Solving Refinery-Planning Problems

By

Arjan Buijssen [S309471]
[B.Sc. Tilburg University 2008]

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Operations Research and Management Science

Faculty of Economics and Business Administration
Tilburg University

Supervisor university: Dr. G. Gurkan
Supervisor ORTEC: Drs. W.J. Lukje
Date: January 2008
Preface

At the end of my study Econometrics and Operations Research at Tilburg University, I had to write a Master’s thesis on the area of my Master’s specialization Operations Research and Management Science. Since I did not have an internship in the first three years of my study, I really preferred to write a practical thesis based on an internship at a well-established company in the area of the Operations Research. ORTEC gave me the possibility for doing this at the Shell department.

I would like to use this opportunity to thank Drs. Sandra Lukje, the supervisor from ORTEC, for all her support and involvement during the internship. There were a lot of other people from ORTEC who have also helped me with advices, information, and comments. Thank you all for helping me writing this thesis!

During my internship my supervisor from the university was Dr. Gul Gurkan. I would really like to thank her for helping me writing this thesis by correcting and reading my thesis, and for pushing me in the right directions. Also special thanks go out to Prof. Dr. Ir. Dick den Hertog for introducing me to ORTEC and for being my second assessor.

I would also like to thank my parents Riny and Henk for supporting me my whole life and giving me the opportunity to study and to develop myself in the way I want to.

Last but certainly not least I want to thank my sister Janneke as well, for supporting me during the whole process.

Arjan Buijsse
Gouda, January 2008
Management Summary

In the refinery industry nonlinear programming (NLP) and linear programming (LP) problems have to be solved in all different segments of the supply chain. One sort of problem, which has to be solved for achieving a maximum profit, is the planning problem at the refinery.

For optimizing these problems the optimal combination of crude oils has to be purchased, blended, and processed for selling the desired end products with a maximum profit. There are a lot of crude oils available on the market, which all have different properties (e.g. octane number, sulphur percentage, or viscosity) and prices. The end products (such as diesel and gasoline), which will be sold to the customers, have to satisfy certain specifications and demands.

Due to specific parts of the optimization process, these planning problems are nonlinear. To solve these problems a software package called PIMS, specifically developed for refinery problems (by AspenTech), is used by Shell. ORTEC has been supporting Shell for more than 20 years by developing a user-interface for PIMS (called PlanStar), building the NLP models, and giving advices on these areas.

Since Shell has started optimizing their refinery-planning problems with PIMS, the software has run in a mode called Distributive Recursion. In this mode the nonlinear programming problems are approximated by linear programming problems. Since a couple of years a new solving mode has been available in PIMS, which is called Extended Nonlinear Programming. This mode has a different manner of solving the nonlinear programming problems (by using a nonlinear solver). Besides this it generates additional features to analyse the obtained solutions afterwards.

These additional features are useful when one wants to test the flexibility of the crude oils, to define a secondary goal (besides the primal goal of maximizing the profit), or to analyse small changes in the objective value by marginally adjusting the prices.

One of the additional features can also be used in combination with a software package called Marginal Analysis Tool (developed by ORTEC and owned by Shell). One of the functionalities of this tool, which can be improved by using one of the additional features of XNLP, is to address uncertainty in the crude oil availability. It can also be utilized as a strategic tool for trading. However, according to this thesis it does not improve the functionality of the main goal of the Marginal Analysis Tool, which is to evaluate whether there are opportunities to increase the refinery margin by purchasing other crude oils than usual. Small adaptations in the feature in PIMS could also make it more useful for this purpose.

A small case study also shows us that for these models XNLP does not obtain better results than DR. In most of the cases the objective values are approximately equal and the
computation times to achieve the solutions are, especially for large refinery models, larger in the XNLP mode.

Hence in this thesis the mathematics and the functionality of the new nonlinear solution method XNLP available in PIMS have been investigated.
Table of Contents

Preface
Management Summary
Table of Contents

1 Introduction ......................................................................................................................... 3
   1.1 The petroleum industry ................................................................................................. 3
   1.2 ORTEC and Shell ......................................................................................................... 4
   1.3 Problem formulation .................................................................................................... 4
       1.3.1 Optimization problems in refineries ................................................................. 4
       1.3.2 Methods of solving nonlinear problems ......................................................... 5
       1.3.3 Main Goal .......................................................................................................... 7
       1.3.4 The approach .................................................................................................... 7

2 Distributive Recursion ...................................................................................................... 9
   2.1 Introduction .................................................................................................................. 9
   2.2 Pooling problem ......................................................................................................... 9
       2.2.1 Introduction (Example 1) .............................................................................. 9
       2.2.2 Distributive Recursion applied on pooling problem (Example 2) ............... 13
   2.3 Delta Based Modeling ............................................................................................... 18
       2.3.1 Introduction ................................................................................................... 18
       2.3.2 Distributive Recursion applied on Delta Based Modeling (Example 3) ....... 19

3 Extended Nonlinear Programming .................................................................................. 29
   3.1 Introduction ................................................................................................................ 29
   3.2 The CONOPT solver ................................................................................................. 30
       3.2.1 GRG method .................................................................................................. 30
       3.2.2 GRG method applied on an NLP problem (Example 4) ......................... 34
   3.3 The XSLP solver ....................................................................................................... 37

4 Other packages for solving nonlinear programs ........................................................... 39
   4.1 Introduction ................................................................................................................ 39
   4.2 The SNOPT solver ................................................................................................... 39
   4.3 The MINOS solver .................................................................................................. 41
   4.4 Comparison of CONOPT, XSLP, SNOPT, and MINOS ..................................... 42

5 Additional output in the XNLP mode .......................................................................... 45
   5.1 Introduction ................................................................................................................ 45
   5.2 Ranging Analysis ....................................................................................................... 45
       5.2.1 Introduction .................................................................................................. 45
       5.2.2 Ranging Analysis applied on a small demo model (Example 5) ............... 47
   5.3 Parametric Analysis .................................................................................................. 49
       5.3.1 Introduction .................................................................................................. 49
5.3.2 Parametric Analysis applied on a small demo model (Example 6) .................... 49
5.4 Goal Programming .......................................................................................................... 51
  5.4.1 Introduction ........................................................................................................ 51
  5.4.2 Goal Programming applied on a small demo model (Example 7) ...................... 51

6 Case comparison ........................................................................................................... 53
  6.1 Introduction ........................................................................................................ 53
  6.2 Case study ......................................................................................................... 53
  6.3 Conclusions .................................................................................................. 54

7 Ranging Analysis in MAT ......................................................................................... 57
  7.1 Introduction .......................................................................................................... 57
  7.2 Current method: Cargo Analysis ......................................................................... 58
  7.3 Differences between both methods ....................................................................... 60
  7.4 Using Ranging Analysis to improve MAT ......................................................... 61
    7.4.1 Algorithm ...................................................................................................... 61
    7.4.2 Example of Ranging Analysis in MAT (Example 7) ..................................... 63
  7.5 Comparison of the current and new method ...................................................... 65
    7.5.1 Comparison of both methods (Example 8) ................................................... 65
    7.5.2 Conclusions and motivation ....................................................................... 67
    7.5.3 Recommendations ...................................................................................... 67

8 Conclusions and recommendations .............................................................................. 70
  8.1 Conclusions ........................................................................................................ 70
  8.2 Recommendations & further research ............................................................. 71
    8.2.1 Recommendations .................................................................................... 71
    8.2.2 Further Research ..................................................................................... 72

References ....................................................................................................................... 73

Appendix A – Continuation of Example 2 ................................................................. 75
Appendix B – Continuation of Example 4 ................................................................. 78
Appendix C – Summary statistics of Example 5 .................................................... 82
Appendix D – Refineries for case study .................................................................... 83
Appendix E – Outcomes of Example 7 .................................................................... 85
Appendix F – Current implementation .................................................................... 88
Appendix G – Outcomes of Example 8 .................................................................... 99
1 Introduction

1.1 The petroleum industry

Oil can be defined as any substance that could be liquid that is in a viscous liquid state at ambient temperatures or slightly warmer, and is immiscible with water and miscible with other oils. It is one of the main energy resources in the world and therefore a good planned process is of great significance. Before the oil reaches its final destination, for instance the filling station, it has had a long process. Such a process is shown in Figure 1.1.

Figure 1.1 Flow diagram of the oil process

At the beginning of the process the crude oil is explored and produced at the oil fields. This will be shipped to a refinery where it will be processed, stored, and blended into different end products, such as Diesel and Premium 98. These products are then shipped (or transported through a pipeline) to a depot, from where it will be distributed to the retail, e.g. filling stations.

In this thesis, the emphasis will be placed on the refinery part of the process. A refinery can be defined as a complex of factories, pipelines, and tanks that transform the crude oil into many useful products like LPG, Diesel, and Gasoline. At such a refinery, many essential decisions have to be taken, in such a way that the correct end products will be produced with a maximum total profit.

In Section 1.2 the role of ORTEC and Shell in this process will be discussed and in Section 1.3 the problem of this thesis will be described and formulated.
1.2 ORTEC and Shell

ORTEC is one of the largest providers of advanced planning and optimization solutions. Founded in 1981, ORTEC currently has more than 500 employees and offices in North America and Europe. Their blue-chip customer base of more than 800 customers includes leading enterprises in manufacturing, transportation, industry and banking. ORTEC is a privately held company that develops, markets, and implements advanced planning and scheduling software, according to http(1).

Most of the employees work at the office in Gouda. One of the business units at this office is ‘ORTEC Logistics Consulting’, in which for small regional companies as well as multinational corporations the logistic operations and processes are improved. With the aid of the experts at ORTEC, unique opportunities can be identified to improve their logistics network, according to http(6). The largest department in this business unit is fully arranged for the support of Shell, a worldwide group of oil, gas, and petrochemical companies. This thesis is written on the basis of an internship at this department. At this department nearly the complete supply chain is optimized with the aid of ORTEC. The part of the supply chain discussed in this thesis is the planning process from the crude oil selection until the selling process at the gate of the refinery (from crude until blending in Figure 1.1), without taking the transportation of the products to the customers into account.

One of the projects in the Shell department is the development of software packages for Shell to help optimizing their crude oil selection, the process of refining, and blending possibilities at the refineries. A more detailed description of the problems that occur at a refinery and the corresponding optimization methods is stated in Section 1.3.

1.3 Problem formulation

1.3.1 Optimization problems in refineries

An optimization problem is a computational problem in which the goal is to find the best of all possible solutions. More formally, the objective is to find a solution in the feasible region, which has the optimal value of the objective function. Such problems occur everywhere and therefore also in the petroleum world. The complete supply chain of the petroleum industry will not be discussed in this thesis, but only the part at the refineries. As explained above, a petroleum refinery is a complex of factories, pipelines, and tanks that transforms raw materials (crude oil) into many useful end products. There are many different crude oils, which all have various properties and molecule structures. When a certain crude oil enters the refinery, it will be processed in several processing units. These units transform the crude oil into intermediates, called components. After these processes the intermediates will be blended in the right proportion to obtain the desired products, specified by the customer. To get an idea of all these processes in a refinery, an example of a flow diagram of a rather complex refinery is shown in Figure 1.2.
As shown in this figure, the crude oils are first processed into components like LPG, Naptha, and Kerosine, using a Crude Distilling Unit. After that, several units, such as a Hydro Treater, Platformer, and High Vacuum Unit, are used to transform these components into products with the user-preferred properties, like sulphur percentage, octane number, and density. The crude oil costs are about 90% of the refinery input cost and therefore the selection of the optimal crude oil mix is extremely important to achieve higher profits. Due to the diversity of the crude oils and the corresponding fluctuating price scenarios, it is very difficult to evaluate all possible scenarios to obtain the optimum crude oil mix for the refinery. Besides this, also the streams inside the refineries are extremely diverse. Modeling these processes therefore results in nonlinear programming problems. In Subsection 1.3.2 the methods for dealing with the nonlinearity of these problems will be discussed in more detail.

Note that also the transportation of the crude oils to the refinery is subject to a lot of uncertainty, such as cargo sizes, sea routes etcetera. However, this will be considered as given in the rest of this thesis.

1.3.2 Methods of solving nonlinear problems

There are many unknown variables in a petroleum refinery that are influencing the total profit of the refinery, such as crude oil purchases, product sales, and the division of the amounts to the different units. To obtain the optimal allocation of these variables, a mathematical representation of the process can be developed. The best approximation of such a process is a nonlinear programming (NLP) problem, since in practice several complex nonlinear parts are involved in the problem. However, nonlinear problems could be very difficult mathematical problems and are most of the time very hard or even impossible to solve.

At this moment many software packages are on the market to deal with these nonlinear problems. One of those packages is called “PIMS”, which has been developed by
AspenTech. Shell utilizes this package to solve and evaluate the planning problems at their refineries worldwide. ORTEC advises Shell on the area of quantitative optimization problems. In Figure 1.3 three programs are shown that are developed by ORTEC and owned by Shell Global Solutions, to simplify the use of PIMS and to analyse solutions obtained by PIMS.

Since PIMS is a complicated software package, PlanStar has been built as a user-interface to simplify PIMS for the users. PlanStar, developed by ORTEC, is a refinery-planning tool, containing a graphical user interface and a database that stores all the required data. It exports the data to PIMS and shows the obtained solution afterwards in a clarifying way. Hence note that PlanStar does not solve the nonlinear problem itself.

Besides PlanStar, ORTEC has also developed the two Microsoft Excel templates ‘Front-End’ (FE) and ‘Marginal Analysis Tool’ (MAT). The Front-End is used for defining and updating primarily commercial data for the monthly plan. Thus it prepares a base case in Excel, which can be imported in PlanStar, followed by the optimization step in PIMS and the import of the solution in PlanStar afterwards.

