**NTNU – Trondheim**
Norwegian University of
Science and Technology

# A Study in MINLP-class Optimization Problems for Simulated Petroleum Production

Håvard Ausen

Master of Science in Engineering Cybernetics
Submission date: June 2012
Supervisor: Bjarne Anton Foss, ITK
Co-supervisor: Bjarne Grimstad, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

# Problem Description

## Background

In an oil and gas production system, where wells are connected to manifolds with multiple pipelines, a complex optimization problem arises. The objective is often to maximize the oil production while respecting constraints on the system, such as limited capacity of processing facilities. The production is dependent on advanced flow dynamics and closing/routing of the wells, which result in nonlinearities and integer variables, respectively. The problem thus belongs to the class of Mixed Integer Nonlinear Programming (MINLP) problems. Furthermore, the nonlinearities in the problem are given by high-fidelity flow simulators and derivative information is unavailable.

In the recent work IFAC (2012)[1], a derivative-free optimization algorithm was implemented and tested on a small instance of the above problem. The algorithm approximates the non-linear simulator models with surrogate models that are fitted to a number of data points from the simulators. In an iterative manner, these models are used to form a MINLP problem that is solved by Bonmin (IBM). The work in IFAC (2012) suggests several improvements of the algorithm, in example incorporating a trust-region and investigating other classes of surrogate models and their convexity properties. It has later been postulated that the algorithm should be modified so that the (local) surrogate models are fitted when solving the NLP subproblems (in the inner loop where the binary variables are fixed). It would be interesting to measure the modified algorithm against the original algorithm in IFAC (2012).

## Description

A literature study on MINLP programming and convexity theory should be conducted as part of the research. The student may also write a short

[1](IFAC 2012) Bjarne Grimstad, Vidar Gunnerud, Dag Ljungquist, Håvard Ausen, and Victoria Lervik. Optimization of a Simulated Well Cluster using Surrogate Models. Pending paper. Automatic Control In Offshore Oil and Gas Production, IFAC Workshop, 2012.

survey on MINLP-solvers (history, development, and application), but this is optional.

The research should be a continuation of previous work IFAC (2012) with focus on refining and testing the proposed algorithm. Since this is ongoing research, the main goal will be to get more insight into the problems outlined above. Hopefully, the thesis might answer some questions and rise new ones.

**Task Description**

- Literature study on convex optimization and MINLP programming. Focus should be on topics related to the above description, e.g. MINLP algorithms (interior point) and trust region.
- Investigate different strategies/schemes for approximating simulator models using convex or non-convex surrogate models. Provide a discussion on different classes of surrogate models and at which point in the algorithm they should be updated (fitted to the simulator data).
- Implement the proposed scheme(-s) and apply it(them) on a simple optimization problem as well as the full optimization problem with binary variables.
- Discuss optimization results and propose further refinements.

**Abstract**

To aid in faster and better decision making it is interesting to couple advanced simulators with optimization tools. Most simulators however does not offer gradients, therefore derivative-free methods must be used. In this thesis optimization of and oil and gas field with free routing is considered. By embedding the structural information in the optimization problem and approximating the simulators by polynomials a MINLP problem is formed which can be solved by gradient based solvers. This approach requires that the polynomial models are updated frequently to fit the simulators. Each update requires several simulations and creates a trade-off between robustness and computation time. Different updating strategies for the models are considered in this thesis. By solving a separate optimization problem to update the models the MINLP problem can be formulated as a convex problem which is solved in a branch and bound framework and with an interior-point. Two approaches to updating the models in respect to the branch and bound method are explored, and it is found to be more robust to update the models for each node of the branch and bound tree, ensuring a local fit before branching.

## Sammendrag

For å lage et verktøy som kan hjelpe til med raskere og bedre avgjørelser er det interessant å prøve og koble sammen avanserte simulatorer med optimaliserings verktøy. De fleste simulatorer tilbyr midlertidig ikke informasjon om gradienter, derfor må gradient frie metoder benyttes. Denne avhandlingen tar for seg produksjons optimalisering på et olje og gass felt med fri routing av brønner. Ved å bygge inn den strukturelle informasjonen i optimaliseringsproblemet og tilnærme simulatorene med polynomer dannes et MINLP problem som kan løses med gradient baserte metoder. Denne tilnærmingen krever at polynom modellene oppdateres ofte for å tilnærme simulatorene. Hver oppdatering krever flere simuleringer og skaper en kompromiss mellom robusthet og beregningshastighet. Ulike oppdaterings strategier for modellene er vurderte i denne avhandlingen. Ved å løse et eget optimaliseringsproblem for å oppdatere modellen kan MINLP problemet formuleres som et konvekst problem som løses med en branch-and-bound metode og en interior-point løser. To tilnærminger til å oppdatere modellene i forhold til branch-and-bound metoden er utforsket, og det ble funnet å være mest robust å oppdatere modellene for hver node av branch-and-bound treet, noe som sikrer en lokal tilpasning før branching.

# Contents

# List of Figures

# List of Tables

# Acknowledgements

I would like to thank Bjarne Grimstad and Vidar Grunnerud for valuable contributions to the reformulation of the optimization problem as well as discussion partners throughout the work on this thesis. I would also like to thank Bjarne Foss for accepting me as his master student.

# Abbreviations

| | |
|---|---|
| BB | Branch and Bound |
| DFO | Derivative-Free Optimization |
| GOR | Gas-to-Oil ratio |
| HSE | Health, Safety and Environment |
| IP | Integer Programming |
| KKT | Karush-Kuhn-Tucker |
| LP | Linear Programming |
| MILP | Mixed Integer Linear Programming |
| MINLP | Mixed Integer Non-Linear Programming |
| MIP | Mixed Integer Programming |
| MPC | Model Predictive Control |
| NLP | Non-Linear Programming |
| PDE | Partial Differential Equation |
| QP | Quadratic Programming |
| SQP | Sequential Quadratic Programming |
| USD | United States Dollar |
| WC | Water-Cut |
| WPC | Well Performance Curve |
| WWII | World War 2 |

# Chapter 1

# Introduction

In engineering advanced simulators have become commonplace to aid in problem solving and for training purposes. Simulators are often used in "what-if" analyses, but problems quickly grow in size to a level where it becomes too time consuming to explore all possible scenarios. To aid in the "what-if" analysis it is interesting to try and couple simulators with optimization techniques. In optimization the possible solutions can be explored in a structured manner, thus guaranteeing an optimal solution. Having a computer tool which can give recommendations together with simulation data can aid in better and faster decision making, the returns can be in the order of hundreds of millions of USD [1].

The most efficient optimization techniques require that the derivatives from the simulators are available. This is often not the case for black-box simulators where a set of input variables produces an output, common when simulators are provided as compiled computer code. Derivative free methods exist for this purpose, they tend to fall into one of two categories: Model-based methods which builds a quadratic or lower order polynomial model of the problem by perpetuating the simulators, and applies line-search or trust-region based methods to calculate the next step. The other category consist of direct search methods which uses a set of tabulated points to determine the next step while implicitly handling the problem structure. Although there

are many examples of simulator-based optimization the maturity of the applications varies a lot [2].
In production optimization problems tend to have a clear structure in the form of production chains, where a product move in one direction through many processes before it if finished and model based methods should be more suitable [2]. Model-based methods can use the structure to quickly converge towards an optimal solution [3].

In oil and gas production the wells are routed through a network of pipes before it reaches processing equipment. This creates a complex optimization problem with non-linear flow dynamics in the wells and pipelines. The objective is to maximize the oil production while upholding capacity constraints such as gas and water handling capacities. The production is given by the discrete routing/closing of the wells. The problem belongs to the class of a mixed integer non-linear programming (MINLP) problems.

## 1.1 Motivation

Production optimization for oilfields can be classified as long-term and short-term optimization, the long term includes decisions on well placements, drainage strategies, design of processing equipment, export strategies etc. The goal is to maximize the net present value of the field [4].
This thesis will focus on short-term optimization; the day to day optimization of oil production. The production has to satisfy production constraints. The constraints tend to move over the lifetime of the field: In the beginning there might be capacity restrictions in the pipelines, while for the tail production it might be the water handling capacity as wells tend to produce a larger amount of water towards the end of the field lifetime.
Today optimization is done by cross-disciplinary teams located onshore and offshore. Decisions are based on well tests, measurements and current performance. An optimization tool can help teams by providing suggestions on how to increase production rates and by giving information on active con-

straints, such as capacity constraints for a given pipeline. It is important to note here that the reliability of the optimization is limited by the accuracy of the simulators and suggestions from the optimization tool could be ignored if there are large uncertainties associated with the reservoir or wells. Several simulators are available to simulate multiphase flow and reservoir dynamics. Some popular ones are PIPESIM and ECLIPSE by Schlumberger [5] [6], LedaFlow by Kongsberg Oil & Gas Technologies [7] and Flow Manager by FMC Technologies [8]. Teams might have access to several of these, and it is therefore desirable to make a general purpose optimization which can combine several simulators in one optimization procedure. Recent advances in MINLP algorithms have made solvers more robust to non-convex problems and they have become readily available for academic and commercial use. Optimization on simulators could be performed by pure derivative free methods, treating the whole system as a black-box. But by splitting the system into separate components, creating many small black-boxes, and explicitly handling the mass and energy balances between each component, a hybrid derivative free optimization method can be created. This should be more efficient than a model-based optimization method as it can exploit the problem structure directly.

## 1.2 Scope

This thesis is a continuation of work performed by the IO center in Trondheim in close collaboration with NTNU. The thesis will explore different formulations of the optimization problem and the best formulation will be used to test how the models should be updated in respect to the integer handling of the optimization procedure. The implementations will be evaluated on performance and robustness criteria.

# Chapter 2

# Problem Description

In production optimization the value chain often consist of a series of processing equipment or stations the product has to move through. Even though the processes which happen inside each station can be very complex, for example the pressure loss in a multiphase pipeline, reactions in a chemical reactor or the development of an oil and gas reservoir, the connections between each station can be modelled by simple energy and mass balances. Or in the case of oil and gas processing; pressure and volumetric flow. By incorporating as much of the structural information as possible in the optimization formulation, the solver can take advantage of this information in each step to bring it closer to the optimum. The complex equipment can be by handled by approximations based on perturbation of the simulators. This creates a hybrid method somewhere between a model-based derivative free method and a gradient based method. The problem can be described as

$$\begin{aligned} \min_{x,b} \; & J(x) \\ s.t. \; & \\ F(x,b) &\leq 0 \\ g(x,b) &= 0 \\ h(x,b) &\leq 0 \\ x \in \mathbb{R}^n \quad & b \in \{0,1\}^m \end{aligned} \tag{2.1}$$

Where all the structural information is given by $g$ and $h$ with known derivatives. $F(x,b)$ is the simulator, and the derivatives are not available. (2.1) will be referred to as the master problem. As the derivatives of $F(x,b)$ are not available an approximation of the simulator which is independent of the binary variables and valid for a short interval around the current iterate, $x_i$, is made.

$$f(x_i + \delta) \approx F(x_i + \delta, b_i) \quad \forall |\delta| \leq \Delta \tag{2.2}$$

For some $\Delta$ greater than zero. The model error is defined as the difference between the model and the simulator at the current iterate.

$$\epsilon_i = |f(x_i) - F(x_i, b_i)| \tag{2.3}$$

When $f$ is an analytical and smooth function the subproblem can be solved by traditional optimization techniques:

$$\begin{aligned} \min_{x,b} \; & J(x) \\ s.t. \; & \\ f(x) &\leq 0 \\ g(x,b) &= 0 \\ h(x,b) &\leq 0 \\ x \in \mathbb{R}^n \quad & b \in \{0,1\}^m \end{aligned} \tag{2.4}$$

By updating the models after each subproblem, or iteration, the model error is expected to decrease and $(x_i, b_i)$ should converge towards the optimal point

of the master problem. To guarantee the same (local) optimal solution for the master problem and the subproblem it is required that the model and the simulator are equal in the optimal point $(x^*, b^*)$

$$f(x^*) = F(x^*, b^*) \tag{2.5}$$

(2.5) is not a sufficient condition for optimality of the master problem, the KKT conditions must be satisfied as well. Therefore it is required that $f$ is also a good approximation in a region around $x^*$ to yield accurate estimates of the derivatives for $F$, as stated in (2.2). This allows us to form a termination criterion for the algorithm: When a solution to (2.4) is found and (2.5) is fullfilled the algorithm has reached a local optimum $(x^*, b^*)$.

### Binary variables

The binary variables form discontinuities in the problem. As the models only form an accurate description in a small area around the current iterate the binary variables can cause the next iterate to be in a completely different area of the feasible region, this causes large model errors, and further optimization without correcting the models will yield a solution that have no relation to the master problem. A challenge is to find a good way to handle the integer variables together with the model updates to increase the chances for the algorithm to quickly converge towards a solution without excluding candidates for the optimal point.

