NTNU

Norwegian University of
Science and Technology

Production Optimization of a field with ESP lifted wells

Øystein Kristoffersen

Master of Science in Engineering and ICT
Submission date: June 2017
Supervisor: Milan Stanko, IGP

Norwegian University of Science and Technology
Department of Geoscience and Petroleum

Summary

Mathematical optimization is a set of techniques used to find the optimal points of some function given known constraints. This work aims to apply such techniques to find optimal production of oil from a network of wells where each wells have an ESP (Electric Submersible Pump) installed for artificial lift. The optimization model will attempt to optimize the total oil production while keeping constraints on water treatment capacity and local conditions at the pump. As optimizing non-linear functions is not efficient, a piecewise linear approximation (PWL) is created for oil rate as a function of frequency of the ESP for each well. This ensures linearity. Integer variables were introduced to the optimization model to turn wells on and off. The optimization model is then a MILP (Mixed Integer Linear Programming) problem, a well known class of problems in optimization theory that in practise can be solved efficiently compared to a non-linear MINLP problem, for which there exist is no general solving algorithm.

The scope of this work is to implement a optimization model in 2 different ways. First Microsoft Excel is used to optimize a 2 well network model where the production merges in a flow line. A third party add-on to Excel called OpenSolver is used to expand the optimization capabilities of Excel. In the second part the optimization model is implemented in AMPL. For this part, the optimization theory is applied on the Peregrino field offshore from Brazil. The field contains 27 producing wells. Pipesim is used as a reservoir model simulator in both parts, however for the Peregrino field the Pipesim model only contains one well, then single branch analysis is performed inputting data for each well in turn on the single branch model using a VBA macro in Excel. The optimization model for the Peregrino field is also slightly lower in complexity compared to the 2 well example field in part 1, as there is no common flowline to merge the flow from all the wells.

The optimization model implemented in AMPL for part 2 is successful. It provides optimal operating points for the ESP of each well while handling multiple operational constraints. All proposed cases ran without problem with very low runtimes. For part 1, OpenSolver in Excel struggled with many aspects of the proposed model. Many simplifications were made, including fixed f_{ESP} and WC for both wells. In the end OpenSolver manged to solve the network, but the simplifications means the optimization model is too low in complexity to have practical application and finding optimal operating points.

Sammendrag

Matematisk optimering handler om å finne optimale verdier for en funksjon ved å bruke forskjellige optimeringsteknikker. Dette arbeidet tar sikte på å bruke slike teknikker til optimalisere oljeproduksjonen fra et nettverk med oljebrønner hvor hver brønn har en ESP (Electric Submersible Pump) installert for ekstra kunstig løft. Optimeringsmodellen vil prøve å optimere den totale oljeproduksjonen samtidig som begrensninger i forhold til maksimalt tillatt vannproduksjon og lokale strømningsbegrensinger ved den installerte ESP blir overholdt. Ettersom optimering av ulineære funksjoner kan være svært vanskelig, ble en stykkevis linearisert approksimasjon introdusert for funksjonen av oljeproduksjon gitt frekvens til brønnens ESP. Dette lineariserer funksjonen og gjør problemet med å finne optimale verdier av funksjonen lettere. Heltallsvariabler ble introdusert til optimeringsmodellen for å kunne skru en brønn av og på. Problemet med å optimere oljeproduksjonen er da et MILP problem (Mixed Integer Linear Programming), en kjent klasse med problemer som kan løses effektivt sammenlignet med ulineære problemer (MINLP), hvor ingen generell algoritme er kjent.

Omfanget av dette arbeidet er å implementere en optimeringsmodell på to forskjellige måter. I første del brukes Microsoft Excel til å optimere et testfelt med 2 brønner hvor strømningen fra hver brønn møtes i en felles rørledning opp til land. Et tredjeparts programvaretillegg for Excel kalt OpenSolver blir brukt for å utvide optimeringsfunksjonaliteten som er tilgjengelig i Excel. I andre del av prosjektet ble en optimeringsmodell implementert i AMPL. I denne delen ble optimeringsteorien prøvd ut på et felt kalt Peregrino som ligger på kontinentalsokkelen utenfor Brasil. Feltet produserer fra 27 brønner. Pipesim brukes som reservoirmodell i begge deler av prosjektet. For Peregrino inneholder Pipesim-modellen derimot bare 1 brønn og et VBA skript som kjøres fra Excel brukes til å sette egenskapene til hver brønn etter tur når hvert datapunkt for lineariseringen av funksjonen genereres. Optimeringsmodellen til Peregrino er også litt mindre komplisert enn testfeltet med 2 brønner ettersom Peregrino ikke har en felles rørledning hvor strømningen fra hver brønn møtes.

Optimeringsmodellen implementert med AMPL for del 2 av prosjektet er en suksess. Optimeringen gir optimale frekvenser for alle pumper og overholder alle begrensningene som produksjon av vann o.l. I alle situasjoner optimeringen ble kjørt for kjørte programmet uten problemer og med lave kjøretider. For del 1, med OpenSolver og Excel, var det flere problemer med å kjøre den foreslåtte optimeringsmodellen. Mange forenklinger ble utført. Disse inkulderer fast f_{ESP} og WC for begge brønnene. Til slutt klarer OpenSolver å løse netverket, men den gjenværende modellen har for lav kompleksitet til å ha praktisk anvendelse i forhold til optimering.

Preface

This work is the thesis of my master's degree at Norwegian University of Science and Technology (NTNU). The thesis is written as a part of the Engineering & ICT (Ingeniørvitenskap & IKT) program with specialization within petroleum technology. The thesis is written at the Department of Geoscience and Petroleum and is a continuation of a preliminary project performed autumn 2016. However the thesis is a standalone work and can be read independently from previous work. The work of this thesis was perfomed between January and June of 2017.

I would like to thank Associate Professor Milan Stanko from the Department of Geoscience and Petroleum who supervised this work, introduced me to the topics of this thesis and gave guidance and useful discussions on the work. I would also like to give thanks to Arnaud Hoffmann of Petrostreamz AS for giving valuable input on optimization theory related to petroleum production systems and who also gave a lot of feedback on the AMPL program. Finally I would like to thank PhD candidate Diana Gonzales at the Department of Geoscience and Petroleum who always answered my questions and provided many useful suggestions.

Trondheim, June 11, 2017
Øystein Kristoffersen

Table of Contents

List of Tables

List of Figures

Abbreviations

$AMPL$	=	A Mathematical Programming Language
API	=	Application programming interface
ESP	=	Electrical Submersible pump
GOR	=	Gas oil ratio[$\frac{scf}{stb}$]
IP	=	Integer Programming
LP	=	Linear Programming
$MILP$	=	Mixed Integer Linear Programming
$MINLP$	=	Mixed Integer Non-Linear Prorgramming
PI	=	Productivity Index
PWL	=	Piecewise Linear
$s.t.$	=	Subject To
VBA	=	Visual Basic for Applications
WC	=	Water Cut [$\%$]

Nomenclature

All flow rates in [$\frac{STB}{d}$]

q_{liq}	=	Liquid rate
q_o	=	Oil rate
q_w	=	Water rate
q_g	=	Gas rate
f_{ESP}	=	ESP frequency [Hz]
f_{ref}	=	ESP reference frequency, usually 60Hz
W_{ESP}	=	ESP power consumption [HP]

All pressure in [psi]

P_{wh}	=	Wellhead pressure
P_{res}	=	Reservoir pressure
P_{sep}	=	Seperator pressure
P_{suc}	=	Pressure at suction of the ESP
P_{in}	=	Pressure at inlet of a pipe
ΔP	=	A difference in Pressure
λ	=	Used to denote SOS2 variables
Ω	=	Used to denote SOS2 variables
$\mathcal{P}, \mathcal{F} \ldots$	=	Denotes a set of breakpoints

Other subscripts and superscripts:

loc	=	Local conditions
tot	=	Total
min	=	Minimum
max	=	Maximum

Chapter 1

Introduction

1.1 Background

During the lifetime of an oil field the water cut in the wells usually increase. As some wells get a high water cut and some new wells may be drilled, there will be variation in the water cut among the wells of the field, and it can be beneficial to turn down the production of some wells producing a lot of water in order to let wells producing more petroleum produce at a higher rate. When wells produce at the range of 96 - 98 % water, closing the well can be considered. Other constrains like water treatment capacity and constraints on local liquid flow and pressure in installed pumps etc. can also be taken into account when deciding which wells produce at what capacity. The fields introduced in this thesis all use ESP pumps for artificial lift. The frequency of the ESP can be increased and decreased to change the production of a well. This becomes a variable when deciding optimal operating point for each well to optimize the oil production given various constraints on the field. As the functions of liquid as a function of pump frequencies are very complicated and time consuming to optimize, this thesis aims to apply various optimization techniques in a way that efficiently calculates the operating points of every well in a oil field.

Constraints are put on fields to operate within local regulations and in general to produce efficiently and maximize profit. Water produced from oil fields is severely polluted and need to be cleaned and properly treated before it can be disposed of in the nature or in the ocean. There is limited water treatment capacity on the production vessels, giving a water treatment constraints. Another constraint could be the capacity of the power supply for the ESP pumps. From this kind of constraint arise the necessity of optimizing the production given the constraints.

1.2 Objective

The project is divided into two major parts.

- The first part aims to be a proof of concept for optimizing a general oil field. A simple field model containing two wells with flows merging in a flowline is used. This part uses Excel and a Pipesim model of the field, and will be referenced to as ”the Excel optimization”. This part is a continuation and wrap up of a preliminary project to this thesis performed autumn 2016 also at NTNU.
- The second part aims to apply the theory on a real world case; modelling and optimizing production of the Peregrino which is located offshore in Brazil. Excel will still be used for generating and storing field data, but a more suitable tool, AMPL, will be used to model the optimization model and run the optimization.

1.3 Previous work

A primary inspiration for this work is Hoffmann and Stanko (2016), which aims to optimize a field with an ESP installed in each well using similar techniques as this project. The paper studies a network of wells located in clusters with the flow off each cluster merging in a flowline. The first part of this project tries to study this type of field by using a simpler 2 well field model. The second part of this project focuses on the Peregrino field. The master thesis (Hollund, 2010) written at UiS (University of Stavanger) studies ESP design for the Peregrino field. Other papers studying the ESP design for the Peregrino field include (Olsen et al., 2011) and (Castro and Leite, 2015). Another master thesis written at NTNU, Røyset (2013) studies optimization of oil fields modeled with Pipesim, using a software called Pipe-It by Petrostreamz AS to run optimization. The preliminary project to this work also worked with the Pipe-It template from Røyset (2013), as well as starting work on automation of generating breakpoints from Excel using the Pipesim API called OpenLink. (Ranjeva et al., 2014) studies the Peregrino field with focus is on completion design.

1.4 Artificial lift and ESP

In the early phase of an oil field, the ΔP (difference in pressure) between reservoir pressure and wellhead pressure might be sufficient for the well to produce on its own. As time progresses the pressure in the reservoir sinks gradually and at some point production cannot be sustained naturally. Artificial lift are techniques to artificially increase the lifting capacity of a well. Pumps are a very common way of introducing artificial lift to the field. Gas lift is another example. All fields in this project have wells that have Electric Submersible Pumps (ESP) installed downhole. The ESP is powered by an electric motor which is also installed downhole. The frequency of the ESP can be varied to change the amount of extra lift provides, and thus the production rate of the well.

1.5 Peregrino

Peregrino is a field located in Brazil. It is an offshore field producing a heavy oil with high viscosity (Castro and Leite, 2015) and very low GOR and bubble point pressure (Hollund, 2010). These properties makes ESP a good choice for the Peregrino field, as they are efficient with high lift capacity and low energy consumption. Gas lift would not be optimal due to the low GOR (Hollund, 2010). There are 27 production wells at around 2500 beneath the ocean floor. The water depth is around 100 - 120 meters Hollund (2010). The wells are categorized in three groups, low flow-rate producers, medium flow-rate producers and high flow-rate producers Castro and Leite (2015). When making a well model for this project, 10 wells of low and medium producers, and 7 high producers where assumed, for a total 27 wells. Table 1.1 and table 1.2 shows some basic properties of the Peregrino field and the fluid produced:

Property	**Value**	**Unit**
Initial Reservoir Pressure	231 (3352)	bar (psi)
Reservoir Temperature	78.6 (173.5)	°C (°F)
Datum	2340 (7678)	m (ft)
Pump Setting Depth	1960 (6431)	m (ft)
Pump Frequency	40 - 65	Hz

Table 1.1: Field properties for the Peregrino field.

Property	**Value**	**Unit**
Oil Gravity	14	°API
Gas Specific Gravity	0.732	air = 1
Water Specific Gravity	1.07	water = 1
Gas Oil Ratio (GOR)	73	SCF/STB
Gas impurities N_2 / H_2S / CO_2	0.23 / 0.00 / 0.07	% mol vol
Oil viscosity @ Initial Reservoir Condition	163	cP
Bubble Point Pressure @ Reservoir Temperature	700	psi
Bo @ Bubble Point	1.053	Bbl / STB

Table 1.2: Baisc fluid properties of Peregrino field.

1.6 Software

1.6.1 Excel

Excel (Microsoft Office, 2016) is a spreadsheet application and part of the Microsoft Office suite. Excel is used to generate and store data for this project. Excel contains a VBA (Visual Basic for Applications) interface, allowing the user to write macros for the spreadsheets, automatizing processes and setting up, importing and exporting data. An add-on to Excel called OpenSolver (Frontline Systems, 2017) is a third party open source software used to set up the optimization case based on the cells in the Excel spreadsheet. A cell can be input as objective cell and in a constraint relation. It works similar to the built in solver, but is more powerful as it can handle more variables, and allows the optimization model to be solved by commercial solver.

1.6.2 Pipesim

Pipesim (Schlumberger, 2012) is a reservoir simulator, or more precisely a petroleum production system modelling and simulation tool. It is currently being developed by Schlumberger. Pipesim can model many aspects of a field and run a simulation to calculate unknown parameters. Both networks with many wells and single wells can be modeled and simulated. There is an API for Pipesim called OpenLink. OpenLink can be programmed with VBA in Excel, making it possible to make macros to automatically input parameters to the field model and run multiple simulations.

1.6.3 AMPL

AMPL (A Mathematical Programming Language) (AMPL Optimization, Inc., 2017) is a programming language designed to solve large scale mathematical problems. The syntax of AMPL is very close to mathematical notation for optimization problems, making it easy to design and implement an optimization model. AMPL is able to use powerful solvers, commercial and open source, to solve a wide variety of problem classes within mathematical optimization theory.

Chapter 2

Theoretical foundation

2.1 Introduction to Optimization

Mathematical optimization is a way of finding an optimal value of a function given certain constraints. The goal is to find a maximum or minimum value given certain constraint. This thesis will deal with maximizing functions, but it is important to note that for a function f

$$maximize\ (f) \equiv minimize\ (-f) \tag{2.1}$$

which means that theory for maximization problems is also valid for minimization problems.

2.1.1 Linear Programming (LP)

Linear Programming is a method of doing optimization were both objective function and constraints in the optimization model are linear. Linear Programming problems can generally be written in the following way:

$$maximize\ \mathbf{c}^T\mathbf{x} \tag{2.2}$$

$$s.t. \tag{2.3}$$

$$A\mathbf{x} \leq \mathbf{b} \tag{2.4}$$

$$\mathbf{x} \geq 0 \tag{2.5}$$

$$\mathbf{x} \in \mathbb{R} \tag{2.6}$$

where x is the vector of variables that will be changed to optimize the objective function, $\mathbf{c}$ and $\mathbf{b}$ are vectors of coefficients that are known, $\mathbf{c}$ is for the objective function and $\mathbf{b}$ represent the constants of the set equations representing the constraints of the model. A is a matrix with coefficients for the equations representing the set of constraint equations.

These definitions of the basic form of a LP problem will be important when the problems are expanded beyond linearity.
Linear programming problems have many useful properties. The feasible region for a LP problem is always a convex region. This means that when a local maximum is found, that maximum if also a global maximum. An example of a non-convex function is seen in figure 2.3, which shows a non-linear problem. A LP problem has no solution if and only if 2 constraints contradict each other, or if the feasible region is unbounded, for instance a maximizing problem where the objective grows to infinity.
Figure 2.1 shows a LP problem where the grey lines represent the surface of the two dimensional function $f(x) = x + y$ and the feasible region where solutions to the optimization problem can be found. The feasible region is constrained by some given functions represented by colored lines (and also the axis). The goal of the optimization is to maximize the value of f within the feasible region.