With an optimal monthly plan available in PlanStar, MAT is used for analysing the optimal solution by marginal changes of decision variables or parameters. The principle of MAT is to change only one item per case, leaving the rest of the model unchanged. MAT consists of two parts: Cargo Analysis and Marginal Steering. In Cargo Analysis the user can analyse cases with a different crude oil (or other feedstock) composition. On the other hand, in Marginal Steering cases can be defined with marginal changes in for instance unit capacities or product demands. The cases setup in MAT can be returned to PlanStar and optimized by using PIMS.

Shell is using PIMS to solve their nonlinear optimization problems at the refinery. The solution method that is used in PIMS is called Distributive Recursion (DR). Since 2003 there has been a new nonlinear method available in PIMS, called Extended Nonlinear Programming (XNLP). According to the developers, this method performs faster and gives more global optima than Distributive Recursion. Shell is questioning whether or not transforming their refinery models to be solved in the XNLP mode, since the functionality and the mathematics behind it can be seen as a black box. Therefore they want to
investigate the (dis)advantages of this method, to decide switching from the DR to the XNLP mode.

1.3.3 Main Goal
The main goal of this thesis can be formulated as follows:
To investigate the mathematics and the functionality of the new nonlinear solution method XNLP available in PIMS.

This main goal can be achieved by answering the following research questions:

- How do the two solution methods Distributive Recursion and Extended Nonlinear Programming work on mathematical level?
- How appropriate are the nonlinear solvers that are available in XNLP for solving refinery-planning problems, compared to other main nonlinear solvers?
- To what extent does XNLP provide more accurate and higher quality solutions in the refinery planning area?
- What are the additional features when using XNLP and how can they be used to improve the functionality of the Marginal Analysis Tool?

1.3.4 The approach
To fulfill the main goal of this thesis, the old method Distributive Recursion first has to be discussed extensively to get a good overview of the current situation. The method will also be applied on some small examples, which contain typical nonlinear parts of a refinery-planning model. This mathematical discussion of the Distributive Recursion is in Chapter 2.

After the overview of the current situation has been given, the mathematical background of the new method XNLP will be discussed. The user can choose between two nonlinear solvers (inside this method) to solve the problems, namely CONOPT and XSLP. They both have to be discussed to get a good overview of the new situation and to provide an answer to the first research question, which is done in Chapter 3 of this thesis.

After this mathematical discussion, the question arises why the solvers CONOPT and XSLP are available in the XNLP mode, although there are many other nonlinear solvers available on the market. It therefore has to be investigated to what extent these two nonlinear solvers are appropriate for refinery-planning problems, compared to two other nonlinear solvers SNOPT and MINOS. To provide a good answer to the second research question, this will be investigated in Chapter 4.

Since Extended Nonlinear Programming is based on a different mathematical method than Distributive Recursion, it may also generate other, additional features to analyse the obtained solution. In Chapter 5 of this thesis these features will be investigated, such that the first part of the fourth research question could be answered.

After these chapters, the basic theoretical background of both solution methods DR and XNLP is clarified. A good continuation of this thesis is now to compare both methods in a practical way by using some nonlinear programming (NLP) problems of refineries of Shell (see Chapter 6). In this chapter the third research question “To what extent does XNLP
provide more accurate and higher quality solutions in the refinery planning area?” will be investigated.

The utility of the additional features, which may be generated by using XNLP, also has to be investigated. This is done in Chapter 7, such that it will be attempt to provide a reliable answer to the second part of the fourth question “How can the additional features be used to improve the functionality of the Marginal Analysis Tool?”.
2 Distributive Recursion

2.1 Introduction

Since the beginning of the petroleum industry, companies have been searching for the best possible method for solving the (nonlinear) problems at the refineries. The main issue of nonlinear programming problems is the fact that a final solution, which has been obtained by a certain solver, in the absence of certain mathematical properties (such as convexity), may be a local optimum. Hence, a solver can distinguish itself from others by obtaining global optima more frequently as final solution.

Since 1980, Shell (one of the largest worldwide group of oil, gas, and petrochemical companies) has used Distributive Recursion in combination with the software package of AspenTech (PIMS). Distributive Recursion is a specific version of Successive Linear Programming (SLP), where the nonlinear problem is represented in PIMS as a linear problem, by presuming some of the variables as known. After solving the linear problem, the presumed value and calculated physical value are compared, which results in a certain “error”. If this error is not sufficiently small, the guessed variables are updated and a new LP is solved.

This method of solving nonlinear problems will be explained in more detail in Section 2.2 and it will be applied on the (nonlinear) pooling problem in refineries. In Section 2.3 it will be applied on a more complicated and advanced problem that occurs in refineries (Delta Based Modeling).

2.2 Pooling problem

2.2.1 Introduction (Example 1)

One of the largest problems in modeling a refinery process is the traditional “pooling” problem, since it includes the modeling of nonlinear ratio constraints and even more general nonlinearities. Pooling can be defined as “the mixing of several streams prior to subsequent use”. In Figure 2.1 a simple pooling example (Example 1) is shown. The discussion in this section is based on Aspen Technology (1998).

![Figure 2.1 Simple pooling problem in a refinery (Example 1)](image)

In this figure, two components, called FTV and TVG, are pooled together in the POOL unit, which results into a new component, called blend. The corresponding amounts of the components in barrels (bbls) are $x_1$ and $x_2$, for FTV and TVG respectively. Blending these...
two components results in an amount $x_3$ of blend, which is the sum of $x_1$ and $x_2$. Furthermore, every component in the process has certain properties, which are typical for that component, like density, sulphur percentage, and viscosity. For the sake of this example, we will focus only on the sulphur percentage ($\%S$) of the components, which is denoted by $s$ and $t$ (in $\%S$), for FTV and TVG respectively. The sulphur percentage of the blend has been restricted to a certain percentage denoted by $u$ (in $\%S$), which means that not every ratio between the amounts of the two incoming components has been allowed. In Figure 2.1 the purchasing prices of the components FTV and TVG are denoted by $c$ and $d$ (in $$/bbl) respectively. The selling price of blend is denoted by $e$ (in $$/bbl). The goal of this problem is to purchase the components FTV and TVG (determining values for $x_1$ and $x_2$) in such a way that the total profit has been maximized and the sulphur constraint does not violate. To illustrate this example, the values for the parameters are shown in Table 2.1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c$</td>
<td>$2$</td>
</tr>
<tr>
<td>$d$</td>
<td>$3$</td>
</tr>
<tr>
<td>$e$</td>
<td>$4$</td>
</tr>
<tr>
<td>$s$</td>
<td>1.5 $%S$</td>
</tr>
<tr>
<td>$t$</td>
<td>0.5 $%S$</td>
</tr>
<tr>
<td>$u$</td>
<td>1 $%S$</td>
</tr>
</tbody>
</table>

Table 2.1 Numerical parameter values (Example 1)

Note that the problem does not take into account properties such as the density of the different components, since such properties would make the example more complicated and thus less comprehensible.

The objective of this problem is to maximize the total profit, thus it has the following objective function:

$$\max 4x_3 - 2x_1 - 3x_2 .$$

Let us assume that the refinery is not allowed to buy more than 200 barrels of each component. Since every component has different properties (in this case: sulphur percentages), the ratio between these quantities is essential for the outcomes of the problem. However, the ratio in this mixture is not known in advance. The optimal ratio, which maximizes the total profit, is restricted by quantity and quality constraints. As the name implies, quantity constraints are restrictions on the quantities of the variables $x_1$, $x_2$, and $x_3$. These constraints are in this example defined as:

$$0 \leq x_1 \leq 200$$
$$0 \leq x_2 \leq 200$$
$$x_1 + x_2 = x_3 .$$

The quality constraints concern some specifications of the components, for instance the sulphur content or the Research Octane Number (RON). In this case, the blend has the
restriction that it is not allowed to contain more than 1 %S sulphur. Hence the quality constraint can be defined as:

\[
\frac{3/2 x_1 + 1/2 x_2}{x_1 + x_2} \leq 1.
\]

Now the total problem can be rewritten into the following LP problem:

\[
\begin{align*}
\max & \quad 2x_1 + x_2 \\
\text{s.t.} & \quad x_1 \leq x_2 \\
& \quad 0 \leq x_1 \leq 200 \\
& \quad 0 \leq x_2 \leq 200.
\end{align*}
\]

To see this, note that the third constraint of (2.1) can be used to simplify the objective function. As one can see, the described problem is linear and therefore solvable with a linear solver. A simple calculation gives us an objective function value of 800 with corresponding values \( x_1 = 200 \) and \( x_2 = 200 \) (maximum allowable values).

However, when the problem of Figure 2.1 is enlarged, a nonlinear constraint will arise. The enlarged problem (Example 2) is shown in Figure 2.2.

Figure 2.2 Enlarged pooling problem in a refinery (Example 2)

The notation that has been used in Figure 2.2 is stated in Table 2.2.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c_1 )</td>
<td>Cost SRS ($/bbl)</td>
</tr>
<tr>
<td>( c_2 )</td>
<td>Cost FTV ($/bbl)</td>
</tr>
<tr>
<td>( c_3 )</td>
<td>Cost TVG ($/bbl)</td>
</tr>
<tr>
<td>( p_a )</td>
<td>Price blend a ($/bbl)</td>
</tr>
<tr>
<td>( p_b )</td>
<td>Price blend b ($/bbl)</td>
</tr>
<tr>
<td>( q_1 )</td>
<td>Sulphur content SRS (%S)</td>
</tr>
<tr>
<td>( q_2 )</td>
<td>Sulphur content FTV (%S)</td>
</tr>
<tr>
<td>( q_3 )</td>
<td>Sulphur content TVG (%S)</td>
</tr>
<tr>
<td>( q_a )</td>
<td>Sulphur content blend a (%S)</td>
</tr>
</tbody>
</table>
In this case, all costs, prices, qualities (sulphur percentages) of the components, and bounds are parameters and thus given. The decision variables to be optimized in this nonlinear problem are basically $x_1$, $x_2$, $x_3$, $x_4$, and $x_5$. Again, these variables have to be chosen in such a way that the total profit will be maximized.

Using Figure 2.2 and Table 2.2, the following mathematical model of the optimization problem can be developed:

$$
\text{max } \quad p_a (x_1 + x_4) + p_b x_5 - c_1 x_1 - c_2 x_2 - c_3 x_3
$$

$$
(1a) \quad \frac{q_1 x_1 + q_b x_4}{x_1 + x_4} \leq a
$$

$$
(1b) \quad \frac{q_b x_5}{x_5} \leq b
$$

$$
(2) \quad \frac{q_2 x_2 + q_3 x_3}{x_2 + x_3} = q_b
$$

$$
(3) \quad x_2 + x_3 = x_4 + x_5
$$

$$
(4a) \quad 0 \leq x_1 \leq u_1
$$

$$
(4b) \quad 0 \leq x_2 \leq u_2
$$

$$
(4c) \quad 0 \leq x_3 \leq u_3
$$

$$
(4d) \quad x_1 + x_4 \geq l_a
$$

$$
(4e) \quad x_5 \geq l_b.
$$

In this model the total profit is once more the difference between the revenue and the total costs restricted to quality and quantity constraints, as shown in (2.2). The constraints (1a) and (1b) are the quality constraints of the two blends; constraint (2) is the quality

<table>
<thead>
<tr>
<th>$q_b$</th>
<th>Sulphur content blend b (%S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>Upperbound quality blend a</td>
</tr>
<tr>
<td>$b$</td>
<td>Upperbound quality blend b</td>
</tr>
<tr>
<td>$u_1$</td>
<td>Upperbound quantity SRS (bbls)</td>
</tr>
<tr>
<td>$u_2$</td>
<td>Upperbound quantity FTV (bbls)</td>
</tr>
<tr>
<td>$u_3$</td>
<td>Upperbound quantity TVG (bbls)</td>
</tr>
<tr>
<td>$l_a$</td>
<td>Lowerbound quantity blend a (bbls)</td>
</tr>
<tr>
<td>$l_b$</td>
<td>Lowerbound quantity blend b (bbls)</td>
</tr>
<tr>
<td>$x_1$</td>
<td>Quantity SRS (bbls)</td>
</tr>
<tr>
<td>$x_2$</td>
<td>Quantity FTV (bbls)</td>
</tr>
<tr>
<td>$x_3$</td>
<td>Quantity TVG (bbls)</td>
</tr>
<tr>
<td>$x_4$</td>
<td>Quantity of POOL directed to blend a (bbls)</td>
</tr>
<tr>
<td>$x_5$</td>
<td>Quantity of POOL directed to blend b (bbls)</td>
</tr>
</tbody>
</table>

Table 2.2 Notation table of the pooling problem (Example 2)
constraint of the pool; constraint (3) is the quantity constraint of the pool; and constraints (4a)-(4e) are the other quantity constraints of this model. To show that this model is nonlinear, it can be simplified into:

\[
\begin{align*}
\max \ (p_a - c_1) x_1 - c_2 x_2 - c_3 x_3 + p_a x_4 + p_b x_5 \\
qu_1 x_1 + \frac{q_2 x_2 + q_3 x_3}{x_2 + x_3} x_4 & \leq a(x_1 + x_4) \\
q_2 x_2 + q_3 x_3 & \leq b(x_2 + x_3) \\
x_2 + x_3 & = x_4 + x_5 \\
0 & \leq x_i \leq u_i \\
0 & \leq x_j \leq u_j \\
0 & \leq x_k \leq u_k \\
x_i + x_j & \geq l_a \\
x_k & \geq l_b.
\end{align*}
\]

(2.3)

To see this, note that constraint (2) of model (2.2) has been substituted into constraint (1a). As one can see, the first constraint of model (2.3) is a nonlinear one and therefore it can be concluded that this pooling problem is a nonlinear one and has to be solved by using nonlinear solving techniques.