## 2.1 Previous Work

Early work includes [4], where Gunnerud used mixed-integer linear programming to maximize the oil production on a semi-realistic model of the Troll west oil rim containing several well clusters and manifolds. Troll is a challenging field due to gas-coning effects which causes rate dependent GOR. The non-linear behavior in the wells and pipelines were handled by linear constraints creating a MILP problem. The linear constraints causes a very

high number of integer variables, but a linear problem has the advantage of providing a duality gap with the solution and very efficient solvers exist for these problems. The problem was solved with two different decomposition strategies to exploit the special structure of the problem were both approaches proved better than a global method for larger problems. The key differences between [4] and this thesis are the coupling with the simulators. In [4] the entire feasible area has to be explored to build linear constraints, this causes excessive amounts of simulations to build tables and breakpoints. For example the pipeline pressure drop is parameterized with three different flow rates, oil, gas and water. This causes the number of breakpoints to increase with the power of three and the number of binary and continuous variables grows extremely rapidly. Further the space was only discretized in respect to pressure drops, the inlet pressure was not considered which could lead to errors.

Later work includes [9], performed by Ausen, Grimstad and Lervik where the models approximating the simulators where updated iteratively, thus only part of the feasible area is simulated. A well routing problem was solved successfully on a simulated oilfield with four wells, one manifold and two pipelines. The simulators were approximated by quadratic functions which lead to a series of non-convex MINLP problems. After a MINLP problem was solved the model with the largest model error was updated to fit around the current optimal point to create a new MINLP problem, which was then solved again, this process was repeated until the model errors came within predefined tolerances. By solving a MINLP problem with free routing between each model update the optimization might fail to explore all possible candidates for a global optimum because the models only offer a local fit of the simulators, this challenge will be described more in detail later in the thesis. The model update was performed as an exact solution to a set of linear equations which provides an exact match between the model and the tabulated points from the simulator but gives no control on convexity or robustness to noise.

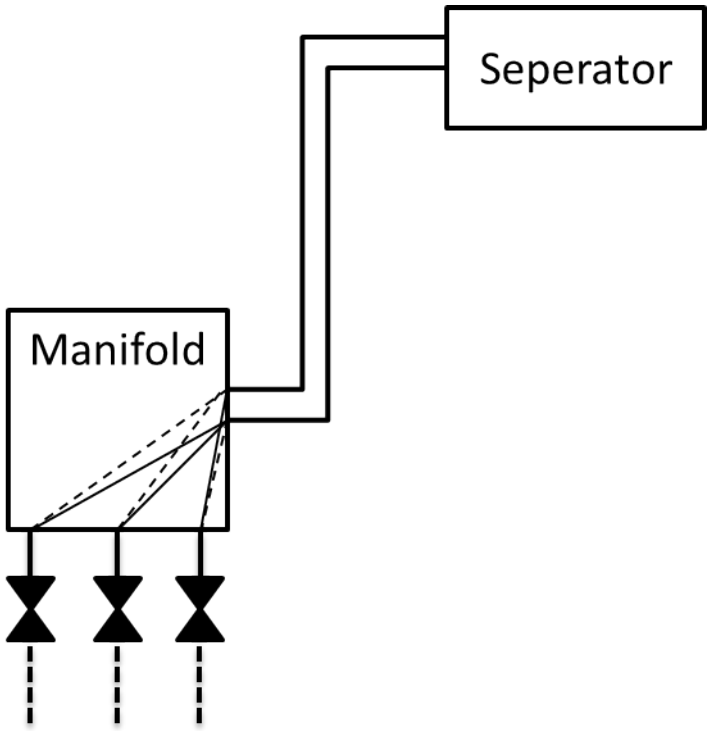



Figure 2.1: Field overview, three wells are routed through a manifold to two pipelines leading to a topside separator.

## 2.2 Case Study

The production optimization is applied to a simulated oilfield with a fixed separator pressure and constant gas-to-oil ratio(GOR), for simplicity it is assumed that no water is present in the system and no capacity constraints for the gas handling capacity. Three wells with different GOR are connected to a manifold and two pipelines lead to the separator located topside as seen in figure 2.1. This is a similar set-up to the one used in [9]. The goal is to find the optimal routing which maximizes oil production. To test the robustness of the algorithm for larger problems more wells will be connected through the same manifold to the two pipelines.

# Chapter 3

# Theoretical Background

Early optimization theory dates back to the beginning of the 20th century [3]. Among the first applications of optimization were resource allocation during WWII [1]. With the advent of computers and algorithmic advances optimization is now commonplace in control theory, such as MPC [10], operations research [1] and logistics planning for firms like FedEx and UPS [1]. LP problems can be solved extremely quickly even for very large problems containing thousands of variables and constraints [3]. Mixed integer programming (MIP) is challenging due to discontinuity and the vast number of possible solutions [1][11]. But some methods have been proven effective and reliable, among them are the Branch and bound method which will be presented in this chapter. In non-linear programming (NLP) SQP has long been very popular, but as the understanding and proofs for interior point algorithms they have been developed it has become equally popular [3], a brief comparison will be given in this chapter.

## 3.1 Convexity

A convex optimization problem guarantees that a global solution is found, and not only a local optimum, and is therefore extremely attractive for optimization purposes. A convex problem also means that the algorithm will

behave in a predictable manner, and not get stuck in stationary points. Convexity is best defined by a convex set, and a set is convex if the line segment joining any two points in the set also lies within the set. Formally it can be stated by letting $C$ be a convex set then

$$ax_1 + (1-a)x_2 \in C$$
$$0 \leq a \leq 1$$

for any $x_1, x_2 \in C$ [12].
A general optimization problem stated as $\min_{x\in\mathbb{R}^n} J(x)$, subject to $c(x) \leq 0$, is convex if the feasible set is a convex set, and the objective function $J(x)$ is convex. This requires that the hessian of fulfils $\nabla^2 J(x) \geq 0$. In the same way we can define the convexity of the feasible set. $c(x) \leq 0$ forms a convex set if $\nabla^2 c(x) \geq 0$ or similarly, $c(x) \leq 0$ forms a convex set if $\nabla^2 c(x) \leq 0$. From this is should be obvious that equality constraint only form a convex set if $\nabla^2 c(x) = 0$ which means that equality constraints must be linear in order to ensure convexity. Constraints on the form

$$x^T Q x + Rx + s \leq 0 \tag{3.1}$$

are convex if and only if $Q \geq 0$, this means that the eigenvalues of Q has to satisfy $\lambda \geq 0, \lambda \in \mathbb{R}$.

## 3.2 Branch and Bound

Integer programming is very difficult in nature compared to normal convex linear and non-linear programming. While continuous problems have an infinite number of feasible points integer problems have a finite number of possible solutions. Intuitively this might lead one to think that MIP is easier, as only a finite number of points needs to be evaluated to guarantee a global optimum. But the number of possible solutions can be astronomically large. Consider a problem with $n$ binary variables the number of possible solutions would be $2^n$, a relatively small problem with 50 variables will have more

than $10^{15}$ possible solutions. So the difficulty of an optimization problem is largely determined by how constrained it is, not the size of the feasible area. And integer variables impose severe constraints on the problem. To solve this some structured way of searching through the possible solutions and excluding a large amount of non-optimal points need to be used. Branch and bound (BB) is one such method. It is well known within the field of IP [1]. As the name indicates it bounds the problem such that only a part of the solutions space is considered. And by branching out it can split a large problem up into many smaller subproblems which can be solved separately. This makes BB very suited for parallelization on computers with multiple cores. To better understand the branch and bound algorithm consider the MILP problem on the form:

$$\begin{aligned} \min_{x\in\mathcal{R}^n, y\in\mathcal{Z}^l} \quad & c^T x + d^T y \\ s.t. & \\ Ax + By &= b \\ x, y &\geq 0 \end{aligned} \tag{3.2}$$

The description that follows is based on [11] and [1]. The algorithm starts by solving a relaxed version of the problem, relaxation here means that the integer constraints on the $y$ variables are removed creating an LP problem. This first relaxed problem is known as the root node.

Next the algorithm selects an element of $y$ from the solution with a non-integer value to branch on. The selection of branching variables can have a big impact on the algorithms performance. No robust method for variable selection exists[11], but a common practice is to use a set of user defined priorities or some form of heuristics [11]. When a branching variable is chosen, two subproblems are generated and added as children to the root node. The child nodes will have constraints added to the problem. Let for example the solution to the root problem be $y = 2.8$, then the first child has the constraint $y \leq 2$ added to the problem and $y \geq 3$ added to second child. If the solution is integer or the relaxed problem is infeasible, no branching is performed.

This process divides the solution space into subspaces, while the part between the two subspaces which does not contain any feasible integer solutions is cut off.
This process is continued with the children inheriting its parents constraints until there are no more unsolved nodes.

When adding more and more constraints as the BB moves down into the search tree created by the making the feasible area smaller BB earns the second part of its name: When the optimal point is left outside of the new feasible region it will, for convex problems, guarantee that all children will have worse optimal solutions than the parent. This allows a bounding to be performed: If a node produces a solution which is worse than the current best integer solution, its children will not yield any better integer solutions, and all its children can be cut off. In other words no branching should be performed on that node. The worst case performance of BB is similar to a brute force algorithm, but in practice it performs much better. For a more thorough introduction including a step-by-step description the author would like to recommend [1] for further reading.

### 3.2.1 Search Strategies

When a branching variable has been chosen and the new nodes created, it still has to determine which node to explore next. Different strategies for searching the tree and picking which node should be considered next have a great influence on the time it takes to find feasible and optimal solutions. If the right choices are made, many nodes can be cut off. Some common search strategies are listed below.

#### Depth-first

Depth-first searches down into the tree, quickly adding more and more constraints. This allows the depth-first search to quickly find feasible solutions to the MIP. These solutions can in turn be used to bound the tree. The strategy can risk to search extensively in a part of the feasible area which

contains few good solutions and thus increasing the number of nodes which has to be explored before and optimum is found.

#### Breadth-first

Breadth-first searches the feasible area quickly; this can be beneficial when a good bound is known before the search is started. It can allow the breadth-first search to quickly eliminate large parts of the feasible area. In general a breadth-first search takes longer to find feasible solutions than depth-first and therefore might risk to explore larger parts of the BB tree.

#### Last solved problem first

Lets the algorithm take advantage of a warm start as children are not very different from the parent node and a lot of the information computed can be reused, such as estimates of active constraints.

#### Best bound

Chooses the node which has the best bounds. As this is more likely to yield a good solution.

As both the choice of branching variable and search strategy greatly influence BB performance the choice for both strategies can be made by closely studying the problem structure, but might as well be determined by trail and error.

## 3.3 Non-Linear Optimization

Two methods for non-linear optimization will be described briefly: Active-set and interior-point. The descriptions given here are based on [3] and[12]. The

description will focus on key differences, advantages and disadvantages for each method.

### 3.3.1 Active-Set Method

Active-set sequential quadratic programming (SQP) was developed in the 1970s. The name refers to how the method handles constraints; an estimate of the active set is used to find candidates for optimal solutions. It solves the non-linear problem by calculating the step direction from a QP problem using linearised constraints, feasibility is handled by line search or trust region methods. Active-set methods are well suited to use together with BB as each problem is very similar to the previous and good estimates of the active set are easy to give.

### 3.3.2 Interior-Point Method

Often called a barrier method, it solves the problem by computing a solution the the primal dual problem based on the KKT conditions. The method earns its name from how it moves towards an optimal point; the method moves along a strictly feasible central path. This makes it better to handle non-convexities [3] and makes interior-point very attractive for real-time implementations such as MPCs [10]. But it is less suited to take advantage of precalculated information, such as estimates of the active se,t which make them less suited to use with BB [3]. Interior-point methods tend to outperform active-set methods in large scale applications, but as they consider all constraints for each step the calculations can be very expensive.

## 3.4 Derivative Free Optimization

Optimization techniques are very robust and efficient when the problem is well formulated, but in many applications the need arises to use optimization where the explicit function and its derivatives are unknown. Derivative

free optimization techniques differ from traditional optimization in the fact that sampled function values are used to determine the next step instead of gradient information and KKT conditions. Model based methods construct a model, usually based on polynomials, by interpreting the sampled values. This is effective in many cases, especially if the underlying problem contains a clear structure [3]. Model based methods are especially sensitive to round off errors and errors due to finite difference models common when solving PDEs with computers.

More general purpose DFO methods exist, the most popular methods are perhaps simulated annealing and Nelder Mead [13]. They use a set of sampled variables to determine a new iterate. The main drawback for DFO methods is that the number of function evaluations can be excessive [3]. They also tend to be very sensitive to scaling.

## 3.5 Chapter Summary

In [9] and [2] the basis were laid out for a hybrid derivative free-method which aims to exploit the structural information in a production network by embedding structural information in the optimization formulation. As the problem belongs to the class of MINLP problems an SQP solver might be more suitable but IPOpt, a interior-point solver, was chosen to handle all NLP optimization due to previous experience and readiness of this solver. Branch and bound was used to handle integer constraints, a BB method is available through BONMIN, but a separate BB method was written by the author to achieve a more customizable method. The BB method was written in Python and employs a breadth-first search strategy. A complete list of options passed to BONMIN and IPOpt is found in appendix A.4.

# Chapter 4

# Data Sets and Approximated Models

In this chapter the simulator data will be presented and a brief discussion on different methods for interpolating techniques will be given. Finally a method for updating the models while controlling convexity is described.

