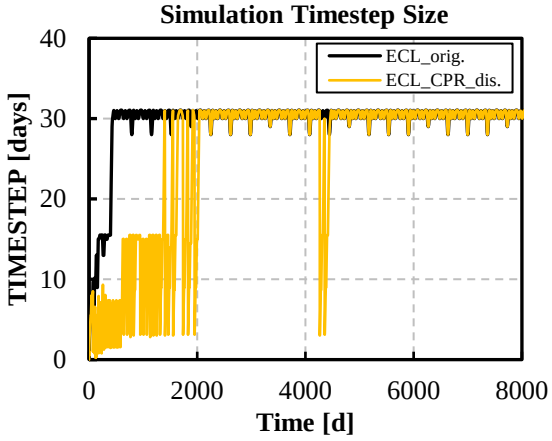


Figure 4.4: Simulation time-step size results for the reference case and CPR solver disabled case.

Fault Threshold Pressure: The *ECL_THPRESFT_disabled* case. Removal of *THPRESFT* keyword and its data items from the simulation deck of the original Eclipse model did not change the model. The field total production and injection volumes presented in Table 4.1 indicate that the deviation of simulation results between the modified case and reference case is negligible. These were the expected results for this case, since the removal of *THPRESFT* keyword has no impact on the fluid flow dynamics in the reservoir model of the Field (see section 3.1.1).

Oil Properties and Rel-Perm Scaling: The *ECL_PVT_modified* case. The impact of modifications made to the PROPS section of the original model is analyzed together, since both of these modifications has to do with the fluid properties in the system. Scaling of the relative permeability curves manually resulted in small deviation of the field production parameters in the modified case. Due to the inadequate implementation of relative permeability curve scaling functionality in Flow, the end-points in of these curves were scaled manually to match the scaled curves generated by Eclipse. The average field pressure predicted by the modified case plotted in Figure 4.5 shows the maximum deviation from the reference values is slightly higher than 4%.

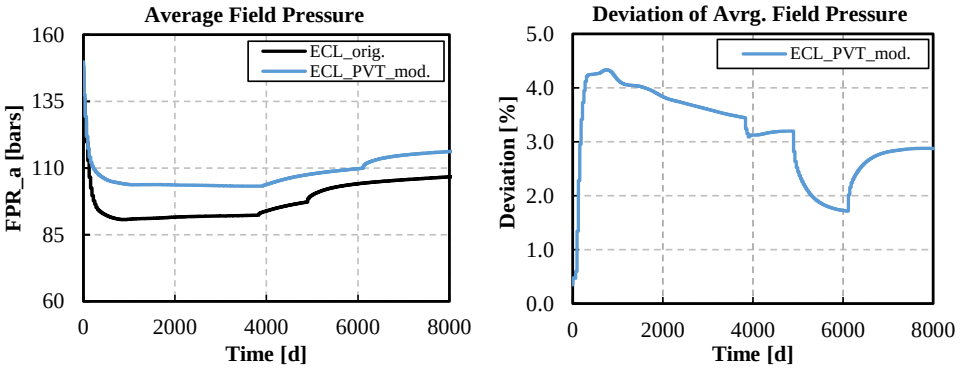


Figure 4.5: The plots show the average field pressure profiles and deviation of the average field pressure of the modified case from the reference. The suffix "a" on the title of y-axis signifies that the pressure values have been normalized for confidentiality reasons.

Well Group Controls and Artificial Lift: The *ECL_GCON_disabled* & *ECL_AQL_disabled* cases. The well group controls (VRR $= 1$) imposed on both of the injector wells in the Field was disabled due to insufficient implementation of this feature in Flow. In this case, the water injection rate targets ($1500\ m^3/d$) and injector well BHP constraints ($500\ bars$) are the only controls of the injector wells. Looking at Figure 4.6, the BHP parameter of the injector wells never reach its upper limit during the simulation run in the reference case since the simulator automatically regulates the water injection rates of the injectors according to the cumulative fluid production rates from the reservoir. On the other hand, it can be observed that the injector wells in the modified case are controlled by the water injection target rate during the first half of the simulation, except for the short time period for the INJ2. During the second half of the simulation, the simulator switches the controls of the injector wells to WBHP parameter, where injector wells operate at maximum BHP.

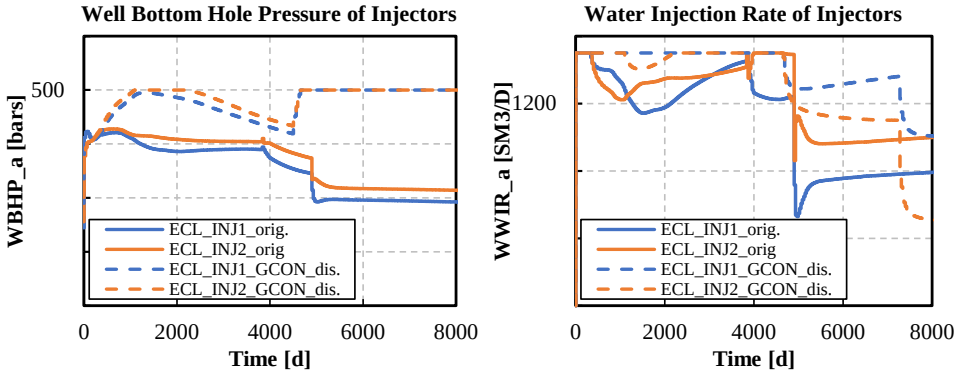


Figure 4.6: The WBHP and WWIR profiles for the injector well INJ1 and INJ2 are shown in the figure. Suffix "a" denotes that the pressure and injection rate axes are anonymized.

Looking at Table 4.1, deactivation of the well group controls for the injector wells leads to cumulative water produced and injected volumes to deviate around 11% and 13% form the reference volumes, respectively. The cumulative oil production volume stayed almost the same since the additional water volume injected into the field did not improve the volumetric sweep efficiency in the reservoir. However, higher cumulative water injection volume caused the average reservoir pressure predictions to deviate significantly from the reference values.

Looking at Figure 4.7, it can be observed that the water injection without the well group controls imposed on the injectors leads to overpressurization of the reservoir, since the VRR is around 1.04 for this case. As a result, the deviation of the average field pressure is slightly higher than 15% by the end of the simulation in this case.

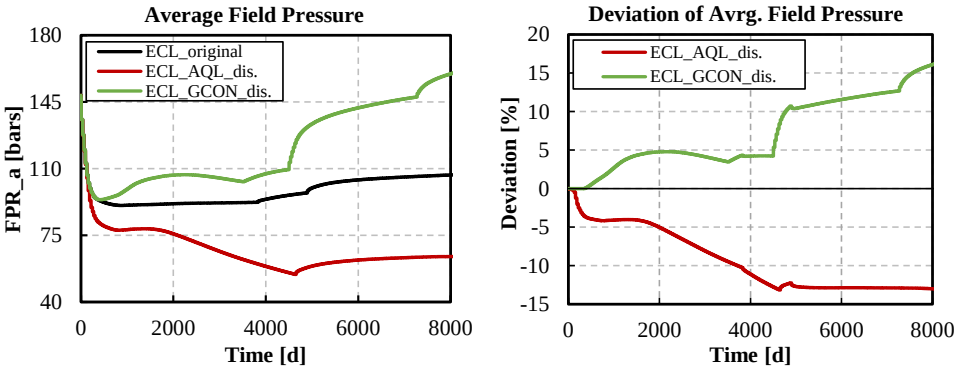


Figure 4.7: The average field pressure profiles (normalized to preserve confidentiality) predicted by the modified cases and reference case is presented in the left plot. The deviation of the average field pressure predictions in the modified cases is shown in the right plot. The suffixes "original", "GCON_dis", and "AQL_dis" signifies the reference case, well group controls disabled case, and gas lift disabled case, respectively. The suffix "a" on the title of y-axis signifies that the pressure values have been normalized for confidentiality reasons.

The artificial lift option was deactivated in the work model of the Field due to the numerical difficulties encountered in Flow. The simulator was not able to interpolate the vertical lift performance relationship (VLP) table, which enables a simulator to calculate the pressure drop in the production tubing as a function of well flow rate, included in the original Eclipse model of the Field. In such situations, well tubing head pressure (WTHP) controls cannot be specified either since the simulator requires a VLP table to calculate this parameter. Therefore, the default value of $0\ bars$ was used for the WTHP parameters of all producer wells in the simulation case where the gas lift option was deactivated.