2.2.2 Distributive Recursion applied on pooling problem (Example 2)

A well-known tool for solving these nonlinear pooling problems is Distributive Recursion (DR), explained by Aspen Technology (1998). Distributive Recursion is a nonlinear technique used in PIMS to model nonlinearities by approximating them with linear segments, which are presumed in advance. Then an “LP matrix” is updated after every recursion. The LP matrix is a matrix where every row denotes a constraint and every column a variable. The updated LP matrix is considered to give a sufficiently good approximation of the nonlinear model when the differences between the presumed and the real values of the variables are within predefined tolerances.

To illustrate this, consider again the pooling problem described in Figure 2.2 (Example 2). Assuming there are no fixed production costs or blending costs, the nonlinear problem (2.3) with the following values of the parameters in Table 2.2 will be used for this example.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_1$</td>
<td>$1$</td>
</tr>
<tr>
<td>$c_2$</td>
<td>$2$</td>
</tr>
<tr>
<td>$c_3$</td>
<td>$3$</td>
</tr>
<tr>
<td>$p_a$</td>
<td>$2$</td>
</tr>
<tr>
<td>$p_b$</td>
<td>$4$</td>
</tr>
<tr>
<td>$l_1$</td>
<td>3.5%</td>
</tr>
</tbody>
</table>
Table 2.3 Numerical parameter values for the pooling problem (Example 2)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_2$</td>
<td>1.5%</td>
</tr>
<tr>
<td>$q_3$</td>
<td>0.5%</td>
</tr>
<tr>
<td>$a$</td>
<td>2</td>
</tr>
<tr>
<td>$b$</td>
<td>1</td>
</tr>
<tr>
<td>$u_1$</td>
<td>200</td>
</tr>
<tr>
<td>$u_2$</td>
<td>200</td>
</tr>
<tr>
<td>$u_3$</td>
<td>200</td>
</tr>
<tr>
<td>$l_a$</td>
<td>100</td>
</tr>
<tr>
<td>$l_b$</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 2.4 Presumed parameter values for recursion one (Example 2)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_b$</td>
<td>1.25</td>
</tr>
<tr>
<td>$d_e$</td>
<td>0.5</td>
</tr>
</tbody>
</table>

As explained earlier, one of the methods to solve a nonlinear problem is Distributive Recursion. The basic idea of Distributive Recursion is to transform this nonlinear problem into a linear problem by guessing some values in advance.

Usually, the first step of this method is to estimate the amount of the pool(s) that will go to the different, in this case two directions. Let us denote the fraction that goes to blend $a$ by $d_e$ and let us presume that 50% of the amount in $POOL$ will go to both blends (so in the default case $d_e = 0.50$ and thus $x_q = x_3$). Since we do not know the total crude composition, we do not know the composition of the pool. However, by guessing the content of sulphur in the pool, the model becomes linear and the ratio between $x_2$ and $x_3$ becomes known. Let us initially guess that the $POOL$ blend contains 1.25 %S sulphur. As shown in Figure 2.2, this equals the sulphur percentage of blend $b$, since the pool and blend $b$ are identical. Thus we first presume the following parameter values as known, although they are actually unknown (see Figure 2.2):

Note that the value of $q_b$ in Table 2.4 does not result in a feasible solution, since blend $b$ is not allowed to contain more than 1 %S sulphur. However, we will see that making such an infeasible guess will still take us to the optimal solution. In general, solving a strictly convex problem will always result in a unique global solution. When a problem is not convex, there is an opportunity of obtaining a local optimum. Hence, a good guess is then extremely important, whereas in this case even a meaningless guess will take us to the global optimum. By presuming this sulphur percentage of $q_b$ as known, an error term is created and should be introduced to the model. This error term is divided between the different directions of the pool by the fraction $d_e$, thus it is equally divided between the two directions. The error term is denoted by $x_e$ and constraints (1a), (1b), and (2) of model (2.2) become:
This error term is necessary for calculating the real sulphur percentage of the pool ($q_b$).

Now the linear problem to be solved by using model (2.2) and assuming the parameter values and guesses discussed above, becomes:

$$
\max x_1 - 2x_2 - 3x_3 + 2x_4 + 4x_5 \\
(1a) \quad 3.5x_1 + 1.25x_4 + 0.5x_5 \leq 2(x_1 + x_4) \\
(1b) \quad 1.25x_5 + 0.5x_5 \leq x_5 \\
(2) \quad 1.5x_2 + 0.5x_3 = 1.25(x_2 + x_3) + x_5 \\
(3) \quad x_2 + x_3 = x_4 + x_5 \\
(4a) \quad x_1 + x_4 \geq 100 \\
(4b) \quad 0 \leq x_1 \leq 200 \\
(4c) \quad 0 \leq x_2 \leq 200 \\
(4d) \quad 0 \leq x_3 \leq 200 \\
(4e) \quad x_5 \geq 100 \\
(5) \quad x_\varepsilon \in \mathbb{R}.
$$

This model can be rewritten into matrix form (with constraints on the rows and variables on the columns) and solved using a linear solver (like Xpress or CPLEX), in which $x_1, x_2, x_3, x_4, x_5,$ and $x_\varepsilon$ are the decision variables. It results in the solution shown in Table 2.5.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Activity (barrels per day)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>61.53</td>
</tr>
<tr>
<td>$x_2$</td>
<td>92.31</td>
</tr>
<tr>
<td>$x_3$</td>
<td>200</td>
</tr>
<tr>
<td>$x_4$</td>
<td>38.47</td>
</tr>
<tr>
<td>$x_5$</td>
<td>253.86</td>
</tr>
<tr>
<td>$x_\varepsilon$</td>
<td>-126.92</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Optimal Value ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objective</td>
</tr>
</tbody>
</table>

Table 2.5 Optimal solution of recursion one (Example 2)

First note that the value of the variable is usually denoted by ‘the activity’ of the variable. Also note that this first solution is in most of the cases not directly the optimal solution of the pooling problem. This is due to the guesses and the nonlinearity of the original problem.
The value of the error term $x_e$ is relevant for calculating the sulphur percentage of the blends in the next recursion. This term compensates the initial guess of the sulphur percentage and is distributed between the different blends according to the distribution guesses (in this case 50% for both blends). Thus using Table 2.5, the sulphur percentages of the two blends become:

Blend a: $q_1 x_1 + q_4 x_4 + d_e x_e \left/ \right. x_1 + x_4 = \frac{3.5 \times 61.53 + 1.25 \times 38.47 - 0.5 \times 126.92}{100} = 2\% S$

Blend b: $q_3 x_5 + \left(1 - d_e\right) x_e \left/ \right. x_5 = \frac{1.25 \times 253.86 - 0.5 \times 126.92}{253.86} = 1\% S$

Using these values one can conclude that both blends contain their maximum allowable sulphur percentage (see model (2.4)).

Now the new values of the guessed variables for recursion two have to be determined (when a second recursion happens to be necessary, as explained below). With the composition of POOL from 92.31 bbls/day FTV ($= x_2$) and 200 bbls/day TVG ($= x_3$) the calculated sulphur percentage of the pool equals

$$\frac{q_2 x_2 + q_3 x_3}{x_2 + x_3} = \frac{1.5 \times 92.31 + 0.5 \times 200}{92.31 + 200} = 0.816\% S.$$ 

(2.5)

To see this, note that the weighted mean of the sulphur content has been calculated in (2.5). Hence the new guessed value for $q_b$ will be $q_b = 0.816\% S$. One can also calculate the new fractions of POOL that goes to both blends:

Blend a: $x_4 \left/ \right. x_2 + x_3 = \frac{38.47}{92.31 + 200} \times 100\% = 13.2\%$

(2.6)

Blend b: $100 - 13.2\% = 86.8\%$

To see this, note that $x_2 + x_3$ equals the total amount in the pool and $x_4$ the amount of the pool directed to blend a. Hence the real distribution value, given these parameter values, should be $d_e = 0.132$.

However, these new guessed values are not necessary when the current solution happens to be a sufficiently good approximation of the real optimal solution. To examine whether this current solution is a good approximation, two different tolerances have been defined, called the Absolute ($ATol$) and Relative Tolerance ($RTol$). They are defined as:

$$ATol = P_C - P_P$$

$$RTol = \left(\frac{P_C - P_P}{P_C}\right) \times 100$$
where $P_c$ denotes the current property value and $P_p$ the previous property value. Usually, the default value for each tolerance is 0.001. In our example, we only focus on the sulphur property, so only for that property these tolerances can be calculated. After recursion one we obtain the following values for $ATol$ and $RTol$:

$$ATol = \left| 0.816 - 1.25 \right| = 0.4342 > 0.001$$

$$RTol = \frac{0.816 - 1.25}{1.25} = 0.3474 > 0.001.$$  

Since neither the $ATol$ nor the $RTol$ is less than the allowed tolerance, a new recursion has to be executed, with the updated values of the non-converged property values. According to (2.5) and (2.6), the new guess of the sulphur percentage in the pool will become 0.816 (instead of 1.25) and the new proportion of the direction of $POOL$ will become 13.2% to blend $a$ and 86.8% to blend $b$ (instead of both 50%). Thus model (2.4) will be adjusted by presuming the following values as known:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_b$</td>
<td>0.816</td>
</tr>
<tr>
<td>$d_e$</td>
<td>0.132</td>
</tr>
</tbody>
</table>

Table 2.6 Presumed parameter values for recursion two (Example 2)

With these values, the new linear programming problem can be solved similarly as describe above. This will result in new activities for all the decision variables, a new value of $d_e$, and new actual sulphur percentages for both blends. These values can be tested again, using to the tolerances described above. If necessary, more recursions can be run until a sufficiently good approximation of the nonlinear programming problem has been obtained. The elaboration of these recursions is similar to the first one and therefore stated in Appendix A.

After three recursions, the obtained solution proves to be a sufficiently good approximation of the NLP problem. The corresponding outcomes are stated in Table 2.7.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Activity (barrels per day)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>40</td>
</tr>
<tr>
<td>$x_2$</td>
<td>200</td>
</tr>
<tr>
<td>$x_3$</td>
<td>200</td>
</tr>
<tr>
<td>$x_4$</td>
<td>60</td>
</tr>
<tr>
<td>$x_5$</td>
<td>340</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Objective</th>
<th>Optimal Value (dollar)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>520</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Guessed values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_b$</td>
</tr>
<tr>
<td>$d_e$</td>
</tr>
</tbody>
</table>

Table 2.7 Optimal solution of the pooling problem, determined with Distributive Recursion (Example 2)
In the next section this example will be enlarged with a so-called “base + delta”-unit, which makes Distributive Recursion a bit more complicated.

2.3 Delta Based Modeling

2.3.1 Introduction

Another complex part of the process at the refineries is addressed by Delta Based Modeling (DBM). DBM is a linear programming technique used for predicting yields of processing units and their corresponding properties, in situations where these yields and properties are a function of feed quality and therefore unknown in advance. Notice once more that the methods explained in this thesis are applied on static models, to plan the purchases and sells of a certain period (e.g. month) for a petroleum refinery. DBM is usually applied on models with a large amount of streams, since it reduces the complexity of the model. However, this complexity reduction causes a loss in information about the use of particular streams in the refinery. We next explain DBM in more detail, based on Aspen Technology (1998).

Delta Based Modeling is applied on a so called “base + delta”-unit. The main difference between a pooling unit (as explained in Section 2.2) and a “base + delta”-unit, is the process inside the unit. In the former unit, a linear process takes place, which results in linear equations to calculate the qualities (like sulphur percentages) and quantities of the outgoing stream. Though note that the complete model is still nonlinear (as described in Section 2.2). On the other hand, in a “base + delta”-unit the qualities of the outgoing streams cannot be calculated by using linear equations, since several chemical reactions take place inside. For instance, the total sulphur of the incoming stream does not have to equal the total sulphur of the outgoing stream, in contrast with the model discussed in Section 2.2. Such a unit is usually preceded (or fed) by a pool of streams, i.e. fed by a component with an unknown composition, which has to be determined by the optimization process (e.g. stream $x_i$ in Example 2). Hence the properties (and yields) of the feed of the unit are unknown for the user. As a consequence, the properties and yields of the outgoing streams of the unit will also be unknown in advance, which makes it more complicated to model. It implies that these processing units have to be modeled in such a way that they can cope with these unknown properties and yields. Therefore, Delta Based Modeling combined with Distributive Recursion is applied to predict the yields and properties of these outlet streams, by approximating the real process by a linearization of the data. This concept is illustrated in Figure 2.3.
In Figure 2.3 the property value (e.g. sulphur percentage) of a certain incoming stream is shown on the x-axis. On the y-axis the corresponding yield of the outgoing stream is shown when it has been processed in the “base + delta”-unit. The real data for a certain property is unknown. However, assuming there is appropriate empirical data, a regression line of this data can be drawn, which will be used to approximate the real data. This regression line is the linearized data as shown in the figure.

As one can see, the linearized data is an approximation of the real data. The user must be satisfied that the slopes of the curves used in this method reflect a linear relationship between the yield and the property value and that this relationship adequately models the real world for each specific DBM application. Since the modeling and solving of a huge amount of streams could be very complex, DBM pre-estimates some unknown variables such that the model can be solved by using Distributive Recursion. Thus in addition to Example 2 (see Section 2.2), more variables will be presumed as known. In Subsection 2.3.2 this will be shown by using an extension of Example 2.

### 2.3.2 Distributive Recursion applied on Delta Based Modeling (Example 3)

To apply the Distributive Recursion method together with Delta Based Modeling, an extension of Example 2 will be used. This extension, Example 3, is shown in Figure 2.4.