## 4.1 Simulator Data

The simulator data was attained using a commercially available simulator on a simulated oilfield equal to the field described in figure 2.1. The data was stored as tables during the optimization. First the concept of water-cut (WC) and gas-to-oil ratio (GOR) is introduced:

**Gas to oil ratio**

GOR is the relation between the amounts of gas flowing from a well divided by the oil flow.

$$\frac{q_{gas}}{q_{oil}} = \text{GOR} \tag{4.1}$$

**Water cut**

WC is used as a measure of how much of the liquid produced is water

Table 4.1: List over well GOR

| Well | GOR |
|---|---|
| 1 | 100 |
| 2 | 150 |
| 3 | 200 |

$$\begin{aligned} q_{water} &= \mathrm{WC} q_{liquid} \\ q_{oil} &= (1 - \mathrm{WC}) q_{liquid} \end{aligned} \tag{4.2}$$

### 4.1.1 Well Performance Curve

The reservoir, well and pipeline leading up to the manifold are all lumped together in one model, called the WPC (Well Performance Curve). It gives the pressure as a function of flow rate, or vice versa. With a fixed GOR and WC the wellhead pressure can be plotted as a function of oilrate, as seen in figure 4.1. The GOR for each well is listed in table 4.1

### 4.1.2 Pipeline

The pressure drop in a pipeline is in general very complex, as it is a function of inlet pressure, viscosity, flow speed, temperature and density as well as the diameter and roughness of the pipe. Different flow regimes can also create different pressure drops even for the same set of input parameters. By fixing the outlet pressure with fixed separator pressure and keeping the WC constant and simulating 10 different flow rates and 3 different values for GOR the pressure loss can be plotted as a function of oilflow for different GORs. The resulting plot can be seen in figure 4.2.

By discretizing the space in terms of GOR and WC instead of using phase flow, there is no need to build tables for non-existing scenarios. For example a table might include the pressure drop for a pipeline with just gas flow,

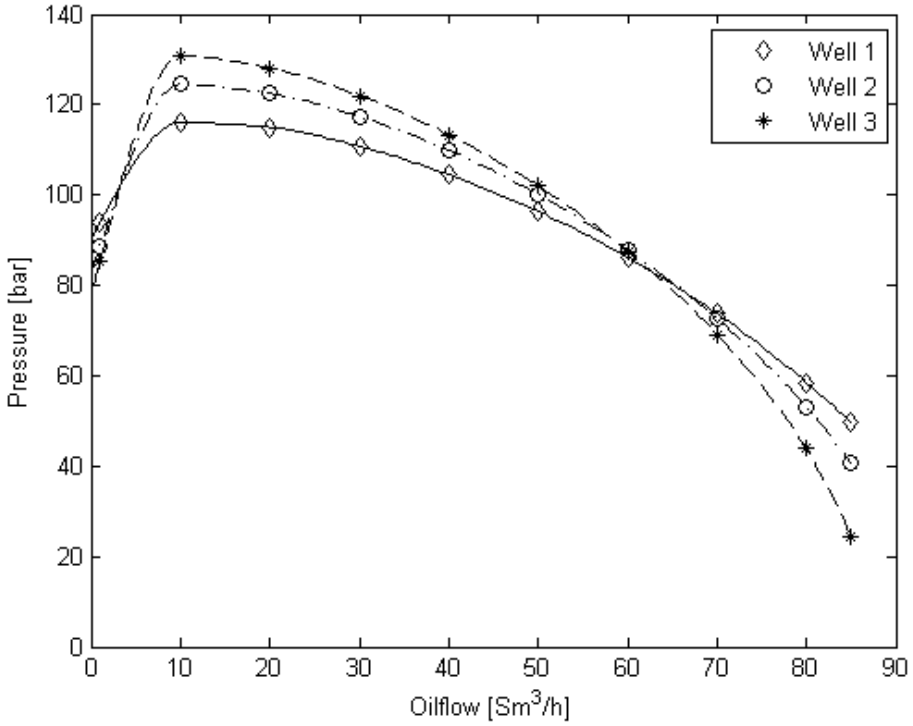



Figure 4.1: The well pressure as a function of oil flow, GOR and WC are constant

although this will never occur in the optimization problem since no wells have a GOR equal to zero.

## 4.2 Different Interpolation Techniques

There are several different ways of approximating data with continuous functions. The most common is probably using polynomial interpolation. The challenge is to capture all the relevant dynamics from the simulator but at the same time, the models must be simple enough to be used in optimization.

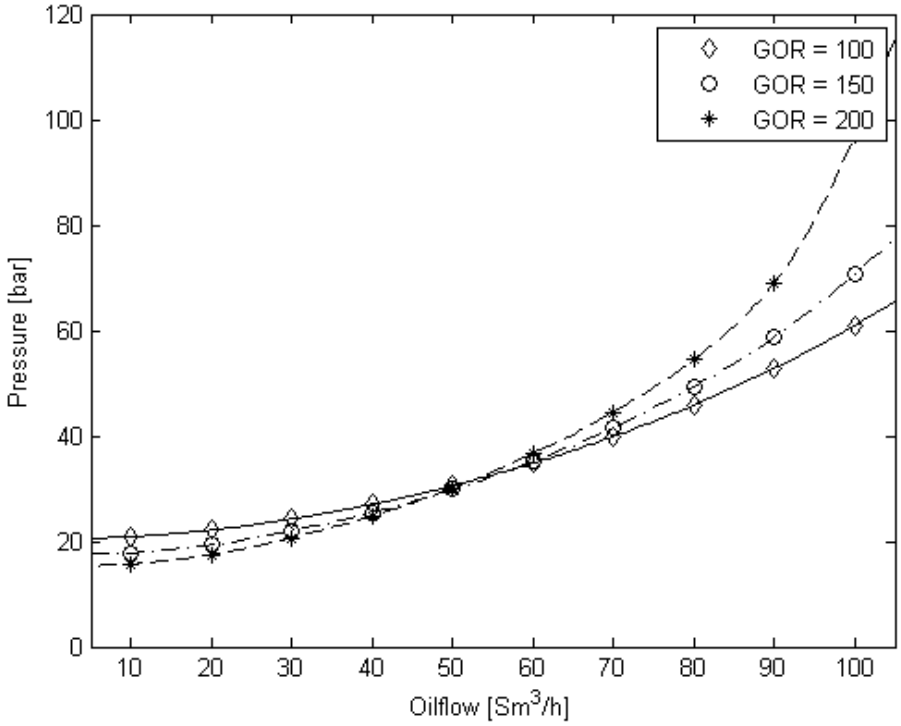



Figure 4.2: The pressure loss in the pipeline for different GOR and a constant WC

### 4.2.1 Polynomial Interpolation

Linear models are the simplest kind of polynomials, these are widely used, as in SQP where constraints are linearised in each step. The shortcoming of these is that the models are (possibly) only valid within a short deviation from the start point, and does not offer any information on second derivatives. For polynomial interpolation the error decreases as the order increases, but only up to a certain point. Increasing the order beyond this results in the error exploding [14].

Higher order Taylor approximations are possible, but often not necessary. It is rare to use higher order than 4th to 5th order polynomials, as the polynomial has a tendency to oscillate wildly between the tabulated values [14].

### 4.2.2 Rational Function Interpolation

Rational functions are sometimes more suited for interpolation than polynomials, as they can model poles. Most commonly they are expressed as a quotient of polynomials [14]. The most famous rational function for interpolation is perhaps the Padé approximation. Rational functions are well suited for high order approximations, as the error tends to decrease as the order is increased [14].

### 4.2.3 Splines

Splines are a special case of polynomial interpolation: The spline is performed by fitting many curves, each valid within a set range. In addition the function, and for higher order splines; the derivatives, are required to be continuous. The most common splines are cubic and linear splines. Splines are most common in cases where the derivatives of the original function are known [14].

### Summary

In optimization the main concern is convexity of the feasible area created by the model, this is controlled by either making models convex or concave. This makes rational functions unsuited for optimization purposes as the convexity becomes unnecessarily complicated to control. Splines provide continuous derivatives but as they are made up of several functions, they are overly complex. Polynomial interpolation provides the best type of fit. A quadratic fit provides sufficient model accuracy as well as easily detectable convexity. A quadratic constraint can be written on the form:

$$x^T Qx + Rx + s \leq 0 \tag{4.3}$$

Assuming the matrix $Q$ is written in a triangular form, the convexity can be asserted by

$$diag(Q) \geq 0 \tag{4.4}$$

where $diag(Q)$ is a vector containing the elements of $Q$'s diagonal.

## 4.3 Updating the Models

First some variables are introduced

| Variable name | Description |
|---|---|
| $x_k$ | The points used to tabulate the simulator with |
| $y_k$ | The resulting values from running the simulator with $x_k$ |
| $x_0$ | The point which the simulator is tabulated around |
| $y_0$ | The resulting value from running the simulator with $x_0$ |

The simplest way to perform a polynomial interpolation is to find the solution to $Ax = b$, where for a 2nd degree polynomial $y = ax^2 + bx + c$

$$A = \begin{bmatrix} 1 & x_0 & (x_0)^2 \\ 1 & x_1 & (x_1)^2 \\ 1 & x_2 & (x_2)^2 \end{bmatrix}, b = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix}, x = \begin{bmatrix} c \\ b \\ a \end{bmatrix} \tag{4.5}$$

As was done in [9]. If more points where used, a least squares solution could be used which gives some robustness to noise, but does not offer any control on convexity. Instead an optimization problem is formulated. The optimization problem is defined as

$$\begin{aligned} \min_{Q,R,s} \sum_{k=0} \left( \left( x_k^T Q x_k + R x_k + s \right) - y_k \right)^2 \\ s.t. \\ {x_0}^T Q x_0 + R x_0 + s = y_0 \\ diag(Q) \geq 0 \\ Q \text{ triangular} \end{aligned} \tag{4.6}$$

The equality constraint is added to ensure equality between the model and simulator when using more points than required. $diag(Q) \leq 0$ can be used if the model has to be a concave.

### 4.3.1 Model Update Robustness in the Presence of Noise

To test the robustness of (4.6) to noise, the number of points for each well simulator was doubled by using interpolation and a small noise signal was added to introduce non-convexity. The noise causes the model to be linear when only three points are used. The linearity is due to the convexity constraint. From figure 4.3 it is clear that robustness to noise can be increased by increasing the number of points used to update the models. It smooths out noise and thus acts as a filter. Choosing how these points should be spaced around $x_0$ is a very difficult problem, but a common way to do it is a fixed grid with equal distance between each point [14]. In this case the simulators are smooth and only a minimum number of points are used to update each model. If the presence of noise is known, a larger number of points should be used.

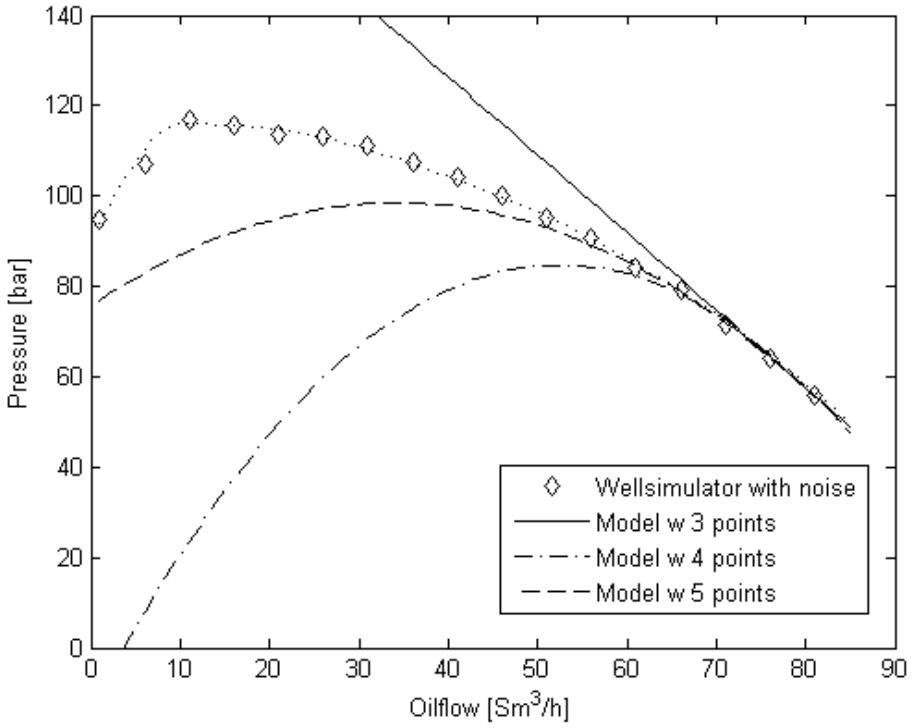



Figure 4.3: Model update robustness to noise

# Chapter 5

# Different formulations

In optimization a lot of emphasis is often put on finding a good solver. But reformulating the problem can prove more effective than changing solver. In this chapter three different formulations of the same problem will be given, all with different properties. Towards the end of the chapter the formulations are tested for robustness and performance. To make the formulations easier to compare with other work, and expanded, the water phase is used in the formulations even when it is assumed no water is present in the system. First two techniques are described for reformulating optimization problems.