In general, the drawdown of a producer well is reduced when the gas lift option is disabled due to a heavier fluid column in the wellbore. On the

other hand, well drawdown is increased when no WTHP constraint is imposed on a producer well. Looking at Figure 4.8, it can be observed that all producer wells in the gas lift deactivated case are operated at the minimum WBPH limit ($60\ bars$). Thus, the producer wells have a higher drawdown in this case in comparison to well drawdown in the reference case.

Table 4.1 indicates that the deviation of the total produced volumes of oil and water in the gas lift deactivated case is around 9% and 15%, respectively. Due to the higher cumulative fluid volume withdrawal, the average reservoir pressure is lower in the modified case compared to the reference case (see figure 4.7). The deviation of the average field pressure is around -13% by the end of the simulation.

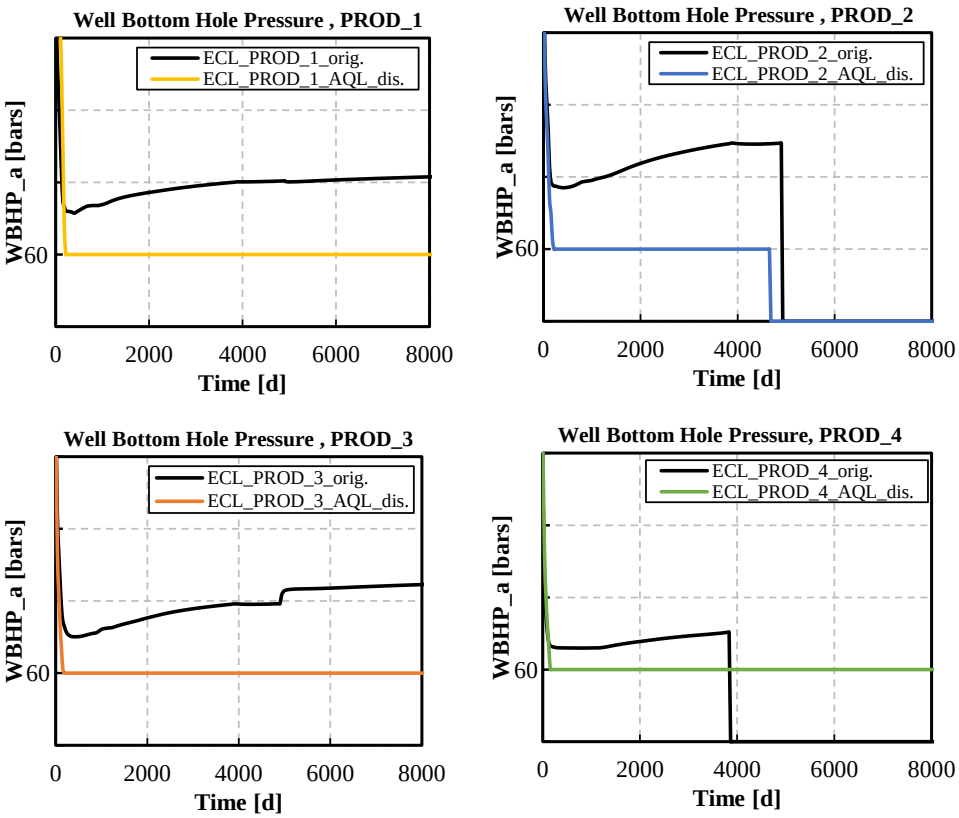


Figure 4.8: Figure illustrates well bottomhole pressures. Suffix "a" denotes that the pressure and injection rate axes are anonymized.

Periodic Testing of Closed Wells: The *ECL_WTEST_disabled* case. Removal of *WTEST* keyword and its data items from the simulation deck of the original Eclipse model did not change the model. The field total production and injection volumes presented in Table 4.1 indicate that the deviation of simulation results between the modified case and reference case is negligible.

4.1.3 Combined effects

The simulation results presented in Table 4.1 show how the various model conversion approximations and modifications impact the total production and injection volumes in each modified case.

Table 4.1: In this table, field total production and injection volumes predicted by the modified cases are presented as a fraction of cumulative volumes predicted by the reference case.

Case	FOPT [-]	FWPT [-]	FWIT [-]	Time [min]
ECL_orig.	1.000	1.000	1.000	52
ECL_PETOPTS_dis.	1.359	1.167	1.180	48
ECL_CPR_dis.	0.983	0.984	0.984	75
ECL_THPRESFT_dis.	1.000	1.000	1.000	52
ECL_PVT_mod.	0.970	1.057	1.040	-
ECL_GCON_dis.	0.999	1.113	1.132	-
ECL_AQL_dis.	1.089	1.148	1.106	50
ECL_WTEST_dis.	1.000	1.000	1.000	52
ECL_reduced	1.368	1.195	1.251	180
ECL_reduced[†]	1.006	1.024	1.060	180
ECL_work[††]	1.000	1.000	1.000	216
OPM_work[§]	1.000	1.000	1.000	252

[†] *ECL_PETOPTS_dis.* case as the reference case.
[††] *ECL_reduced* case as the reference case.
[§] *ECL_work* case as the reference case.

Combined Effects: The *ECL_reduced* case. The combined impact of all of the modifications, which were mentioned in the previous section, was also investigated by applying all of these approximations and modifications together to the original simulation deck of the Field. Looking at Figure 4.9, the average field pressure profile predicted by the reduced model seems to deviate significantly from the reference case pressure profile. It is evident that the *PETOPTS* feature in the original simulation deck has a dominant impact on the simulation results. The maximum deviation of the average field pressure is around 38% by the end of the simulation.

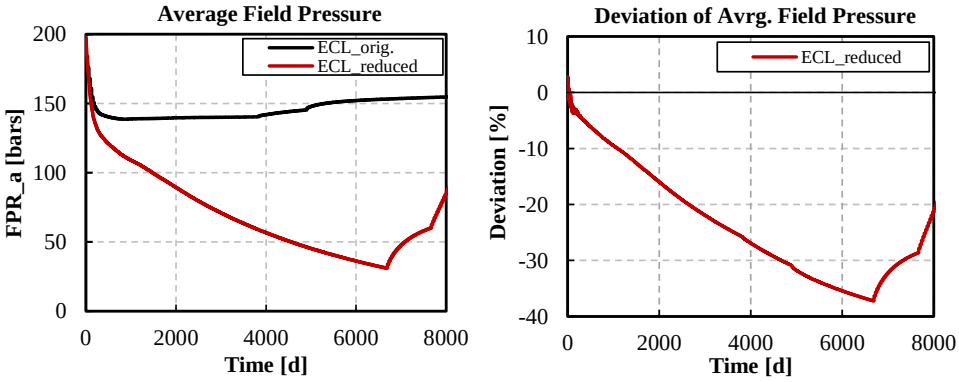


Figure 4.9: The plots show the average field pressure profiles and deviation of the average field pressure of the reduced model from the reference. The suffix "a" on the title of y-axis signifies that the pressure values have been normalized for confidentiality reasons.

4.1.4 Flow vs Eclipse

The aim of the validation process was to validate the simulation results predicted by OPM-Flow simulator. In order to do so, the reduced model of the Field was run in both simulators, and the results were compared.

The work model: The *ECL_work* & *OPM_work* cases. Finally, the work model of the Field was developed by implementing all of the modifications, which were mentioned previously, on the original simulation deck. Looking at the field production results and pressure profiles presented in Table 4.1 and Figures 4.10 and 4.11, the match between the results predicted by both of the simulators gave a close match, i.e., the deviation between the results is less than 1%. Thus, the work model results predicted in Flow is validated, and this concludes the validation part of this work.

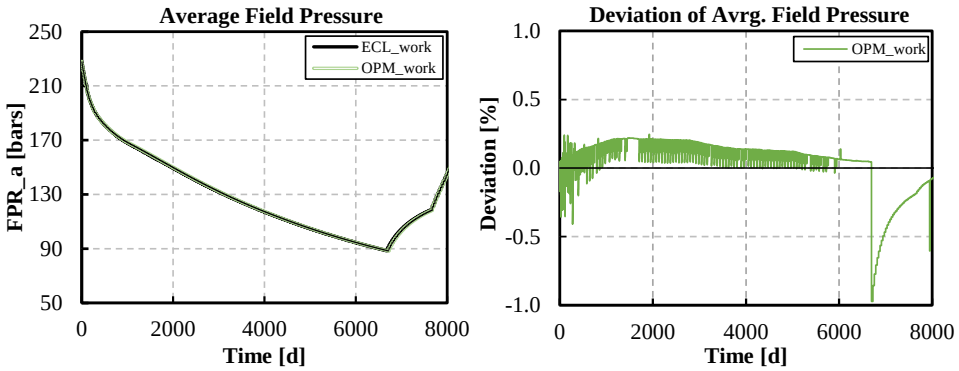


Figure 4.10: The plots show the average field pressure profiles and deviation of the average field pressure of the work model run in Flow. The suffix "a" on the title of y-axis signifies that the pressure values have been normalized for confidentiality reasons.