In contrast with Example 2, one of the two directions downstream the pool cannot be sold directly to the customer, but is progressed to a “base + delta”-unit. Inside this unit a certain chemical process takes place. The details of the exact process are beyond the scope of this
thesis. An example of such a unit is a ‘Hydro-Desulphurizer unit’, where the pool stream \(x_5\) (with a high sulphur percentage) is converted into fuels with lower sulphur percentages \((x_6\) and \(x_7)\). As a consequence of this new unit, blend \(b\) is not sold anymore. Instead, two new end products DIESEL and GASOLINE have been added and can be produced and sold now. Note that these products also have quantity and quality (sulphur percentage) constraints. The notation in Figure 2.4 is comparable with those of Figure 2.2. For simplicity, the parameters with corresponding values and the decision variables are shown in Table 2.8.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(c_1)</td>
<td>Cost SRS ($/bbl)</td>
<td>$1</td>
</tr>
<tr>
<td>(c_2)</td>
<td>Cost FTV ($/bbl)</td>
<td>$2</td>
</tr>
<tr>
<td>(c_3)</td>
<td>Cost TVG ($/bbl)</td>
<td>$3</td>
</tr>
<tr>
<td>(p_a)</td>
<td>Price blend (a) ($/bbl)</td>
<td>$2</td>
</tr>
<tr>
<td>(p_D)</td>
<td>Price DIESEL ($/bbl)</td>
<td>$6</td>
</tr>
<tr>
<td>(p_G)</td>
<td>Price GASOLINE ($/bbl)</td>
<td>$5</td>
</tr>
<tr>
<td>(q_1)</td>
<td>Sulphur content SRS (%S)</td>
<td>3.5 %S</td>
</tr>
<tr>
<td>(q_2)</td>
<td>Sulphur content FTV (%S)</td>
<td>1.5 %S</td>
</tr>
<tr>
<td>(q_3)</td>
<td>Sulphur content TVG (%S)</td>
<td>0.5 %S</td>
</tr>
<tr>
<td>(q_p)</td>
<td>Sulphur content outgoing stream of the pool (thus of (x_4) and (x_5)) (%S)</td>
<td>Decision variable</td>
</tr>
<tr>
<td>(q_D)</td>
<td>Sulphur content DIESEL (%S)</td>
<td>Decision variable</td>
</tr>
<tr>
<td>(q_G)</td>
<td>Sulphur content GASOLINE (%S)</td>
<td>Decision variable</td>
</tr>
<tr>
<td>(u_1)</td>
<td>Upperbound quantity SRS (bbls)</td>
<td>200</td>
</tr>
<tr>
<td>(u_2)</td>
<td>Upperbound quantity FTV (bbls)</td>
<td>200</td>
</tr>
<tr>
<td>(u_3)</td>
<td>Upperbound quantity TVG (bbls)</td>
<td>200</td>
</tr>
<tr>
<td>(a)</td>
<td>Upperbound quality blend (a) (%S)</td>
<td>2 %S</td>
</tr>
<tr>
<td>(b)</td>
<td>Upperbound quality DIESEL (%S)</td>
<td>0.4 %S</td>
</tr>
<tr>
<td>(c)</td>
<td>Upperbound quality GASOLINE (%S)</td>
<td>1.5 %S</td>
</tr>
<tr>
<td>(l_a)</td>
<td>Lowerbound quantity blend (a) (bbls)</td>
<td>100</td>
</tr>
<tr>
<td>(l_D)</td>
<td>Lowerbound quantity DIESEL (bbls)</td>
<td>0</td>
</tr>
<tr>
<td>(l_G)</td>
<td>Lowerbound quantity GASOLINE (bbls)</td>
<td>0</td>
</tr>
<tr>
<td>(x_4)</td>
<td>Quantity out of POOL directed to blend (a) (bbls)</td>
<td>Decision variable</td>
</tr>
<tr>
<td>(x_5)</td>
<td>Quantity out of POOL directed to UNIT (bbls)</td>
<td>Decision variable</td>
</tr>
<tr>
<td>(x_6)</td>
<td>Quantity out of POOL directed to DIESEL (bbls)</td>
<td>Decision variable</td>
</tr>
<tr>
<td>(x_7)</td>
<td>Quantity out of POOL directed to GASOLINE (bbls)</td>
<td>Decision variable</td>
</tr>
</tbody>
</table>

*Table 2.8 Notation table (Example 3)*
This model can be represented as a nonlinear programming problem, where the total profit has to be maximized without violating the quality and quantity constraints. It results in the following NLP problem:

$$\max \quad p_G(x_1 + x_2) + p_D x_6 + p_G x_7 - c_1 x_1 - c_2 x_2 - c_3 x_3$$

(1a) \quad \frac{q_1 x_1 + q_P x_4}{x_1 + x_4} \leq a

(1b) \quad \frac{q_D x_6}{x_6} \leq b

(1c) \quad \frac{q_G x_7}{x_7} \leq c

(2) \quad \frac{q_2 x_2 + q_3 x_3}{x_2 + x_3} = q_P

(3a) \quad x_3 = x_4 + x_5

(3b) \quad x_5 = x_6 + x_7

(4a) \quad q_D = f(q_P, x_6)

(4b) \quad q_G = g(q_P, x_7)

(5a) \quad 0 \leq x_1 \leq u_1

(5b) \quad 0 \leq x_2 \leq u_2

(5c) \quad 0 \leq x_3 \leq u_3

(5d) \quad x_1 + x_4 \geq l_a

(5e) \quad x_6 \geq l_D

(5f) \quad x_7 \geq l_G,$$

where $f$ and $g$ are unknown, nonlinear functions. In comparison with the model of Example 2 (model (2.4)), some constraints have been changed and some new ones have been defined. To clarify these modifications, all constraints are briefly discussed below:

- **Constraints (1a) – (1c):**
  These constraints are the quality (sulphur) constraints of the different products, i.e. for every product there is a maximum percentage of sulphur allowed. Quality constraint (1a) is the one for blend $a$, (1b) for DIESEL, and (1c) for GASOLINE.

- **Constraint (2):**
  This constraint is the quality constraint of the pool unit. It can be interpreted as the condition that the weighted sulphur percentage of the pool has to equal the sulphur percentage of the outgoing stream (which is a decision variable).

- **Constraints (3a) – (3b):**
  These are the quantity constraints of the units POOL and UNIT respectively. In these constraints it is stated that the total weight of the incoming stream has to equal the total weight of the outgoing stream.

- **Constraints (4a) – (4b):**
These two constraints are the quality constraints for the two end products DIESEL and GASOLINE. In these constraints it is stated that the sulphur percentages of DIESEL ($q_D$) and GASOLINE ($q_G$) are (nonlinear) functions of the quality of the input in the unit ($q_P$) and the amount that is directed to the product ($x_6$ or $x_7$). These functions are unknown (see the real data in Figure 2.3), due to the fact that the process inside the “base + delta”-unit is not given. Notice again that in such a unit, the total sulphur of the incoming stream does not have to be equal to the total sulphur of the outgoing stream, since it could be for instance a ‘Hydro-Desulphurizer unit’. Therefore, the function is unknown and will be linearized by using Delta Based Modeling (see the linearized data in Figure 2.3). In addition, it is known that when we estimate (or guess) the quality of the input ($q_P$), the functions $f$ and $g$ become linear. Thus $f$ and $g$ are linear in $x_6$ and $x_7$ respectively.

- Constraints (5a) – (5f):
  These quantity constraints are the sell and purchase constraints. So, for all the crude oils that can be purchased, there is the ability of choosing a maximum allowed purchase amount and for all the products a minimum demand that has to be satisfied.

In model (2.7) not only constraint (2) is nonlinear, but also constraints (4a) and (4b). To transform model (2.7) into a linear problem, some variables have to be presumed as known again. In this case (in contrast with Example 2), three variables have to be guessed to obtain an LP problem, namely the variables $q_P$, $q_D$, and $q_G$. Then constraints (2), (4a), and (4b) become linear. As a consequence, the number of error terms also increases from one to three (Table 2.9).

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{e,P}$</td>
<td>Error term due to guessing the sulphur percentage of POOL</td>
</tr>
<tr>
<td>$x_{e,D}$</td>
<td>Error term due to guessing the sulphur percentage of DIESEL</td>
</tr>
<tr>
<td>$x_{e,G}$</td>
<td>Error term due to guessing the sulphur percentage of GASOLINE</td>
</tr>
</tbody>
</table>

*Table 2.9 Notation table of the error terms (Example 3)*

As in the previous section one can start with presuming the sulphur content $q_P$ of the pool (and hence of stream $x_4$ and $x_5$). Let us presume that $q_P = 1\%S$. As a consequence, we assume an equal amount of FTV and TVG in the pool (since the sulphur percentages of these components are 1.5 %S and 0.5 %S respectively). One main difference with Example 2 is the guess of two other sulphur percentages. Not only the outgoing stream of the pooling-unit has to be presumed as known, but also the sulphur percentages of the streams out of the “base + delta”-unit, i.e. $q_D$ and $q_G$. Let us presume that $q_D = 0.4\%S$ and $q_G = 1.5\%S$. As explained above, since three sulphur percentages are presumed as known, three error terms have to be incorporated into the model. For recursion one of the Distributive Recursion algorithm, these error terms will again be equally divided between the directions. This means that, as in Chapter 2, it is presumed that the error term $x_{e,P}$ (see Table 2.9) of the guessed sulphur content of the pool will be equally divided in the first recursion, thus again it is presumed that $d_{i} = 0.5$. After the first recursion, this fraction will be adjusted by using the outcomes of the previous recursion. In addition, the error
terms $x_{e,D}$ and $x_{e,G}$ do not have to be divided into fractions, since the production of **DIESEL** and **GASOLINE** has only one outgoing stream ($x_6$ and $x_7$ respectively). The values of the four guesses are stated in Table 2.10.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_p$</td>
<td>1</td>
</tr>
<tr>
<td>$q_D$</td>
<td>0.4</td>
</tr>
<tr>
<td>$q_G$</td>
<td>1.5</td>
</tr>
<tr>
<td>$d_\varepsilon$</td>
<td>0.5</td>
</tr>
</tbody>
</table>

**Table 2.10** Guessed parameter values for recursion one (Example 3)

To apply Delta Based Modeling it is sufficient to have data of one base with corresponding delta, yields, and properties, as described in Subsection 2.3.1. A unit also has the opportunity to produce different end products, by processing in different modes. Notice that more bases and/or modes result in a more complicated model. Hence the unit in this example can produce in only one mode and one base with corresponding deltas and yields. The corresponding collected data are stated in Table 2.11.

<table>
<thead>
<tr>
<th>Feed sulphur</th>
<th>Yield</th>
<th>Outgoing sulphur</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base $B = 0.75 %$S</td>
<td><strong>DIESEL</strong> $BY_D = 65%$</td>
<td>$BP_D = 0.50 %$S</td>
</tr>
<tr>
<td></td>
<td><strong>GASOLINE</strong> $BY_G = 35%$</td>
<td>$BP_G = 1.50 %$S</td>
</tr>
</tbody>
</table>

| Delta $D = 0.05 \%$S | **DIESEL** $DY_D = -2\%$ | $DP_D = 0.05 \%$S |
|                       | **GASOLINE** $DY_G = 2\%$ | $DP_G = 0.08 \%$S |

**Table 2.11** Data of the “base + delta”-unit (Example 3)

In this table one base with corresponding yields and sulphur properties is shown. It has to be interpreted as follows: when the feed of the “base + delta”-unit has a sulphur percentage that equals the sulphur base ($B$) of 0.75 \%S, the (base) yield of the outgoing stream contains 65\% **DIESEL** ($BY_D$) and 35\% **GASOLINE** ($BY_G$) of the total weight, i.e. $x_6 = 0.65x_5$ and $x_7 = 0.35x_5$. The corresponding sulphur percentages (base property values) of these products are then 0.50 \%S ($BP_D$) and 1.50 \%S ($BP_G$), for **DIESEL** and **GASOLINE** respectively. This scenario is denoted by Scenario 1 and shown in Figure 2.5. When the sulphur percentage of the incoming stream happens to be different than the base value of 0.75 \%S, the lower part of Table 2.11 becomes relevant. Let $z \in \mathbb{R}$ denote the number of delta values ($D$) between the real sulphur percentage of the feed and the base sulphur percentage ($B$), i.e.

$$
z = \frac{q_{\text{input}} - B}{D} \quad (2.8)
$$
where \( q_{\text{input}} \) denotes the sulphur percentage of the incoming stream. The value of \( z \) is called the shift factor. If, for instance, the sulphur percentage of the input equals 0.85 \( \% \)S, then the value of \( z \) equals (by using (2.8)):

\[
z = \frac{q_{\text{input}} - B}{D} = \frac{0.85 - 0.75}{0.05} = 2.
\]

Thus to approximate the yields and sulphur percentages of this feed, the delta values have to be added two times to the base values, using the corresponding \( DY \)- and \( DP \)-columns of Table 2.11. Thus given the input contains 0.85 \( \% \)S sulphur, 61% \( (BY_D + 2 \cdot DY_D) \) of the total output stream will be \( \text{DIESEL} \), which contains a sulphur percentage of 0.60 \( \% \)S \( (BP_D + 2 \cdot DP_D) \). Similarly, 39% \( (BY_G + 2 \cdot DY_G) \) of the total output stream will be \( \text{GASOLINE} \), which contains a sulphur percentage of 1.66 \( \% \)S \( (BP_G + 2 \cdot DP_G) \). This scenario (Scenario 2) is also shown in Figure 2.5. Note that all the values of Table 2.11 are approximations and become worse when the shift factor \( (z) \) increases.