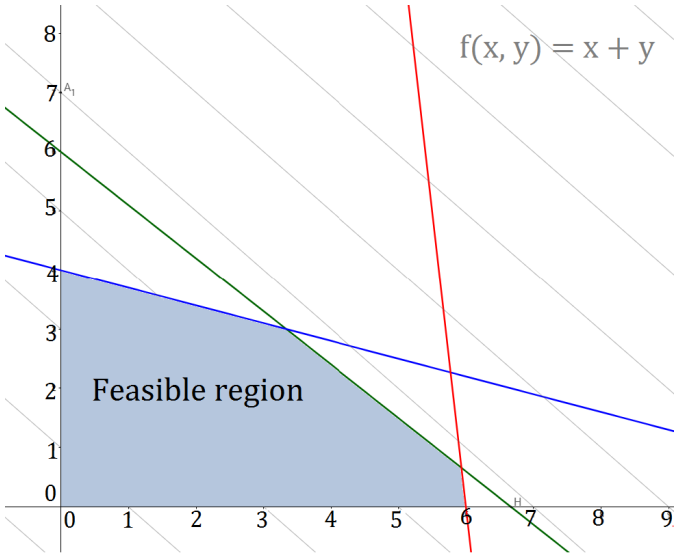


Figure 2.1: Linear programming: Linear objective function $f(x, y)$ and linear constraints represented by colored lines.

A common algorithm to solve linear programming problems is called Simplex. Simplex solves this class of problems efficiently, that is, in practise it solves the problem in polynomial time. There are however problems for which the Simplex performs poorly, giving a theoretical exponential worst case runtime.

2.1.2 Integer Programming (IP)

Integer programming is a method of optimization where only integer values can be valid solutions.

$$maximize\ \mathbf{c}^T\mathbf{x} \tag{2.7}$$

$$s.t. \tag{2.8}$$

$$A\mathbf{x} \leq \mathbf{b} \tag{2.9}$$

$$\mathbf{x} \geq 0 \tag{2.10}$$

$$\mathbf{x} \in \mathbb{Z}^n \tag{2.11}$$

There is no demand for linearity in IP in the definition, but it is common to assume linear objective and constraints when talking about IP, in which case it can be relevant to refer to it as ILP (Integer Linear Programming). In the figure below the same problem as figure 2.1 is shown, however the feasible region now only consist of integer values marked as red dots.

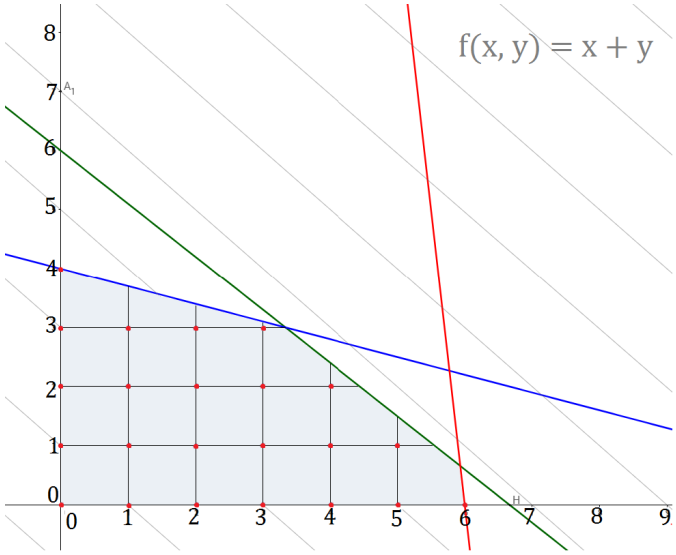


Figure 2.2: Integer programming: Linear objective function $f(x, y)$ and constraints, but only integer values (red dots) are feasible solutions

Solving IP problems is harder than solving LP problems. Solving IP problems is NP-hard, meaning there is no algorithm that can efficiently solve a general IP problem in polynomial time.

2.1.3 Mixed Integer Linear Programming (MILP)

When some variables are integers and others are continuous, the problem is a Mixed Integer Linear Programming problem, or MILP. This class of program also requires linear objective and constraint functions. A general form is

$$maximize\ \mathbf{c}^T\mathbf{x} \tag{2.12}$$

$$s.t. \tag{2.13}$$

$$A\mathbf{x} \leq \mathbf{b} \tag{2.14}$$

$$\mathbf{x} = \{\mathbf{x}_C, \mathbf{x}_I\} \tag{2.15}$$

$$\mathbf{x}_C \geq 0, \mathbf{x}_C \in \mathbb{R} \tag{2.16}$$

$$\mathbf{x}_I \in \mathbb{Z}^n \tag{2.17}$$

where $\mathbf{x}_C$ are continuous variables, and where $\mathbf{x}_I$ are integer variables. MILP problems are very useful, and the main interest for this thesis. Models introduced in this thesis will contain binary variables, that is integers between 0 and 1, and continuous variables while still keeping both objective and constraints linear.

2.1.4 Mixed Integer Non-Linear Programming (MINLP)

If either objective function or a constraint is not linear, we have a non-linear problem. Like a MILP, the variables of a MINLP can be a mix of integer variables and continuous variables. Solving this class of problems is very hard, and there is no general algorithm to solve every problem in this class. There exist techniques to handle sub-classes of MINLP, but this project focuses on linearizing the functions to end up with MILP.

$$maximize\ \mathbf{c}^T\mathbf{x} \tag{2.18}$$

$$s.t. \tag{2.19}$$

$$A\mathbf{x} \leq \mathbf{b} \tag{2.20}$$

$$\mathbf{x} = \{\mathbf{x}_C, \mathbf{x}_I\} \tag{2.21}$$

$$\mathbf{x}_C \geq 0, \mathbf{x}_C \in \mathbb{R} \tag{2.22}$$

$$\mathbf{x}_I \in \mathbb{Z}^n \tag{2.23}$$

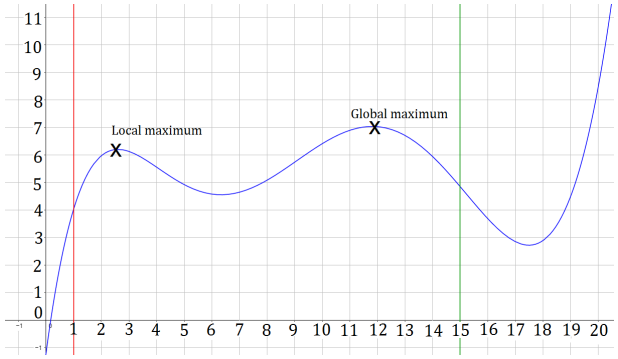


Figure 2.3: Non-Linear programming: A found maximum is not necessarily a global maximum

2.2 Piecewise Linear Approximations (PWL)

A *Piecewise Linear Approximation* of a function is a way of linearizing a function. While there is no general way of solving a MINLP, that is a non-linear optimization problem, by linearizing the function a MILP is obtained. This works by dividing the function into breakpoints. A piecewise linear approximation of the function in figure 2.3 can be seen in 2.4 below. The black dots represent the break-points, while the black line segments between them represent the new linear function between to given break points. The continuous original graph can be seen as a blue line in the background. To obtain function values for input between two breakpoints, linear interpolation is used. To model this for the optimization problem, we introduce variables that work together as a *special ordered set* to interpolate between the breakpoints.

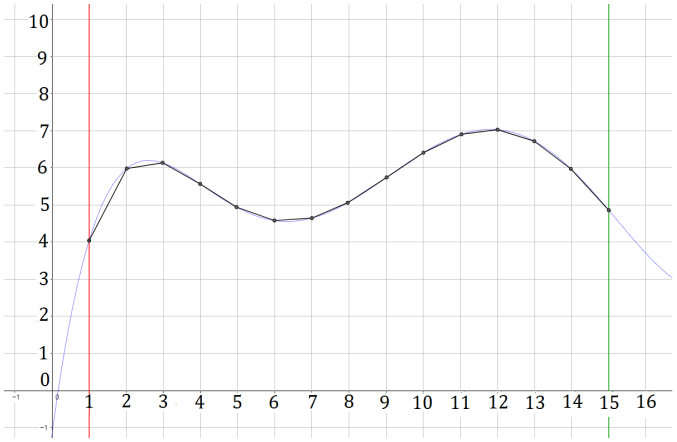


Figure 2.4: Black line shows Piecewise Linear approximation of the blue function on the range constrained by the red and green lines. Each segment in the PWL can be treated as a linear function.

2.2.1 SOS2 variables

A *Special Ordered Set* is a set of variables for optimization models that are designed to choose optimal breakpoints of a piecewise linear approximation of a function. A SOS2 is a type of special ordered set that is useful for finding optimal values from the breakpoints of a PWL function. The rules of the set is

$$\sum_{SOS2} \lambda = 1,\ \lambda \geq 0 \tag{2.24}$$

and also that only two consecutive variables in the set can be greater than 0, meaning the rest of the variables in the set are 0. There is one SOS2 variable for each breakpoint. If an optimal solution $f_{optimal}$ is located halfway between breakpoint i and $i+1$, then $\lambda_i = 0.5$ and $\lambda_{i+1} = 0.5$ and the rest of the SOS2 variables are 0. The solution to the problem is then the sumproduct of the function-values of the breakpoints and the SOS2-variables.

$$f_{optimal} = \sum_{p \in \mathcal{P}} f(p) * \lambda_p \tag{2.25}$$

where $\mathcal{P}$ is the set of breakpoints. In this way the SOS2 variables are interpolating between the breakpoints. This can also be done in 2 dimensions as illustrated in figure 2.5. In 2 dimensions, there will be one set of SOS2 variables for each dimension $\lambda_{i,j}$. The figure plots data generated for a well with liquid rates as a function of P_{wh} and ESP frequency. Notice that the optimal value is not necessarily the highest value for liquid rate in the plot, as the optimization model have constraints over all wells that are not seen in the illustration. For instance, if the water cut of a well is high the optimization may reduce output from that well to keep a water treatment capacity constraint, and let another well produce a higher liquid rate.

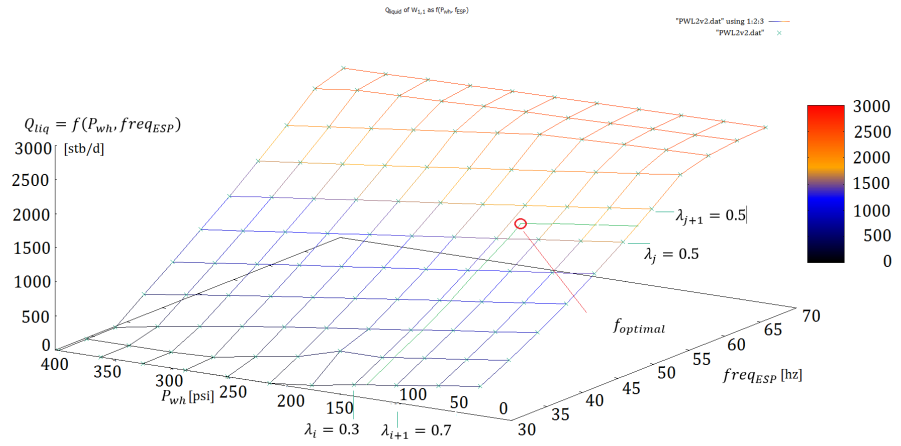


Figure 2.5: Data-points making up a Piecewise Linear approximation for a well: The liquid rate is a function of P_{wh} and ESP frequency. Two-dimensional SOS2 variables $\lambda_{i,j}$ are introduced to interpolate values between the data points.

2.3 Black Box optimization

To generate the data to use for the piecewise linear approximation of the objective function, a black box simulator is used. As the equations describing liquid rate as a function of ESP frequency is very complicated, the black box allows for abstracting away that complexity. A given ESP frequency is input to the black box, and a liquid rate comes out. The black box used in this project is Pipesim. Multiple variables can be input depending on the variables of the objective function.

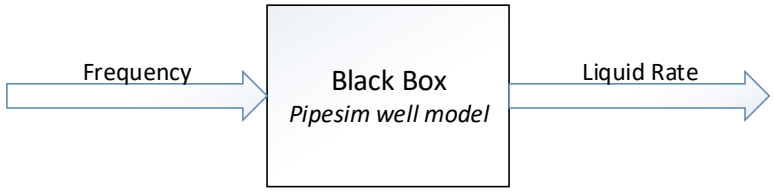


Figure 2.6: Generating data with a black box simulator.

2.4 ESP boost production system optimization

Two models for optimizing the oil production from petroleum fields will be presented. Fist the model used for the Peregrino will be presented. The Peregrino produces straight to the production vessel, with no common flowline. This means there is no need to do solve a network to make flow from many wells merge in a flowline. Secondly, the theory for solving the network will also be presented.

2.4.1 Peregrino-like fields, no network

The objective function will be the produced oil. Given a function

$$q_{liq}^{i} = f(F_{ESP}), i \in [wells] \tag{2.26}$$

where F_{ESP} is the frequency of the pump and the function f maps to the breakpoints for the Q_{liq} in the PWL table, the oil rate Q_o can be optimized as following:

$$maximize\ q_o^{tot} \tag{2.27}$$

$$subject\ to \tag{2.28}$$

$$q_{liq}^{i} = \sum_{p \in \mathcal{P}} q_{liq,p}^{i} * \lambda_p^i,\ i \in [wells] \tag{2.29}$$

$$q_o^{tot} = \sum_{i \in Wells} q_{liq}^{i} * (1 - WC^i) \tag{2.30}$$

$$q_w^{tot} = \sum_{i \in Wells} q_{liq,i} * WC_i \tag{2.31}$$

$$P_{suc}^{i} = \sum_{p \in \mathcal{P}} P_{suc,p}^{i} * \lambda_p^i,\ i \in [wells] \tag{2.32}$$

$$W_{esp}^{i} = \sum_{p \in \mathcal{P}} W_{ESP,p}^{i} * \lambda_p^i,\ i \in [wells] \tag{2.33}$$

where $\mathcal{P}$ is the set of breakpoints. In general, any property that is in the PWL table, can be defined in the same way as Q_{liq}, P_{suc} and W_{ESP} have been defined in equation 2.29, 2.32 and 2.33.
Constraints are defined as following:

$$q_w^{tot} \leq q_w^{max} \tag{2.34}$$

$$P_{suc}^{i} \geq P_b * sf,\ i \in [wells] \tag{2.35}$$

where sf is a safety factor for the bubble point pressure constraint.

In addition, the local flowrate at the ESP is restricted by the following conditions (Hoffmann and Stanko, 2016)

$$q_{liq}^{min,ESP}|_{f=f_{ref}} * \frac{f_{ESP}}{f_{ref}} \leq q_{liq}^{local,ESP} \leq q_{liq}^{max,ESP}|_{f=f_{ref}} \frac{f_{ESP}}{f_{ref}} \tag{2.36}$$

$$\forall\, i \in Wells, \sum_{F \in p} \lambda_F^i = z^i \tag{2.37}$$

Where and z is a binary variable turning the well on or off.