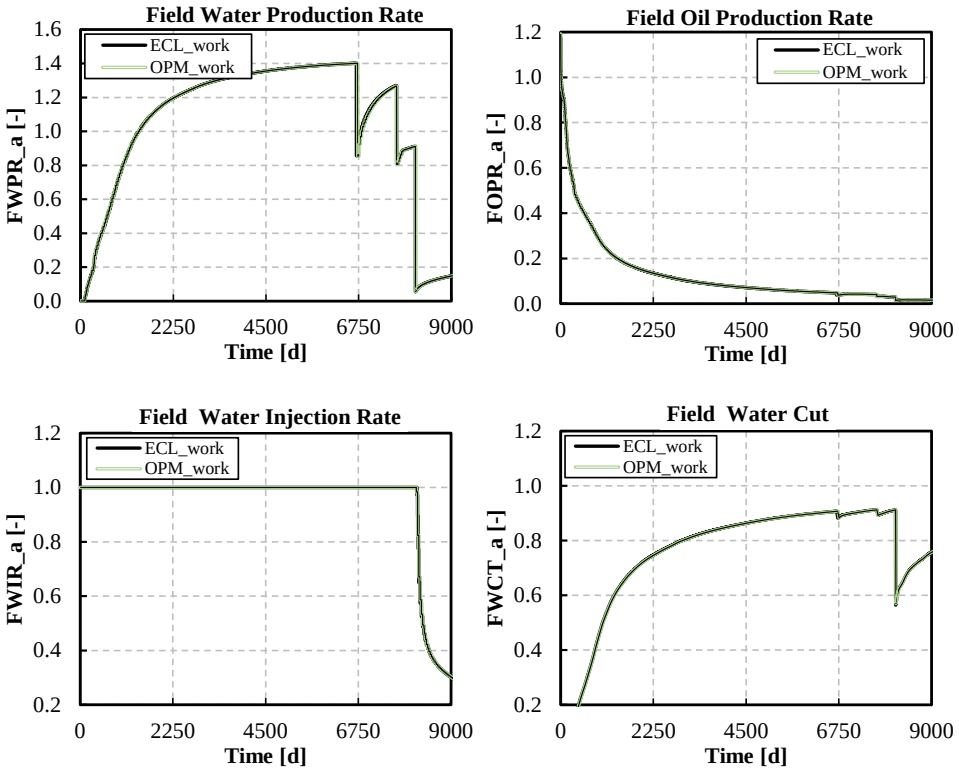


Figure 4.11: Field production and injection rates. The suffix "a" on the title of y-axis signifies that the pressure values have been normalized for confidentiality reasons.

The run-time of the work model in Flow on a single core setup was 36 minutes (17%) longer compared to run-time in Eclipse. The time-step size plot in Figure 4.12 indicates that Flow takes larger time-steps than Eclipse without accumulating significant truncation error. It is a remarkable performance from an open-source reservoir simulator given the complexity of the simulation model of the Field.

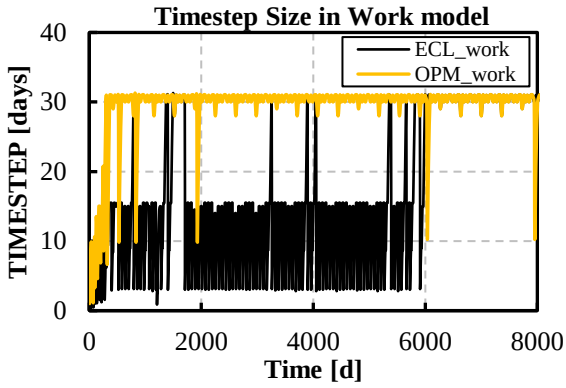


Figure 4.12: Simulation time-step size results for the work model in both Eclipse and Flow.

4.2 Optimization Results

This section describes the results obtained from FieldOpt by utilizing the optimization workflow described in the previous chapter (see section 3.2.2). The main results of each well optimization run will be described in terms of final objective value and algorithm performance. Moreover, the implementation of these solutions in the work model of the Field is also presented. The best placement solutions obtained for each well were implemented together in the work model to develop an optimized well placement scenario of the Field. The results of the optimum and base case development scenarios (introduced in section 3.1) are compared and presented in terms of the final objective function values, field production quantities, and saturation map plots at different time steps.

4.2.1 Optimization solutions

The procedure utilized throughout the optimization part of this work produced a total of 16 well placement solutions in the work model of the Field. That is, the placements of all four producer wells were optimized by using two optimization algorithms launched with two different configurations in FieldOpt. The optimization runs using the APPS optimization algorithm with the initial step-length scaling (*AutoStepInitScale*) configuration set to 0.25 and 0.75 are denoted by the name of the well which is optimized, followed by "APPS1" and "APPS2", respectively. Similarly, the runs using the PSO algorithm with swarm size of 6 and 12 particles are denoted by the name of the well which is optimized, followed by "PSO1" and "PSO2", respectively. Values associated with the initial well configuration is referred to as "Base Case" in the tables. All results reported in this section are predicted by the work model of the Field run in Flow.

Optimization results for well *PROD_4*. Figure 4.13 and Table 4.2 show the objective function evolution versus the number of function evaluations (simulations) of each optimization case for the *PROD_4*. For APPS algorithm, the improvement of NPV in the cases where the initial step-length scaling configuration with 0.25 and 0.75 is 16.19% and 16.56%, respectively. However, the APPS algorithm with the former configuration (APPS1) converged to a solution (maximum NPV) value within 298 function evaluations, which is twice as fast compared to the latter APPS (APPS2) algorithm

configuration. Applied to the same the problem, the PSO algorithm with 6 particles yields 14.36% NPV improvement after 531 function evaluations. Similarly, the PSO algorithm configuration with 12 particles yields 17.48% NPV improvement after 574 function evaluations.

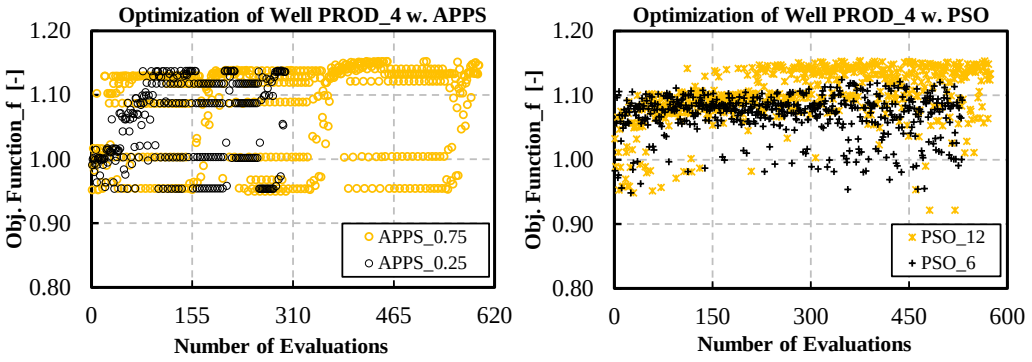


Figure 4.13: The figure shows the objective function evolution versus the number of evaluations for both configurations of APPS (left) and PSO (right) algorithms. The suffix "f" denotes that the values are presented as a fraction of base case objective function value.

Table 4.2 shows the results of production quantities after 300 days predicted by each case. It can be observed that the total oil production in all of the optimized cases is higher than the base case prediction. Total water production and injection volumes are approximately close to base case prediction volumes except for *P4_PSO2* case. The total water production volume in the *P4_PSO2* case is significantly higher when compared to all other cases. The reason for such performance is that the optimum placement found in this case is too close to the WOC in the Western segment of the Field. Hence, water-breakthrough occurs much earlier in this case compared to all other optimization cases for well *PROD_4*. In general, all four optimization cases for this well yields a higher oil recovery because the optimized well trajectories target the unswept regions (in the base case) and drain the Western segment of the reservoir more efficiently (see figure 4.16).

Table 4.2: Optimization results for well *PROD_4*.