![Figure 2.5 Scenario 1 and 2 (Example 3)](image)

To incorporate this method of modeling a “base + delta”-unit to the base model (2.7), some additional constraints and variables have to be defined. The model that will be used to start the first recursion of Distributive Recursion is shown below.
max \( p_a(x_1 + x_4) + p_D x_6 + p_G x_7 - c_1 x_1 - c_2 x_2 - c_3 x_3 \)

\[ (1a) \quad q_1 x_1 + q_p x_4 + d \varepsilon x_{e,p} \leq a(x_1 + x_4) \]

\[ (1b) \quad q_D x_6 + x_{e,D} \leq b x_6 \]

\[ (1c) \quad q_G x_7 + x_{e,G} \leq c x_7 \]

\[ (2a) \quad q_2 x_2 + q_3 x_3 = q_p (x_2 + x_3) + x_{e,p} \]

\[ (2b) \quad q_p x_5 + (1 - d) x_{e,p} = B x_5 + D \delta \]

\[ (3a) \quad x_2 + x_3 = x_4 + x_5 \]

\[ (3b1) \quad BY_D x_5 = x_6 + DY_D z \]

\[ (3b2) \quad BY_G x_5 = x_7 + DY_G z \]

\[ (4a) \quad BP_D BY_D x_5 + \Delta_D z = q_D x_6 + x_{e,D} \]

\[ (4b) \quad BP_G BY_G x_5 + \Delta_G z = q_G x_7 + x_{e,G} \]

\[ (5a) \quad 0 \leq x_1 \leq u_1 \]

\[ (5b) \quad 0 \leq x_2 \leq u_2 \]

\[ (5c) \quad 0 \leq x_3 \leq u_3 \]

\[ (5d) \quad x_1 + x_4 \geq l_a \]

\[ (5e) \quad x_6 \geq l_D \]

\[ (5f) \quad x_7 \geq l_G \]

\[ (5g) \quad z, x_{e,P}, x_{e,D}, x_{e,G} \in \mathcal{R}, \]

where

\[ \Delta_D = (BY_D + DY_D)(BP_D + DP_D) - (BP_D \cdot BY_D), \]

and

\[ \Delta_G = (BY_G + DY_G)(BP_G + DP_G) - (BP_G \cdot BY_G). \]

This model is an elaborate version of model (2.7), which can be applied on Distributive Recursion. In comparison with model (2.7), some constraints have been changed and some new ones have been defined. To clarify these modifications, all constraints will again be discussed below:

- **Constraints (1a) – (1c):**
  These constraints are the sulphur constraints (quality constraints) of the different products, i.e. for every product there is a maximum percentage of sulphur allowed. Notice the added error terms in these constraints due to the guessed quality variables \( q_p \), \( q_D \), and \( q_G \). Notice once more that only the fraction \( d \varepsilon \) of the error term \( x_{e,p} \) is used in constraint (1a), since only stream \( x_4 \) has been directed to blend \( a \) and not the other outgoing stream of the pool \( x_3 \). For the constraints (1b) and (1c), the complete error term is used, since the total outgoing stream is directed to that product.

- **Constraints (2a) – (2b):**
These constraints are the quality constraints for the two different units. For the unit *POOL*, constraint (2a) can be interpreted as the condition that the weighted sulphur percentage of the pool has to equal the guessed sulphur percentage of the outgoing stream plus the error term $x_{e,p}$, which is correct by definition. As explained earlier, this error term is divided between the two directions by using the fraction $d_e$. Constraint (2b) is the quality constraint for *UNIT*. The error term $x_{e,p}$ is again partly being used due to the guess of $q_p$. Thus the left hand side of the constraint is the expected sulphur content in the stream $x_5$ and the right hand side is the linear approximation, using the base and delta values of Table 2.11.

- **Constraint (3a):**
  This is again the quantity constraint of the pool unit, equal to constraint (3a) of model (2.7).

- **Constraints (3b1) – (3b2):**
  These two constraints are the substitutions of constraint (3b) of model (2.7), since the outgoing stream of the unit is not known in advance when using DBM. These constraints are the quantity constraints for the two directions *DIESEL* and *GASOLINE*. In words, constraint (3b1) means for instance that the total part of the incoming streams that will be directed to *DIESEL*, assuming the data of the base case, has to equal the total outgoing stream directed to *DIESEL*, adapted with the corresponding delta values.

- **Constraints (4a) – (4b):**
  These two constraints resemble the previous two; however, one should note that these two are quality constraints whereas (3b1) and (3b2) are quantity constraints. The right hand side of for instance constraint (4a) is the outgoing yield $y_{6\times}$ times the approximated sulphur percentage $q_D$ (with corresponding error term $x_{e,D}$) of the outgoing stream. Thus the left hand side of the constraint has to contain the yield times the sulphur percentage of the incoming stream. The expected yield of the incoming stream that will become *DIESEL* will be $B_Y_{D\times}x_5$ plus a certain term for the number of deltas that the feed property deviates from the base value. To see this, note that $B_Y_{D\%}$ of the incoming stream in the base case flows to *DIESEL*. This whole term at the left hand side has to be multiplied with the base property value for *DIESEL* (similar to the right sight). The extra term $\Delta_D z$ is added for the number of deltas that the feed property deviates from the base value. Notice the coefficient $\Delta_D$ in front of the shift vector. This value has been calculated with the following expression:

$$\Delta_D = (B_Y_D + D_Y_D)(B_P_D + D_P_D) - (B_P_D \cdot B_Y_D),$$

and equivalently for the *GASOLINE* case. The reason of this expression is beyond the scope of this thesis and will be assumed as given.

- **Constraints (5a) – (5f):**
  These quantity constraints are the sell and purchase constraints. Thus, for all the crude oils that can be purchased, there is the ability of choosing a maximum allowable purchase amount and for all the three products a minimum demand that has to be satisfied.

Using the values of Table 2.8, 2.10, and 2.11, model (2.9) can be filled in and simplified into the following LP problem:
Model (2.10) is an arbitrary LP problem and solvable with an LP-solver, such as CPLEX or Xpress, with decision variables $x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_{e,p}, x_{e,D}, x_{e,G}$, and $z$. On a similar way as in Section 2.2, the recursion steps of Distributive Recursion can be executed. In this case, it takes four recursions to obtain a solution that proves to be a sufficiently good approximation of the optimal solution of the nonlinear problem. The final relevant activities are shown in Table 2.12.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Activity (barrels per day)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>47.55</td>
</tr>
<tr>
<td>$x_2$</td>
<td>32.50</td>
</tr>
<tr>
<td>$x_3$</td>
<td>200</td>
</tr>
<tr>
<td>$x_4$</td>
<td>52.45</td>
</tr>
<tr>
<td>$x_5$</td>
<td>180</td>
</tr>
<tr>
<td>$x_6$</td>
<td>125.01</td>
</tr>
<tr>
<td>$x_7$</td>
<td>55.08</td>
</tr>
<tr>
<td>$q_b$</td>
<td>2</td>
</tr>
<tr>
<td>$q_P$</td>
<td>0.638</td>
</tr>
<tr>
<td>$q_D$</td>
<td>0.400</td>
</tr>
<tr>
<td>$q_G$</td>
<td>1.287</td>
</tr>
</tbody>
</table>

**Optimal Value (dollar)**

| Objective | 512.91 |

*Table 2.12 Optimal solution (Example 3)*

As one can see, the purchase of TVG ($x_3$) has reached its maximum allowable value, while the other two crude oils ($x_1$ and $x_2$) are purchased less than their maximum. The production of *blend a* is at its minimum ($x_1 + x_2$), while the other two products (whose minima are zero) are also sold.

In the next chapter a second solution method for nonlinear programming problems in the petroleum industry, named Extended Nonlinear Programming (XNLP), will be discussed and applied on a small example.
3 Extended Nonlinear Programming

3.1 Introduction

In this chapter Extended Nonlinear Programming (XNLP) will be discussed in a theoretical way. First the main difference with Distributive Recursion will be elaborated and after that two versions of XNLP will be discussed. The first version uses the nonlinear solver CONOPT to solve the refinery-planning problems. This method will be extensively discussed in Section 3.2 and applied on an example. The second one uses the nonlinear solver XSLP and will be briefly discussed in Section 3.3.

Extended Nonlinear Programming is a new nonlinear solution method available in the PIMS software package. At this moment Shell is still using Distributive Recursion at its petroleum refineries worldwide, since they are not (yet) familiar with XNLP. The main difference between these two methods is the way of using and implementing the information from every recursion. This difference between the two solution methods is shown in Figure 3.1.

![Figure 3.1 Flow diagrams of DR and XNLP](image)

In the case of Distributive Recursion (which has extensively been explained in Chapter 2), a nonlinear programming problem is sent to PIMS where the user also has to define an initial guess (arrows 1) to transform the nonlinear problem into a linear one. Then the linear programming problem is expressed in a matrix and sent to a linear solver (like Xpress and CPLEX) (arrow 2). After that it will return to PIMS (arrow 3) and will be checked whether the solution is appropriate enough. When the solution does not prove to be a sufficiently good approximation, new guesses will be chosen and a new input will be sent to the solver (arrow 4). When, after several recursions, the solution finally proves to be a sufficiently good approximation of the optimal solution of the NLP problem, the method stops and PIMS produces the “optimal” solution as the output (arrow 5).
The new solution method in PIMS, Extended Nonlinear Programming, has a different way of calculating the optimal solution, which is also shown in Figure 3.1. Again a nonlinear problem is sent to PIMS (arrow 1), together with an initial guess specified by the user. This will again be sent to the solver (arrow 2), expressed in a matrix. After that, the solver sends the first approximation back to PIMS (arrow 3). Thus until this step there is no difference between DR and XNLP. Though in this case PIMS does not send a new guess back to the linear solver, but sends the linear information as a first guess, together with the nonlinear model to a nonlinear solver (CONOPT or XSLP) (arrow 4). Inside that solver several iterations will be executed to obtain a sufficiently good solution (arrow 5). After that, it will be returned to PIMS (arrow 6), where the output will be accessible for the user (arrow 7). Thus, the recursion steps are now executed outside PIMS and inside the nonlinear solver. The working of the two different nonlinear solvers will be discussed in the rest of this chapter.

### 3.2 The CONOPT solver

One of the available solvers for the XNLP method is CONOPT. CONOPT is a commercial solver used in software packages such as AIMMS, Matlab, and PIMS. This solver uses the Generalized Reduced Gradient (GRG) method to solve the (nonlinear) problems, which is described by de Klerk et al. (2005). The GRG method is a generalization of the reduced gradient method. The Reduced Gradient method was developed by Wolfe (1963) to solve a nonlinear problem with linear constraints. The basic idea of this Reduced Gradient method is that under certain assumptions all variables can be expressed in terms of an independent subset of the variables. Hence the number of variables in the model will be reduced to approximate the optimal solution. The Generalized Reduced Gradient method can also be applied on nonlinear problems with nonlinear constraints. The basic idea of the GRG method, which has been developed by Abadie and Carpentier (1969), is to replace the nonlinear constraints by their linear Taylor approximation at the current value of $x$, and then apply the reduced gradient method. Hence the Generalized Reduced Gradient method has basically the same structure as the Reduced Gradient method. Since the nonlinear problems at the refineries contain several nonlinear constraints, the generalized version of the method is used in PIMS and will be discussed below, based on de Klerk et al. (2005).

#### 3.2.1 GRG method

First let us assume that the problem at the refinery can be written as:

\[
\min f(x) \\
\text{s.t. } h_j(x) = 0, \quad j = 1, \ldots, m \\
x \geq 0,
\]

(3.1)

where the functions $f, h_1, \ldots, h_m$ are supposed to be continuously differentiable. To illustrate this method with an example, the method will first be explained in general. Therefore some notation and assumptions are required.
Let us assume that the gradients of the constraint functions \( h_1, \ldots, h_m \) are linearly independent at every point \( x \geq 0 \) and that each feasible value of \( x \) has at least \( m \) positive components. Let \( F \) denote all values of \( x \) for which the problem has been solved. Note that when the constraints are nonlinear, the feasible region \( F \) may not be convex. This could cause extra effort, which could result in executing more iterations before a sufficiently good solution has been found.

In the initialization step of the method \( (t = 0) \), a feasible solution \( x_{r=0} \geq 0 \) with \( h_j(x_{r=0}) = 0 \) for all \( j \) has to be chosen. This solution is usually equal to the first solution obtained in Distributive Recursion (as shown in Figure 3.1). The vector \( x \) can be divided into a basic \( x_B \) and a non-basic part \( x_N \), i.e. \( x_{r=0} = (x_B_{r=0}, x_N_{r=0}) \) for \( t \geq 0 \). Note that in first case the choice of \( x_B_{r=0} \) is restricted to nonnegative values. Furthermore, the Jacobian matrix of the constraints \( H(x) = (h_1(x), \ldots, h_m(x))^T \) at each \( x \geq 0 \) is denoted by \( JH(x) \) and has full rank. Consequently, the matrix \( JH(x) \) can also be divided into a basic part \( B \) and a non-basic part \( N \), i.e. \( JH(x) = (B \quad N) \) with the \( x_B \)-columns stated in part \( B \) and the \( x_N \)-columns in part \( N \).

Now the constraints \( H(x) = 0^T \) can be linearized by the first-order Taylor approximation\(^1\), using the Jacobian matrix:

\[
H(x) = H(x_{r=0}) + JH(x_{r=0})(x - x_{r=0}) = JH(x_{r=0})(x - x_{r=0}) = 0^T.
\]

To see this, note that \( H(x_{r=0}) = 0^T \). Using this equation, the following expression can be derived:

\[
JH(x_{r=0})x_{r=0} = Bx_B + Nx_N . \tag{3.2}
\]

To see this, note that the matrix \( JH(x) \) and the vector \( x \) both have been divided into two parts: the basic and non-basic part. Hence, multiplying the matrix \( JH(x) \) with the vector \( x \) results in a summation of both products as shown in (3.2). This equation can now be simplified, by denoting the left hand side of the expression by \( b \), so \( b = JH(x_{r=0})x_{r=0} \). Then by using equation (3.2), the variables in the basis \( B \) can be written as

\[
x_B = B^{-1}b - B^{-1}Nx_N . \tag{3.3}
\]

Using this expression, model (3.1) can be written as

\[
\begin{align*}
\min & \quad f_N(x_N) \\
\text{s.t.} & \quad x_B = B^{-1}b - B^{-1}Nx_N \geq 0 \\
& \quad x_N \geq 0,
\end{align*} \tag{3.4}
\]

\(^1\) In general the first-order Taylor expansion for a function \( f(x) \) in a point \( \bar{x} \) is defined by \( f(x) \approx f(\bar{x}) + \nabla f(\bar{x})(x - \bar{x}) \), where \( \nabla f(\bar{x}) \) denotes the gradient of the function in \( \bar{x} \).
where \( f_N(x_N) = f(B^{-1}b - B^{-1}Nx_N, x_N) \). To see this, note that the constraints \( H(x) = 0^T \)
of model (3.1) are now substituted by expression (3.3), which results in a model without restrictions (except the non-negativity constraints). Thus the objective function of model (3.4) has to be minimized, taking into account that the variables stay positive.