2.4.2 Solving the network

When wells are producing to a flowline, additional constraints have to be introduced to the optimization model in order to make sure the pressure and rates from each well converges in the flowline. Each well can produce at different water cut, making the calculations more complicated than a case were each well produce without a network. For each well, data must be available for different wellhead pressures P_{wh} to match the pressure into the flowline P_{in}. P_{in} is calculated by doing a counter-current pressure calculation on the flowline from the separator P_{sep}. This means that q_{liq}^i for each well i will be dependent on 2 variables in the data set.

$$q_{liq}^i = f(P_{wh}, F_{ESP}), i \in [wells] \tag{2.38}$$

Our goal is to have the flow of each well merge in the flowline, meaning that

$$P_{wh}^i = P_{in}, i \in [wells] \tag{2.39}$$

$$P_{in} = P_{sep} - \Delta P \tag{2.40}$$

$$\sum_{i \in wells} q_{liq}^i = q_{liq,flowline} \tag{2.41}$$

$$\sum_{i \in wells} WC^i = WC_{flowline} \tag{2.42}$$

The ΔP in the flowline is dependent on WC and $q_{liq,flowline}$ in the flowline, and can be calculated using a PWL and another two dimensional set of SOS2 variables.

$$\Delta P = \sum_{q \in \mathcal{Q}} \sum_{wc \in \mathcal{WC}} \Delta P_{q,wc} * \Omega_{q,wc} \tag{2.43}$$

The same strategy is used to find $q_{liq,flowline}$ and $WC_{flowline}$ for equation 2.41 and 2.42 from the same flowline PWL table.

$$q_{liq,flowline} = \sum_{q \in \mathcal{Q}} \sum_{wc \in \mathcal{WC}} q_{liq,q,wc} * \Omega_{q,wc} \tag{2.44}$$

$$WC_{flowline} = \sum_{q \in \mathcal{Q}} \sum_{wc \in \mathcal{WC}} q_{liq,q,wc} * \Omega_{q,wc} \tag{2.45}$$

$\mathcal{Q}$ and $\mathcal{WC}$ represent the two dimensions of the flowline table: Breakpoints for liquid rate and water cut. Now a P_{wh} for both wells have been established, leaving the two dimensional function for q_{liq} to only have solution at that P_{wh}. Then an optimal frequency at that P_{wh} for both wells is sought after in the original PWL table. The SOS2 variables for that table is also 2 dimensional as shown in equation 2.38. Except for the added dimension for searching for P_{wh}, the rest of the variables are found like in the previous section.

$$q^i_{liq} = \sum_{p \in \mathcal{P}} \sum_{f \in \mathcal{F}} q^i_{liq,pf} * \lambda^i_{p,f}, \; i \in [wells] \tag{2.46}$$

$$\vdots$$

Where $\mathcal{P}$ and $\mathcal{F}$ now represent the sets of breakpoints for P_{wh} and f_{ESP} respectively. As solutions with higher frequencies are tried by the solver, the q_{liw} increases, affecting the ΔP. A solver has to take both dimensions of both the $q_l iq$ PWL table and the ΔP flowline table in to account continuously, making this more complicated than a field with no network to solve.

Chapter 3

Methods

The project consist of two major parts. Optimizing a 2 well network using Excel and OpenSolver will be reffered to as ”Excel-optimization” and is the first part. The second part is using AMPL for optimization, while still using Excel to generate the tables of data from the well. This will be referred to as ”AMPL-optimization”.

3.1 2 well Excel optimization with network

For the first part of the project, Excel was used together with Pipesim and a third party solver addon for Excel called Open Solver. By using the API for Pipesim called Open Link, macros were written in Excel to automatize the process of running the optimization. This part is a continuation and completion of techniques studied in the preliminary project prior to the work on this thesis. Excel is used with OpenSolver as this combination provides a good way to learn and understand optimization theory by manually setting up the optimization in Excel. The OpenSolver provides a colored interface overlay on the Excel spreadsheet to indicate roles and relation of the different cells, see figure 3.3, 3.4 and 3.5. Basic properties of the field like reservoir pressure, fluid properties, dimension on pipes etc. are input directly in Pipesim when setting up the field. The parameters that are to change for each breakpoint, in this case P_{wh} and f_{ESP} are input with OpenLink commands in a script when running the VBA macro to generate data for many different operating points. The VBA macro will store the data in the Excel spreadsheet itself. Then a second VBA macro will use the stored data to set up PWL tables and OpenSolver. An overview of the workflow of the Excel optimization can be seen in figure 3.1.

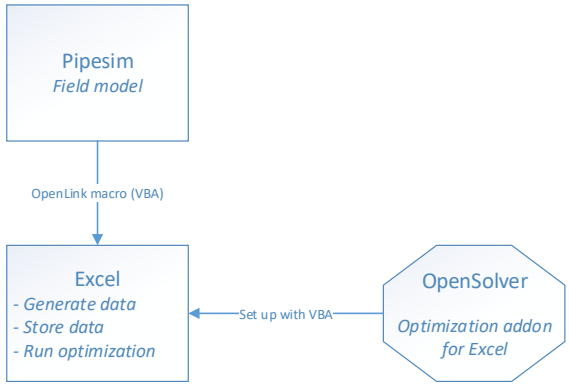


Figure 3.1: Workflow for running optimization on the Peregrino field with Excel

3.1.1 Pipesim model

The Pipesim model contains 2 wells. The properties of the field can be input to the field in Pipesim or by a VBA macro using the OpenLink API for Pipesim. The VBA macro will run the network file and extract data at the ”sink” node. Figure 3.2 shows the layout of the Pipesim model.

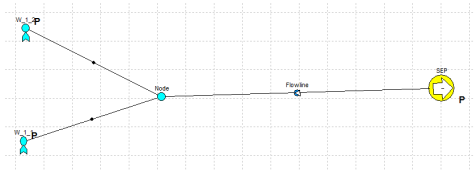


Figure 3.2: The Pipesim model used for

3.1.2 Sheet layout

The Excel document consist of 2 sheets. The first sheet called ”Data” is where raw data generated from Pipesim is stored. The second sheet is called ”PWL” and a macro will automatically convert the raw data from the ”Data” sheet into PWL tables in the ”PWL” sheet. The macro also sets up the optimization model for the OpenSolver add-on.
Figure 3.3 shows a section of the ”PWL” sheet where the breakpoints are set up with the SOS2 variables in the ”LAMBDA” column. The colored boxes indicate that the cells are part of the OpenSolver model and the cells role in the model. Notice that only data for one frequency has been used to make the PWL table one dimensional and simplify the optimization by only solving for $P^i_{wh} = P_{sep} - \Delta P_{flowline}$. Some simplifications were made for the Excel optimization. This is discussed further in the result section.

Figure 3.4 shows the section for the results in the ”PWL” sheet. The result for oil production for a given well, will be sum product of the oil rate at all data points for that well, with the SOS2 variables for that well, as explained in the chapter ”Theoretical foundation”. This section also contains the values for constraints on water treatment and P_{suc} for the ESP.
Figure 3.5 shows the flowline table set up with SOS2 variables (”OMEGA” column). The sum product of each column (”WC”, ”Qliq” and ”deltaP”) with the SOS2 variable column will give the wanted ΔP based on q_{liq} and WC from the two wells. The result is shown in the result-section of the sheet shown in figure 3.4. The constraints to OpenSolver make sure that ΔP in the flowline matches the water cut and total liquid rate coming from the two wells. Then the P_{wh} for both well is $P_{sep} - \Delta P_{flowline}$. The SOS2 variables for the well data (figure 3.3) will choose data only at this P_{wh}.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	DATA - PWL TABLES																
2	Well	Pwh	Frequency	Power	Psuc	Qlocal	Qliq	WC	GLR	Qo	Qw	Qg	LAMBDA				
3	M4:M25,C	[Psi]	[Hz]	[HP]	[Psi]	[STB/d]	[STB/d]	[%]	[scf/STB]	[STB/d]	[STB/d]	[STB/d]					
4	W_1_1	27	66	122.98	37	3175.50	2545.63	22	3.90	1985.59	560.04	99.28	1	1	1	sumx	1= 1
5	W_1_1	64	66	124.92	40	3083.00	2535.42	22	3.90	1977.63	557.79	98.88	0	1	0	sumy	1= 1
6	W_1_1	101	66	126.67	44	3002.40	2525.22	22	3.90	1969.67	555.55	98.48	0	0	0		
7	W_1_1	138	66	128.42	48	2923.20	2513.86	22	3.90	1960.81	553.05	98.04	0	0	0		
8	W_1_1	175	66	130.09	52	2843.20	2501.83	22	3.90	1951.43	550.40	97.57	0	0	0		
9	W_1_1	212	66	131.75	57	2776.20	2488.41	22	3.90	1940.96	547.45	97.05	0	0	0		
10	W_1_1	249	66	133.27	62	2707.30	2474.11	22	3.90	1929.81	544.31	96.49	0	0	0		
11	W_1_1	286	66	134.69	68	2642.50	2458.97	22	3.90	1918.00	540.97	95.90	0	0	0		
12	W_1_1	323	66	136.08	75	2578.50	2442.24	22	3.90	1904.95	537.29	95.25	0	0	0		
13	W_1_1	360	66	137.4	82	2517.30	2424.33	22	3.90	1890.98	533.35	94.55	0	0	0		
14	W_1_1	397	66	136.41	109	2440.20	2353.80	22	3.90	1835.97	517.84	91.80	0	0	0		
15	W_1_2	27	66	104.5	569	3325.70	3269.49	94	0.30	196.17	3073.32	9.81	1	1	1	sumx	1= 1
16	W_1_2	64	66	107.61	575	3274.70	3213.41	94	0.30	193.16	3026.25	9.66	0	1	0	sumy	1= 1
17	W_1_2	101	66	110.59	581	3223.20	3168.83	94	0.30	190.13	2978.70	9.51	0	0	0		
18	W_1_2	138	66	113.36	588	3172.90	3119.49	94	0.30	187.17	2932.32	9.36	0	0	0		
19	W_1_2	175	66	116.1	594	3120.40	3067.88	94	0.30	184.07	2883.80	9.20	0	0	0		
20	W_1_2	212	66	118.67	600	3068.20	3016.71	94	0.30	181.00	2835.70	9.05	0	0	0		
21	W_1_2	249	66	121.06	607	3016.60	2966.05	94	0.30	177.96	2788.08	8.90	0	0	0		
22	W_1_2	286	66	123.28	613	2965.60	2915.99	94	0.30	174.96	2741.03	8.75	0	0	0		
23	W_1_2	323	66	125.41	619	2913.10	2864.40	94	0.30	171.86	2692.53	8.59	0	0	0		
24	W_1_2	360	66	126.92	625	2861.5	2813.745	94	0.30	168.82	2644.92	8.44	0	0	0		
25	W_1_2	397	66	126.5	632	2810.3	2763.458	94	0.30	165.81	2597.65	8.29	0	0	0		
26																	

Figure 3.3: Data from the data sheet organized into PWL tables with SOS2 variables in the ”LAMBDA” column.

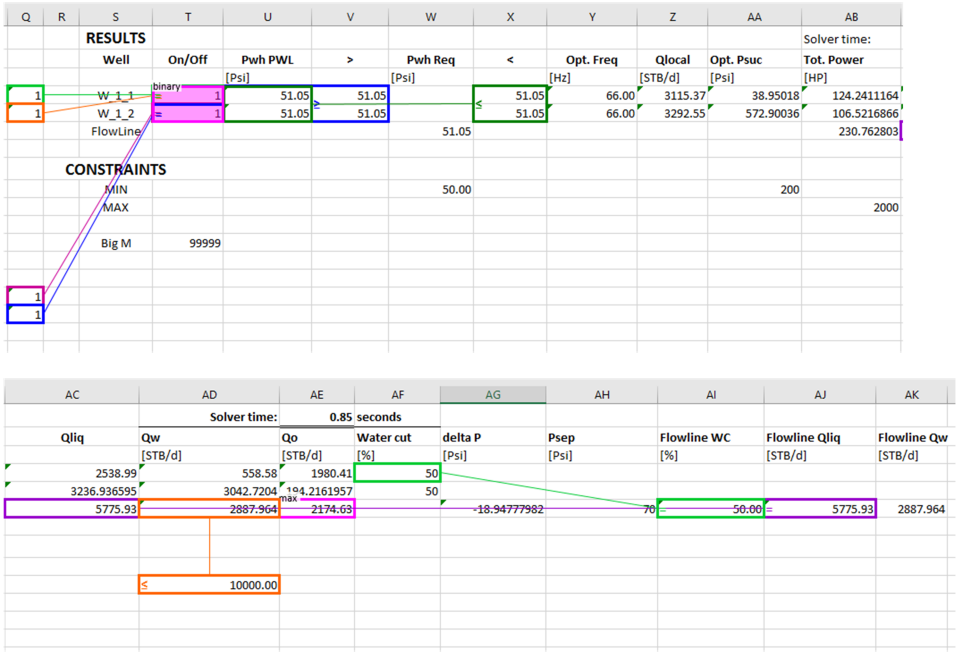


Figure 3.4: Result rows and figure of pipesim network

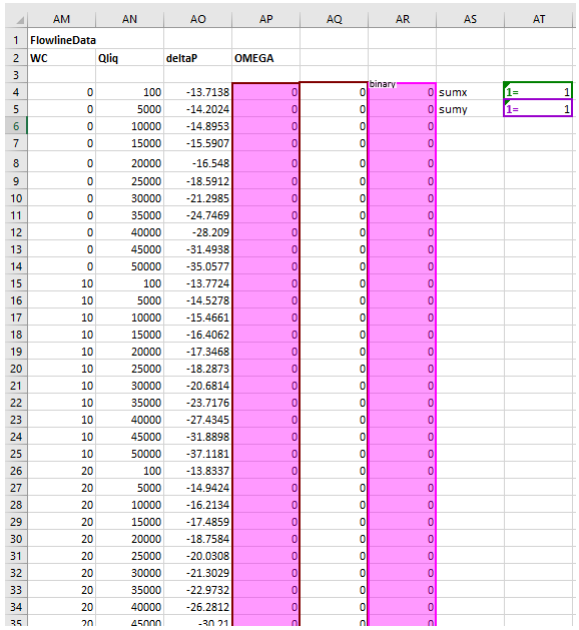


Figure 3.5: Flowline table

3.1.3 Variables and constants

This section shows the different variables and constants used in the optimization sheet. The colors indicate roles and properties of the different variables and constants.

Colors:
variable for OpenSolver
constant
varies through Excel-relations
objective (also vary through Excel-relation)

λ_p , used to calculate P_{wh} per well
$\Omega_{p,wc}$, used to calculate $P_{in,flowline}$ and ΔP from Flow Line Table

Pressures:
P_{wh}^j , $j \in wells$ (here: 2 wells)
$P_{in,flowline}$, Pressure into flowline
P_{sep} , Pressure in the sepeartor

Liquid flow by well:
Q_{liq}^j , $j \in wells$, Liquid flow per well
Q_w^j , $j \in wells$, Water flow per well
Q_o^j , $j \in wells$, Water flow per well
$Q_{liq,tot}$, Sum liquid flow
$Q_{o,tot}$, Sum oil flow
$Q_{w,tot}$, Sum water flow

Liquid flow in flow line:
$Q_{liq,flowline}$, Liquid flow in flowline
$Q_{w,flowline}$, Water flow in flowline

Water cut:
WC^j ,$j \in wells$
$WC_{flowline}$
$BigM$, a big constant (set to for instance 9999)
$onOff$, binary variable to to turn well one or off

3.1.4 Equations

Maximize $Q_{o,tot}$
By changing λ^j , $\Omega_{p,wc}$
Realtions in Excel sheet
From well data:

$$P^j_{wh} = \sum_{p \in \mathcal{P}} \lambda^j \cdot P^j_{wh,p},\ j \in wells \tag{3.1}$$

$$Q^j_{liq} = \sum_{p \in \mathcal{P}} \lambda^j \cdot Q^j_{liq,p},\ j \in wells \tag{3.2}$$

$$Q^j_{w} = \sum_{p \in \mathcal{P}} \lambda^j \cdot Q^j_{w,p},\ j \in wells \tag{3.3}$$

$$Q^j_{o} = \sum_{p \in \mathcal{P}} \lambda^j \cdot Q^j_{o,p},\ j \in wells \tag{3.4}$$

$$Q_{liq,tot} = \sum_{j \in wells} Q^j_{liq,tot} \tag{3.5}$$

$$Q_{o,tot} = \sum_{j \in wells} Q^j_{o} \tag{3.6}$$

$$Q_{w,tot} = \sum_{j \in wells} Q^j_{liq} \cdot WC^j \tag{3.7}$$

From the flowline table:

$$\Delta P_{flowline} = \sum_{q \in \mathcal{Q}} \sum_{wc \in \mathcal{WC}} \Omega_{q,wc} \cdot p_{q,wc} \tag{3.8}$$

$$P_{in,flowline} = P_{sep} - \Delta P_{flowline} \tag{3.9}$$

$$Q_{liq,flowline} = \sum_{q \in \mathcal{Q}} \sum_{wc \in \mathcal{WC}} \Omega_{q,wc} \cdot q \tag{3.10}$$

$$WC_{flowline} = \sum_{q \in \mathcal{Q}} \sum_{wc \in \mathcal{WC}} \Omega_{q,wc} \cdot wc \tag{3.11}$$

$$Q_{w,flowline} = Q_{liq,flowline} \cdot WC_{flowline} \tag{3.12}$$

where $\mathcal{P}$ represent the breakpoints for the generated data, while $\mathcal{Q}$ and $\mathcal{WC}$ represent the breakpoints for $Q_{liq,flowline}$ and $WC_{flowline}$, and $p_{q,wc}$ is a corresponding ΔP at given values for Q_{liq} and WC in the flowline table.