Case	*PROD_4* WOPT [-]	*PROD_4* WWPT [-]	FOPT [-]	FWPT [-]	FWIT [-]	#Eval.	Δ NPV[%]
Base Case	1.000	1.000	1.000	1.000	1.000	-	-
P4_APPS1	4.509	10.989	1.157	0.999	1.000	298	16.19
P4_APPS2	4.537	10.643	1.160	0.919	1.000	596	16.56
P4_PSO1	4.267	6.638	1.139	0.994	1.000	531	14.36
P4_PSO2	5.171	5258.124	1.179	6.150	1.000	574	17.48

Optimization results for the rest of the wells. Tables 4.3, 4.4, and 4.5 show the optimum placement results for wells *PROD_1*, *PROD_2*, and *PROD_3*, respectively. For well *PROD_1*, both *P1_APPS1* and *P1_PSO2* cases yield around 12% NPV improvement after 472 and 624 function evaluations, respectively.

Table 4.3: Optimization results for well *PROD_1*.

Case	*PROD_1* WOPT [-]	*PROD_1* WWPT [-]	FOPT [-]	FWPT [-]	FWIT [-]	#Eval	Δ NPV[%]
P1_APPS1	1.800	11.639	1.132	8.178	1.000	472	12.24
P1_APPS2	1.841	17.632	1.104	12.007	1.000	494	8.64
P1_PSO1	1.683	12.535	1.091	8.687	1.000	528	7.92
P1_PSO2	1.710	14.198	1.136	9.761	1.000	624	12.33

For well *PROD_2*, *P2_PSO2* case yields the highest NPV increase around 11.8% after 977 function evaluations.

Table 4.4: Optimization results for well *PROD_2*.

Case	*PROD_2* WOPT [-]	*PROD_2* WWPT [-]	FOPT [-]	FWPT [-]	FWIT [-]	#Eval	Δ NPV[%]
P2_APPS1	1.156	1.598	1.048	1.163	1.000	407	4.93
P2_APPS2	1.172	1.875	1.054	1.239	1.000	500	5.49
P2_PSO1	1.259	5.483	1.080	2.283	1.000	828	7.99
P2_PSO2	1.186	1.453	1.115	1.153	1.000	977	11.86

For well *PROD_3*, the PSO algorithm cases with 6 and 12 particles yield around 6.3 NPV increase after 584 and 758 function evaluations.

Table 4.5: Optimization results for well *PROD_3*.

Case	*PROD_3* WOPT [-]	*PROD_3* WWPT [-]	FOPT [-]	FWPT [-]	FWIT [-]	#Eval	Δ NPV[%]
P3_APPS1	1.867	15.924	1.020	0.976	1.000	552	2.06
P3_APPS2	1.863	14.807	1.020	0.970	1.000	1013	2.07
P3_PSO1	1.297	2.012	1.062	1.073	1.000	584	6.36
P3_PSO2	1.264	2.223	1.061	1.061	1.000	758	6.25

Combined well placement solution. The optimum well placement configuration of the producer wells in the Field was assembled by using the optimum placement solutions for each producer well generated by FieldOpt (described in the previous section). Since each well had four optimized placement solutions, only the solution that was considered to be practically applicable (e.g., not too close to other wells or completed outside of target formations) and yielded a higher NPV improvement was picked for each well. In this part, the simulation results for the combined well placement scenario are presented and described. Throughout this section, the "Optimized Case" represents the results of the scenario where the optimized well configuration is implemented to drain the reservoir. The length of each producer well in both of the cases is shown in Table 4.6.

Table 4.6: Well lengths for Base Case and Optimized Case scenarios.

Case	Well Length[m]			
	PROD_1	PROD_2	PROD_3	PROD_4
Base Case	2456	1384	1247	896
Optimized Case	1796	1545	1037	1176

The description of each producer well in the optimized well placement scenario is presented below:

- ***Well PROD_1:*** A dedicated Formation Four producer in the Eastern segment of the Field. It was a dedicated Formation Three producer in the base case scenario.

- ***Well PROD_2:*** A commingled producer from Formation Three and Formation Four in the Eastern segment of the Field. It was a dedicated Formation Four producer in the base case scenario.

- ***Well PROD_3:*** A dedicated Formation Four producer in the Eastern segment of the Field, which is the same description as in the base case scenario.

- ***Well PROD_4:*** A commingled producer from Formation Three and Formation Four in the Western segment of the Field, which is the same description as in the base case scenario.

The total production quantities from the reservoir for the optimized case are shown in Table 4.7. This case with optimized well configuration yields 31% higher cumulative oil production after 300 days, and the NPV improvement is around 30.8%. The cumulative water production volume is 7.8 times higher compared to the base case. Looking at the field average pressure profiles depicted in Figure 4.14, it can be observed that the pressure depletion predicted in the optimized case is significantly higher compared to the base case. This is a reasonable outcome since the extra volume of fluids withdrawn from the reservoir is not compensated for by the injectors (i.e., VRR is lower in the optimized case).

Table 4.7: In this table, field total production and injection volumes predicted by the optimized well configuration case are presented as a fraction of cumulative volumes predicted by the base case.

Case	FOPT [-]	FWPT [-]	FWIT [-]	Δ NPV[%]
Base Case	1.000	1.000	1.000	-
Optimized Case	1.311	7.842	1.000	30.81

Another feature of the optimized well placement scenario worth noting is the water-breakthrough time. The water production plots in figures 4.14 and 6.1 indicate that water-breakthrough time is shorter in the optimized case (after 50 days) compared to the base case (after 110 days).

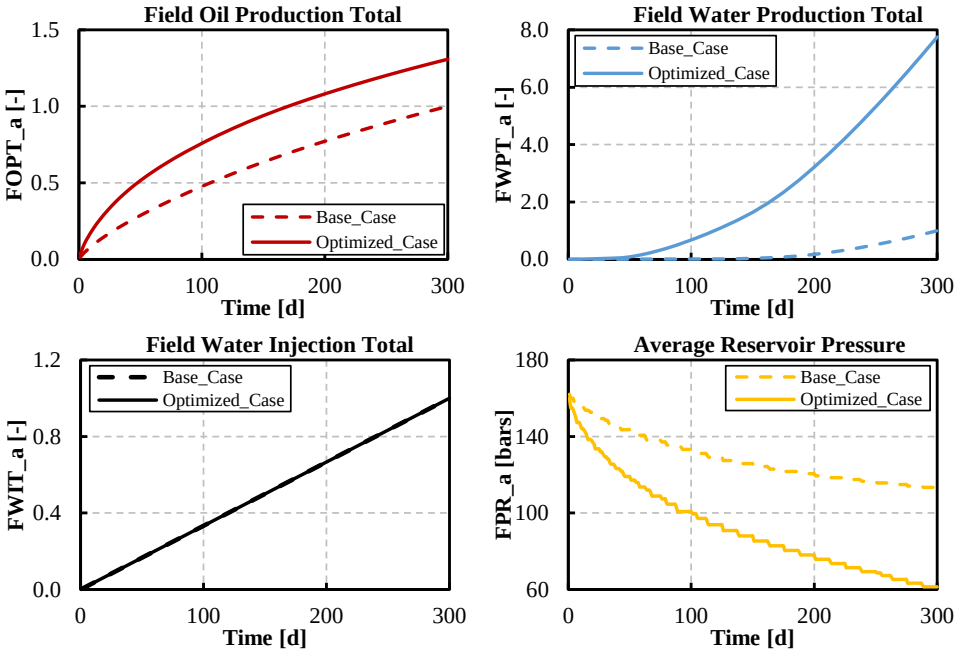


Figure 4.14: Field production totals and average pressure for the base case (dashed) and optimized case (solid) over a production time frame of 300 days. The suffix "a" on the title of y-axis signifies that the pressure values have been normalized for confidentiality reasons.

The total fluid production from each well in the optimized case is shown in Table 4.8. It can be seen that except for well *PROD_2*, the optimized scenario yields higher total oil and water production volumes for each well over the base case.

Table 4.8: Total production volumes for each well in the optimized well configuration case are presented as a fraction of cumulative volumes predicted for the corresponding well in the base case.