After these steps, the method continues with calculating the search directions for the basic and non-basic variables by using the steepest descent method. The steepest descent method is a line search method that moves in the opposite direction than the derivative of the objective function. This is the most obvious choice for a search direction, since the objective value will always decrease maximally when \( x \) increases with a sufficiently small \( \varepsilon > 0 \). Thus the search direction of the nonbasic variable is defined as

\[
s_N = -\frac{\partial f_N(x_N)}{\partial x_N}.
\]  

(3.5a)

Converting \( f_N(x_N) = f(B^{-1}b - B^{-1}Nx_N, x_N) \) into a function of \( x_B \) results in \( f_B(x_B) = f(x_B, N^{-1}b - N^{-1}Bx_B) \). Now the search direction of the basic variables can be written as

\[
s_B = -\frac{\partial f_B(x_B)}{\partial x_B} = -\frac{\partial f_B(x_N(x_B))}{\partial x_B}.
\]

Now by using the chain rule\(^2\) this becomes

\[
s_B = -\left(-\frac{\partial f_N(x_N)}{\partial x_N} \cdot (-B^{-1}N) \right) = -B^{-1}Ns_N.
\]

(3.5b)

These search directions are necessary to determine the new values for \( x_N \) and \( x_B \). One uses the formulas

\[
x_N^{\text{NEW}} = x_N^{\text{OLD}} + \lambda s_N,
\]

(3.6a)

\[
x_B^{\text{NEW}} = x_B^{\text{OLD}} + \lambda s_B,
\]

(3.6b)

for the non-basic and basic variables respectively. Let \( \lambda^* \) denote the value of \( \lambda \) for which the objective function of model (3.4) has been minimized (without taking into account the nonnegativity constraints). This can be done by filling in expression (3.6a) in the objective function of (3.4) and optimizing this function with respect to \( \lambda \). Let \( \lambda_{\text{max}} \) denote the maximum absolute value of \( \lambda \) for which both variables are still nonnegative (by using (3.6)). Now the optimal value of \( \lambda \) equals \( \lambda^* \) when \( \lambda^* \leq \lambda_{\text{max}} \), and \( \lambda_{\text{max}} \) when \( \lambda^* > \lambda_{\text{max}} \).

With this optimal value of \( \lambda \), the new values of \( x_N \) and \( x_B \) can be calculated by using

\(^2\) If \( y = f(x) \) and \( x = g(t) \) then \( \frac{dy}{dt} = \frac{dy}{dx} \cdot \frac{dx}{dt} \).
expressions (3.6). After that, these new values have to be checked for feasibility by using model (3.1). When they are not feasible, the basic variable usually has to be adjusted and the whole procedure has to be repeated with these values, until a good solution has been found.

The steps that have to be followed for executing the GRG method for a given nonlinear model with nonlinear constraints of the form as model (3.1) are summarized below.

Initialization:
- Start with iteration \( i = 0 \).
- Find a feasible solution \( x_{i=0} \) and divide this solution into a basic (\( x_B \)) and a non-basic (\( x_N \)) part.
- Define a stopping value \( \varepsilon > 0 \) such that the algorithm stops when \( |H(x)| < \varepsilon \).

Step 1:
- Calculate the Jacobian \( A \) and corresponding vector \( b \) by using
  \[
  A = JH(x_{i=0}),
  \]
  and
  \[
  b = Ax_{i=0}.
  \]
  The matrix \( A \) will now be divided into a set of columns \( N \) and a set of columns \( B \). The columns in \( N \) are the columns that are differentiated with respect to a non-basic variable and the columns in \( B \) are those that are differentiated with respect to a basic variable.

Step 2:
- Write \( x_B \) as a function of \( x_N \) by using equation (3.3).

Step 3:
- Rewrite the starting model into a model similar as (3.4), with only non-basic variables by using the obtained expression(s) in Step 2.

Step 4:
- Determine the search directions, using equations (3.5a) and (3.5b).

Step 5:
- Determine the solutions for the next iteration, using
  \[
  x_{N|i+1} = x_{N|i} + \lambda s_{N|i}
  \]
  and
  \[
  x_{B|i+1} = x_{B|i} + \lambda s_{B|i}.
  \]

Step 6:
- Calculate the maximum allowable value \( \lambda_{\text{max}} \) of \( \lambda \), such that \( x_{N|i+1} \) and \( x_{B|i+1} \) are still nonnegative.
• Calculate the value $\lambda^*$ that optimizes the objective function of model (3.4) (without taking into account the nonnegativity constraints).

• If $\lambda^* \leq \lambda_{\text{max}}$, then $\lambda_{i+1} = \lambda^*$.
  
  If $\lambda^* > \lambda_{\text{max}}$, then $\lambda_{i+1} = \lambda_{\text{max}}$.

**Step 7:**
• Fill in $\lambda_{i+1}$ in the equations of Step 5 and calculate $x_{Bi+i}$ and $x_{Ni+i}$ by using the equations of Step 5.

**Step 8:**
• While $|H(x_{i+1})| \geq \epsilon$, adapt $x_{Bi+i}$ such that $H(x_{i+1}) = 0, i=i+1$ and start again at Step 1. When elements of $x_{Bi+i}$ are positive, those variables will remain as basic ones. Otherwise, new basic variables have to be chosen for the next iteration. The non-basic variable with the largest absolute value of gradient is selected to enter the basic variable set, according to Lee et al. (2004).

• When $|H(x_{i+1})| < \epsilon$, the vector $x_{j;i+1}$ has to be adjusted in the same way as above (to deal with the constraints). This vector is the optimal solution with corresponding objective function value.

### 3.2.2 GRG method applied on an NLP problem (Example 4)

To illustrate the GRG method with an example, we apply it on the following nonlinear programming problem:

\[
\begin{align*}
\text{min} & \quad 2x_1^2 + 4x_2^2 - 2x_l \\
\text{s.t.} & \quad x_1^2 + 2x_2^2 - 36 = 0 \\
& \quad x_1, x_2 \geq 0.
\end{align*}
\]

**Iteration 1:**

**Initialization:** Let us start from the point $x_{i=0} = (x_{Bi=0}, x_{2i=0})^T = (2, 4)^T$ with an objective value of 68 and let us assume that for this recursion, $x_2$ is the basic variable. Let us also define the stop value: $\epsilon = 0.16$.

**Step 1:** The Jacobian matrix of the constraints equals:

\[
A = JH(x_{i=0}) = \begin{pmatrix} N & B \end{pmatrix} = \begin{pmatrix} \frac{\partial h_1}{\partial x_1} & \frac{\partial h_1}{\partial x_2} \\ \frac{\partial h_2}{\partial x_1} & \frac{\partial h_2}{\partial x_2} \end{pmatrix} = \begin{pmatrix} 2x_{Bi=0} & 4x_{2i=0} \end{pmatrix} = \begin{pmatrix} 4 & 16 \end{pmatrix}.
\]

From this, we get

\[
b = Ax_{i=0} = \begin{pmatrix} 4 & 16 \end{pmatrix} \begin{pmatrix} 2 \\ 4 \end{pmatrix} = 72.
\]

**Step 2:** Now by using equation (3.3), this basic variable can be written as a function of the non-basic variable:
\[ x_2 = B^{-1}b - B^{-1}Nx_i = \frac{1}{16} \left( 72 - \frac{1}{16} 4x_i \right) = 4.5 - 0.25x_i. \]

**Step 3:** Hence the objective function becomes

\[ f_i(x_i) = f(x_i, B^{-1}b - B^{-1}Nx_i) = f(x_i, 4.5 - 0.25x_i). \]

Using this objective function, model (3.4) can be written as

\[
\begin{align*}
\min & \quad 2x_i^2 + 4(4.5 - 0.25x_i)^2 - 2x_i \\
\text{s.t.} & \quad 4.5 - 0.25x_i \geq 0 \\
& \quad x_i \geq 0.
\end{align*}
\]

This equals

\[
\begin{align*}
\min & \quad 2.25x_i^2 - 11x_i + 18 \\
\text{s.t.} & \quad 4.5 - 0.25x_i \geq 0 \\
& \quad x_i \geq 0.
\end{align*}
\]

Hence this model can now be optimized by solving this model, i.e. minimizing this objective function, taking into account the nonlinearity of the variable \( x_i \) and indirectly of \( x_2 \).

**Step 4:** Now the search directions of the basic and the non-basic variable, denoted by \( s_B \) and \( s_N \), respectively, have to be determined:

\[ s_{N|t=0} = s_{B|t=0} = -\frac{\partial f_N(x_{N|t=0})}{\partial x_i} = -4.5x_{N|t=0} + 11 = 2, \]

and

\[ s_{B|t=0} = s_{2r=0} = -B^{-1}Ns_{N|t=0} = -\frac{1}{16} 4 \cdot 2 = -0.5. \]

**Step 5:** Thus now the new variables can be calculated by using the search directions determined above.

\[
\begin{align*}
x_{2r=1} &= x_{2r=0} + \lambda s_{B|t=0} = 4 - 0.5\lambda \geq 0, \quad (3.9b)
\end{align*}
\]

\[
\begin{align*}
x_{|r=1} &= x_{|r=0} + \lambda s_{N|t=0} = 2 + 2\lambda \geq 0, \quad (3.9a)
\end{align*}
\]
Step 6: Using expressions (3.9), one can arrive at $\lambda_{\text{max}} = 8$, because larger values result in a negative value for $x_{2|r=1}$. Now using formulas (3.9), the objective function of model (3.8) can be rewritten into:

$$\min \quad 2.25 x_{[r=1]}^2 - 11x_{[r=1]} + 18.$$ 

In terms of $\lambda$ it becomes

$$\min \quad 9\lambda^2 - 4\lambda + 5.$$ 

This objective function can now be minimized with respect to $\lambda$, by solving the equation

$$\frac{\delta}{\delta \lambda} \left(9\lambda^2 - 4\lambda + 5\right) = 0.$$ 

This results in the following optimal value of $\lambda^*$:

$$\lambda^* = 2.22 \cdot 10^{-1}.$$ 

Since $\lambda^* < \lambda_{\text{max}}$ the optimal value of $\lambda$ now equals $\lambda_1 = 2.22 \cdot 10^{-1}$.

Step 7: Inserting this value of $\lambda$ into expressions (3.9), results in

$$x_{[r=1]} = (x_{[r=1]} \quad x_{2|r=1})^T = (2.444 \quad 3.889)^T.$$ 

Step 8: As expected, the constraint of model (3.8) (and thus of model (3.7)) will not hold when using these values. This is due to the nonlinearity of the constraint. Since $|H(x_{[r=1]})| = 0.222 > \epsilon$, the basic variable $x_2$ will be adjusted such that the constraints of the original model (3.7) hold, i.e. $x_{[r=1]} = (x_{[r=1]} \quad x_{2|r=1})^T = (2.444 \quad 3.875)^T$ with an objective value of 67.111. Consequently, only $x_{[r]}$ changes and $x_N = x_1$ stays fixed. These (feasible) values of $x$ are the starting points for the next iteration, which will be similar to Iteration 1. Now $x_2$ will again be the basic variable, since $x_{[r=1]}$ stayed positive. All iteration steps, necessary to obtain a sufficiently good solution, are similar to the one above and hence elaborated in Appendix B.