## Linearizing mixed integer constraints

Non-linear constraints are more difficult to handle than linear and it is a big advantage to have linear constraints. Consider a constraint on the form

$$xy \leq 0 \tag{5.1}$$

Where $y$ is a binary variable ensuring that $x \leq 0$ only if $y = 1$. It can be rewritten as

$$x - M(1 - y) \leq 0 \tag{5.2}$$

This is called a big-M formulation, where $M$ is a constant. As long as $M$ is greater than $x$ this is the same constraint. At first glance it might seem

like a good idea to use a large as possible value for $M$, but this can cause the problem to be poorly scaled. This not only affects performance, but can cause the problem to be ill conditioned to a level where the solver fails to find a solution.
A large value for $M$ increases the solution space and hence the way the algorithm behaves. Therefore $M$ should be chosen as small as possible, while still guaranteeing that the original feasible area remains unchanged. To achieve the best formulation, different values have to be used for the different inequalities. Optimization problems are in general bounded and a bound on $M$ is normally possible to find.

### Elimination of variables

Equality constraints can be used to eliminate variables, this is straight forward for linear constraints, and most solvers have methods for handling this [3]. But non-linear constraints can impose implicit bounds on variables. For example the constraint

$$(x_1 - 1)^3 = x_2^2 \tag{5.3}$$

Implicitly holds the constraint $x_1 \geq 1$ and if $x_2^2$ is eliminated by substitution $x_1 \geq 1$ has to be added to the problem to ensure mathematical equality.

## 5.1 Sets and Notation

To define the problem, sets of variables are used. The definitions are given in table 5.1, 5.2 and 5.3. All variables are defined as non-negative. It is assumed that the convexity of the models is controlled by using the method described in section 4.3.

Table 5.1: Indexes and sets

| Index and set | Description |
|---|---|
| $j \in \mathcal{J} = \{1, 2, 3\}$ | Wells |
| $l \in \mathcal{L} = \{1, 2\}$ | Pipelines |
| $p \in \mathcal{P} = \{oil, gas, water\}$ | Phases |

Table 5.2: Variables. All variables are non-negative

| Variable | Description | Units |
|---|---|---|
| $b_{j,l}$ | Binary routing variable for flow from well $j$ to pipeline $l$ | [] |
| $q_{j,p}^{Well}$ | Flowrate of phase $p$ from well $j$ | $[m^3/h]$ |
| $q_{l,p}^{Pipe}$ | Flowrate of phase $p$ in pipeline $l$ | $[m^3/h]$ |
| $q_{j,l,p}^{Man}$ | Flowrate of phase $p$ from well $j$ to pipeline $l$ | $[m^3/h]$ |
| $p_j^{Well}$ | Wellhead pressure for well $j$ | $[bar]$ |
| $p_l^{Man}$ | Manifold pressure for pipeline $l$ | $[bar]$ |
| $\Delta p_l^{Pipe}$ | Pressure drop over pipeline $l$ | $[bar]$ |

## 5.2 Basic Formulation

This formulation is called basic because it is intuitive to set up and understand, but it is not suited for optimization due to many non-linear equality constraints, making the problem non-convex. Apart from minor differences it is the same formulation which was used in [9].

### Objective

The objective is to maximize the oil production.

$$\max \sum_{l \in \mathcal{L}} q_{l,oil}^{Pipe} \tag{5.4}$$

Table 5.3: Constants

| Parameter | Description | Units |
|---|---|---|
| $P_{Sep}$ | Separator inlet pressure | $[bar]$ |
| $C_{gas}$ | Topside gas handling capacity | $[m^3/h]$ |
| $C_{water}$ | Topside water handling capacity | $[m^3/h]$ |
| $r_j^{GOR}$ | Gas to oil ratio for well $j$ | [] |
| $r_j^{WC}$ | Water cut for well $j$ | [] |
| $M_p^q$ | big-M for volumetric flow of phase $p$ | $[m^3/h]$ |
| $M^P$ | big-M for pressure | $[bar]$ |

### Topside production restrictions

The production facilities have constraints on how much water and gas they can treat

$$\sum_{l \in \mathcal{L}} q_{l,gas}^{Pipe} \leq C_{gas} \tag{5.5}$$

$$\sum_{l \in \mathcal{L}} q_{l,water}^{Pipe} \leq C_{water} \tag{5.6}$$

A fixed separator pressure is required

$$p_l^{Man} - \Delta p_l^{Pipe} = P_{Sep} \quad \forall l \in \mathcal{L} \tag{5.7}$$

### Pipeline

The pressure loss in the pipeline is given by the quadratic model and is a function of phase flow.

$$\Delta p_l^{Pipe} = P_l(q_{l,oil}^{Pipe}, q_{l,gas}^{Pipe}, q_{l,water}^{Pipe}) \quad \forall l \in \mathcal{L} \tag{5.8}$$

$$\begin{aligned} P_l(q_{l,oil}^{Pipe}, q_{l,gas}^{Pipe}, q_{l,water}^{Pipe}) =& \beta_{0,l} + \beta_{1,l} q_{l,oil}^{Pipe} + \beta_{2,l} q_{l,gas}^{Pipe} + \beta_{3,l} q_{l,water}^{Pipe} \\ &+ \beta_{4,l} (q_{l,oil}^{Pipe})^2 + \beta_{5,l} (q_{l,gas}^{Pipe})^2 + \beta_{6,l} (q_{l,water}^{Pipe})^2 \\ &+ \beta_{7,l} q_{l,oil}^{Pipe} q_{l,gas}^{Pipe} + \beta_{8,l} q_{l,oil}^{Pipe} q_{l,water}^{Pipe} + \beta_{9,l} q_{l,gas}^{Pipe} q_{l,water}^{Pipe} \end{aligned} \tag{5.9}$$

This can also be written as a matrix formulation on the form $x^TQx + Rx + s$

$$P_l = \begin{bmatrix} q_{l,oil}^{Pipe} \\ q_{l,gas}^{Pipe} \\ q_{l,water}^{Pipe} \end{bmatrix}^T \begin{bmatrix} \beta_{4,l} & \beta_{7,l} & \beta_{8,l} \\ 0 & \beta_{5,l} & \beta_{9,l} \\ 0 & 0 & \beta_{6,l} \end{bmatrix} \begin{bmatrix} q_{l,oil}^{Pipe} \\ q_{l,gas}^{Pipe} \\ q_{l,water}^{Pipe} \end{bmatrix} + \begin{bmatrix} \beta_{1,l} \\ \beta_{2,l} \\ \beta_{3,l} \end{bmatrix}^T \begin{bmatrix} q_{l,oil}^{Pipe} \\ q_{l,gas}^{Pipe} \\ q_{l,water}^{Pipe} \end{bmatrix} + \beta_{0,l} \quad (5.10)$$

**The manifold**

The flow in to the manifold has to be equal to the flow out from the manifold, this is handled by a simple mass balance

$$q_{l,p}^{Pipe} = \sum_{j \in \mathcal{J}} b_{j,l} q_{j,p}^{Well} \quad \forall l \in \mathcal{L}, p \in \mathcal{P} \quad (5.11)$$

The outlet pressure from the manifold cannot be greater than the pressure from the wells routed to that pipeline:

$$p_l^{Man} b_{j,l} \leq p_j^{Well} \quad \forall l \in \mathcal{L}, j \in \mathcal{J} \quad (5.12)$$

The routing variables are used to turn the constraints on and off. In the case that there exist a pressure loss between the well and manifold pressure this difference could be used to calculate the correct choke position. Thus the chokes are implicitly handled. This is a great advantage as chokes would introduce more nonlinearities to the problem. Routing constraints ensures that a well can only be routed to a maximum of one pipeline.

$$\sum_{l \in \mathcal{L}} b_{j,l} \leq 1 \quad \forall j \in \mathcal{J} \quad (5.13)$$

**The wells**

$$q_{j,oil}^{Well} \leq M_{oil}^{q} \sum_{l \in \mathcal{L}} b_{j,l} \quad \forall j \in \mathcal{J} \quad (5.14)$$

Ensures that the flow from the well is zero if it is not routed to any pipelines. Because of the fixed GOR and WC, the relationship between the phases is linear

$$q_{j,gas}^{Well} = r_j^{GOR} q_{j,oil}^{Well} \quad \forall j \in \mathcal{J} \quad (5.15)$$

$$q_{j,water}^{Well} = r_j^{WC} q_{j,oil}^{Well} \quad \forall j \in \mathcal{J} \tag{5.16}$$

$r^{WC}$ can be defined as $r^{WC} = \frac{WC}{1-WC}$ to be equal to the definition of WC given in (4.2) which complies with the industry understanding of water cut. As the relationship between oil, gas and water flow is linear the well and reservoir models are reduced to

$$p_j^{Well} = W_j(q_{j,oil}^{Well}) \quad \forall j \in \mathcal{J} \tag{5.17}$$

$$W_j(q_{j,oil}^{Well}) = \alpha_{0,j} + \alpha_{1,j} q_{j,oil}^{Well} + \alpha_{2,j}(q_{j,oil}^{Well})^2 \tag{5.18}$$

## 5.3 Reduced Formulation

By exploiting the many equality constraints in the basic formulation the number of variables and constraints can be reduced greatly. A close look at the basic formulation reveals that all the pressure variables are functions of flow (the topside separator pressure is a constant parameter, not a variable). It should therefore be possible to reduce the problem to a form without pressure variables. The most important property of the reduced formulation is that it is a convex problem when all the binary variables are fixed.

**Objective**

$$\max \sum_{l \in \mathcal{L}} q_{l,oil}^{Pipe}$$

**Pressure**

By inserting (5.8) into (5.7) $p_l^{Man}$ can be eliminated.

$$(\Delta p_l^{Pipe} + P_{Sep}) b_{j,l} \leq p_j^{Well} \quad \forall l \in \mathcal{L}, j \in \mathcal{J} \tag{5.19}$$

(5.8) and (5.17) are non-linear equalities by eliminating these, the problem becomes more convex. This is achieved by inserting (5.8) and (5.17):

$$\left(P_l(q_{l,oil}^{Pipe}, q_{l,gas}^{Pipe}, q_{l,water}^{Pipe}) + P_{Sep}\right) b_{j,l} \leq W_j(q_{j,oil}^{Well}) \quad \forall l \in \mathcal{L}, j \in \mathcal{J} \tag{5.20}$$

As mentioned earlier the elimination of variables must be done with some care, as the pressure variables were non-negative, a bound has to be placed on the models

$$P_l(q_{l,oil}^{Pipe}, q_{l,gas}^{Pipe}, q_{l,water}^{Pipe}) \geq 0 \quad \forall l \in \mathcal{L} \tag{5.21}$$

$$W_j(q_{j,oil}^{Well}) \geq 0 \quad \forall j \in \mathcal{J} \tag{5.22}$$

The pressure variables have now successfully been eliminated. The rest of the problem is formulated in the same way as for the basic formulation.

**Topside production restrictions**

$$\sum_{l \in \mathcal{L}} q_{l,gas}^{Pipe} \leq C_{gas}$$

$$\sum_{l \in \mathcal{L}} q_{l,water}^{Pipe} \leq C_{water}$$

**The manifold**

$$q_{l,p}^{Pipe} = \sum_{j \in \mathcal{J}} b_{j,l} q_{j,p}^{Well} \quad \forall l \in \mathcal{L}, p \in \mathcal{P}$$

$$\sum_{l \in \mathcal{L}} b_{j,l} \leq 1 \quad \forall j \in \mathcal{J}$$

**The wells**

$$q_{j,oil}^{Well} \leq M_{oil}^{q} \sum_{l \in \mathcal{L}} b_{j,l} \quad \forall j \in \mathcal{J}$$

$$q_{j,gas}^{Well} = r_j^{GOR} q_{j,oil}^{Well} \quad \forall j \in \mathcal{J}$$

$$q_{j,water}^{Well} = r_j^{WC} q_{j,oil}^{Well} \quad \forall j \in \mathcal{J}$$

## 5.4 Convex Formulation

The non-convexity that existed in the basic formulation was in the models and in the mass balance equation. A partial solution to this is given in the reduced formulation which is convex for fixed routing. But the problem can be further improved to make the problem convex.

### Objective

$$\max \sum_{l \in \mathcal{L}} q_{l,oil}^{Pipe}$$

### Pressure

Again the need to eliminate the nonlinear equality constraints is present. But to simplify the equation further, and thereby simplifying the Jacobian of the constraints, a rewriting of (5.12) is needed

$$p_l^{Man} b_{j,l} \leq p_j^{Well} \quad \forall l \in \mathcal{L}, j \in \mathcal{J}$$

can be rewritten to

$$p_l^{Man} \leq p_j^{Well} + M^P(1 - b_{j,l}) \quad \forall l \in \mathcal{L}, j \in \mathcal{J} \tag{5.23}$$

Following the same procedure as for the reduced problem by inserting (5.7), (5.8) and (5.17) we arrive at

$$P_l(q_{l,oil}^{Pipe}, q_{l,gas}^{Pipe}, q_{l,water}^{Pipe}) + P_{Sep} - W_j(q_{j,oil}^{Well}) + M^P(b_{j,l} - 1) \leq 0 \quad \forall l \in \mathcal{L}, j \in \mathcal{J} \tag{5.24}$$

Which is a convex constraint given that $P_l$ is a convex function and $W_j$ is concave. As with the reduced formulation the pressures must be positive

$$P_l(q_{l,oil}^{Pipe}, q_{l,gas}^{Pipe}, q_{l,water}^{Pipe}) \geq 0 \quad \forall l \in \mathcal{L}$$
$$W_j(q_{j,oil}^{Well}) \geq 0 \quad \forall j \in \mathcal{J}$$

### The manifold

The mass conservation in (5.11) is the last remaining non-convex function. This can be rewritten to a linear function by introducing a new variable, the flow within the manifold, $q^{Man}_{j,l,p}$, the flow from well $j$ to pipe $l$ of phase $p$. A graphical representation is given in figure 5.1.