Quantity	PROD_1	PROD_2	PROD_3	PROD_4
WOPT [-]	1.693	0.644	1.225	4.432
WWPT [-]	10.974	1.142	1.721	9.926

The oil saturation map of the Field after 300 days of production are shown in Figures 4.15 and 4.16 for the base case and optimized case well configuration scenarios, respectively. In Figure 4.16, it can be seen that well *PROD_4* is longer in the optimized case, and its new location targets the oil accumulation near the WOC in the Western segment of the Field. The increased WOPT in this well by such re-positioning is a significant factor contributing to the overall improvement in NPV for the optimized scenario. It can also be observed that wells *PROD_1*, *PROD_2*, and *PROD_3* are relocated towards the east of the Field. As a result of this relocation, the central part of reservoir in the Eastern segment has a better sweep compared to the base case. The water-breakthrough time in producer well *PROD_1* is reduced significantly (see figure 6.1) in this scenario because its new trajectory is closer to injector well *INJ_1*. The well *PROD_3* has greater drainage area in the optimized case since it moved further towards the east of the Field, away from the central fault.

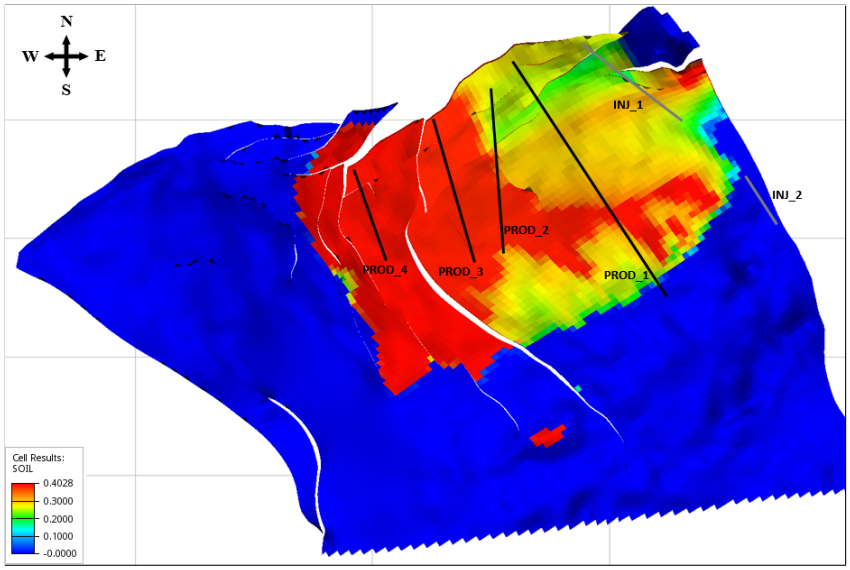


Figure 4.15: Oil saturation map of Formation Four (K-slice:41-44) after 300 days of production in the Base Case. The initial producer trajectories are shown as black lines. Injector trajectories are marked with grey lines. *The wells are not in the same formation.*

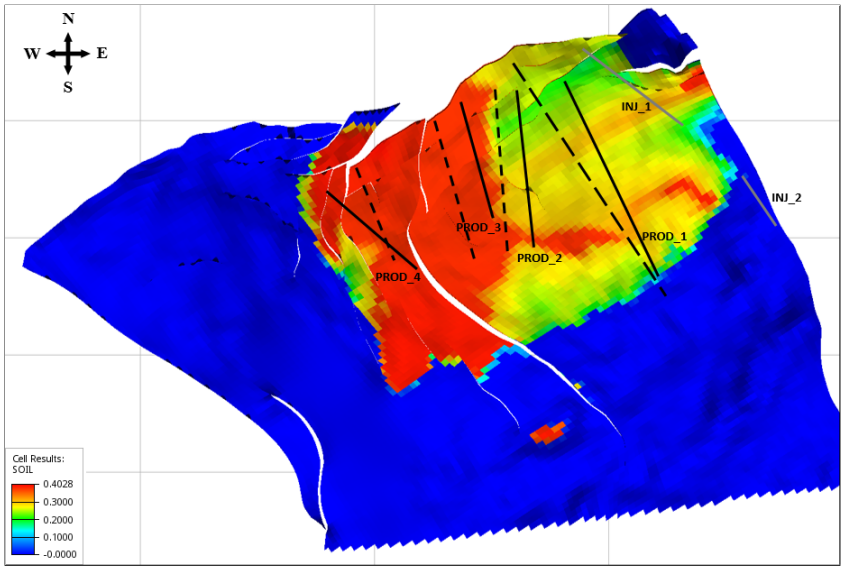


Figure 4.16: Oil saturation map of Formation Four (K-slice:41-44) after 300 days of production in the Optimized Case. The initial and optimized producer trajectories are shown as dashed and solid black lines, respectively. Injector trajectories are marked with grey lines. *The wells are not in the same formation.*

Chapter 5

Discussion

This chapter expands upon the results that were presented in the previous chapter, and tries to address the main research questions. Similar to the previous chapter, this chapter also consists of two sections.

Simulation model conversion and validation. The simulation model conversion process was focused on developing the work model of the Field. During this process, a number of approximations and modification were made to the original Eclipse model since Flow does not support all of the technical features implemented in Eclipse reservoir simulator. The impacts of these modifications on the simulation prediction results were presented in section 4.1.1 and Table 4.1. The items mentioned in that section can be summarized and discussed as below:

PETOPTS Keyword. Deactivation of this option impacts the fluid dynamics in the reservoir simulation model. The cumulative volumes predicted for oil and water in the *PETOPTS* deactivated case deviated by approximately 36% and 17%, respectively. The inspection of the pressure maps revealed non-uniform depletion of the reservoir in the original Eclipse model of the Field, where significant pressure differences along the faults were observed. No physical reason was found for such behavior of the model since the horizontal wells passing through the faults should have prevented the occurrence of such high-pressure differences, due to the cross-flow through the well annulus. On the other hand, the pressure drainage predicted by the modified case is more uniform and physically reasonable. It was hard to in-

vestigate why this might be the case since Eclipse does not provide details of this feature beyond the reference manual and technical description.

CPR Solver. Disabling the *CPR* solver generated a significant deviation in the prediction results for this particular reservoir model. Such a deviation between the results was not expected. Preconditioning the linear equations via the *CPR* feature in a simulation model means that the same flow equations are solved by using a different mathematical technique. In this case study, it was not exactly clear why preconditioning the linear equation solver yielded an observable deviation in the prediction results. As expected, disabling this feature increased the run-time of the simulation. This result indicates that the *CPR* solver feature can potentially reduce the computational cost of running this reservoir model.

End-point Scaling. The manual scaling of the relative permeability curves of the original model did not yield a significant deviation in the prediction results. However, it is worth mentioning that the scaling of connate water saturation to 60% in all of the grid cells in a simulation model of a conventional oil reservoir seems to be an unusual practice. Such high connate water saturation throughout the reservoir is improbable.

Well Group Controls and Artificial lift. Deactivation of the well group controls for the injector wells and lift gas injection for the producers yield a significant deviation in the prediction results of this model when implemented individually. Hence, fixing the software implementation of these features in Flow would be desirable.

The impacts of deactivation of the gas lift option and well group controls on the field-wide production could have been mitigated in the work model by imposing additional control parameters on the wells. This can be accomplished by extracting the resulting well bottom-hole pressures from the reference case, discretizing these quantities at different time intervals, and using these discretized values as well bottom-hole pressure constraints for the entire production time frame in the work model of the Field. However, it seemed reasonable to assume that the impacts of these modifications would have a dampening effect on each other when applied together in the work model of the field. Therefore, no further action was taken to reduce the deviation of the simulation results predicted by these modified cases in

this work.

During the model conversion and validation process, some drawbacks and software bugs in Flow (v.2018.10) simulator were found. The following items were identified as the potential improvement areas in Flow:

- Implement an adequate CPR solver feature.
- Expand the keyword library items in order to make model conversion process from Eclipse to Flow more user-friendly (e.g., *THRPRESFT*).
- Fix the software bugs in SCHEDULE section items, such as the well group controls (*GCONINJE*) option, VLP table interpolation technique, and periodic testing of closed wells (*WTEST*) feature.

Validation. After investigating the impacts of each individual modification made to the original model, a benchmark test was conducted by using the work model of the Field. With this test, an impressive match (i.e., the deviation is less than 1%) was demonstrated between the prediction results generated by Flow and Eclipse. The simulation time-step size comparison presented in Figure 3 revealed that Flow takes longer time-steps compared to Eclipse, which indicates that each reservoir simulator has different convergence criteria for solving linear equations. However, taking longer time-steps to solve the linear equations in the simulation model does not translate into a shorter simulation run-time. The run-time comparison of the work model presented in Table 4.1 showed that the single core performance of Flow is 17% slower than Eclipse.