Constraints to the solver:

$$P_{wh}^{j} \leq P_{in,flowline} + (1 - onOff) \cdot bigM, \; j \in wells, \;\; onOff = 1 \tag{3.13}$$

$$P_{wh}^{j} \geq P_{in,flowline} - (1 - onOff) \cdot bigM, \; j \in wells, \;\; onOff = 1 \tag{3.14}$$

$$Q_{liq,tot} = Q_{liq,flowline} \tag{3.15}$$

$$Q_{w,tot} = Q_{w,flowline} \tag{3.16}$$

$$\sum_{p \in P} \lambda^{j} = 1, \;\; \lambda^{j} \in SOS2 \tag{3.17}$$

$$\sum_{q \in \mathcal{Q}} \sum_{wc \in \mathcal{WC}} \Omega_{p,wc} = 1, \;\; \Omega_{q,wc} \in SOS2 \tag{3.18}$$

were λ is a SOS2 variable for the PWL well data (3.3), and Ω is a SOS2 variable for the two dimensional PWL table for the flowline data (3.5). Also note that in stead of setting an equality relation petween P_{wh} and P_{in}, a set of two inequalities have been introduced to model this in order to have binary variables be able to turn of wells.

3.1.5 How the SOS2 sets are ensured with OpenSolver

In order to have the SOS2 variables act as a SOS2-set, a few extra variables and constraints must be given to the OpenSolver. Each PWL-table will need the following variables (one for each variable in the SOS2-set):

BinaryHelpingVar $\in \{0, 1\}$

EnsureSOS2

$$\sum_{\lambda \in SOS2} BinaryHelpingVar_{\lambda} = 1 \tag{3.19}$$

$$EnsureSOS2_{1} = BinaryHelpingVar_{1} \tag{3.20}$$

$$EnsureSOS2_{k} = BinaryHelpingVar_{k} + BinaryHelpingVar_{k-1} \tag{3.21}$$

for a breakpoint $k \neq 1, k \in \mathcal{P}$

$$\forall p \in \mathcal{P} \;\; \lambda_{p}^{j} \leq EnsureSOS2_{p} \tag{3.22}$$

3.2 AMPL optimization - Peregrino field

While the Excel optimization is a proof of concept and a good and visual way to study and learn about optimization, the AMPL optimization of this project is aimed at using the theory by running optimization on the Peregrino field. The Peregrino field have a different setup than the two well network from the Excel optimization case. The Peregrino field has 27 producing wells, and they all produce straight to a production vessel without having the flow merge in a common flowline. This means there is no need to solve the network to match the properties of the wells with the merged fluid in the flowline. This simplifies the optimization model. AMPL was chosen to implement the optimization model for this part as it is more powerful and simple to implement compared to Excel and OpenSolver.

3.2.1 About AMPL

Learning AMPL

Previous to this project knowledge of AMPL had been acquired by attending the course TTK16 at NTNU autumn 2016. The course teaches optimization and modelling and also how to implement some simple models using AMPL. The concepts of PWL and SOS2 variables was also introduced in the course, making it a very good course for preparing for this project.

Acquiring AMPL

For this course a 30-day free trial license for students was used to implement the optimization model and run the optimization for cases. Thorough preparation and design of the optimization of model was carried out in advance to make the most out of the 30-day trial license. A full license can be granted to use for a university course, however a single student off a class will not be eligible. The 30-day trial was sufficient for this project. An alternative to getting a full license of AMPL can be to try a open source program like GLPK (GNU Linear Programming Kit).

Syntax

The syntax of AMPL is very similar to natural mathematical optimization notation, making it simple to go from the designed model to implemented code. Examples of code is attached in chapter 3.2.5 and appendix D.

3.2.2 Optimization workflow

As with the Excel optimization already discussed, Pipesim is used to model the field and work as a black box simulator, and Excel will be used to automatize Pipesim, run simulations and store the data. However now AMPL will be used as optimization engine in stead of OpenSolver in Excel. A macro in Excel automatically takes the raw data generated from Pipesim and outputs it to data files that AMPL reads in to the program structure.

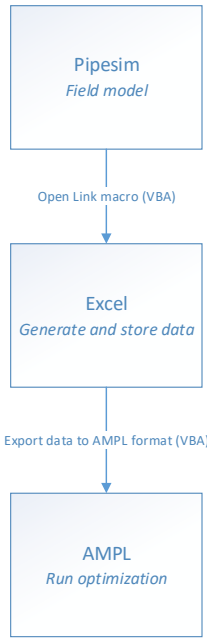


Figure 3.6: Workflow for running optimization on the Peregrino field with AMPL

3.2.3 Pipesim model

The Pipesim model is created in a different way then the two well model. In stead of creating 27 separate wells, the model only contains one well, and the properties for each of the 27 individual wells are input to the single well model through the VBA macro in Excel. The VBA macro will input the correct ESP model, PI, and a water cut for a well based on if it is a low, medium or high producer. Then it will input the different frequencies in turn to output the breakpoint data for that well. The VBA macro will use the single branch object of Pipesim to perform analysis in stead of the network object. This means the analysis is performed at tubing level rather than considering the whole network. This is in order to be able to configure the ESP, fluid model etc. from the macro. However this method also have some drawbacks which will be discussed in section 5.3.

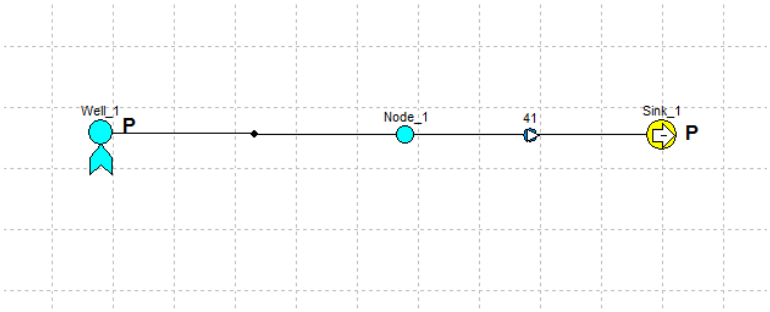


Figure 3.7: Schematic of Pipesim model used to generate data for AMPL optimization

The ESP models were chosen based on Castro and Leite (2015) and Olsen et al. (2011). The PI for each class of wells was chosen by following the rules seen in table 3.1, using the PI for the medium producer as a starting point. Originally the PI of a low producer well was to be $\frac{1}{7}$ of the medium producer well, and the PI of the high producer 4 times bigger than that of the medium producer well. The high producer wells had a tendency too produce at higher rates and the low producer wells at lower rates then the ESP local rate constraints allow. The PI of low and high producer wells were tuned to $PI_{med} * \frac{1}{6}$ and $PI_{med} * 3.8$ respectively. The PI for the medium producers was then set to 20 [STB/d/psi] after a lot of tuning. Using these parameters, field conditions similar to the Peregrino could be obtained from the Pipesim model, with rates staying within the operating range for the ESP at all time steps.

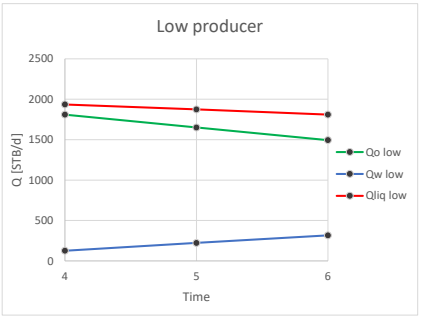
Well Class	Pump	Flow q $[STB/d]$	PI PI_{med}=20$STB/d/psi$
Low	538P100	2000 - 5000	$PI_{med} * \frac{1}{6}$
Medium	HC12500	5000 - 13500	PI_{med}
High	HC20000	13500 - 20000	$PI_{med} * 3.8$

Table 3.1: Information about pump and PI for each well class

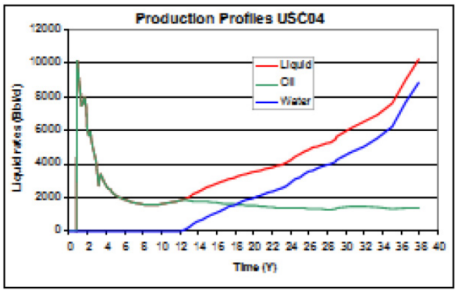
3.2.4 Generating PWL tables in Excel

The tables are generating using a macro written in VBA to load the Pipesim model, input the well information like pump, PI and WC, then generate breakpoints. As the Peregrino field pumps are to be at a setting between 40Hz and 65Hz (Castro and Leite, 2015), breakpoints were created from 39Hz to 65Hz by steps of 2Hz for each well. This gives a total of 14 breakpoints for each well. For 30 wells generating 14 breakpoints means Pipesim has to generate data for a total of 420 breakpoints per time step. Then data as generated for 3 time steps to see how the field develops with time. Generating 420 breakpoints for one time step takes 2 hours and 52 minutes. This gives an average of 24.5 seconds per

simulation run by Pipesim. Total runtime for all the data generated over 3 time steps is 8 hours and 36 minutes.
To get data close to the real Peregrino field, water cuts, production rate and PI was tuned to get typical liquid rates for the Peregrino field. Typical liquid rates are based on Castro and Leite (2015). Figure 3.8, 3.9 and 3.10 shows data generated for each class of well for this project compared with the well profiles presented by Castro and Leite (2015). Note that *this is not accurate data for the Peregrino field*, but generated to be as close as possible to the typical well profiles from in Castro and Leite (2015) for the sake of having realistic data to perform optimization. All time steps mentioned in this thesis are referring to the figures from Castro and Leite (2015). For example, ”time 4” will be time 4 in the figures shown below.

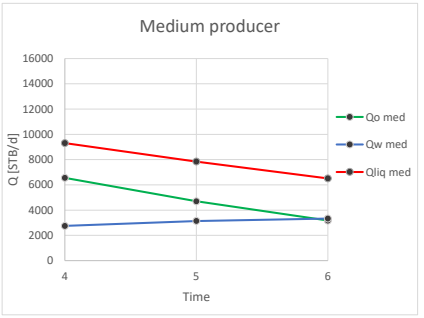


(a) Generated data

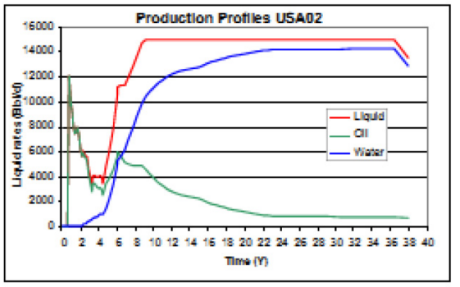


(b) Well profile Castro and Leite (2015)

Figure 3.8: Generated data for low flow-rate producer well to the left compared to Castro and Leite (2015) on the right. Orange lines represent the 3 time-steps for the generated data, timestep 4, 5 and 6

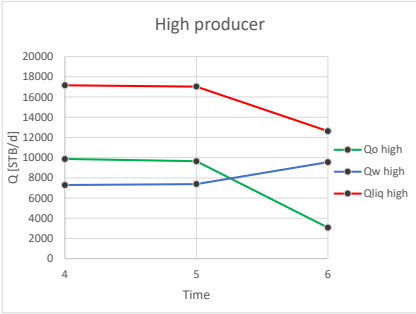


(a) Generated data

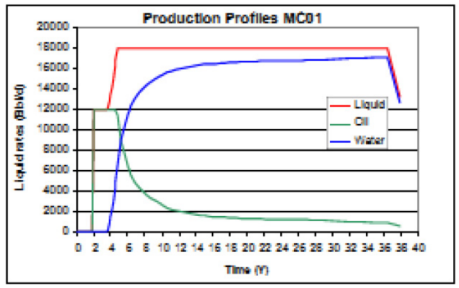


(b) Well profile Castro and Leite (2015)

Figure 3.9: Generated data for medium flow-rate producer well to the left compared to Castro and Leite (2015) on the right. Orange lines represent the 3 time-steps for the generated data, timestep 4, 5 and 6



(a) Generated data



(b) Well profile Castro and Leite (2015)

Figure 3.10: Generated data for high flow-rate producer well to the left compared to Castro and Leite (2015) on the right. Orange lines represent the 3 time-steps for the generated data, timestep 4, 5 and 6

3.2.5 AMPL program

This section gives an overview of the most important functions, modules, layout and program flow of the AMPL program.

Data

The data for each breakpoint is written to the Excel sheet. A VBA macro then writes that data to data files for AMPL. There is one file that contains fixed data for the wells, that is PI, WC, GOR, and minimum and maximum flow through that wells ESP. This file is called ”well_info.tab”. In addition each well has a ”.tab” file for the data at each breakpoint of that well. The file ”read_table.run” will read these data files and input the data into the data structures in AMPL and ready to be used by the program.

Model

The optimization model itself is written into two ”.mod” files. ”Declaration.mod” will simply declare all variables, parameters and sets. ”Optimization.mod” is where the optimization model is written. The following short piece of code shows a part of the ”Optimization.mod” file to show how the file is organized. The code also shows the syntax of AMPL, which is very similar to natural mathematical optimization notation as mentioned earlier. The file ”Optimization.mod” is attached as appendix D.1.

```
# OBJECTIVE :
maximize oilProduction: var_qo_tot;

# CONSTRAINTS:
subject to definition_qliq{i in WELLS}:
  var_qliq[i] = sum{F in SFesp[i]} Qliq[i, F] * var_Lambda[i, F];

subject to definition_qo_tot:
  var_qo_tot = sum {i in WELLS} var_qliq[i] * (1-WC[i]);

subject to Max_Water:
  var_qw_tot <= qw_max;
```

Peregrino.run

The file ”Peregrino.run” can be seen as the main function of this program. This is where all data files and model files are loaded in, other run-files are run from, and were the *solve* command is executed. Also which solver to use can be set here, and configuration of the solver can also be done here. The program flow starts with loading the model files and reading the data files. Then the ”SOS2.run” file is run, which sets up the SOS2 variables based on the well data which is now in the program. It then solves the optimization problem. ”output.run” is run to write the results in a textfile called ”out.txt”. This file is attached as appendix D.2.

Figure 3.11 illustrates the program flow and relation between the different modules of the program.

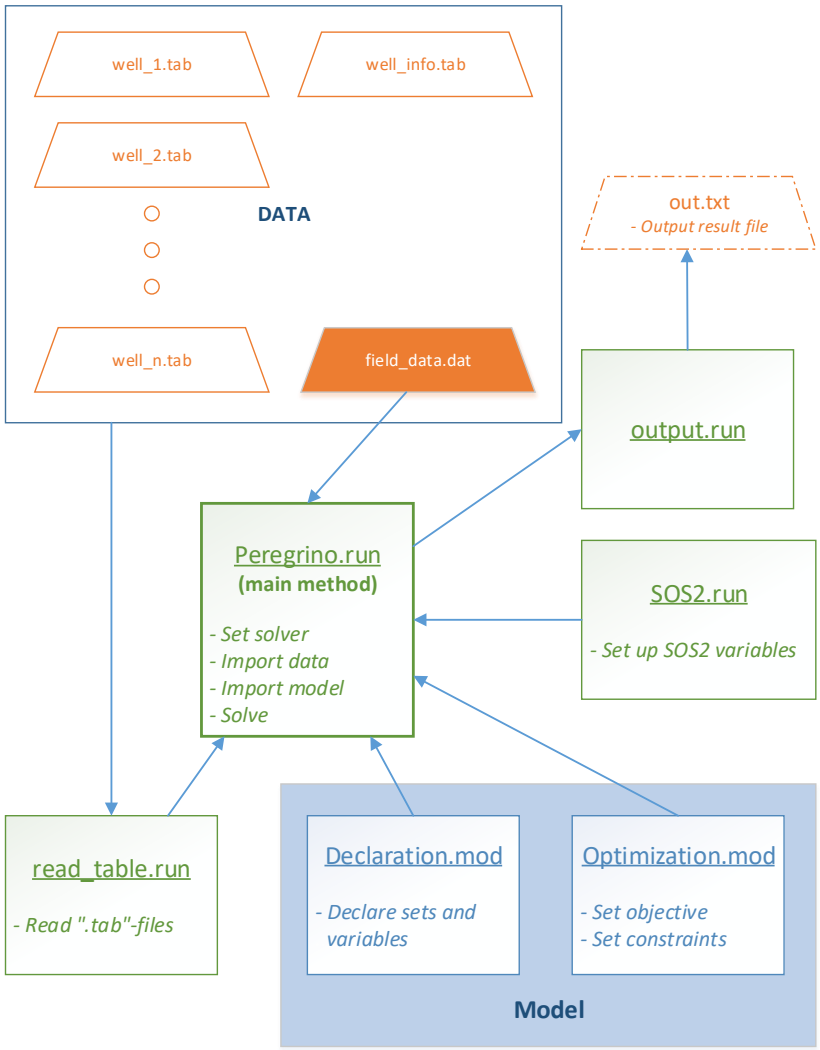


Figure 3.11: Workflow for AMPL program

Chapter 4

Results

4.1 2 well network optimization, Excel

Simplifications were done to the model in order to lower the complexity. The most drastic simplification is fixing the f_{ESP} to make the PWL table with breakpoints generated from Pipesim one dimensional. This means the optimization will only search for P_{wh} to match the P_{in} of the flowline and solve the network. Another simplification is fixed WC in both wells. Table 4.1 shows the result obtained from running the solver.