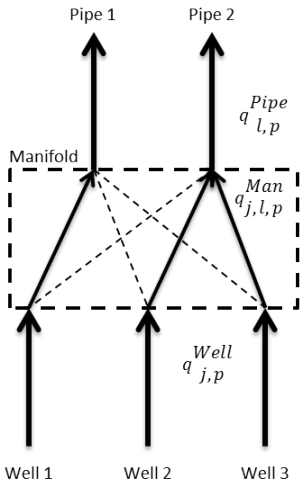



Figure 5.1: A graphical interpretation of the manifold flow

Flowing into the manifold we have

$$q^{Well}_{j,p} = \sum_{\forall l \in \mathcal{L}} q^{Man}_{j,l,p} \quad \forall j \in \mathcal{J}, p \in \mathcal{P} \tag{5.25}$$

And similarly out of the manifold

$$q^{Pipe}_{l,p} = \sum_{\forall j \in \mathcal{J}} q^{Man}_{j,l,p} \quad \forall l \in \mathcal{L}, p \in \mathcal{P} \tag{5.26}$$

The routing is handled by

$$q^{Man}_{j,l,p} \leq M^q_p b_{j,l} \quad \forall j \in \mathcal{J}, l \in \mathcal{L}, p \in \mathcal{P} \tag{5.27}$$

Note that all these constraints are linear. The rest remains as in the basic problem:

Routing constraints

$$\sum_{l \in \mathcal{L}} b_{j,l} \leq 1 \quad \forall j \in \mathcal{J}$$

**Topside production restrictions**

$$\sum_{l\in\mathcal{L}} q_{l,gas}^{Pipe} \leq C_{gas}$$

$$\sum_{l\in\mathcal{L}} q_{l,water}^{Pipe} \leq C_{water}$$

**The wells**

$$q_{j,oil}^{Well} \leq M_{oil}^{q} \sum_{l\in\mathcal{L}} b_{j,l} \quad \forall j \in \mathcal{J}$$

$$q_{j,gas}^{Well} = r_j^{GOR} q_{j,oil}^{Well} \quad \forall j \in \mathcal{J}$$

$$q_{j,water}^{Well} = r_j^{WC} q_{j,oil}^{Well} \quad \forall j \in \mathcal{J}$$

## 5.5 Big-M

The big-M notation is used extensively through the convex formulation, but also for ensuring the closing of wells in the other formulations. The transformation from $xy \leq 0$ to $x + M(y-1) \leq 0$ is only valid if $M \geq \max\{x\}$.
Two M's have been used in the formulation $M_p^q$ for volume flow with subscript $p$ determining the phase, and $M^P$ for pressure.

**Volume flow**

From (5.14) and (5.27) it should be clear that

$$M_p^q \geq \max_{j\in\mathcal{J}}\{q_{j,p}^{Well}\} \quad \forall p \in \mathcal{P} \tag{5.28}$$

$$M_p^q \geq \max_{j\in\mathcal{J},l\in\mathcal{L}}\{q_{j,l,p}^{Man}\} \quad \forall p \in \mathcal{P} \tag{5.29}$$

but as $q_{j,p}^{Well} = \sum_{l\in\mathcal{L}} q_{j,l,p}^{Man}$ it is clear that $q_{j,p}^{Well} \geq q_{j,l,p}^{Man}$ and (5.28) and (5.29) can be reduced to just $M_p^q \geq \max_{j\in\mathcal{J}}\{q_{j,p}^{Well}\}$. The maximum value can be found by using the fact that we cannot have negative pressures, we can say that $M_{oil}^q \geq q_{oil}^{max}$ where $q_{oil}^{max}$ is solution to:

$$\begin{aligned} &\max \quad q \\ &s.t. \\ &\max_{j\in\mathcal{J}}\{W_j(q)\} \geq 0 \\ &q \geq 0 \end{aligned} \tag{5.30}$$

Or $q_{oil}^{max}$ could be calculated directly from the roots of $W_j(q)$. The limits for the other two phases are easily found by applying the linear relationship between them:

$$M_{gas}^{q} \geq \max_{j\in\mathcal{J}}\{r_j^{GOR}\}q_{oil}^{max} \tag{5.31}$$

$$M_{water}^{q} \geq \max_{j\in\mathcal{J}}\{r_j^{WC}\}q_{oil}^{max} \tag{5.32}$$

**Pressure**

For pressure it is slightly more complex, to simplify the notation, the binary variables are left out. From (5.24) it is clear that $M^P$ has to be bigger than the largest pressure:

$$M^P \geq \max_{q_{l,p}^{Pipe},q_{j,p}^{Well}\in\mathcal{J},\mathcal{L},\mathcal{P}}\{P_l(q_{l,oil}^{Pipe}, q_{l,gas}^{Pipe}, q_{l,water}^{Pipe}) + P_{Sep} - W_j(q_{j,oil}^{Well})\} \tag{5.33}$$

As we cannot have negative pressures $\max\{-W_j(q_{j,oil}^{Well})\} = 0$. The pressure loss in the pipeline cannot be higher than the pressure supplied by the wells. This can be proven by rearranging (5.24) to

$$P_l(q_{l,oil}^{Pipe}, q_{l,gas}^{Pipe}, q_{l,water}^{Pipe}) + P_{Sep} \leq W_j(q_{j,oil}^{Well}) \quad \forall l \in \mathcal{L}, j \in \mathcal{J} \tag{5.34}$$

Using this it is possible to write

$$M^P \geq \max_{q_{j,oil}^{Well},j\in\mathcal{J}}\{W_j(q_{j,oil}^{Well})\} \tag{5.35}$$

As $W_j$ is a concave 2nd order polynomial the maximum can be calculated directly by

$$M^P \geq \max_{j\in\mathcal{J}}\left\{W_j(-\frac{\alpha_{1,j}}{2\alpha_{2,j}})\right\} \tag{5.36}$$

Table 5.4: Table for number of variables and constraints for different formulations

| | Basic | Reduced | Convex |
|---|---|---|---|
| Variables | 23 | 16 | 28 |
| Constraints | 26 | 24 | 42 |
| Non-linear constraints | 15 | 15 | 11 |

or if $-\frac{\alpha_{1,j}}{2\alpha_{2,j}} < 0$

$$M^P \geq \max_{j \in \mathcal{J}} \{W_j(0)\} \tag{5.37}$$

Equation (5.29) and (5.35) lets us form tight bounds on the values for big-M. These bounds will change after each model update but they remove the need for a user set value, and make the algorithm much more reliable. This is very useful in a setting where the user has little, or no knowledge of optimization.

## 5.6 Comparison of Different Formulations

Table 5.4 shows that the reduced number of variables for the reduced problem does not lead to a significant reduction in the number of constraints, this is due the implicit bounds which were held by the non-negative variables. The convex formulation displays more variables and constraints than the other formulation. This is an example of how the feasible area can be improved by introducing more variables and constraints.

The formulations were tested by solving a subproblem using models with a known optimum. The problems where formulated in AMPL [15] and solved with BONMIN, a powerful MINLP solver released by COIN-OR [16]. The results are shown in table 5.6 and 5.5. For table 5.5 it should be noted that with the routing locked, BONMIN found the optimal point for all three formulations with heuristics, therefore no information is given on the number

of iterations or nodes. The basic formulation fails in the start-point and performance test. Ausen, Grimstad and Lervik argues in [9] that a feasible start point will in general be available for oilfields as it is possible to use the current implementation of routing and choke positions as a starting point for the algorithm. It would also be possible to do some simple precalculations to find a feasible starting point. But it is clear that the basic and reduced formulations are far less robust than the convex formulation even though some computational advantage is seen in table 5.5. The convex formulation guarantees to find the global optimum for each subproblem, as long as this property is maintained any errors which occur will be due to true infeasibility of the problem or errors due to model updates. Although the reduced formulation is convex for a fixed routing it is still non-convex which means that the bounding property in BB is not preserved. This means that children for a given node might yield better solutions than the parent node. As a result bounding cannot be performed in the same way as for convex problems. This makes the convex problem the best suited for further implementation.

Table 5.5: Comparing the performance for the different model formulations. Number of interior-point iterations/Number of nodes searched

| Routing | Basic | Reduced | Convex |
|---|---|---|---|
| Free | 367/18 | 357/18 | 637/34 |
| 2 free wells | Failed | 70/2 | 179/10 |
| 1 free well | Failed | 37/2 | 35/0 |

## 5.7 Exploiting Problem Structure

Even though the branch and bound algorithm is effective in solving MINLP problems, it still has to explore a large number of nodes to find integer solutions. In order to help the solver, problem structure can be exploited to reach good integer solutions quickly.

Table 5.6: Testing for robustness with respect to starting-points

| | Basic | Reduced | Convex |
|---|---|---|---|
| Strictly feasible point | OK | OK | OK |
| Feasible point | Failed | Failed | OK |
| Infeasible | Failed | Failed | OK |

### 5.7.1 Search Direction

Because each well can only be routed to one pipeline, the problem is left with a special structure which could be exploited to quickly find integer solutions. To demonstrate start with a simple version of the routing constraint (5.13):

$$b_1 + b_2 \leq 1$$

In BB $b_1$ and $b_2$ will be handled as continuous variables in the root node. When the branching is performed on either one of the binary variables they will be locked to either one or zero. But if $b_1 = 1$ $b_2$ must be zero. On the other hand when $b_1 = 0$ $b_2$ can take any value less than one. In summary by locking binary variables to one, the problem structure implicitly locks two binary variables, which makes it more efficient to search in this direction as integer solutions are higher up in the search tree.

### 5.7.2 Heuristics

To quickly find a good feasible solution close to the optimum, heuristics can be employed before BB is run. This would allow BB to quickly cut away large portions of the feasible region. Two things can be assumed by an optimal solution: First it is suboptimal to close all wells, by taking this statement to its extreme it can be claimed that all wells should be open. Second the wells should be spread evenly across the pipelines in an effort to minimize the flow rates through each pipeline thus minimizing pressure loss due to friction. A simple heuristics for using these principles is stated below.

**Heuristics:**

$b_{\mathcal{J},\mathcal{L}} \leftarrow 0$ *Initialize routing*
for $j \in \mathcal{J}$:
| for $l \in \mathcal{L}$:
| | if $\sum_{j\in\mathcal{J}} b_{j,l} \leq \min_{l\in\mathcal{L}}\{\sum_{j\in\mathcal{J}} b_{j,l}\}$:
| | | $b_{j,l} \leftarrow 1$
| | | break for
| | end if
| end for $l$
end for $j$
**return** $b_{\mathcal{J},\mathcal{L}}$ *Return routing*

## 5.8 Extension

In a commercial application a certain flexibility is required of the software. This is to allow users to explore different solutions and uphold demands set by maintenance or HSE issues. In this section a short description of some such scenarios are represented.

### 5.8.1 Target Production Rate

Some fields have objectives to meet target production rates instead of maximum production, often due to long term drainage strategies. This can easily be implemented by specifying the target production rate $q_{oil}^{Target}$ and adding the constraint

$$\sum_{l\in\mathcal{L}} q_{l,oil}^{Pipe} \leq q_{oil}^{Target} \tag{5.38}$$

Or by changing the objective function to

$$\min \left( \sum_{l\in\mathcal{L}} q_{l,oil}^{Pipe} - q_{oil}^{Target} \right)^2 \tag{5.39}$$

(5.38) is probably preferred as this does not introduce more nonlinearities to the problem.

### 5.8.2 Gas lift

Gas lift is a common way to increase production rates, and it is important that a production optimization scheme can handle this. The simplest way is to add a term $q_j^{Lift}$ to the well flow:

$$q_{j,gas}^{Well} = r_j^{GOR} q_{j,oil}^{Well} + q_j^{Lift} \quad \forall j \in \mathcal{J} \tag{5.40}$$

Other fluids which are common to inject can be handled in the same manner. For example water injection used to reduce viscosity and anti-freeze agents used to prevent the formation of hydrates. It should also be possible to formulate the problem to optimize on gas lift.

### 5.8.3 Specific Routing

This is perhaps most easily handled. It simply implies to put extra constraints on the binary variables and is very straight forward.

### 5.8.4 Compressors and Pumps

Other subsea equipment could be added to the formulation by simple pressure gains or by including more approximated models. This is outside the scope of this master thesis but there is nothing in the formulations given that prohibits this type of equipment to be included.

### 5.8.5 Daisy Chains

To provide accurate field descriptions the formulation must be able to handle daisy chains and well clusters where several manifolds are connected in series to reach the topside processing facilities.