Well Placement Optimization. In this section , the results for the optimization part of this work are discussed and analyzed. The following topics are discussed: optimization algorithm performance; the applicability of optimization solutions; technical limitations and challenges regarding FieldOpt.

The results for the optimization part of this work are presented in section 4.2. For each oil producer, two optimization algorithms were used. Both of the algorithms were run from the same initial position with two different configurations for all four producers in the Field. The results are presented

in Tables 4.2 through 4.5. Through the results presented in these tables, a clear optimization incremental in the objective function (i.e., NPV) can be seen in all of the cases. The highest incremental objective function improvement was observed to be around 17.5% for well *PROD_4*. For the rest of the wells, the NPV incremental was between 5 to 12 $\%$. When compared in terms of the number of function evaluations, APPS algorithm with the smaller step-length configuration utilized for optimizing well *PROD_4* performed the least amount of function evaluations (in 298 evaluations) before converging to an optimum solution.

On average, it was observed that the PSO algorithm yields a higher incremental NPV improvement in this particular case study. Due to its stochastic nature, this algorithm searches the bound region more thoroughly compared to the APPS algorithm. That is, the PSO algorithm tries a broader solution span with different well configurations and orientations in the search space before converging to an optimum solution. However, upon a closer examination of the optimum solutions obtained for each well, it was revealed that most of the well trajectories found by the PSO algorithm were not realistic enough to be applicable in field development projects.

As presented in Table 4.2, it was observed that the optimization case with the highest NPV incremental also had the highest total water production volume over the base case. In this particular case, the optimized well trajectory found by the PSO algorithm was very close to the WOC at the Western segment of the reservoir. Such cases indicate that more realistic constraints have to be introduced into the optimization problem in order to take full advantage of the capabilities of PSO algorithm. Thus, using the simplified weighted sum of the cumulative fluid production volumes as the objective function led to impractical solutions in this optimization problem. The occurrence of these cases can be reduced by increasing the water handling cost parameter used in the objective function and imposing an upper limit for the water production rate depending on the water handling capacity of topside facilities.

For this particular optimization problem, the optimum placement solutions generated using the APPS algorithm were found to be more realistic, and this algorithm reached its solutions in a smaller number of evaluations in general. Such an outcome is reasonable since APPS is a deterministic al-

gorithm, where the performance of such algorithms is highly dependent on the initial well coordinates provided. Therefore, the performance and placement solutions of this algorithm could have been better or worse depending on the initial well coordinates provided.

Based on the optimum placement solutions obtained for each producer, an improved well placement configuration was compiled for developing the Field (see figure 4.16). As a result, the improved well configuration yielded 30.8% increase in NPV after 300 days of production over the base case well configuration. The lengths of the optimized wells were not significantly different compared to their lengths in the base case. Moreover, the mechanistic study of the new drainage strategy revealed that the producers were re-positioned towards the oil accumulations that were left undrained in the base case scenario. The well orientations were kept perpendicular to the orientation of the channels in the Eastern segment of the Field since such well orientation increases the channel exposure to the producers. These observations indicate once again that the objective function improvements achieved by FieldOpt are systematic in this optimization problem.

One important point worth mentioning is that the geological uncertainty of the Field was not accounted for in this work. This could have been done by employing a number of different geological realizations of the Field and running an ensemble of simulations, which would enable the probabilistic analysis of the optimization solutions. However, such a workflow will increase the computational cost of the work significantly.

The main technical limitation was the computational resources available for this work. With an optimization horizon of 300 days, the average runtime for each function evaluation (i.e., simulation) was around 50 minutes. It should be noted that 300 days of production time is too short compared to the production time frame of 25 years in the base case scenario. Furthermore, given the scope and time restrictions for delivering this work, only one well placement was optimized in each optimization case. However, combining separately optimized well placement solutions do not necessarily equate to the *optimum* well placement solution for a reservoir. The optimum well placement configuration of the Field can only be found by optimizing all of the wells simultaneously.

Throughout the optimization work, a number of impediments and challenges were encountered regarding FieldOpt. Some potential improvement areas in FieldOpt are summarized below:

- *Bound region constraints.* Currently, the reservoir bound constraints can only be specified as a rectangular shape containing the initial well coordinates. This makes fitting deviated well coordinates into the feasible region very challenging in faulted and curvy reservoir formations, such as the formations in Field. Having an option to set up a bounding region with a customized shape would significantly ease the optimization workflow.

- *Driver file configuration.* The driver files in FieldOpt are configured manually, which leaves a lot of room of for human errors. In addition, knowing how to read and write these files might not be so straightforward for an average user. Therefore, developing a graphical user interface and user manual would make FieldOpt more user-friendly and attract new collaborators.

- *Drilling schedule optimization.* The schedule for drilling the wells is an important decision during the development stage of petroleum fields. The optimum placement of a well in the reservoir might vary depending on the drilling order of the wells since the placement variables in well placement optimization problems are inter-related. Implementing drilling schedule optimization functionality in FieldOpt would make it a more comprehensive field optimization software package.

Chapter 6

Conclusions

This thesis presents an open-source approach for well placement optimization of a marginal petroleum field on the NCS. Two open-source simulators, OPM-Flow and FieldOpt, were employed for this task. This work consists of two parts: the first part dealt with converting the Eclipse simulation model of the reservoir to run in Flow, and the second part focused on improving the oil recovery by optimizing the well placement in the reservoir by using FieldOpt.

The main intention of the first part of the work was to convert the original Eclipse model of the Field to run in Flow, and validate the simulation results against the reference simulator. The technical limitations of Flow combined with a complex simulation model of the Field required several approximations and modifications to be made to the original model in order to develop the work model. The impact of each modified item in the simulation deck was investigated by comparing the results in each case to the reference case results. Lastly, simulation results from the work model of the Field run in Flow was validated by comparing them to the reference simulator results. The computational performance test on a single-core setup indicated that the predictive simulation run-time of the work model in Flow was only 17% slower than Eclipse. Overall, this part of the work showed that the open-source reservoir simulator can be a reliable and competitive alternative to a proprietary reservoir simulator, such as Eclipse.

Throughout the second part of this work, the placement of the producer wells was optimized by using two optimization algorithms, which were implemented in FieldOpt. An apparent improvement in the objective function (i.e., NPV) was observed in all of the optimization cases. The highest incremental NPV improvement yielded by the APPS and PSO algorithms in single well optimization cases was 16.2% and 17.5%, respectively. The optimum well trajectories found by the APPS algorithm were considered to be more practical and realistic compared to solutions found by the PSO algorithm due to the objective function constraints applied for this particular optimization problem. The final well placement configuration of the Field compiled by combining the individual placement solutions generated 30.8% incremental increase in the NPV over the base case scenario. Through this case study, FielOpt's ability to optimize the placement of wells in a reservoir was confirmed.

Overall, this thesis as a whole demonstrated that it is possible to use open-source workflow to make well placement decisions for petroleum field development projects. The main benefit of such a workflow is that it reduces the cost of automatic optimization projects by eliminating hefty subsurface software fees. In addition, it provides programming flexibility so that the end-user can custom fit the software to their specific needs. The potential beneficiary of this workflow would be operators looking to reduce costs associated with field development optimization projects.

Recommendations for further work. The full potential of neither Flow nor FieldOpt was exploited in this work due to the work scope, technical difficulties with simulation software, and computational power restrictions. Hence, there is still a lot of room to further expand the scope of this work. Some of the recommended topics are listed below:

- Use different realization models (i.e., net-based and facies based models) to investigate the uncertainty in the simulation results related to the static properties of the Field, and repeat the workflow presented in this work.

- Investigate the reason for the significant deviation of the simulation results caused by removing the *PETOPTS* and *CPR* solver features from the original Eclipse model of the Field.

- Some authors have reported that Flow has slightly outperformed Eclipse in dual-core benchmark study conducted on the Norne field model. It would be interesting to test the dual-core performance of Flow with the work model of the Field.
- The continuation of this work could investigate an optimization case where all six of the wells, that is four oil producers and two water injector, are optimized simultaneously in FieldOpt.
- Test the performance of the remaining optimization algorithms implemented in FieldOpt on the well placement optimization problem of the Field. Further, solve the placement optimization problem over a longer optimization horizon.
- Impose more realistic constraints and penalty functions on the problem variables by implementing minimum inter-well distance, maximum well length, and maximum water production rate. Moreover, use a realistic discount rate to account for the time value of the future cash flow.