The final solution of the GRG method for this example has been obtained after 5 iterations. This optimal solution has an objective function value of 64.162 with a corresponding optimal value of $x$:

$$x_{[r=5]} = (3.922 \quad 3.211)^T.$$ 

The method explained in this section is used in the solver CONOPT in the XNLP mode and has a structure that differs significantly from the solution method used in Distributive Recursion. In the next section, the second solver available in the XNLP mode will be described.
3.3 The XSLP solver

XSLP is a solver developed by AspenTech (developers of PIMS) and based on the sequential or successive linear programming (SLP) method. This method is about solving a nonlinear programming problem by a sequence of linear approximations and by using linear programming to solve each subproblem, as explained by Pike (2001). These subproblems are problems in which the constraints and objective function of the original model are linearized by using the first-order Taylor expansion. Notice that the linear solution method Distributive Recursion (Chapter 2) is also a variant of the SLP method, but significantly differs from this nonlinear solver (as will be clear in this section). Let us assume that the nonlinear problem of the refinery is again of the following structure:

\[
\begin{align*}
\min & \quad f(x) \\
\text{s.t.} & \quad h_j(x) = 0 \quad j = 1, \cdots, m \\
& \quad x \geq 0,
\end{align*}
\]

(3.10)

where the functions \( f, h_1, \cdots, h_m \) are supposed to be continuously differentiable and may be nonlinear. This model can be linearized around a feasible point \( x_k \) at each iteration \( k \), using the Taylor expansion as defined in Section 3.2. This means that the nonlinear constraint functions in the subproblems can be locally approximated by

\[
h_j(x) \approx h_j(x_k) + \nabla h_j(x_k)(x - x_k) \quad j = 1, \cdots, m.
\]

The objective function of the subproblem will be linearized in the same manner as the constraints. However, the first term at the right hand side \( f(x_k) \) will be removed; since it is a constant in the objective function, it does not influence the optimal value of \( x \). However, note that it does influence the objective function value, but since we are interested in the optimal value of \( x \) for each subproblem, it is not significant for that computation. Using these linearizations, the subproblem of model (3.10) becomes

\[
\begin{align*}
\min & \quad \nabla f(x_k)(x - x_k) \\
\text{s.t.} & \quad h_j(x_k) + \nabla h_j(x_k)(x - x_k) = 0 \quad j = 1, \cdots, m \\
& \quad x \geq 0.
\end{align*}
\]

This can be rewritten in the form

\[
\begin{align*}
\min & \quad \nabla f(x_k)(x - x_k) \\
\text{s.t.} & \quad \nabla h_j(x_k)(x - x_k) = -h_j(x_k) \quad j = 1, \cdots, m \\
& \quad x \geq 0.
\end{align*}
\]

(3.11)

This submodel can now be solved with an LP algorithm, using \( (x - x_k) \) as the decision variable, such as the Simplex algorithm. However, since the Simplex algorithm usually works with non-negative variables and since the values of \( (x - x_k) \) can take on both positive and negative values (depending on the location of the optimum), a change of...
variables has to be done. Griffith & Stewart (1961) introduced the following substitution to deal with this issue:

\[(x - x_k) = (x - x_k)^+ - (x - x_k)^- , \]  

(3.12)

where

\[(x - x_k)^+ = \begin{cases} 
(x - x_k) & \text{if } (x - x_k) \geq 0 \\
0 & \text{if } (x - x_k) < 0,
\end{cases} \]

\[(x - x_k)^- = \begin{cases} 
-(x - x_k) & \text{if } (x - x_k) \leq 0 \\
0 & \text{if } (x - x_k) > 0.
\end{cases} \]

Substituting equation (3.12) into model (3.11) results in the following linear problem, which can be solved using the Simplex algorithm:

\[
\begin{align*}
\text{min} & \quad \nabla f(x_k)(x - x_k)^+ - \nabla f(x_k)(x - x_k)^- \\
\text{s.t.} & \quad \nabla h_j(x_k)(x - x_k)^+ - \nabla h_j(x_k)(x - x_k)^- = -h_j(x_k) & j = 1, \cdots, m \\
& \quad (x - x_k)^+ - (x - x_k)^- \geq -x_k.
\end{align*}
\]

(3.13)

After this problem has been solved, it will be checked whether or not the corresponding value of \(x\) is a sufficiently good approximation of the optimal solution of the original model (3.10) and it will be decided whether or not a new iteration will be started with the current value of \(x\) as new starting point \(x_{k+1}\). We omit the details of how these can be done. Finally, note that the feasible starting point for the first recursion can be obtained by running the first recursion of the Distributive Recursion algorithm.

Besides CONOPT and XSLP there are other appropriate nonlinear solvers available on the market, such as SNOPT and MINOS. These packages for solving nonlinear programming problems are not available in PIMS, but to give a good overview and to compare the solvers described above with other ones, they will be briefly discussed and compared in the next chapter.
4 Other packages for solving nonlinear programs

4.1 Introduction

Two other major nonlinear solvers, which are not available in PIMS, will be discussed in this chapter, to give a good overview of different solvers for nonlinear programming problems available on the market. The solver SNOPT (based on sequential quadratic programming) and the solver MINOS (based on sequential linearly constraints), both developed by Stanford Business Software Inc., will be discussed in Section 4.2 and 4.3 respectively. Our discussion will mainly be based on Nocedal and Wright (1999) and Moré and Wright (1993).

Next, all four nonlinear solvers discussed in this thesis will be compared in Section 4.4, by giving an overview of which sort of problem is suitable for which solver.

4.2 The SNOPT solver

The SNOPT (Sparse Nonlinear Optimizer) solver is a sparse line-search method based on the sequential or successive quadratic programming (SQP) algorithm with quasi-Newton approximations and with an augmented Lagrangian merit function. In such an SQP algorithm, quadratic programming is used to solve a nonlinear programming problem, by solving a sequence of quadratic subproblems. These subproblems are problems in which the constraints of the original model are linearized by using the first-order Taylor expansion and in which the objective function is a quadratic approximation of a Lagrangian, using the second-order Taylor expansion.

The objective function of each subproblem will be a quadratic approximation of a Lagrangian around a point \( x_k \) at iteration \( k \), using the Taylor expansion:

\[
\Lambda(x, \lambda_k) = \Lambda(x_k, \lambda_k) + \nabla_x \Lambda(x_k, \lambda_k) (x - x_k) + \frac{1}{2} \nabla^2_{x,x} \Lambda(x_k, \lambda_k) (x - x_k)^2,
\]

where \( \nabla^2_{x,x} \Lambda(x_k, \lambda_k) \) is the Hessian of the Lagrangian. Once more, the first term of the right hand side will be omitted in the subproblem, since \( \Lambda(x_k, \lambda_k) \) is a constant and it does not influence the optimal value of \( x \). The subproblem of model (3.10) now becomes

---

3 The Lagrangian for a model with objective function \( f(x) \) and constraints \( h_j(x) = 0 \) for \( j=1,\ldots,m \) is defined as \( \Lambda(x, \lambda) = f(x) + \sum_j \lambda_j h_j(x) \).

4 In general the second-order Taylor expansion for a function \( f(x) \) in a point \( \overline{x} \) is defined by \( f(x) \approx f(\overline{x}) + \nabla f(\overline{x})(x - \overline{x}) + \frac{1}{2} \nabla^2_{x,x} f(\overline{x})(x - \overline{x})^2 \), where \( \nabla f(\overline{x}) \) denotes the gradient of the function in \( \overline{x} \) and \( \nabla^2_{x,x} f(\overline{x}) \) the Hessian of the function in \( \overline{x} \), with respect to \( x \).
\[
\min_x \nabla_x \Lambda(x_k, \lambda_k)(x - x_k) + \frac{1}{2} \nabla^2_{x,x} \Lambda(x_k, \lambda_k)(x - x_k)^2 \\
\text{s.t.}\quad h_j(x) \approx h_j(x_k) + \nabla h_j(x_k)(x - x_k) = 0 \quad j = 1, \ldots, m \\
x \geq 0.
\] (4.1)

The sequential QP approach outlined above requires the computation of \( \nabla^2_{x,x} \Lambda(x_k, \lambda_k) \) at iteration \( k \). Most of the SQP methods (including SNOPT) use the quasi-Newton approximation of this Hessian, i.e. it replaces this matrix with the Broyden-Fletcher-Goldfarb-Shanno (BFGS) approximation \( B_k \). This approximation is defined as

\[
B_k = \nabla^2_{x,x} \Lambda(x_1, \lambda_1),
\]

and for iteration \( k > 1 \)

\[
B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k + y_k y_k^T}{s_k^T B_k s_k + y_k^T y_k},
\]

where \( s_k = x_{k+1} - x_k \) and \( y_k = \nabla \Lambda(x_{k+1}, \lambda_{k}) - \nabla \Lambda(x_k, \lambda_k) \). Explaining why such an approximation is suitable for these problems is beyond the scope of this thesis.

Solving the QP subproblem (4.1) with approximated Hessian is itself an iterative procedure. Again, the details of how this iterative procedure is carried out are beyond the scope of this thesis.

After the QP subproblem has been solved, the next step is to choose a new starting point for the next iteration, i.e. \( x_{k+1} \). This point will be obtained by using \( x_{k+1} = x_k + \alpha_k (x - x_k) \) and should not only decrease the objective value, but also move towards satisfying the constraints. Since \( x_k \) and \( x \) are already given, one only has to determine the distance \( \alpha_k \) to move along the direction \( x - x_k \). Therefore we first have to define a merit function \( \ell \), which determines whether or not a certain point is better than another. This function \( \ell \) (called Augmented Lagrangian) is defined as

\[
\ell(x, \lambda; \nu) = f(x) + \sum_{j \in [\ell,m]} \lambda_j h_j(x) + \frac{1}{2} \sum_{j \in [\ell,m]} \nu_j h_j^2(x),
\] (4.2)

where \( \nu \) is a vector of positive penalty parameters. Relative large values for the penalty parameters mean large weights assigned to constraint satisfaction relative to minimization of the objective. Given the starting point \( x_k \), the search direction obtained by solving the \( k^{th} \) QP subproblem \( x - x_k \), and the corresponding Lagrange multipliers \( \lambda_k \) and \( \lambda_{k+1} \), the optimal step length \( \alpha_k^* \) at iteration \( k \) is now found by (approximately) minimizing

\[
\ell(x_k + \alpha_k (x - x_k), \lambda_k + \alpha_k (\lambda_{k+1} - \lambda_k); \nu).
\]
Note that the computation of $\lambda_{k+1}$ can be done in several ways, but to avoid the details of this solver, these computations are beyond the scope of this thesis, but elaborated in Nocedal en Wright (1999). The optimal step length can now be used to determine the new starting point, $x_{k+1}$. After that, one needs to check whether this value is an appropriate solution of the original problem (3.10). The value is tested on certain tolerances that are specified by the user in advance. If this new starting point proves to be a good approximation of the optimal solution (i.e. no tolerances are violated) the solver is finished. Otherwise, a new iteration can be executed with starting point $x_{k+1}$.

4.3 The MINOS solver

The MINOS (Modular In core Nonlinear Optimization System) solver is based on a sequential linearly constrained (SLC) method. The principal idea is not to simply use a merit function as a test of suitability for steps (as (4.2) is used), but to optimize the merit function directly. Thus this method generates a step by minimizing a Lagrangian subject to linearizations of the constraints. That means again that the constraints will be linearized, using the first-order Taylor expansion. In contrast with an SQP solver that minimizes a quadratic approximation of the original objective function subject to linear constraints (such as SNOPT), the subproblem in SLC methods optimizes a nonlinear approximation of the Lagrangian. Note that MINOS only uses an SLC method when the constraints of the problem are nonlinear (which is indeed the case in the petroleum industry). The discussion below is based on Nocedal and Wright (1999).

Let us again consider the equality-constrained problem (3.10). At iteration $k$ the subproblem with linearized constraints becomes

$$
\begin{align*}
\min_x \quad & F_k(x) \\
\text{s.t.} \quad & h_j(x_k) + \nabla h_j(x_k)^T (x - x_k) = 0 \quad j = 1, \ldots, m \\
& x \geq 0,
\end{align*}
$$

where the objective function $F_k(x)$ is defined differently for different SLC solvers that are available on the market. To define these different objective functions, let us first denote the difference between $h_j(x)$ and its linearization at $x_k$ by $\Delta h_j^k(x)$ for every constraint $j$, i.e.

$$
\Delta h_j^k(x) = h_j(x) - h_j(x_k) - \nabla h_j(x_k)^T (x - x_k),
$$

at iteration $k$. Now the objective function $F_k(x)$ is in some of the SLC solvers, especially the first generation SLC solvers that were available on the market, defined as

$$
F_k(x) = f(x) - \sum_j \lambda_j^k \Delta h_j^k(x),
$$

where $\lambda^k_j$ is the current Lagrange multiplier estimate. In addition, the MINOS solver adds an extra term to the Lagrangian. In the MINOS solver $F_k(x)$ has been defined as
\[ F_k(x) = f(x) - \sum_j \lambda^k_j \Delta h^k_j(x) + \frac{1}{2\mu} \sum_j \left[ \Delta h^k_j(x) \right]^2, \]  
(4.4)

where \( \mu \) is a positive penalty parameter, which determines the weight that we assign to constraint satisfaction relative to minimization of the objective. Note that a Lagrangian usually contains the constraint functions \( h_j(x) \) and not the differences \( \Delta h^k_j(x) \).

Intuitively, since both have to equal zero in the original model, one expects that choosing one of these will not influence the outcomes significantly. The exact reason why MINOS uses the differences \( \Delta h^k_j(x) \) is on algorithmic detail and beyond the scope of this thesis.

Since the nonlinear subproblem (4.3) with objective function (4.4) has to be minimized in every step, several iterations (and thus several function and constraints evaluations) are required to generate the new iterate. MINOS also uses quasi-Newton iterations to solve this linearly constrained subproblem. When such a subproblem has been solved and therefore an optimal \( x_k \) and \( \lambda^k \) (for iteration \( k \)) have been obtained, a test of tolerance has to be executed again to decide whether or not a new iteration has to be run. In the next iteration, we take the multiplier estimate \( \lambda^{k+1} \) equal to be the multiplier of the subproblem (4.3) at the previous iteration. Then after several iterations one can show that \( x_k \) converges to an approximated solution \( x^* \) of the optimal solution of the NLP problem.

Thus the main difference between MINOS and SNOPT is that MINOS directly optimizes a Lagrangian, whereas SNOPT optimizes a quadratic Taylor approximation of the objective function. Thus SNOPT executes two steps at each iteration, whereas MINOS only executes one. The main similarities between both solvers are the fact that they both use linearized constraints in the subproblems and that they both use the quasi-Newton approximation of the Hessian.

### 4.4 Comparison of CONOPT, XSLP, SNOPT, and MINOS

Each nonlinear programming problem has its own characteristics. Some problems only have a nonlinear objective function with linear constraints and others have a linear objective function with extremely nonlinear constraints. Also the number of nonlinear constraints differs for each problem.

In this section, the four nonlinear solvers outlined above will be evaluated in a theoretical way on some of those typical characteristics. This overview is stated in Table 4.1 and will be discussed in the rest of this section.