Excel	**results**			
		well 1	well 2	Flowline
P_{wh}	[psi]	55.81	55.81	55.81 (P_{in})
f_{ESP}	[Hz]	66	66	-
q_{liq}	[STB/d]	2537.68	3230.49	5768.18
q_w	[STB/d]	1551.74	1642.99	2884.089
q_o	[STB/d]	985.94	1587.50	2573.44
WC	[%]	50	50	50
ΔP	[Psi]	-	-	-14.19
P_{sep}	[Psi]	-	-	70

Table 4.1: Excel result

The OpenSolver used 1.55 seconds to obtain this result using a linear solver called *COIN-OR CBC*.

4.2 Peregrino (no network), AMPL

In this chapter the result of the Peregrino optimization is presented. Each case presents the total liquid for each time step as calculated by the AMPL program in a table. Then the frequency of all wells are presented in a radar plot to show which wells the AMPL program choose to produce at high or low frequency, or turn off completely. In some cases, the plot of previous cases are added to the plot for comparison.

4.2.1 Cases

The optimization was tested on 4 cases designed to see how the program would choose to distribute the production over all the wells in different situation. Case 1 is a case to see how the field will operate without a constraint on water produced. Case 2 is the base case and represents normal operating conditions for the Peregrino field. Water production is constrained to 120000 STB/d. Case 3 represents maximum water treatment capacity cut by 58.34% to 50000 STB/d. Case 4 studies how the program compensates for turning off wells with good properties. Finally case 5 shows how the program choose operating points when switching the objective from maximizing oil production to maximizing pump efficiency. Except mentioned variation in constraints, all other conditions are the same for all cases. Table 4.2 shows an overview of the cases.

Case	Description
1	Unconstrained water production
2	Base case, $q_w \leq 120000 STB/d$
3	Reduced water capasity, $q_w \leq 50000 STB/d$
4	Well intervention, 4 wells turned off, $q_w \leq 50000 STB/d$
5	Change objective. Maximize pump efficiency. Constraints as base case

Table 4.2: Case matrix

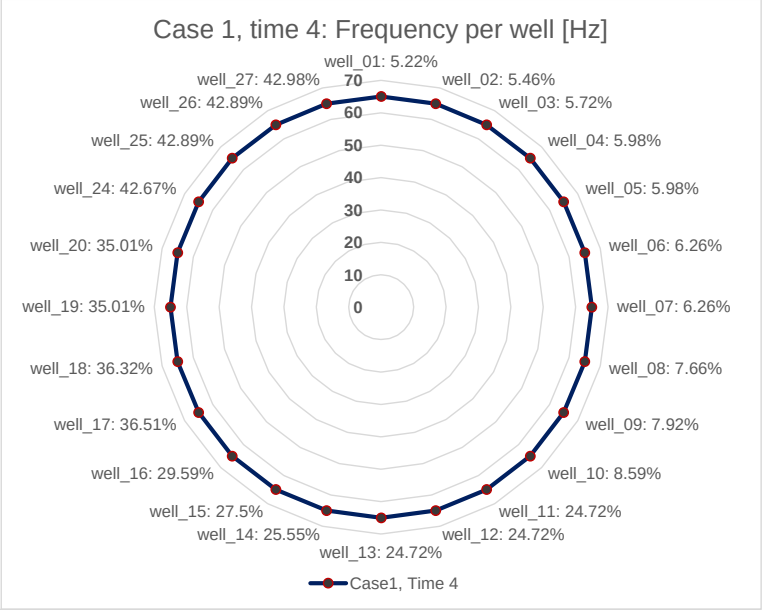
The Peregrino field is currently producing at around 100 000 STB/d. For every optimization case (except case 1), if the production exceeds this value, q_o =100 000 STB/d of oil production becomes a constraint and the objective switches to minimizing water production. Then, as oil production decreases and water production increases with time, the proposed constraints of the case will activate and maximizing oil production will be the objective.

4.2.2 Case 1, Unconstrained water production

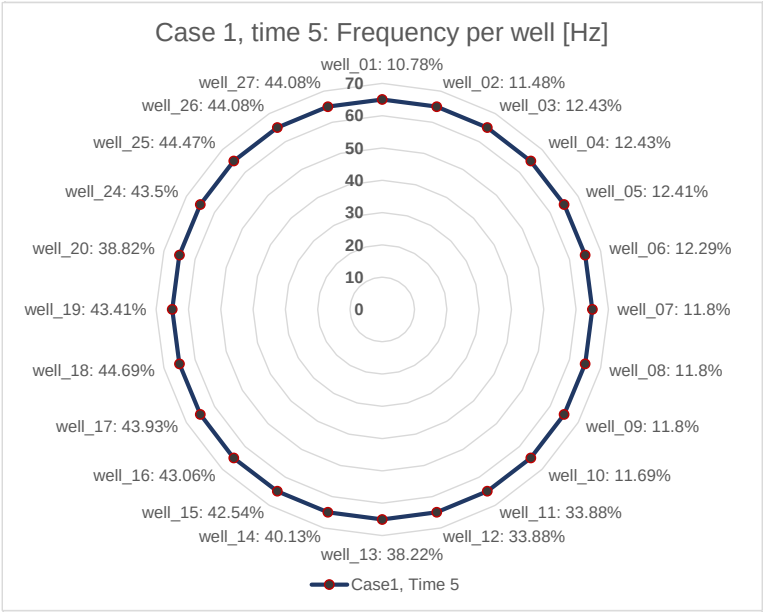
This case demonstrates how the program will maximize the production without a constraint on water or oil production. This is useful for comparison with the base case. As there are no constrains on how much water or oil can be produced, the pumps will all run at the maximum frequency 65Hz. There is a drop in oil rate with time as the water cuts drop, however the total liquid rate also experience a slight drop with time. The liquid rate drop is 17.89% from time 4 to 6, while oil rate drops by 49.93%, showing that the water cut increasing with time is a significant factor and that more water will be produced with time.

Case 1			
All q in [STB/d]	q_o	q_w	q_{liq}
Time 4	188219	96939	285158
Time 5	166588	107550	274139
Time 6	94227	139916	234143
$\Delta q_{t=4\ to\ 6}$	-93992 (-49.93%)	+42977 (+44.33%)	-51015 (-17.89%)

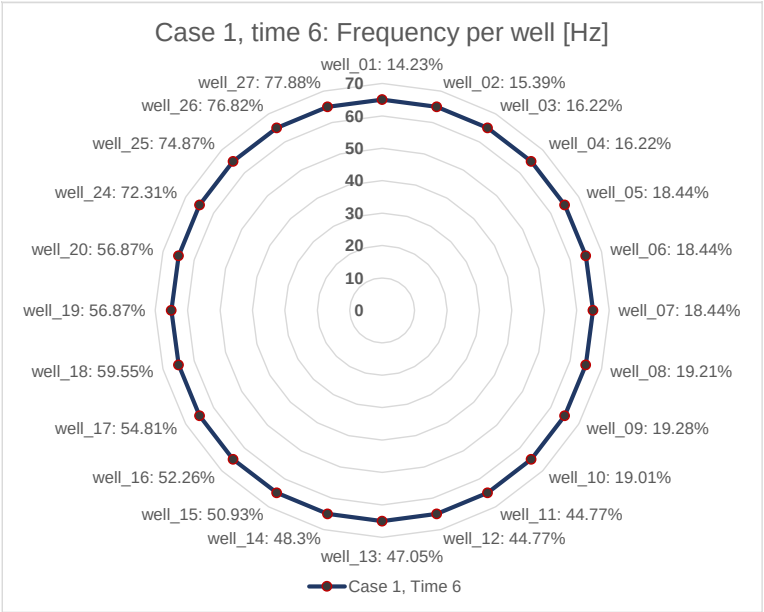
Table 4.3: Case 1: Result from running optimization on the 3 time step.



(a) Case 1: Time 4



(b) Case 1: Time 5



(c) Case 1: Time 6

Figure 4.1: Case 1: Frequency of each well over three time steps. The percentage after each well name represents the water cut of that well.

4.2.3 Case 2, Base case

This is the base case, representing normal operating conditions. Water treatment capacity is at $120000 STB/d$. Table 4.4 shows the resulting field rates over all time steps as well as a comparison with case 1.

Case 2			
All q in [STB/d]	q_o	q_w	q_{liq}
Case 1: Time 4	188219	96939	285158
Case 2: Time 4	100000	32959.6	132960
Δq	*-88219 (-46.87%)*	*-63980 (-66.00%)*	*-152198 (-53.37%)*
Case 1: Time 5	166588	107550	274139
Case 2: Time 5	100000	55196.8	155197
Δq	*-66588 (-39.97%)*	*-52353.2 (-48.68%)*	*-118942 (-43.39%)*
Case 1: Time 6	94227	139916	234143
Case 2: Time 6	88475	120000	208475
Δq	*-5752 (-6.10%)*	*-19916 (-14.23%)*	*-25668 (-10.96%)*
Case 2: $\Delta q_{t=4-6}$	-11525 (-11.52%)	+87041 (+264.08%)	+75515 (-56.80%)

Table 4.4: Case 2: Comparing results from case 1 and case 2 and the difference in liquid rates.

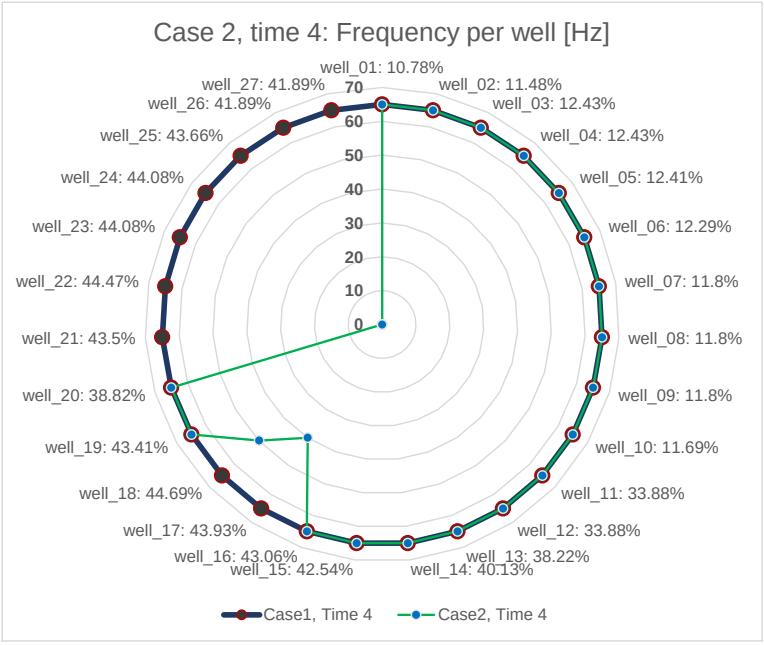
The result show that by doing optimization and choosing operating points for each pump using the proposed model, will choose operating points so that wells producing little water are chosen to produce more and wells producing a high water rate are turned down. This can be seen as for instance at time 6, a 14.23% reduction in water production compared to case 1 leads to only 6.10% reduction in the oil rate. The exact state of each well in case 2 are presented in figure 4.2.

A nonoptimal strategy for comparison

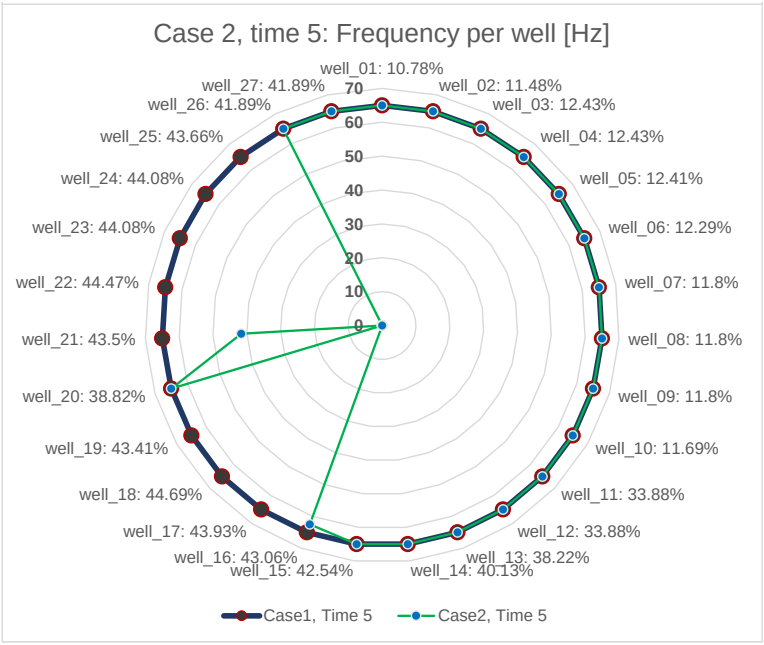
As an alternative to doing optimization, a more straight forward strategy may be to simply keep all wells running but at a lower pump frequency to lower the rates to keep constraints. Table 4.5 shows resulting liquid rates from putting all pumps at 55Hz, 10Hz below the maximum setting. This strategy could be implemented in a more intelligent way, as for time steps 4 and 5 there is more potential as far as water production goes, and the field produce over the current plateau level of the Peregrino. However by time 6 it is clear that oil production is worse then compared to case 2 while simultaneously breaking the water production constraint.

All f_{ESP} @ 55Hz			
All Q in [STB/d]	Q_o	Q_w	Q_{liq}
Time 4	182861	101427	284288
Time 5	160776	107154	267930
Time 6	81113.3	131869	212983

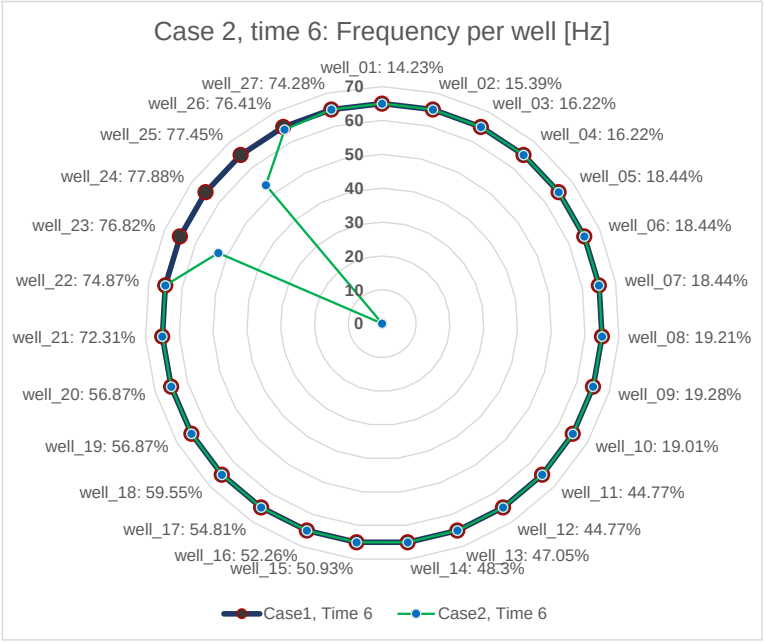
Table 4.5: Nonoptimal strategy for choosing operating points for comparison with case 2



(a) Case 2: Time 4



(b) Case 2: Time 5



(c) Case 2: Time 6

Figure 4.2: Case 2: Frequency of each well over three time steps. The percentage after each well name represents the water cut of that well.