Further details on this work. All the data that was used for this thesis will be stored for future references for those who would like to investigate further, ask questions, or view the results manually with respect to terms and conditions of the confidentiality agreement. Further inquiries can be directed to: sadigov@ualberta.ca

Bibliography

Arukhe, J. O., Khelaiwi, F., Isichei, O., Dhubaiki, A., et al., 2017. Smart Well Completion Optimization in Multilateral Wells. In: Abu Dhabi International Petroleum Exhibition & Conference. Society of Petroleum Engineers.

Aziz, K., Settari, A., 1979. Petroleum Reservoir Simulation. Applied Science Publ. Ltd., London, UK.

Baumann, E. J., Dale, S. I., Bellout, M. C., 2019. FieldOpt: A Powerful and Effective Programming Framework Tailored for Field Development Optimization. Preprint submitted to Computers and Geosciences.

Baumann, E. J. M., 2015. Fieldopt: Enhanced software framework for petroleum field optimization-development of software support system for the integration of oil production problems with optimization methodology. Master's thesis, NTNU.

Bellout, M. C., 2014. Joint Optimization of Well Placement and Controls for Petroleum Field Development. Ph.D. thesis, Norwegian University of Science and Technology.

Brownlee, J., 2011. Clever Algorithms: Nature-Inspired Programming Recipes. Jason Brownlee.

Cavazzuti, M., 2012. Optimization Methods: From theory to design scientific and technological aspects in mechanics. Springer Science & Business Media.

Clark, R. A. J., Ludolph, B., et al., 2003. Voidage replacement ratio calculations in retrograde condensate to volatile oil reservoirs undergoing eor processes. In: SPE Annual Technical Conference and Exhibition. Society of Petroleum Engineers.

Ebadi, F., Davies, D. R., et al., 2006. Should" Proactive" or" Reactive" Control be Chosen for Intelligent Well Management? In: Intelligent Energy Conference and Exhibition. Society of Petroleum Engineers.

Eberhart, R., Kennedy, J., 1995. Particle Swarm Optimization. In: Proceedings of the IEEE international conference on neural networks. Vol. 4. Citeseer, pp. 1942–1948.

Gould, N., 2006. An introduction to algorithms for continuous optimization. Oxford University Computing Laboratory Notes.

Jansen, J. D., Brouwer, D., Naevdal, G., Van Kruijsdijk, C., 2005. Closed-loop reservoir management. First Break 23 (1), 43–48.

Jesmani, M., Bellout, M. C., Hanea, R., Foss, B., et al., 2015. Particle swarm optimization algorithm for optimum well placement subject to realistic field development constraints. In: SPE Reservoir Characterisation and Simulation Conference and Exhibition. Society of Petroleum Engineers.

Jin, C., 2015. A sequential process monitoring approach using hidden Markov model for unobservable process drift. Ph.D. thesis, University of Cincinnati.

Kennedy, J., Mendes, R., 2002. Population structure and particle swarm performance. In: Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600). Vol. 2. IEEE, pp. 1671–1676.

Kolda, T. G., Lewis, R. M., Torczon, V., 2003. Optimization by direct search: New perspectives on some classical and modern methods. SIAM review 45 (3), 385–482.

Kolda, T. G., Torczon, V. J., 2003. Understanding Asynchronous Parallel Pattern Search. In: High Performance Algorithms and Software for Nonlinear Optimization. Springer, pp. 323–342.

Kristoffersen, B. S., 2017. An open-source approach to Integrated Operations. Master's thesis, NTNU.

Lie, K.-A., 2014. An Introduction to Reservoir Simulation Using MATLAB: User Guide for the Matlab Reservoir Simulation Toolbox (MRST). Sintef Ict.

Muggeridge, A., Cockin, A., Webb, K., Frampton, H., Collins, I., Moulds, T., Salino, P., 2014. Recovery rates, enhanced oil recovery and technological limits. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 372 (2006), 20120320.

Nocedal, J., Wright, S., 2006. Numerical Optimization. Springer Science & Business Media.

OPM, 2017. Open Porous Media.
URL https://opm-project.org

OPM, 2018. OPEN POROUS MEDIA Flow Documentation Manual (2018-10).

Peaceman, D., 1977. Fundamentals of Numerical Reservoir Simulation, Elsevier Scientific Publishing.

Peaceman, D. W., et al., 1983. Interpretation of well-block pressures in numerical reservoir simulation with nonsquare grid blocks and anisotropic permeability. Society of Petroleum Engineers Journal 23 (03), 531–543.

Sarma, P., Aziz, K., Durlofsky, L. J., et al., 2005. Implementation of adjoint solution for optimal control of smart wells. In: SPE reservoir simulation symposium. Society of Petroleum Engineers.

Satter, A., Varnon, J. E., Hoang, M. T., et al., 1994. Integrated reservoir management. Journal of Petroleum Technology 46 (12), 1–057.

Schlumberger, 2014. ECLIPSE User Manual, Technical Description. Version 2014.1. Schlumberger Ltd.

Sethi, S. P., 1983. Deterministic and stochastic optimization of a dynamic advertising model. Optimal Control Applications and Methods 4 (2), 179–184.

Sharaf, A. M., Elgammal, A. A., 2018. Novel AI-based soft computing applications in motor drives. In: Power Electronics Handbook. Elsevier, pp. 1261–1302.

Shi, Y., et al., 2001. Particle Swarm Optimization:Developments, Applications and Resources. In: Proceedings of the 2001 congress on evolutionary computation (IEEE Cat. No. 01TH8546). Vol. 1. IEEE, pp. 81–86.

Van der Poel, R., Jansen, J., 2004. Probabilistic analysis of the value of a smart well for sequential production of a stacked reservoir. Journal of Petroleum Science and Engineering 44 (1-2), 155–172.

Volkov, O., Bellout, M. C., 2017. Gradient-based production optimization with simulation-based economic constraints. Computational Geosciences 21 (5-6), 1385–1402.

Appendix A

Presented in Listing 6.1 is a complete JSON driver file for launching an optimization run in FieldOpt.

Listing 6.1: Configuration of the JSON file for launching well placement optimization run with PSO algorithm. The heel and toe coordinates of the wells are anonymized for confidentiality reasons (relative distances are preserved).

```
{
    "Global": {
        "Name": "OPM_FM_RELPR_MS",
        "BookkeeperTolerance": 1e-3
    },
    "Optimizer": {
        "Type": "PSO",
        "Mode": "Maximize",
        "Parameters": {
            "MaxGenerations": 138,
            "PSO-LearningFactor1": 2,
            "PSO-LearningFactor2": 2,
            "PSO-SwarmSize": 12,
            "PSO-VelocityScale": 0.25
        },
        "Objective": {
            "Type": "WeightedSum",
            "UsePenaltyFunction": false,
            "WeightedSumComponents": [
                {
                    "Coefficient": 377.389,
                    "Property": "CumulativeOilProduction",
                    "TimeStep": -1,
                    "IsWellProp": false
                },
                {
                    "Coefficient": -18.869,
                    "Property": "CumulativeWaterInjection",
                    "TimeStep": -1,
                    "IsWellProp": false
                },