# Chapter 6

# Solution Methodology

In [9] the models were updated after each complete run of the MINLP problem. This is very easy to implement when a MINLP solver, such as BONMIN is available. The greatest disadvantage of this approach is that by not ensuring a good model fit before choosing to lock well routing, branching might exclude parts of the problem since the model fit there is poor. An example of what can happen is shown in figure 6.1, here a model fit was performed prior to a changed routing. While the simulators clearly shows that this is a feasible routing the models does not create a feasible area, recall that $W_j \geq P_l + P_{Sep}$.

A solution to this problem is to make sure the models fit the simulator before branching. This can be achieved by updating the models for each NLP problem. This should give better robustness. An added benefit is that once an integer solution has been found it is guaranteed to be a valid solution for the master problem and if the algorithm has to be aborted this solution could be implemented. As there are very many NLP problems to be solved for each MINLP problem, the number of simulator calls and model updates are expected to be a lot higher for this approach.
For simplicity the two approaches will be referred to as the MINLP and NLP approach. The basic differences should be clear by studying figure 6.2. Complete flowcharts for both approaches are given in appendix A.3. To separate

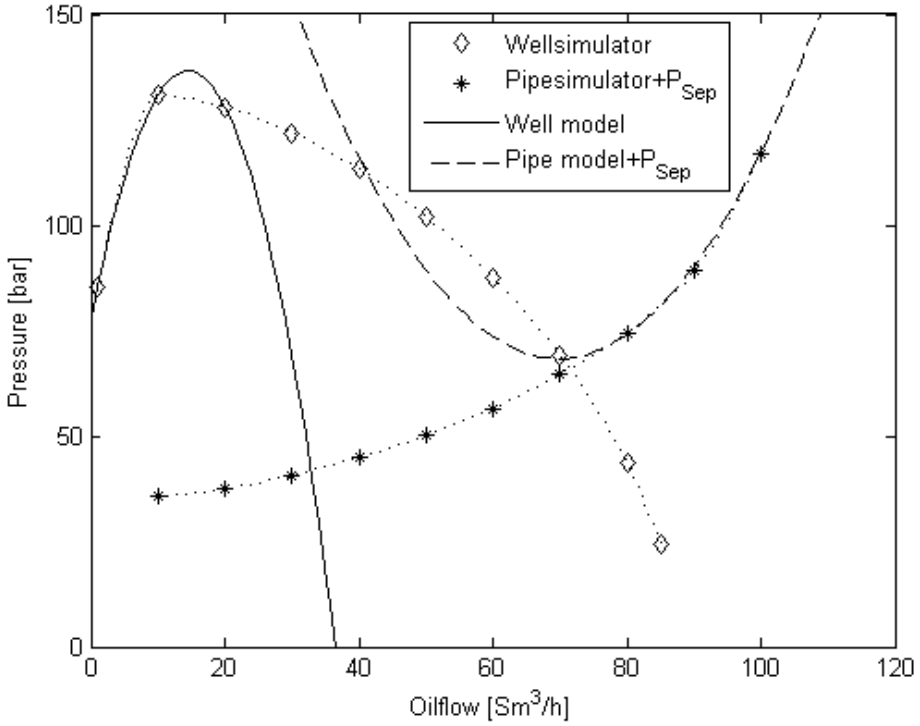



Figure 6.1: An example of what can happen when the routing is changed after a model fit is performed. The master problem has a feasible solution but the models make the subproblem infeasible. (Recall that the well pressure must be larger than the pressure drop in the pipeline.)

the MIP and the NLP part of a BB method a simple BB program was written in python, the NLP problems were solved by IPOpt. The convex formulation was implemented in AMPL with a connection to python to update the models, add constraints and bounds on big-M. The model updates are performed by updating all the models which does not meet a predefined tolerance, this is different from what was done in [9] where only the worst model was updated. Both approaches were solved with the same solvers, tolerances and options using the convex formulation from section 5.4. A complete overview over implementation specifics can be found in appendix A.

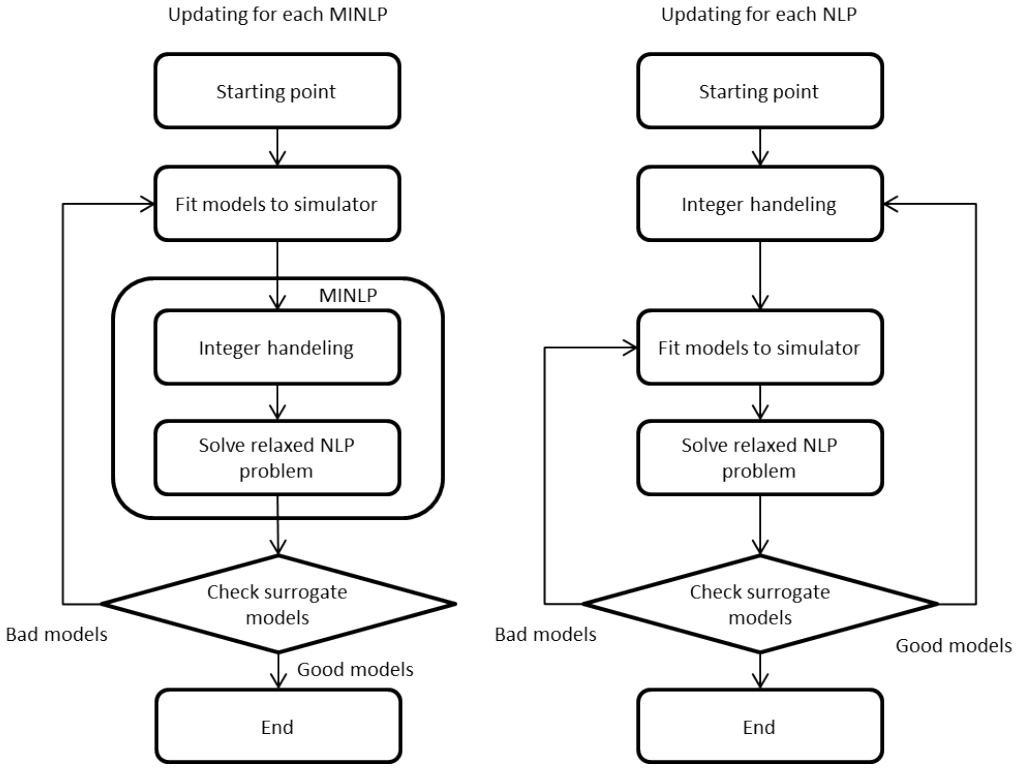
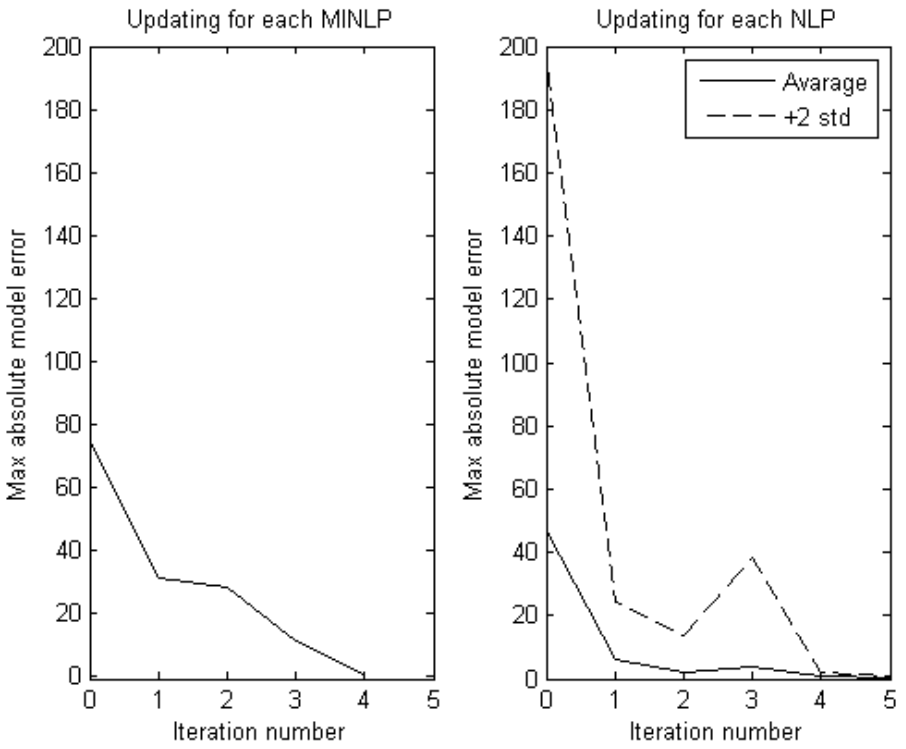



Figure 6.2: Implementation strategy for the two approaches

## 6.1 Preliminary Results

Both approaches successfully found the global optimum for the 3 well case. As predicted, updating the models for each NLP uses many more simulations, a total of 1674 simulations were used for the NLP approach while only 73 was used for the MINLP approach. Heuristics could help bring down the number of simulator calls. It is important to note that simulations in this respect mean component simulations; one pipeline or one well, not the entire system which consists of several pipelines and wells. The greatest model error in each iteration is shown in figure 6.3. The data for the NLP approach is represented as an average and standard deviation. It is not entirely fair to compare the two cases as the MINLP approach only contains one sample. But by assuming this is valid for other cases as well, the models for the NLP approach converges faster on average, although some large errors occur these




s

Figure 6.3: Model errors, the y axis represent the maximum absolute error for all the surrogate models. For the NLP approach the data is displayed using statistics, where 97.8% of the nodes are below the +2 std line.

drop quickly. The search tree generated by the BB method for the NLP approach is shown in figure 6.4, the grey nodes represent integer solutions for the master problem. The special problem structure becomes clearer here as nodes can be found fairly high up in the tree at the bottom, while a deeper search is required elsewhere in the tree. The maximum tree depth is 6 levels and a total of 41 out of a total of 63 nodes were searched. The MINLP approach encountered several problems when trying to solve the NLP problem in each node, among them infeasibility and exceeding maximum number of interior-point iterations.

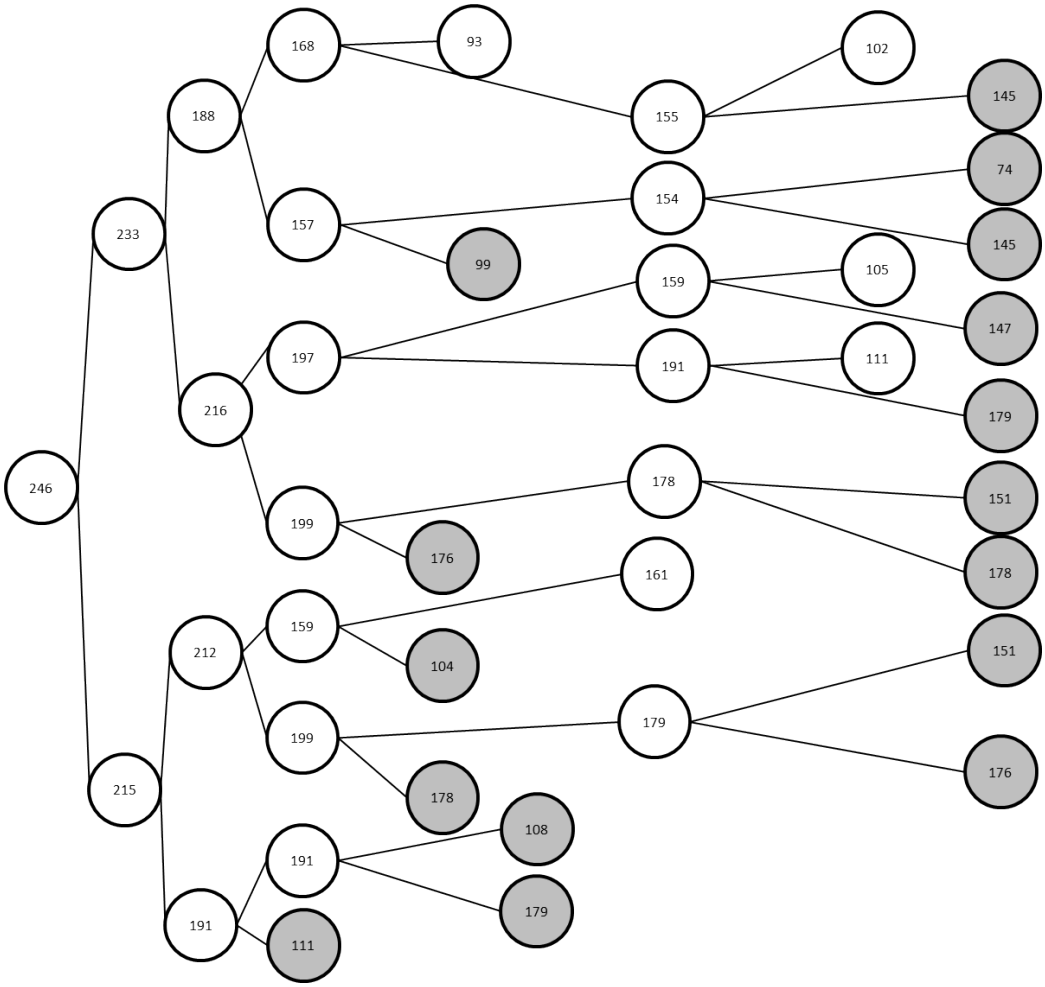



Figure 6.4: Search tree from the branch and bound algorithm. The grey nodes are integer solutions. The numbers are the optimal value in $[m^3/h]$ found in each subproblem when the model fit is within tolerances

# Chapter 7

# Performance and Robustness

To test both approaches for performance and robustness they are tested with multiple wells and with non-convex simulators, creating a non-convex master problem. Updating the models for each NLP is the most robust method, but due to the high number of simulator calls the NLP method was also tested with heuristics in an attempt to bring down the number of calls.