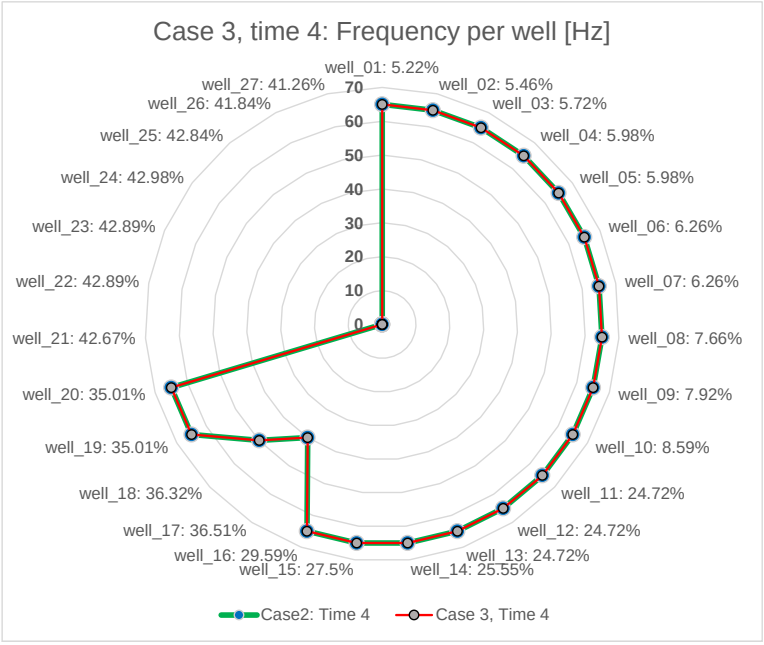
4.2.4 Case 3, Reduced water capacity

This case represent a sudden drop in water treatment capacity. There can be many reasons for a decrease in the ability to process or dispose of the water produced from an oil well. For instance if a injector well is plugged, a temporary restriction on produced water could be reasonable. As for case 3, all other conditions are the same as in case 2, but no more then $50000 STB/d$ of water can be produced. This is 41.5% of the capacity in case 2; normal operating conditions.

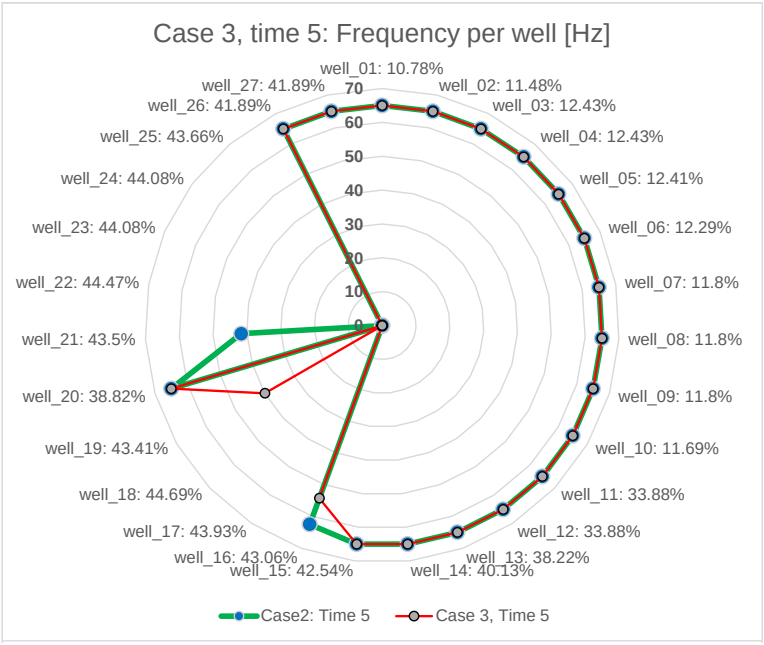
Case 3			
All q in [STB/d]	q_o	q_w	q_{liq}
Case 2: Time 4	100000	32959.6	132960
Case 3: Time 4	100000	32959.6	132960
Δq	*0 (0%)*	*0 (0%)*	*0 (0%)*
Case 2: Time 5	100000	55196.8	155197
Case 3: Time 5	93231.9	50000	143232
Δq	*-6768.1 (-6.77%)*	*-5196.8 (-9.42%)*	*-11965 (-7.70%)*
Case 2: Time 6	88475	120000	208475
Case 3: Time 6	64270.1	50000	114270
Δq	*-24204.9 (-27.35%)*	*-70000 (-58.34%)*	*-94205 (-45.19%)*
Case 3: Δq $_{t=4-6}$	-35729.9 (-35.73%)	+17040.4 (+51.70%)	-18690 (-14.06%)

Table 4.6: Case 3: Comparing results from case 2 and case 3 and the difference in liquid rates.

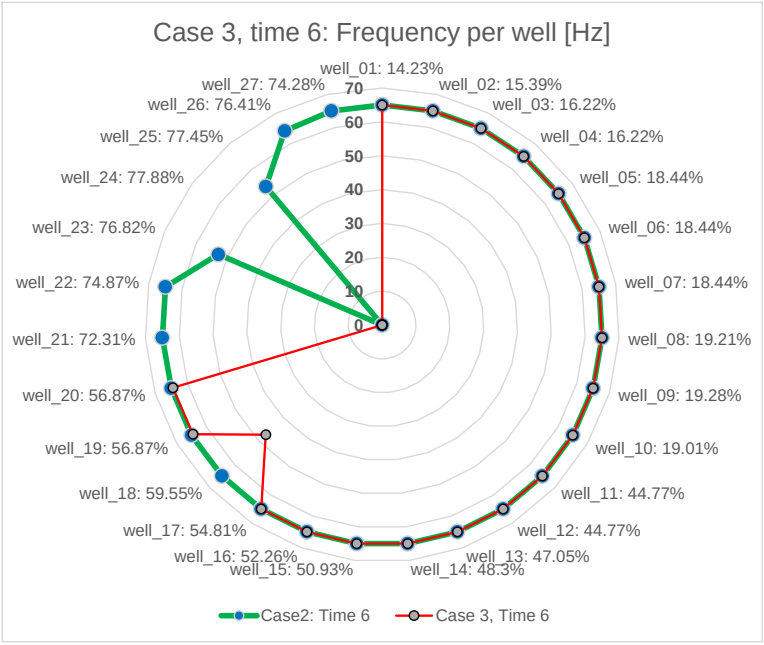
For time step 4 under normal operating conditions (case 2) the water production is low enough that the reduced water treatment capacity does not change the state of the any wells when reducing the constraint value for case 3. From time step 5 and 6 the optimization has to take the water production constraint into account. In every run, the AMPL program manages to reduce the oil production significantly less compared to the reduction in water production. for time 6, water is down 58.34%, while oil is down 27.35% compared to the base case. Figure 4.3 shows the frequency each well is set at for case 3 compared with the base case.



(a) Case 3: Time 4



(b) Case 3: Time 5



(c) Case 3: Time 6

Figure 4.3: Case 3: Comparing the operating points of each well for case 2 and case 3. The percentage after each well name represents the water cut of that well.

4.2.5 Case 4, Well intervention

This case is designed to see how the optimization program handles a situation where multiple wells are turned off for various reasons. In this case, 4 wells with low water cuts are turned off. Two wells are low producers and two wells are medium producer. The starting point is case 3 time 6 where the field can already only produce 50000 STB/d. This case is chosen because it had most high producing wells turned very low or off, and so the comparison between this and case 4 when four wells producing at maximum capacity are turned off is interesting. Running the optimization for only one time step is sufficient to see the result of intervening wells. It is clear that the optimization program choose to increase the best (low WC) of the wells already turned low / off to compensate for the loss of 4 good wells. In figure 4.4 case 2 and case 3 are plotted in the background for comparison with case 4.

Case 4			
All q in [STB/d]	q_o	q_w	q_{liq}
Case 3: Time 6	64270.1	50000	114270
Case 4: Time 6	53214.1	50000	103214
Δq	*-11056 (-17.20%)*	*0 (0.00%)*	*-11056 (-9.66%)*

Table 4.7: Case 4: Comparing results from case 3 and case 4 at time 6

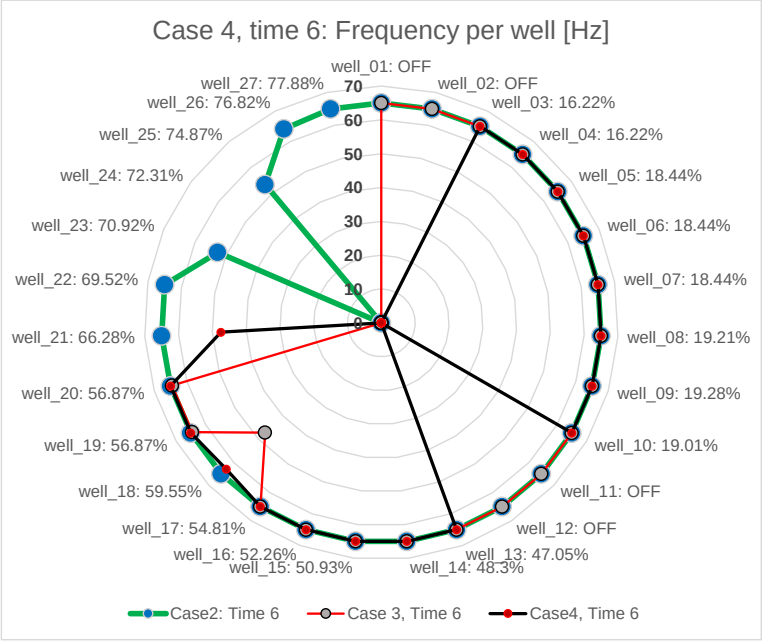


Figure 4.4: Case 4: Comparing the operating points of each well for case 3 and case 4, time step 6. The percentage after each well name represents the water cut of that well.

4.2.6 Case 5, maximize ESP efficiency

This case is designed to see how the optimization program choose operating points for the wells given an objective unrelated to liquid rates. In stead the objective is set to pump efficiency. To make sure each pump is at its maximum efficiency, the "total efficiency" will be defined as following

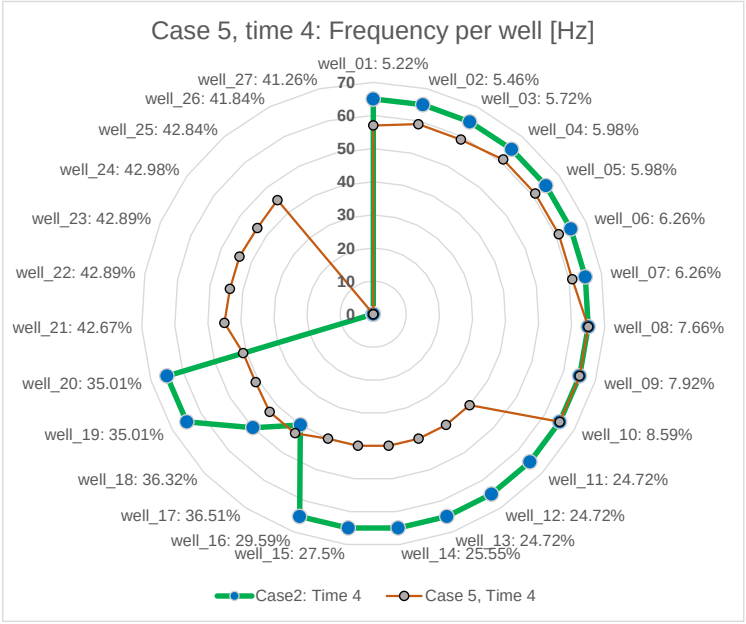
$$maximize \;\; ESP^{tot}_{efficiency} = \sum_{i \in Wells} pump\_efficiency^i \tag{4.1}$$

where $pump\_efficiency^i$ is defined from the same PWL table and in the same manner as P_{suc} etc. (see chapter 2.4.1 and equation 2.32). Constraints on water constrained is as case 2 and a constraint on $q_o \leq 100000 STB/d$ is added. Table 4.8 shows the resulting field rates from case 5 compared to case 2.

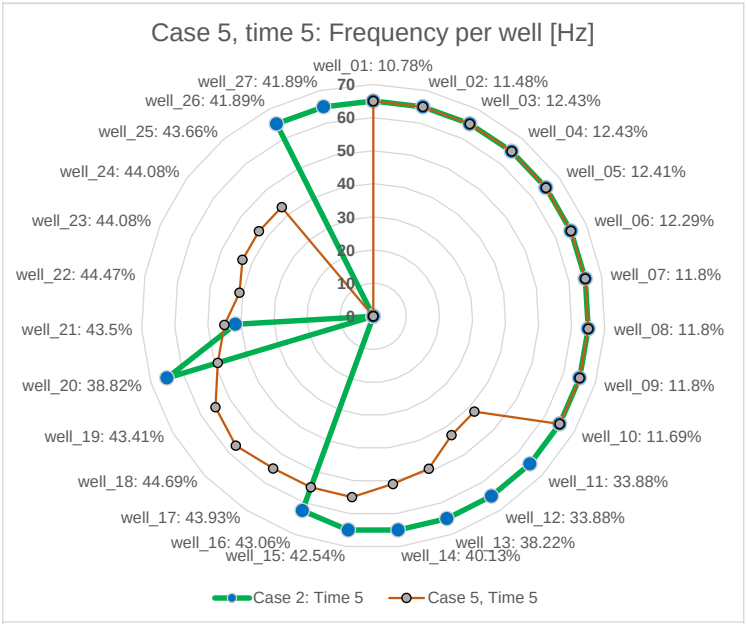
Case 5			
All q in [STB/d]	q_o	q_w	q_{liq}
Case 2: Time 4	100000	32959.6	132960
Case 5: Time 4	100000	47780.7	147781
Δq	*0 (0.00%)*	*+14821.1 (+14.66%)*	*+14821 (+11.15%)*
Case 2: Time 5	100000	55196.8	155197
Case 5: Time 5	100000	60348.2	160348
Δq	*0 (0.00%)*	*+5151.4 (+9.33%)*	*+5151 (+3.32%)*
Case 2: Time 6	88475	120000	208475
Case 5: Time 6	85296.2	112462	197758
Δq	*-3178.8 (-3.59%)*	*-7538 (-6.22%)*	*-10717 (-5.14%)*
Case 5: Δq $_{t=4-6}$	-14703.8(-14.7%)	+64681.3 (+135.37%)	+49977 (+33.82%)

Table 4.8: Case 5: Comparing results from case 2 and case 5 and the difference in liquid rates.

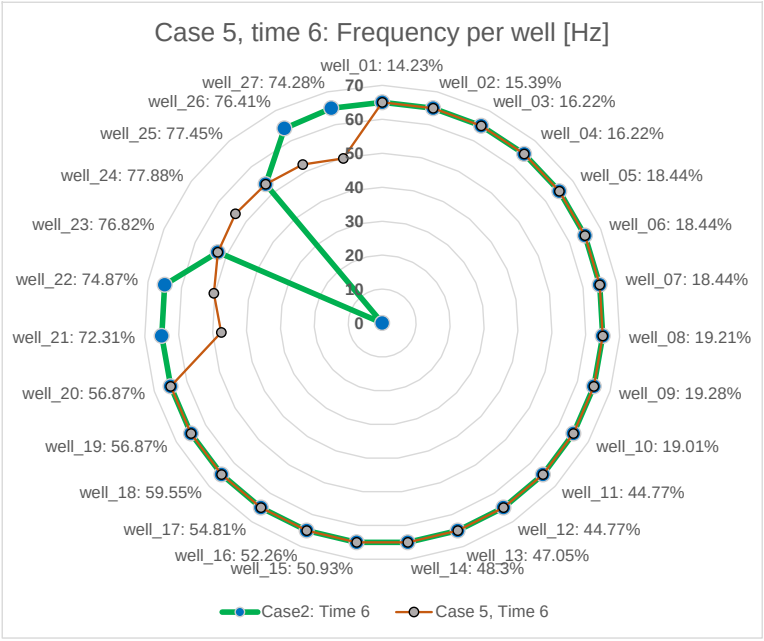
The results regarding field rates are similar to case 2, barely differing by a little more than 10% for each time step. As far as producing a lot of oil and little water is an ideal, the strategy of case 5 is always a little behind in optimality compared to case 2. Although the resulting field rates are similar, optimizing pump efficiency leads to very different points for each well. See figure 4.5 for the operating points of each well in case 5 compared to case 2.