```

```
                {
                    "Coefficient": -31.449,
                    "Property": "CumulativeWaterProduction",
                    "TimeStep": -1,
                    "IsWellProp": false
                }

            ]
        },
        "Constraints": [
            {
                "Wells": ["P1"],
                "Type": "ReservoirBoundary",
                "BoxImin": 86,
                "BoxImax": 102,
                "BoxJmin": 29,
                "BoxJmax": 62,
                "BoxKmin": 10,
                "BoxKmax": 44
              }
        ]
    },
    "Simulator": {
        "Type": "Flow",
        "FluidModel": "BlackOil",
        "ExecutionScript": "bash_flow.sh",
        "ScheduleFile": "OPM_FM_SCH.INC"
    },
    "Model": {
"ControlTimes": [
            0, 25, 50, 75, 100, 125, 150, 175, 200, 225, 25
               0, 275, 300],
        "Reservoir": {
            "Type": "Flow"
        },
        "Wells": [
            {
                "Name": "P1",
                "Group": "G1",
                "Type": "Producer",
                "DefinitionType": "WellSpline",
                "PreferredPhase": "Oil",
                "WellModel": "Projection",
                "WellboreRadius": 0.1905,
                "SplinePoints": {
```

```
                    "Heel": {
                        "x": 100000.00,
                        "y": 1000000.00,
                        "z": 1000.00,
                        "IsVariable": true
                    },
                    "Toe": {
                        "x": 98724.38,
                        "y": 1002071.07,
                        "z": 870.48,
                        "IsVariable": true
                    }
                },
                "Controls": [
                    {
                        "TimeStep":    0,
                        "State": "Open",
                        "Mode": "BHP",
                        "BHP": 60.0,
                        "IsVariable": false
                    }
                ]
            },
            {
                "Name": "P2",
                "Group": "G2",
                "Type": "Producer",
                "DefinitionType": "WellSpline",
                "PreferredPhase": "Oil",
                "WellModel": "Projection",
                "WellboreRadius": 0.1905,
                "SplinePoints": {
                    "Heel": {
                        "x": 200000.00,
                        "y": 2000000.00,
                        "z": 2000.00,
                        "IsVariable": false
                    },
                    "Toe": {
                        "x": 199946.02,
                        "y": 2001375.49,
                        "z": 1871.05,
                        "IsVariable": false
                    }
                },
```

```
                "Controls": [
                    {
                        "TimeStep":    0,
                        "State": "Open",
                        "Mode": "BHP",
                        "BHP": 60.0,
                                "IsVariable": false
                    }
                ]
            },
            {
                "Name": "P3",
                "Group": "G3",
                "Type": "Producer",
                "DefinitionType": "WellSpline",
                "PreferredPhase": "Oil",
                "WellModel": "Projection",
                "WellboreRadius": 0.1905,
                "SplinePoints": {
                  "Heel": {
                      "x": 300000.00,
                      "y": 3000000.00,
                      "z": 3000.00,
                      "IsVariable": false
                  },
                  "Toe": {
                      "x": 299635.01,
                      "y": 3001285.53,
                      "z": 2919.47,
                      "IsVariable": false
                  }
                },
                "Controls": [
                    {
                        "TimeStep":    0,
                        "State": "Open",
                        "Mode": "BHP",
                        "BHP": 60.0,
                                "IsVariable": false
                    }
                ]
            },
            {
                "Name": "P4",
                "Group": "G4",
```

```
                "Type": "Producer",
                "DefinitionType": "WellSpline",
                "PreferredPhase": "Oil",
                "WellModel": "Projection",
                "WellboreRadius": 0.1905,
                "SplinePoints": {
                  "Heel": {
                      "x": 400000.00,
                      "y": 4000000.00,
                      "z": 4000.00,
                      "IsVariable": false
                  },
                  "Toe": {
                      "x": 399714.17,
                      "y": 4000845.09,
                      "z": 3981.07,
                      "IsVariable": false
                  }
                },
                "Controls": [
                    {
                        "TimeStep":    0,
                        "State": "Open",
                        "Mode": "BHP",
                        "BHP": 60.0,
                                "IsVariable": false
                    }
                ]
            },
            {
                "Name": "I1",
                "Group": "I1",
                "Type": "Injector",
                "DefinitionType": "WellSpline",
                "PreferredPhase": "Water",
                "WellModel": "Projection",
                "WellboreRadius": 0.1905,
                "SplinePoints": {
                  "Heel": {
                      "x": 500000.00,
                      "y": 5000000.00,
                      "z": 5000.00,
                      "IsVariable": false
                  },
                  "Toe": {
```

```
                    "x": 499146.29,
                    "y": 5000555.74,
                    "z": 5010.48,
                    "IsVariable": false
                }
            },
            "Controls": [
                {
                    "TimeStep":    0,
                    "State": "Open",
                    "Mode": "BHP",
                    "BHP": 500.0,
                            "Rate": 1500.0,
                            "IsVariable": false
                }
            ]
        },
        {
            "Name": "I2",
            "Group": "I2",
            "Type": "Injector",
            "DefinitionType": "WellSpline",
            "PreferredPhase": "Water",
            "WellModel": "Projection",
            "WellboreRadius": 0.1905,
            "SplinePoints": {
              "Heel": {
                  "x": 600000.00,
                  "y": 6000000.00,
                  "z": 6000.00,
                  "IsVariable": false
              },
              "Toe": {
                  "x": 599678.81,
                  "y": 6000508.2,
                  "z": 6017.65,
                  "IsVariable": false
              }
            },
            "Controls": [
                {
                    "TimeStep":    0,
                    "State": "Open",
                    "Mode": "BHP",
                    "BHP": 500.0,
```

```
                                    "Rate": 1500.0,
                                    "IsVariable": false
                                }
                            ]
                        }

                ]
            }
}
```

Appendix B

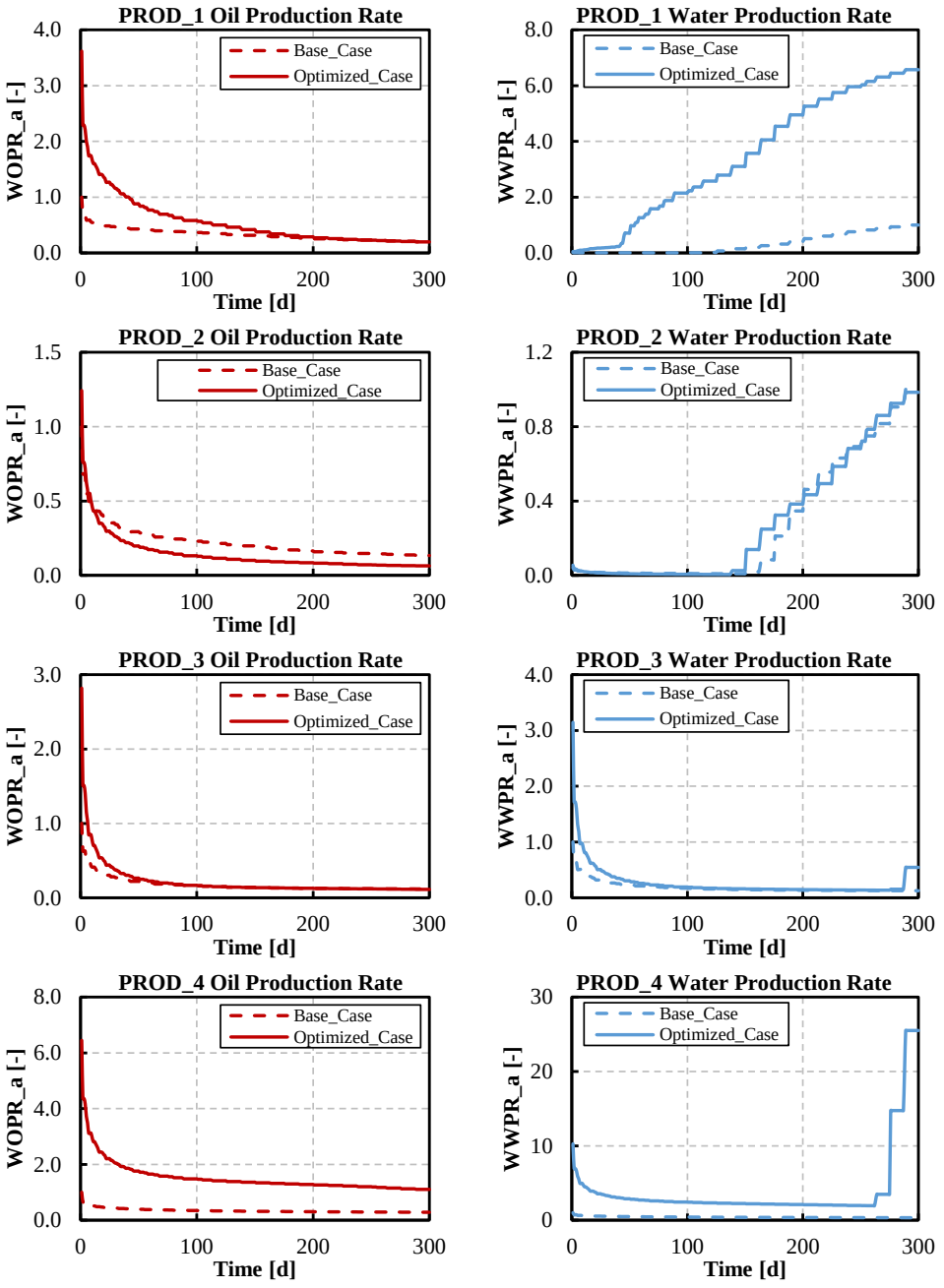


Figure 6.1: Well oil and water production rates in the optimized well placement scenario (as a fraction of corresponding rates in the base case). The suffix "a" on the title of y-axis signifies that the values have been normalized for confidentiality reasons.