## 7.1 Oscillations

For the NLP approach model oscillations where detected in some of the nodes. The model updates oscillated between two or more solutions. This poses a major problem as oscillations in general are extremely hard to detect. The algorithm was written in a way which allows the algorithm to continue branching when the maximum number of model updates has been performed. The oscillations did not occur for any integer solutions, and the local solution found in the NLP problem was closer to the optimum than the start point in all cases. The oscillations were only observed for the pipeline model, and only with one free integer variable, the others were either implicitly or explicitly locked. It is unknown why this pattern emerges, but it is noted that the pipeline flow is extremely sensitive to changes in routing.

## 7.2 Non-Convex Simulators

In the problems up to now the simulators have created a convex-hull for the master problem. This is not true in general, in many cases the simulators form non-convex hulls, this could be a change in flow-regime, numerical errors due to discretization of PDEs or round-off errors.
A relatively large non-convexity was added to the well simulators by manually inserting a data point in a region which would not change the global optimum, the resulting simulator data for well 1 can be seen in figure 7.1. This was done for all three wells. The NLP approach successfully found the global optimum, even when some model updates were performed on non-convex data, this was detected by checking for linear models. It demonstrates that the NLP approach shows some robustness to non-convex simulators. The MINLP approach failed to converge to a solution altogether, the pipeline models oscillated between two different models and failed to reduce the model error further. It is important to note that this was not caused by the non-convex simulators directly, as no linear models were detected. However this shows that both approaches are prone to model oscillations.

## 7.3 Multiple Wells

The extra wells where simply added by copying the three existing wells so that well 1, 4 and 7 are equal the same goes for well 2, 5 and 8 and well 3, 6 and 9. Although this is an artificial way of including more wells it should test the robustness of the algorithm for bigger problems.

### 7.3.1 Heuristics

In section 5.7 two methods where proposed to decrease the number of nodes which has to be explored, prioritizing search direction and a simple heuristics scheme. By looking at the tree in figure 6.4, the effect of the two methods can be assessed. As the two pipelines are equal only three different combinations

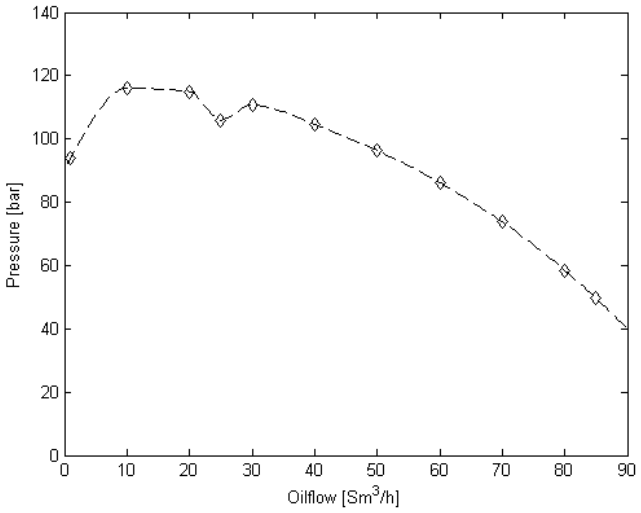



Figure 7.1: Non-convex well simulator for well 1, the non-convexity was added manually

are candidates for the heuristic: Two wells routed to the same pipeline, and one well to the other. The optimal solutions to these routings are 179, 178 and 176 $m^3/h$, or 100%, 99% and 98% of the optimal solution. It is not possible for the prioritized search method to find better bounds than these. The heuristics is also far easier to implement and therefore the prioritized search has not been considered.

### 7.3.2 Results

The NLP approach was solved both with and without heuristics to test the effect it had on the number of simulator calls. In all cases the algorithm successfully terminated with the model error within tolerances. The results are listed in table 7.1. Iterations in this context means the number of times the models where updated, and several models can be updated at once. The number of nodes searched is given as an average for each iteration for the MINLP approach to be comparable to the NLP approach. The improvement gained by applying the heuristic is 39% for 3 wells, and 48% and 19% for 6

and 9 wells respectively, calculated as the average reduction in the number of nodes and simulator calls used.

Table 7.1: Solving with multiple wells and heuristics, H = heuristics

| Number of wells | 3 | | | 6 | | | 9 | | |
|---|---|---|---|---|---|---|---|---|---|
| Approach | MINLP | NLP | NLP w/H | MINLP | NLP | NLP w/H | MINLP | NLP | NLP w/H |
| Optimal value $[m^3/h]$ | 179 | 179 | 179 | 220 | 222 | 222 | 231 | 232 | 232 |
| Iterations | 5 | 198 | 112 | 6 | 4700 | 2460 | 11 | 47642 | 38609 |
| Well simulations | 33 | 690 | 411 | 60 | 30071 | 15725 | 115 | 465465 | 375318 |
| Pipeline simulations | 44 | 984 | 566 | 54 | 26734 | 14628 | 64 | 239056 | 193154 |
| Nodes searched | 27 | 41 | 27 | 419 | 687 | 345 | 4546 | 7753 | 6257 |
| % of tree explored | 21% | 32% | 21% | 5.1% | 8.4% | 4.2% | 0.9% | 1.5% | 1.2% |

# Chapter 8

# Discussion

This chapter aims to summarize the key discussion points presented throughout this thesis. The case studied in this thesis is very simple, and offers a minimum of complexity for optimization purposes. This allows easy analyses as the total number of possible solutions and dimensions can be explored manually. By dividing the field into its basic components, wells, pipelines and manifolds, and treating each competent as a black-box, the simulators can be treated individually. The relations between each component can then be modeled explicitly by linear constraints and the solver can exploit the structure directly. This means that each component can be treated individually and there is no need to simulate the entire system, just the separate components. This formulation is very well suited for parallelization of the simulators and could give significant speed-ups compared to other methods. Both these properties should make this hybrid derivative free optimization technique superior to normal derivative free methods, a comparison of different optimization methods is outside the scope of this thesis and is left for further work. Three different problem formulations were developed to describe the case; a basic, reduced and convex formulation, each with different properties. The basic formulation is easy to understand and set up but yields a non-convex optimization problem. The reduced formulation has a reduced number of variables and constraints and is convex for a fixed routing. It

offers some computational advantage over the convex formulation but does not offer sufficient robustness or convex properties, such as bounding in BB. The convex formulation guarantees a global optimum for each subproblem, which lets us abstract away this part of the optimization as any error which occurs during optimization will happen due to errors originating in the model update or infeasibility of the master problem.

The formulations includes tight bounds on big-Ms, the value for each big-M can be calculated during runtime and does not rely on user-set values which might cause the problem to be poorly scaled. This is an advantage since poorly scaled problems can cause problem infeasibility and numerical instability. An added bonus is less tuning and reduced knowledge of optimization is required of the user which reduces training time, reduces uncertainty and can make the users more confident in the results. The convexity of the reduced and convex formulations was made possible by the method described in section 4.3. The best fit is ensured by solving a convex optimization problem, which results in a convex or concave model, depending on what is needed to form a convex feasible set for the subproblem. The method was shown to provide some robustness to noise and local non-convexities by including more points in the model fit. The number of points used in each update was kept at the minimum for simplicity as the simulators formed smooth curves. But as oscillations where observed for the pipeline models for the NLP and the MINLP approach, this is clearly not a robust way to update the models. The pipeline flow is very sensitive to changes in routing as a small change in the binary variables will cause large changes in pipeline flow. This could mean that the pipeline models must be fitted over a larger area and possibly with more points, to ensure a smaller error when changes in the routing occur.

The MINLP approach to updating the models, where an entire BB tree is searched between each model update, fails to find the optimal value when more wells are introduced. This is most likely because some parts of the feasible area were never explored due to a large model error in this area. This effect was illustrated in figure 6.1. It is also possible to see some of this

effect reflected in table 7.1 where the average number of nodes searched for each tree is different from the number of nodes explored by the NLP approach without heuristics. This means that MINLP approach has searched the tree differently and therefore does not explore the same routing combinations. Further for three wells there where three candidates for an optimal solution ranging at 100%, 99% and 98% of the optimal value, so the effect of having an optimization tool is likely to be no more than a couple percent, and by finding a solution at 99% for 6 wells and 99.5% for 9 wells. It is clear that the MINLP approach does not offer sufficient gains or robustness to be used as an optimization tool.

Updating the models for each node in the BB tree is more robust as a perfect fit is ensured with the simulators before any parts of the feasible area is excluded. As stated earlier this approach also has the benefit that all integer solutions will be valid solutions for the master problem, so if the algorithm is aborted during runtime the best solution found so far could be implemented, this makes this approach more suited for time sensitive applications. The excessive number of simulations required has to be reduced if it is to be used in commercial applications. The total number of simulations for 9 wells was over 840 000, if each simulation took an average of 0.01 seconds the total runtime of the simulations alone would exceed two hours, which makes time sensitive applications severely limited. The heuristics was very successful by nearly halving the number of simulations needed in the 6 well case. The number of simulations needed is closely related to the number of nodes explored, so an efficient branch and bound algorithm capable of exploiting the problem structure more efficiently could also bring down the total number of simulations. Storing results from the simulations in tables which could be used instead of calling the simulators would probably be most effective. For the wells a total of 20-100 points, the number of points used by Gunnerud in [4], would be able to capture all relevant dynamics, and the total number of well simulations needed in the 9 well case could be reduced from 375 318 to 180-900, a very significant reduction.

# Chapter 9

# Conclusion and Further Work

The main contribution of this thesis is the formulation of a convex optimization problem, made possible by solving a separate optimization problem to fit the quadratic models to the simulators. Updating the polynomial approximations for each MINLP problem was shown to lack robustness and failed to find the optimal point for larger problems. Updating the models for each node in the BB tree and resolving the NLP problem until the model error was below a predefined value proved more robust and was able to find better optimal values than the MINLP approach. By updating the models iteratively, as was done in [9], it was hoped that the number of simulations could be reduced compared to the method used by Gunnerud [4], where an excessive amount of simulations were needed to build piecewise linear constraints. But the number of simulations was still excessive, especially for larger problems. A simple heuristics proved very effective by nearly halving the number of simulations and the number of NLP problems solved. The number of simulations could be brought down considerably by storing results from each simulation and by exploiting the problem structure more efficiently.

Model oscillations were detected for both approaches to updating models; the models oscillated between two or more solutions and failed to bring the model error within tolerances. Model oscillations were only observed for the

pipeline models. The pipeline flow is known to be very sensitive to changes in routing and a new approach to selecting the number of points, and which points to tabulate the simulators with, might be needed to solve this problem.

## 9.1 Further Work

The author would not recommend any further work on the MINLP approach to updating the models as it was found to be unsuitable to use in an optimization tool for production chains. The convex formulation should be expanded to support daisy-chains and other types of processing equipment while still remaining convex. To solve the problem with model oscillations it would be interesting to use some sort of merit function or filter to use together with the model update strategy to pick points or weights in each update. The number of simulations must be brought down considerably before this method can be used every day by the oil and gas industry. The heuristic was very effective and improved heuristics and better branching strategies should be tested together with storing values from the simulators to bring down the number of simulator calls. New implementations might want to use an SQP solver to take advantage of 'warm-start' capabilities.

Trust-region is used in many other types of optimization algorithms and could help to increase the robustness by limiting the maximum model error in each step. The size of the trust region could be calculated from the quadratic error found in the objective function after updating each model.

The convergence of the model updates is not proved in any sense, but it should be of academic interest to develop one. Likewise a termination proof should be of interest. An informal termination proof was given in this thesis, but formal poofs would legitimate this simulator based optimization method as a viable optimization technique for production chains.

To validate this method it should be tested against other derivative free methods preferably against commercial production optimization tools already in use by the oil and gas industry.

# Bibliography


[1] Lieberman Hiller. *Introduction to operations research, 9th edition.* Mc Graw Hill, 2010.

[2] V Gunnerud A Conn, B Foss. Embedding structural information in simulation-based optimization. unpublished.

[3] Stephen J Wright Jorge Nocedal. *Numerical Optimization.* Springer, 2006.

[4] Bjarne Foss Vidar Gunnerud. Oil production optimization - a piecewise linear model, solved with two decomposition strategies. computer and chemical engineering (2009).

[5] PIPESIM. Schlumberger software http://www.slb.com/services/software/production_software/prod_analysis_diagnostics/pipesim.aspx 25.05.12.

[6] ECPLIPSE. Schlumberger software http://www.slb.com/services/software/reseng.aspx 25.05.12.

[7] LedaFlow. Kongserg oil and gas technologies http://www.kongsberg.com/en/kogt/offerings/software/ledaflow/ 25.05.12.

[8] Flow Manager. Fmc technologies software http://www.fmctechnologies.com/en/subseasystems/technologies/

advancingtechnologies/productionmonitoringopt/flowmanagement.aspx 25.05.12.