(a) Case 5: Time 4



(b) Case 5: Time 5



(c) Case 5: Time 6

Figure 4.5: Case 5: A comparison between the operating points of all wells for case 2 and case 5. The percentage after each well name represents the water cut of that well.

4.2.7 Runtimes

The program runs very efficiently. For all cases the program runs in less than a second, usually in the range off 0.010 to 0.100 seconds. As runtimes have not been an issue at the level of complexity of the current optimization model, little time was spent measuring the runtime. In table 4.9 an overview of 3 different solvers that were tried out is presented with the runtime for running the proposed optimization model in AMPL.

Solver	n Breakpoints per well	Solvetime
CPLEX	14	0.047
BARON	14	0.593
XPRESS	14	0.375

Table 4.9: Time per solver.

As BARON is a solver for non-linear problems, the strategies it uses to solve the optimization model seems to take slightly more time than CPLEX. However all run times are less then one second. For this project, only CPLEX was used for the actual case study.

Chapter 5

Discussion

5.1 Excel optimization

The Excel optimization is a direct continuation of the preliminary project done one semester previous to the work on this thesis. The goal was to be a proof of concept and show that optimization can be performed with simple tools using Excel and the OpenSolver add-on. The OpenSolver is easy to install and provides a intuitive way to design and setup the optimization case. However debugging is difficult. As the complexity grows it also gets increasingly difficult to keep track and do edits to the model, and the interface for inputting variable cells and constraints can be a bit tricky sometimes, not saving all the changes or even crashing when saving changes to a model. Save often. The easiest way by far to set up OpenSolver is by using a VBA macro as was done for this project. OpenSolver can then be programmed to setup the model based on the PWL tables created. As for the optimization itself, a lot of simplifications were made to the model as OpenSolver reported a lot of infeasibility. Only the linear COIN-OR CBC was used. The simplifications included fixing the water cut to be the same for both wells and only choosing one frequency. This means the optimization was not finding a optimal operating point for the ESP, but rather solving the network searching for a wellhead pressure to match the pressure at the flowline using a fixed operating point. At one point binary variables to turn the wells on and off were removed from the model, but these were reintroduced and the model kept working. After all the simplifications, a runtime of 1.55 seconds to solve the network was achieved. Compared to the runtimes of AMPL for running the Peregrino cases, this is not impressive (difference of complexity of the 2 optimization models not taken into account). To summarize, Excel and OpenSolver can be a great combination to learn about optimization. Simple cases can be easy to design and implement. However as complexity grows, AMPL (or software at the same level) may be a better option.

5.2 AMPL optimization

Compared to OpenSolver, AMPL requires more programming skills and general skills with a computer, like using a terminal. If a user is somewhat comfortable with these concepts then AMPL would be recommended to learn. The course TTK16 at NTNU was useful as a preparation for this course as it teaches both design and modelling of optimization models, and also implementation with AMPL. The syntax is very different from other programming languages as it is designed to mimic optimization notation. It takes some time to get used to, but is very powerful for implementing and solving optimization problems. All cases run for this project was run by AMPL in less than a second as mentioned in chapter 4.2.7. The optimization using AMPL was a success, and all cases could be run without problem and providing solid results both at a field level and well level.

5.2.1 AMPL results

Case 1 is the unconstrained case. As expected the program set all the pumps to produce at maximum value when there no constraints on oil or water production. This happened at every time step. It is also clear from the data that total liquid rate drops with time, more on this in section 5.3. However the effect of increasing water cuts was more significant, leading to a decrease on oil rate of 49.93% as opposed to a 17.89% decrease in total liquid rate.

Case 2 represents normal operating conditions and water production was capped at 120 000 STB/d. For early time steps (4 and 5) the liquid rate is so high that more than the 100 000 STB/d of oil that Peregrino is producing is obtained. Due to this the AMPL program was set to minimize water production for these time steps and constrain the oil rate to 100 000 STB/d. Due to this the program turns off or sets at a low frequency many of the high flow-rate producers. At time step 6 oil rate is below 100 000 STB/d and water rate exceeds 120 000 STB/d, and the proposed model of optimizing oil production given the water production constraint was applied. Now only one well is turned off. This is the high producer with the highest water cut. This well is simply producing too high water rate, and the program turns it off to let wells with lower oil rate that also have lower water rate produce in stead. 2 other high producers are operating at reduced frequency.

Case 3 shows reduced water treatment capacity due to a plugged injector wells. The reduction is drastic in this case to demonstrate the ability of the optimization model. For time step 4 the produced water is so low for the base case, that case 3 doesn't have any change from the base case. At time step 5 it is seen that water production have to be capped by 9.42%. The program choose operating points for the ESP that result in only a 6.77% of the oil rate. For time step 6 the water cuts are so high that the constraint reduces water production by 58.34%. The oil rate is reduced by 27.73%.

Case 4 is based on case 3 time step 6. 4 producing wells, 2 low producers and 2 medium producers are closed down and the program can compensate by filling up the water production constraint by turning up or turning on wells which until now have been restricted. The program performs as expected and compensates by turning up well 18, a medium producer with WC = 59.55%, from 47Hz to 63Hz and turning on well 21, a high producer with WC = 66.28%, to 47Hz. Among the restricted wells from case 3 time 6, these 2 wells were the wells with lowest water cuts.

Case 5 demonstrates how the program choose different operating points when optimizing for a different objective. The results show that while the resulting field rate are similar although slightly less efficient as far as low water rate and high oil rate is concerned, the chosen operating points are chosen differently than for the base case. Less wells are turned off, and more wells are turned at a lower ESP frequency setting. At time step 6, no wells are turned off, instead high producers are turned low to meet water production constraint. This differs from case 2 where 1 high producer is turned off entirely.

5.3 Regarding Pipesim and OpenLink

When creating a macro in Excel using OpenLink for Pipesim in VBA, it is important to obtain the *PIPESIM Open Link Reference Manual* (Schlumberger, 2014). This manual gives a good overview of the API for OpenLink as well as code examples. Figuring out how to run simulations and configure objects and extract data from objects is not intuitive. In appendix C.1 key commands to create breakpoints used for this project is shown. Pipesim can be run with relatively high stability. As the breakpoint generation had to be run using VPN when not connected to the universities network, instability due to license problems could occur. As generating all the breakpoints takes many hours, an error underway is not wanted. The most common error was no license found. This stopped the program for that one breakpoint. A dialog box appears, and by pressing ”ok” the program continues as normal. The breakpoint in question will not update, and have same values as the previous one. For this project no such errors in the data exist. A second error that can occur is conflicting read / write attributes for all the Pipesim files generated by running the Pipesim simulation. If the Pipesim file is located in a folder synced by a cloud service like Dropbox or OneDrive, temporarily turning these off while running Pipesim is recommended to avoid such errors. The errors are very sporadic, and when running Pipesim for hours, an error could occur at any time, delaying progress.

5.4 Further work

As for optimizing Peregrino like fields, more complexity can be added to the optimization model. Extra constraints can consider gas rates and power consumption of the pumps and various other factors. The Peregrino does also have injection of demulsifier to change the viscosity of the oil. This is also not studied in this work. Regarding power consumption, the data is already extracted to the breakpoints and a variable in the AMPL program W_{ESP} already contains the values for the consumed power by the pump at every breakpoint. The unit is for the power consumption is [HP].

More work can be done on generating data from Pipesim. By using the single branch analysis strategy, liquid rates would drop by time due to increasing water cuts. This happens because when analysis is performed considering the tubing at constant wellhead pressure, increased WC will increase the density and viscosity of the fluid, increasing pressure drop in the system and thus reduced liquid rates with time. This is the reason high producer wells could not have lower WC then the ones chosen. Below 40% WC would have the high producers produce over the local ESP flow rate constraints. Lowering the PI (for all classes according to table 3.1) would have the low producers produce too low rate for the local ESP flow rate constraints. A more correct model would have a constant q_{liq} for all times, then only q_w and q_o changes while maintaining

$$q_{liq} = q_o + q_w \tag{5.1}$$

A suggestion can be to try to create the breakpoints using the OpenLink network object in stead of the single branch object. A way to configure the tubing while considering to run the network must be devised. Another strategy could be to design the Pipesim model with all 27 wells in one network file and then run analysis on the network while configuring the tubing and ESP manually in Pipesim.

Chapter 6

Conclusion

Optimizing the Peregrino field using the proposed model was successful. All cases run without problem in AMPL, and the model was able to handle all constraints with ease, even with binary variables for turning on and off the wells. Optimal ESP frequency settings were found for all cases in very short runtimes. The input to the AMPL program are PWL tables for each well containing breakpoints with q_{liq} for a range of f_{ESP} and relevant properties at each breakpoint. The breakpoints were created using Excel, running Pipesim simulations via a VBA macro and the Pipesim OpenLink API. The proposed methodology seems solid and more complexity can be added to account for more field constraints for Peregrino-like fields.

The Excel optimization model was simplified a lot before it was able to run. It was able to solve a network of two wells, that is find the P_{wh} for each well to match the inlet pressure of the flowline given by $P_{sep} - \Delta P_{flowline}$. However, adding another dimension in the PWL table by introducing variable ESP frequency to find optimal frequency given the found P_{wh} was not possible with the current setup. OpenSolver is a easy and intuitive way to learn about optimization and setup simple cases, however AMPL is preferred when the complexity of the optimization model increases.

Bibliography

AMPL Optimization, Inc., 2017. AMPL. www.ampl.com.

Castro, V., Leite, D., 2015. Esp application on heavy oil in peregrino field. In: SPE Artificial Lift Conference. Salvador, Brazil.

Frontline Systems, 2017. OpenSolver. http://opensolver.org/.

Hoffmann, A., Stanko, M. E., 2016. Real-time production optimization of a production network with espboosted wells: A case study. In: SPE Middle East Artificial Lift Conference and Exhibition. Manama, Kingdom of Bahrain.

Hollund, B., 2010. Artificial lift electrical submerged pump, best practice and future demands. Master's thesis, University of Stavanger (UiS), Norway.

Microsoft Office, 2016. Excel. www.microsoftstore.com/Excel/2016.

Olsen, H., Sheth, K., Pessoa, R. F., Okita, R., Crossley, A., Martinez, I., 2011. Esp assisted production allocation in peregrino field. In: Offshore Technology Conference. Rio de Janeiro, Brazil.

Ranjeva, J.-M., Dahl, E., Martinho, F., Goncalves, B., Ramos, A., 2014. Multilateral level-5 dual long horizontal openhole gravel pack completion on the peregrino field. In: SPE Annual Technical Conference and Exhibition. Amsterdam, The Netherlands.

Røyset, O.-m. L., 2013. Simulating and optimizing production from a multi-well field. Master's thesis, Norwegian University of Science and Technology (NTNU), Norway.

Schlumberger, 2012. Pipesim. https://www.software.slb.com/products/pipesim.

Schlumberger, 2014. PIPESIM 2012.2 Open Link Reference Manual.

Appendices

Appendix A

Well Water Cut Overview

Figure A.1 plots the WC of all wells. Each dot represent a well, and each set of connected wells represent a class of wells for different time steps. Black dot means well at time step 4, grey dot means time step 5, and white dot is time step 6. The color indicate well class. Blue is low producer, green is medium producer and red is high producer. Water cut is along the y-axis. The x-axis has no function other than being the variable for the normal distribution for each class. A random x was input to an appropriate normal distribution for each class to output a WC.

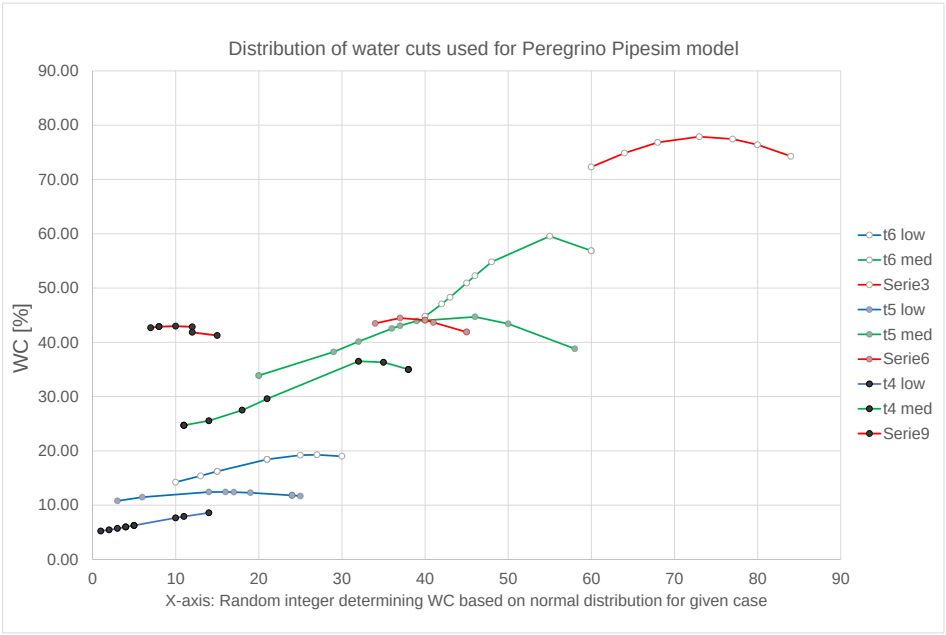


Figure A.1: Plot showing the water cuts of all wells

Appendix B

Pipesim model properties

This appendix shows some settings and properties used to configure the Pipesim model used for part 2 (AMPL optimization) of this project in figure B.1, B.2, B.3 and B.4.