[9] Bjarne Grimstad Håvard Ausen, Victoria Lervik. *Optimization of a Simulated Well Cluster using Surrogate Models.* IFAC Workshop on Automatic Control in Offshore Oil and Gas Production NTNU May 31 - June 1, 2012.

[10] J M Maciejowski. *Predictive control with constraints.* Pearson Prentice Hall, 2002.

[11] Floudas. *Nonlinear and Mixed Integer Optimization.* Oxford University Press, 1995.

[12] Stephen Boyd. *Convex Optimization.* Cambridge University Press, 2004.

[13] R Mead J A Nelder. A simplex method for function minimization. comput. j.,7,1965.

[14] Wiliam H Press et al. *Numerical Recipies. The Art of Scientific Computing, 3rd edition.* Cambridge University Press, 2007.

[15] AMPL. A modeling language for mathematical programming
http://www.ampl.com/
02.02.12.

[16] COIN-OR. Computational infrastructure for operations rsearch,
http://www.coinor.org/
02.02.2012, 2012.

# Appendix A

# Implementation

When implementing an algorithm many issues tend to arise. This chapter will try to clarify said issues in and a way that the results presented in this thesis can be reproduced.

## A.1 Problem Formulation

To make the problem simpler to work with a series of assumptions was performed. All capacity constraints where eliminated, thus the problem is pressure constrained, which is common for mature fields.
For simplicity it is also assumed that there is no water present in the system. Interestingly this gives the same mathematical formulation as a WC of 50% or $r_j^{WC} = 1$. This can be seen by setting $q_{oil} = q_{water}$. As was done in [9].

## A.2 Updating Surrogate Models

The simulators where stored as tables during the optimization. This leads to a discrete simulator. To check if a model was within tolerances a simple interpolation was performed between neighboring points. This interpolated value was used in the equality constraint for the model update method given in section 4.3.

## A.3 Algorithm

Although BONMIN is an MINLP solver it lacks the ability to divide the MIP and NLP layers. Therefore a simple branch and bound algorithm was written in python. The NLP problem was solved using the same NLP solver as BONMIN; IPOpt. It uses a breadth first search to explore the tree. For the MINLP approach, the model updates are performed after the branch and bound algorithm has found an optimal node, a flowchart is given in figure A.1. In contrast the NLP approach makes sure the models fit the simulator before branching, this can be seen in the flowchart given in figure A.2. If the heuristics is applied it is applied before the root problem is solved. Other types of tree searches could be used by manipulating the "Get next node from queue" block in the flow chart. The python implementation requires the python libraries NumPy, SciPy and matplotlib to be installed on the system.

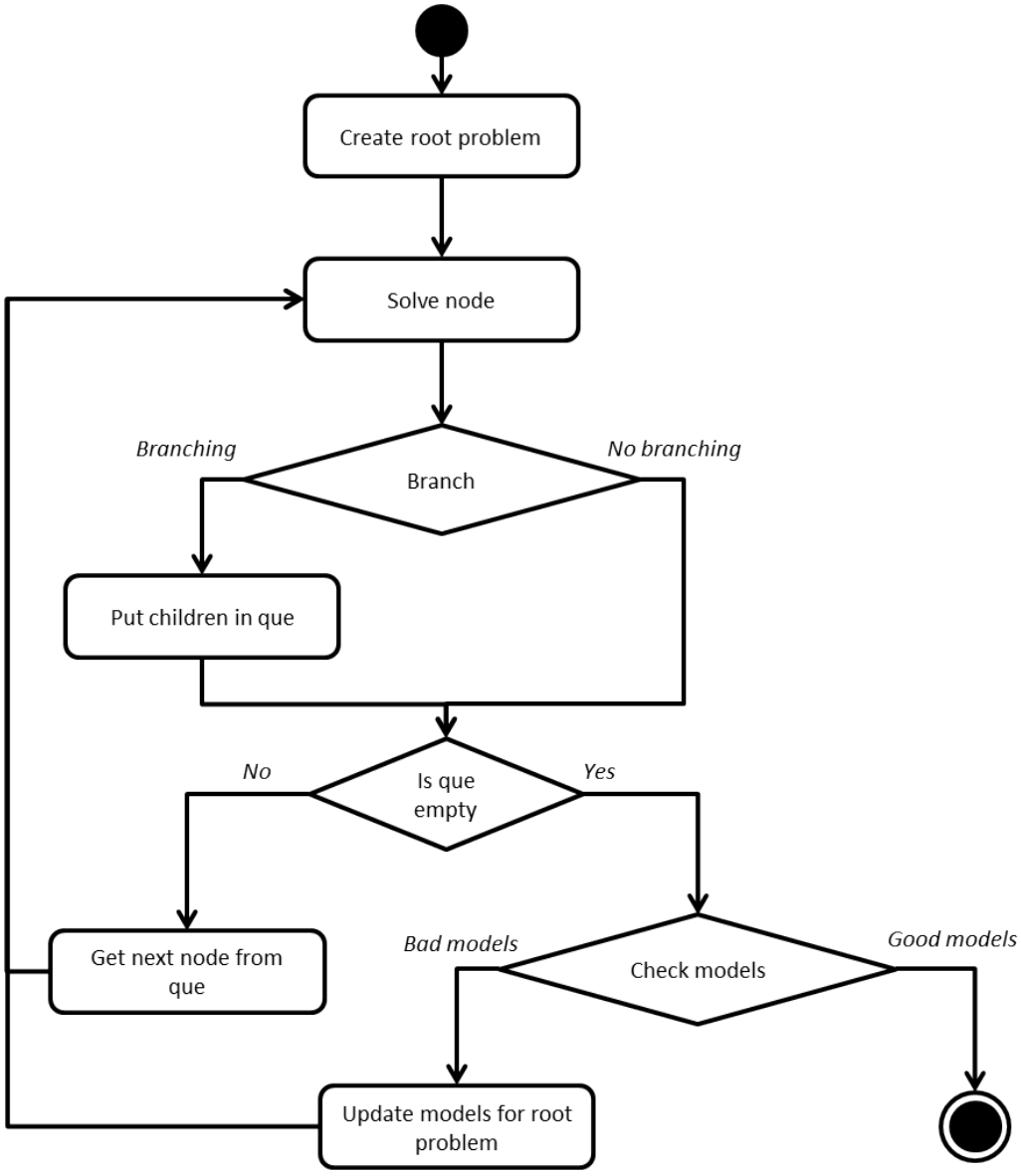



Figure A.1: Flowchart for the MINLP approach

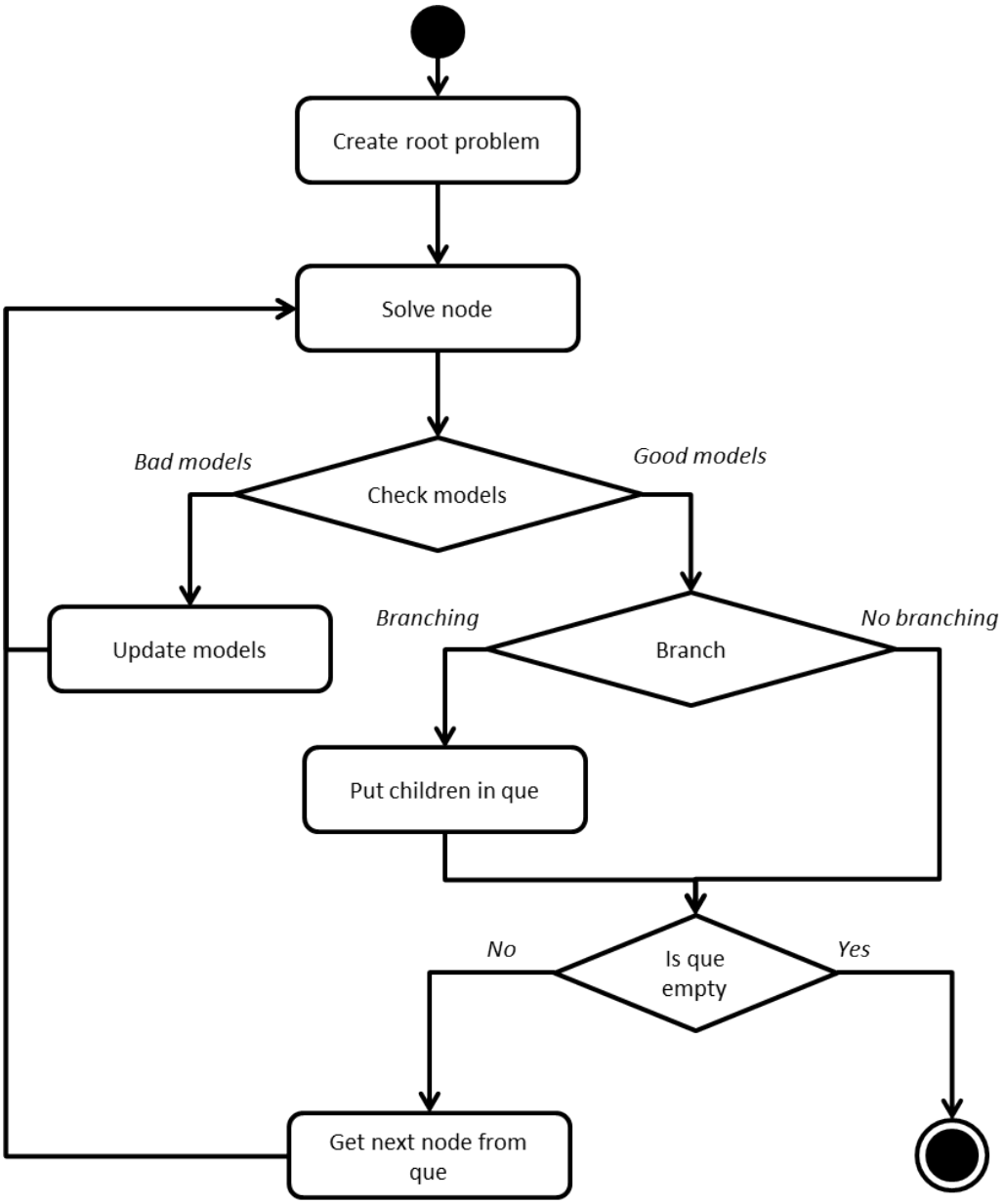



Figure A.2: Flowchart for the NLP approach

## A.4 Tolerances and Options

An overview over tolerances and constants is given in table A.1. The options passed to BONMIN and IPOpt are listed in table A.2. Apart from these the default settings where used. A list of possible options can be found on the COIN-OR website [16].

Table A.1: Tolerances and constants

| | Value |
|---|---|
| Maximum number of model updates | 20 |
| Integer tolerance | $10^{-3}$ |
| Model tolerance | 1 [bar] |
| $P_{Sep}$ | 20 [bar] |

Table A.2: Solver options

| Option passed | Value |
|---|---|
| nlp_scaling_method | none |
| hessian_approximation | limited-memory |
| bonmin.algorithm | B-BB |

# Appendix B

# Folder Structure

The code used in this thesis is organized on the beehive.ad.itk.ntnu.no computer located at NTNU.

## B.1 Different Formulations

The different formulations are found in */media/STORAGE/master_thesis_ausen/different_formualtions*. The formulations can be executed by typing *ampl name.run* in the terminal, for example *ampl basic.run* runs the basic formulation.

| Name | Path |
|---|---|
| Basic | basic.run |
| Reduced | reduced.run |
| Convex | convex.run |

## B.2 MINLP Optimization

The problems are organized in the structure given in table B.1 and can be found in the folder */media/STORAGE/master_thesis_ausen/MINLP_optimization*. Each folder contains the structure given in table B.2. In addition to these the folder *MINLP/model_fit* contains a script which only performs a model

updates at user specified points. The folder *MINLP/static* is a pure BB method without model updates. The classes and functions can be found in the *MINLP/classes_and_functions* folder.

Table B.1:

| Folder name | Description |
|---|---|
| MINLP/dyn | Base case with 3 wells |
| MINLP/multiple_wells/6wells | Case with 6 wells |
| MINLP/multiple_wells/9wells | Case with 9 wells |
| MINLP/non_convex | Robustness test with non-convex well simulators |

To turn the heuristics on or off, uncomment or comment in line 76 in *bb_NLP.py*.

Table B.2:

| File or folder name | Description |
|---|---|
| figures | figures from model updates |
| simulator | simulator data stored as *.csv |
| tab | contains *.tab files used for start and optimal points |
| tab/backup | backup of tab folder |
| tab/startpoint | *.tab files for the startpoint |
| 2pipe3well.dat | *.dat file used to specify the problem size |
| AMPLout.txt | text file generated by bb_MINLP.py or bb_NLP.py |
| bb_MINLP.py | solves the problem using the MINLP approach |
| bb_NLP.py | solves the problem using the NLP approach |
| convex.mod | the mathematical formualtion of the optimization problem |
| convex.run | specifies how the NLP problem should be executed and which files to include |
| extra_binary_const.mod | extra constraint on the binary variables added by BB |
| extra_param.dat | contains values for $\alpha_j$, $\beta_l$, $M_p^q$ and $M^P$ |
| output\ | output file generated by bb_NLP.py |
| pipe_opt.run | specifies the optimization problem needed to fit pipeline models |
| well_opt.run | specifies the optimization problem needed to fit well models |