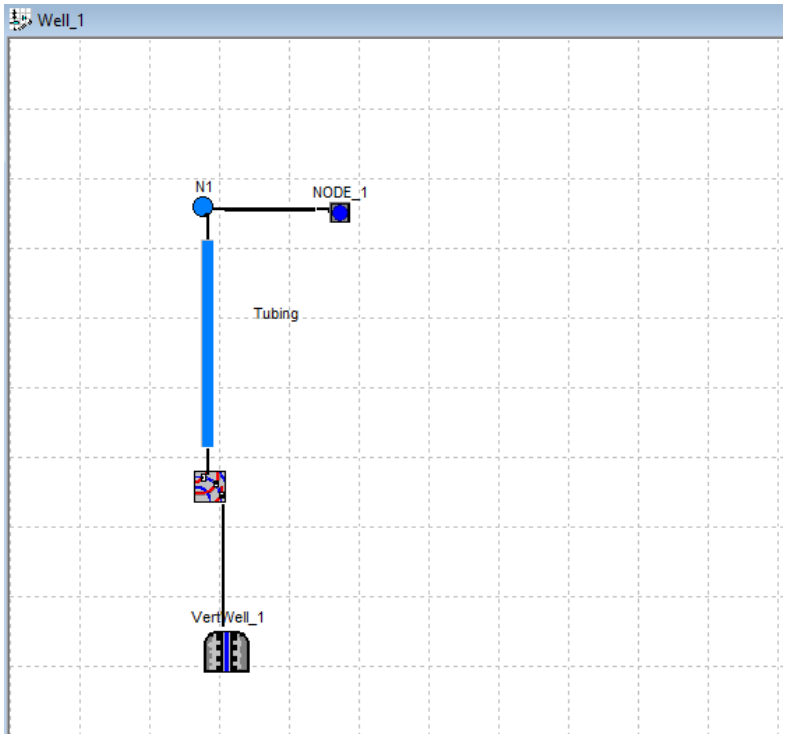


Figure B.1: Well and tubing layout of the Pipsim model

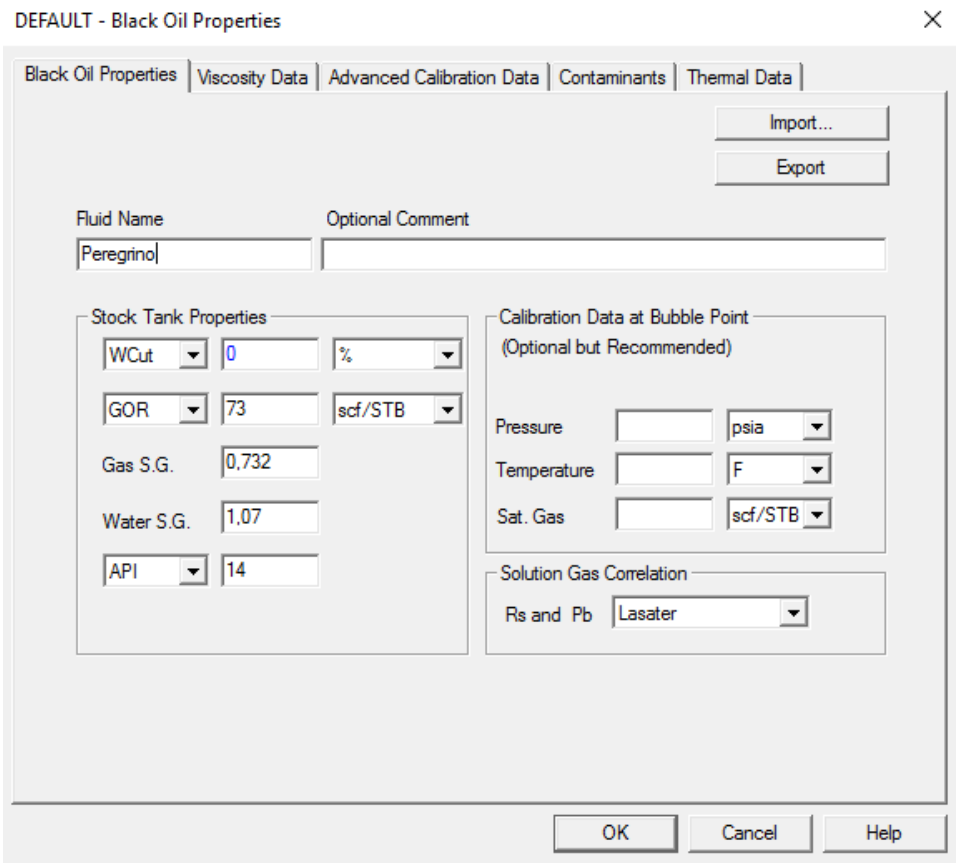


Figure B.2: Black Oil properties of the Pipesim model

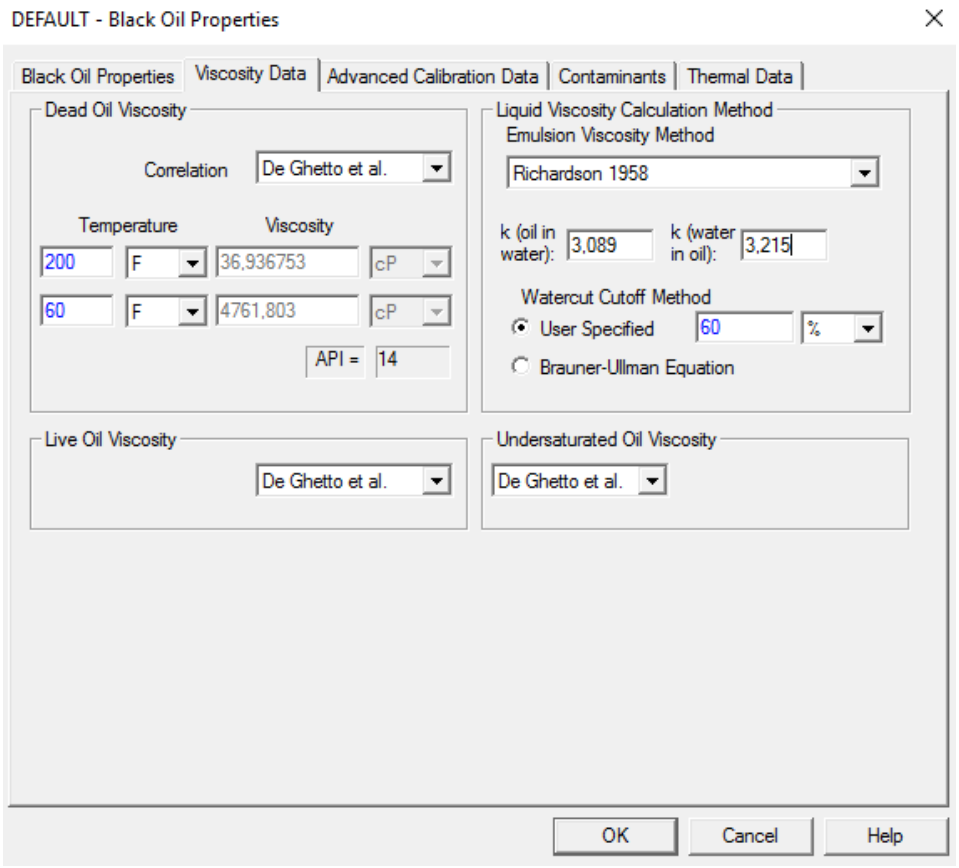


Figure B.3: Viscosity properties of the Pipesim model

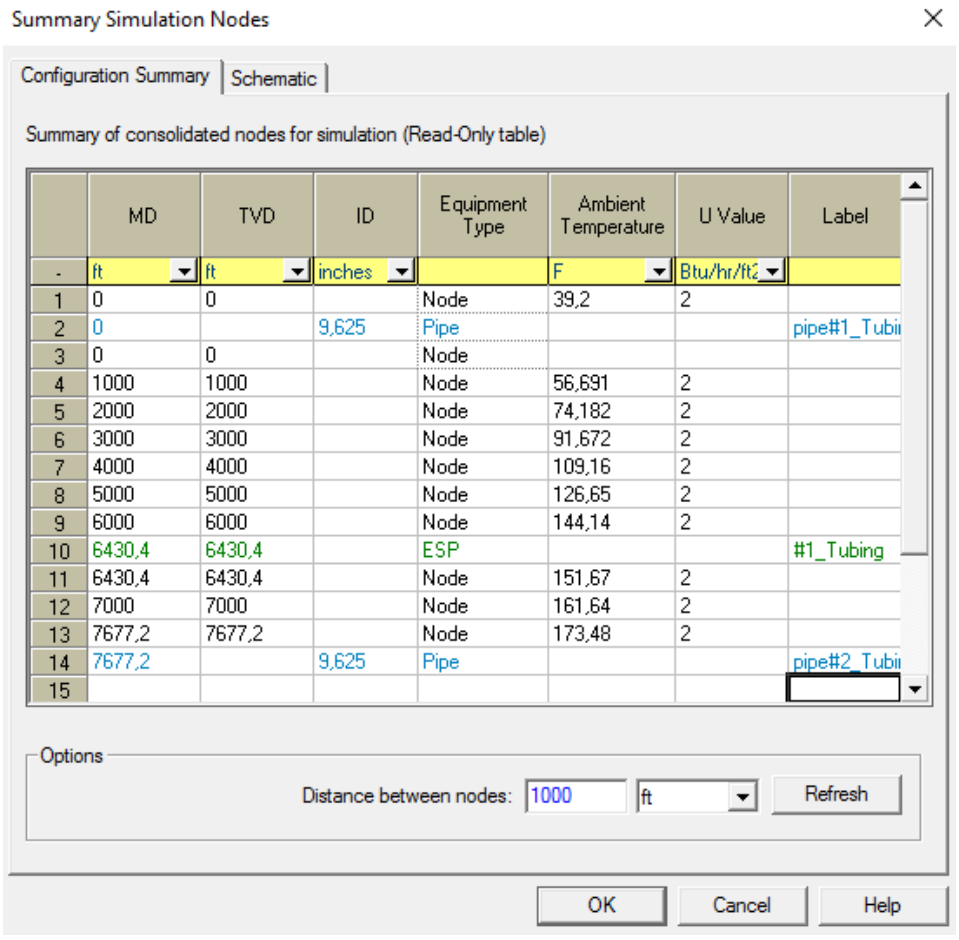


Figure B.4: Summary table for the tubing

Appendix C

Excel VBA macro

This Appendix includes core functionality to manipulate and run a Pipesim model. Note that this is not the full code. C.1 contains the main loop of the macro to create a single breakpoint. Then C.2 and C.3 includes code for core functionality; running the simulation of a Pipesim model and selecting and configuring the pump of a OpenLink single branch object.

C.1 Main loop to run Pipesim simulation

```
' Set water cut through the black oil fluid model object
BOfluidModel.Watercut = WC
well.BlackOil = BOfluidModel

' Choose pump depending on producer class (low, medium, high)
choosePump well, PIcount, pumpMinRate, pumpMaxRate

' Set Frequency
well.SetPropertyVal "Tubing", "ESP SPEED NODE", freq_step, "hz"

' Set PI of well
well.SetPropertyVal "VertWell_1", "WELLPI LIQPI", PI_step, "STB/d/psi"

' Path for temporary saving of well
wellPath = folder & wellname & "Saved.bps"

' Save model
well.SaveModel wellPath

' RUN SIMULATION (Single Branch)
runPipesimSingle well, wellPath

' Read sumfile, extract data (Hoffman & Stanko, 2016)
readSumfile wellname & "Saved", Qliq, Qloc, Psuc, espPower, efficiency

' Close simulation
well.KillSimulationProcess
```

```
   well.GetPropertyStringAtObjectIndex
      "Tubing", "ESP MANUFACTURER", pumpManuf, 0
   well.GetPropertyStringAtObjectIndex "Tubing", "ESP MODEL", pumpModel, 0
   well.GetPropertyVal "Tubing", "ESP STAGES NODE", pumpStages, pumpStages
   well.GetPropertyVal "Tubing", "ESP MD NODE", pumpMD, pumpMD
   well.GetPropertyVal "VertWell_1", "WELLPI LIQPI", PI, "mmscf/d/psi2"
```

C.2 Run simulation

```
Sub runPipesimSingle(model, path) 'run single branch object

   Dim Running As Boolean
   Dim newHour
   Dim newMinute
   Dim newSecond
   Dim waitTime
   Dim errorStr As String

   Dim DataSheet
   DataSheet = "Data"

   model.RunSingleBranchModel2 True, "-v0 -B", False
   Running = model.GetIsModelRunning()
   model.GetLastError errorStr
   doWrite DataSheet, 1, 25, errorStr

   While Running = True
      Running = model.GetIsModelRunning
      newHour = Hour(Now())
      newMinute = Minute(Now())
      newSecond = Second(Now()) + 1
      waitTime = TimeSerial(newHour, newMinute, newSecond)
      Application.Wait waitTime
   Wend

End Sub
```

C.3 Select pump

```
Sub choosePump(well, PIcount, pumpMinRate, pumpMaxRate)
' PIcount = 1: Low producer
' PIcount = 2: Medium producer
' PIcount = 3: High producer
If (PIcount = 1) Then
   well.SetPropertyStringAtObjectIndex
      "Tubing", "ESP MANUFACTURER", "Centrilift", 0
   well.SetPropertyStringAtObjectIndex "Tubing", "ESP MODEL", "538P100", 0
   well.SetPropertyVal "Tubing", "ESP STAGES NODE", 17, ""
   pumpMinRate = 2000
   pumpMaxRate = 5000
ElseIf (PIcount = 2) Then
   well.SetPropertyStringAtObjectIndex
      "Tubing", "ESP MANUFACTURER", "Centrilift", 0
   well.SetPropertyStringAtObjectIndex "Tubing", "ESP MODEL", "HC12500", 0
   well.SetPropertyVal "Tubing", "ESP STAGES NODE", 12, ""
   pumpMinRate = 5000
   pumpMaxRate = 13500
ElseIf (PIcount = 3) Then
   well.SetPropertyStringAtObjectIndex
      "Tubing", "ESP MANUFACTURER", "Centrilift", 0
   well.SetPropertyStringAtObjectIndex "Tubing", "ESP MODEL", "HC20000", 0
   well.SetPropertyVal "Tubing", "ESP STAGES NODE", 12, ""
   pumpMinRate = 13500
   pumpMaxRate = 20000
End If
End Sub
```

Appendix D

AMPL program

This Appendix shows the code for the core functionality of the AMPL program, that is the files ”Optimization.mod” which contains the optimization model, and ”Peregrino.run” which contains the main method to run the AMPL program.

D.1 Optimization.mod

```
#-----------------------------------#
# OBJECTIVE :
# remove # from the objective line to set objective
# 3 possible objectives: max oil, min water, max pump efficiency
#-----------------------------------#

maximize oilProduction: var_qo_tot;
#minimize waterProduction: var_qw_tot;
#maximize pumpEfficiency: var_pump_eff_tot;

#-----------------------------------#
# CONSTRAINTS:
#-----------------------------------#
# DEFINING VARIABLES:
#-----------------------------------#

subject to definition_qliq{i in WELLS}:
  var_qliq[i] = sum{F in SFesp[i]} Qliq[i, F] * var_Lambda[i, F];

subject to definition_qw_tot:
  var_qw_tot = sum {i in WELLS} var_qliq[i] * (WC[i]);

subject to definition_qo_tot:
  var_qo_tot = sum {i in WELLS} var_qliq[i] * (1-WC[i]);

subject to definition_f_ESP{i in WELLS}:
  var_fesp[i] = sum{F in SFesp[i]} F * var_Lambda[i, F];

subject to definition_p_suc_ESP{i in WELLS}:
  var_p_suc[i] = sum{F in SFesp[i]} Psuc[i, F] * var_Lambda[i, F];

subject to definition_ESP_efficiency{i in WELLS}:
```

```
  var_peff[i] = sum{F in SFesp[i]} PumpEff[i, F] * var_Lambda[i, F];

subject to definition_max_efficiency:
  var_pump_eff_tot = sum{i in WELLS} var_peff[i];

subject to definition_qo{i in WELLS}:
  var_qo[i] = var_qliq[i] * (1-WC[i]);

#-----------------------------------#
# CONSTRAINTS ON LOCAL ESP RATE:
#-----------------------------------#

subject to definition_q_local_ESP{i in WELLS}:
  var_q_esp[i] = sum{F in SFesp[i]} Qesp[i, F] * var_Lambda[i, F];

subject to definition_q_min_esp_ESP{i in WELLS}:
  var_q_pump_min[i] = Qesp_min_at_ref[i] * (var_fesp[i] / 60);
  #F_reference = 60hz

subject to definition_q_max_esp_ESP{i in WELLS}:
  var_q_pump_max[i] = Qesp_max_at_ref[i] * (var_fesp[i] / 60);

subject to q_esp_min_flow{i in WELLS}:
  var_q_esp[i] >= var_q_pump_min[i];

subject to q_esp_max_flow{i in WELLS}:
  var_q_esp[i] <= var_q_pump_max[i];

#-----------------------------------#
# FIELD CONSTRAINTS:
#-----------------------------------#

subject to Max_Oil:
  var_qo_tot <= qo_max;

#subject to Min_Oil:
# var_qo_tot >= qo_max;

subject to Max_Water:
  var_qw_tot <= qw_max;

subject to P_bubble_point {i in WELLS}:
  var_p_suc[i] >= p_b * 1.1 * var_on_off[i]; #1.1 = safety factor

subject to Max_ESP_freq {i in WELLS}:
  var_fesp[i] <= ESP_max_freq;#* var_on_off[i];

subject to Min_ESP_freq {i in WELLS}:
  var_fesp[i] >= ESP_min_freq * var_on_off[i];

#-----------------------------------#
# WELL INTERVENTION:
# remove # to turn off "well_n"
#-----------------------------------#

#subject to Intervene_Well_1{i in WELLS}:
# i == "well_1" ==> var_on_off[i] = 0;

#-----------------------------------#
# SOS2 CONSTRAINTS:
#-----------------------------------#
subject to SOS2_sum_constraint{i in WELLS}:
  sum{F in SFesp[i]} var_Lambda[i, F] = var_on_off[i];
```

D.2 Peregrino.run

```
#----------------------------------#
# Peregrino.run (main method)
#----------------------------------#

reset;

option solver cplex;
option show_stats 1;
option cplex_options 'mipdisplay 2 sos 1 mipgap=5e-18 timelimit=6000';
option log_file 'Peregrino-Optimization.log';

# Include Variables, Parameters and Sets
include Declaration.mod;

# include field data
include field_data.dat;

# Read tables
include read_table.run;

model;
include Optimization.mod;

# SOS2 constraints
include SOS2.run;

solve;
display solve_message;

display var_qo_tot;
display var_qw_tot;

include output.run;
```