<table>
<thead>
<tr>
<th>#</th>
<th>Models with …</th>
<th>CONOPT</th>
<th>XSLP</th>
<th>SNOPT</th>
<th>MINOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>extremely nonlinear constraints, objective function, or both</td>
<td>+</td>
<td>?</td>
<td>+</td>
<td>?</td>
</tr>
<tr>
<td>2</td>
<td>a large number of nonlinear constraints</td>
<td>+</td>
<td>?</td>
<td>+</td>
<td>?</td>
</tr>
<tr>
<td>3</td>
<td>few nonlinearities outside the objective function</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>4</td>
<td>many more variables than equations (many</td>
<td>-</td>
<td>-</td>
<td>+/-</td>
<td>+</td>
</tr>
</tbody>
</table>
Table 4.1 Summarized comparison of the four nonlinear programming solvers

<table>
<thead>
<tr>
<th>degrees of freedom</th>
<th>5 ... the optimum of the problem is known to lie on a linear constraint</th>
<th>?</th>
<th>+</th>
<th>?</th>
<th>?</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 ... an objective and its gradients that are expensive to evaluate</td>
<td>?</td>
<td>?</td>
<td>+</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

**Case 1**

Some nonlinear problems are extremely difficult or even not yet possible to solve with one of the four solvers. Others are more trivial and easy to solve. According to http(3), http(8), and Cai et al. (2001), CONOPT and SNOPT perform in general better on models with extremely nonlinear constraints and objective function. Extremely nonlinear models are models that have higher order nonlinearities than second order (quadratic), i.e. models with nonconvex expressions. Moreover, when either the constraints or objective function is extremely nonlinear, CONOPT and SNOPT perform again well on average. Note that new versions of CONOPT (CONOPT2 & CONOPT3) also have an SLP and SQP component, which makes CONOPT more efficient on easier and mildly nonlinear models as well. However, these versions of CONOPT are not discussed in this thesis. Note that the problems at the refineries have a linear objective function and have constraints that are usually not extremely nonlinear.

**Case 2**

On the other hand, one can also look at the quantity of nonlinear constraints in the model. Then the CONOPT and SNOPT solvers do also perform well, according to http(4) and http(5). In refinery-planning problems, only the constraints regarding the pooling and delta based modeling cases are nonlinear, i.e. most of the constraints are linear. Hence outperforming other solvers in this case is not relevant for the petroleum industry.

**Case 3**

However, when there are only a few nonlinearities outside the objective function, SNOPT still performs well, according to http(4). Furthermore, MINOS and XSLP are also appropriate solvers (http(5) and http(7)). CONOPT proves to be less appropriate for less complex models, according to http(3).

**Case 4**

If there are much more variables than equations in the model, i.e. there are many degrees of freedom, then, according to http(5), MINOS proves to be an appropriate solver. Also SNOPT can accommodate these problems, according to http(4). On the other hand, XSLP and CONOPT perform less in this case, based on http(5). Though XSLP is very suitable for problems with nearly as many constraints as variables, according to Bazarraa et al. (1993). Also CONOPT has a fast method for finding a first feasible solution that is particularly well suited for models with few degrees of freedom, i.e. for models where the number of variables is approximately the same as the number of constraints. The models in the petroleum industry are usually of this form. In these cases, CONOPT is likely to outperform MINOS, while for models with many more variables than equations MINOS is usually more suited. In the refinery models, the number of variables is approximately the same as the number of constraints. This means that CONOPT and XSLP are in this case suitable for the problems in the refinery world.
Case 5
In a linear programming problem, the optimum is always obtained on a constraint. If the optimum of the original model is known to lie on a linear constraint, XSLP may be a good solver, according to Poku et al. (2003). Hence for nonlinear programming problems where the (local) optimum is known to lie on a linear constraint, XSLP may be a good and fast solver. Since in refinery problems the optimum usually lies on a linear constraint, XSLP may be appropriate for these kinds of problems.

Case 6
According to http(2), SNOPT generally requires fewer evaluations of the nonlinear functions than MINOS. This means that when the objective and its gradients are expensive to evaluate, SNOPT may be more efficient than MINOS. In refinery problems the objective function is linear and thus cheap to evaluate, which means that in this case SNOPT has less advantage of the fact that it needs fewer evaluations than MINOS.

In this section it has been reported that in six different cases certain solvers seem to perform better than others. In PIMS there is only the opportunity to use either CONOPT or XSLP. Most of the problems can be solved quickly and accurately by both or either one of those. Especially the nonlinear problems in the petroleum industry are appropriate for these two solvers, since it is known that the optimum usually lies on a linear constraint, the objective function is cheap to evaluate, and the number of variables is approximately the same as the number of constraints. Using CONOPT or XSLP in PIMS (in the XNLP mode) generates additional features to analyse and adjust the optimal solution. In the next chapter, the three additional features will be discussed.
5 Additional output in the XNLP mode

5.1 Introduction

When using the XNLP option in PIMS, additional features become available for the user. The most frequently used option is the Ranging Analysis tool, explained by Varvarezos (2003). This option gives the user the opportunity to analyse the range under optimality conditions for each feedstock (the general name for the raw materials that are purchased at the refinery, such as crude oils), product, or capacity. This tool will be described in Section 5.2. A second additional feature, called Parametric Analysis, will be discussed in Section 5.3. This analysis tool can be used to evaluate cases in which variables are analysed when prices of components or products are changed. In Section 5.4 the Goal Programming feature will be discussed. Using this feature, the model can be run with a secondary (and tertiary) goal in addition to the primal goal of the plan. A secondary goal could be for instance maximization of the throughput for a process or minimization of the product quality giveaway. Notice that all these new features are only available in the XNLP mode of PIMS.

5.2 Ranging Analysis

5.2.1 Introduction

The Ranging Analysis tool is an extension of marginal values. It gives a better overview of the flexibility of a feedstock, product, or capacity than marginal values. A marginal value refers to the change in economic value associated with an additional unit of for instance the purchase of a feedstock. It does not give information about large increments, especially not for nonlinear programming problems. In linear problems the marginal values are piecewise constant. This means that they represent the additional value until a certain increment. For larger increments, a different marginal value occurs. Marginal values can therefore not be used to estimate what happens when large amounts of e.g. a feedstock are replaced by other one(s). In nonlinear problems the marginal values are not even piecewise constant or linear. This means that the marginal value is only valid for an extremely small increase of the corresponding variable. For every additional value, the marginal values may be different. Hence comparing marginal values gives only an indication where potential benefits can be achieved and no guarantees for larger adaptations. On the other hand, Ranging Analysis provides better information about larger changes in the variables of the model. It gives more information about the interchangeability of e.g. the feedstocks, with respect to the objective function value. It checks, for instance, whether a feedstock can be replaced by one or more other feedstocks, without a significant, user-specified drop of the objective value. It also checks the ability of using the feedstock for replacing other feedstocks. The principle of Ranging Analysis is described in the rest of this section, based on Varvarezos (2003).
Let us first assume that the nonlinear programming model that will be discussed in this section is of the form

\[
\begin{align*}
\min & \quad f(x) \\
\text{s.t.} & \quad h_j(x) = 0 \quad j = 1, \cdots, m \\
& \quad x^L \leq x \leq x^U,
\end{align*}
\]

(5.1)

where the only differences with model (3.10) are the explicit lower and upper bounds on the variables, denoted by \( x^L \) and \( x^U \) respectively. To perform Ranging Analysis the so-called utility and flexibility indices are necessary. The utility index of a certain feedstock is the fraction of the remaining total feed that this particular feedstock can displace for a predefined marginal drop in the objective value. The formula for this fraction equals

\[
\text{Utility Index} = \frac{X_{\text{MAX}} - X_{\text{BASE}}}{1 - X_{\text{BASE}}},
\]

(5.2)

where \( X_{\text{BASE}} \) is the fraction of the total feed that this feedstock represents at the base optimal solution and \( X_{\text{MAX}} \) is the maximum fraction of the total feed that this feedstock can reach at near optimal conditions. The fraction \( X_{\text{BASE}} \) can be computed by solving model (5.1) and calculating the fraction for each crude oil. Let us denote the component of the vector \( x \) that corresponds to the feedstock \( F \) that is analysed by \( x_F \). The fraction \( X_{\text{MAX}} \) can be computed by solving the following nonlinear problem:

\[
\begin{align*}
\max & \quad x_F \\
\text{s.t.} & \quad h_j(x) = 0 \quad j = 1, \cdots, m \\
& \quad f(x) \geq f^{(5.1)} \cdot (1 - \varepsilon_M) \\
& \quad x^L \leq x \leq x^U,
\end{align*}
\]

(5.3)

where \( \varepsilon_M \) is a user-defined constant that represents the marginal relaxation of the objective value for the Ranging Analysis and \( f^{(5.1)} \) the optimal objective function value of model (5.1). Thus the purchase amount of the analysed feedstock is maximized, such that the objective value is still larger than the base solution multiplied with a constant close to one.

A utility index equal to zero means that the corresponding feedstock, product, or capacity cannot displace other ones within a predefined marginal drop in the objective value. This means that, according to (5.2), \( X_{\text{BASE}} = X_{\text{MAX}} \). When for instance (a part of) a certain feedstock cannot be delivered, it cannot be displaced by other feedstocks with a utility index of zero. A utility index of one means that it can displace the whole remaining total feed and still keeps the objective value above a predefined level. Usually there are no feedstocks that have a utility index of one. If a feedstock has a relatively high utility index, it means that it can displace a high fraction of the total feed, with the objective function value not lower than a predefined level.
On the other hand, the flexibility index is the fraction of the optimal amount of a feedstock that can be displaced for a predefined marginal drop in the objective value. The formula for this fraction equals

\[ \text{Flexibility Index} = \frac{X_{\text{BASE}} - X_{\text{MIN}}}{X_{\text{BASE}}}, \]  

(5.4)

where \( X_{\text{MIN}} \) is the minimum fraction of the total feed this feedstock can reach at near optimal conditions. This fraction is calculated in the same way as \( X_{\text{MAX}} \), except the fact that it minimizes the scalar variable that represents the purchased quantity \( x_F \).

If a feedstock has a flexibility index of zero, it cannot be displaced by other feedstocks without a too large drop of the objective value. On the other hand, a flexibility index of one means that it can be fully displaced by others. Thus a high flexibility index for a certain crude oil refers to a high fraction of its optimal amount that can be displaced. All feedstocks that are not in the base solution, i.e. not purchased, have a flexibility index equal to one, since they can be fully displaced. This is an obvious consequence and therefore quite meaningless.

5.2.2 Ranging Analysis applied on a small demo model (Example 5)

The Ranging Analysis tool will now be clarified with a small example. The summary statistics of all crude oils in this example are shown in Table C.1 of Appendix C. For this base solution a Ranging Analysis can be performed, which results in Figure 5.1 and 5.2.

![Figure 5.1 Flexibility and Utility index of each feedstock (Example 5)
The utility and flexibility indices for the different crude oils are shown in Figure 5.1 and the corresponding $X_{\text{MIN}}$, $X_{\text{BASE}}$, and $X_{\text{MAX}}$ are shown in Figure 5.2. This last figure gives information about the range of the quantity of all crude oils. According to Figure 5.1, there are two crude oils that have a utility index of zero in this scenario, namely NSF and TJI. Hence these crude oils cannot displace other crude oils within a predefined marginal drop in the objective value. Thus when (a part of) a certain feedstock cannot be delivered, it cannot be displaced by NSF or TJI. On the other hand, there is no crude oil with a utility index of one, which is as expected. It means that there is no crude oil that can displace the whole remaining total feed and still keeps the objective value at a certain level.

If we have a look at the flexibility indices, there is one feedstock with a flexibility index of zero and one with an index equal to one. The crude oil BAC cannot be displaced by other crude oils, without a too large drop of the objective value. On the other hand, the crude oil KUW can be fully displaced by other crude oils. Unfortunately this is quite meaningless, since it is an obvious consequence of the fact that KUW has an activity of zero in the optimal solution. Note that on the other hand, KUW can substitute the original crude slate up to 8% (the utility index value) of the total feed. According to Figure 5.2, the feedstock AHV has the widest range, which means that purchasing this feedstock brings a high degree of flexibility without a user-specified drop in the objective function value.

The advantage of Ranging Analysis is that when a certain crude oil can only be purchased less or even not (due to uncertainty in the supply chain), it can determine which crude oil can replace this lack to minimize the decrease in objective function value. Thus it gives a more global view than marginal values on small adjustments in the basic solution. As explained in this section, Ranging Analysis could be an important tool for analysing an optimal solution. Note that instead of feedstocks, the same analysis can be done for end products and capacities. Two other additional features of XNLP will be discussed in the next two sections.
5.3 Parametric Analysis

5.3.1 Introduction
The Parametric Analysis tool can be used to evaluate cases that parameterize an objective function coefficient of a variable, i.e. to investigate the behaviour of the model as a function of some price changes of a raw material, product, or other variables over a defined range. For instance, it can be investigated what happens with the purchase of a certain crude oil when the price of that crude oil or a different one changes. To clarify this tool, the data of Example 5 will again be used in the next example (Example 6).

5.3.2 Parametric Analysis applied on a small demo model (Example 6)
A Parametric Analysis of the crude oil ‘Arabian Light’, which is also used in Example 5, is shown in Figure 5.3.

The price of Arabian Light is shown on the x-axis of Figure 5.3, whereas the purchase (activity) of this crude oil is shown on the y-axis. First notice that the purchase of Arabian Light has only been measured at the integer values of the price, i.e. the lines in Figure 5.3 are meaningless.

Let us again start with the values of Table C.1. When the price of Arabian Light equals 17 $/bbl, the quantity of Arabian Light equals 23.8 bbls/day, which is also shown in Figure 5.3. When the price increases by one, the purchase of Arabian Light decreases directly to 20 bbls/day, which is according to Table C.1 its minimum quantity. On the other hand, when the price decreases by one, the purchase jumps directly to its maximum of 40 bbls/day.

A similar figure can be developed in which the consequences of the price changes of Arabian Light for the activity of a different crude oil or end product are shown. In Figure 5.4, such a graph is shown for the purchase of the crude oil ‘Alaskan N Slope’.
When the price of Arabian Light increases, the purchase of the crude oil Alaskan N Slope slightly increases, according to the figure above. This is due to the fact that the purchase of Arabian Light decreases and it is therefore more attractive to buy Alaskan N Slope. On the other hand, when the price of Arabian Light increases further, the purchase of Alaskan N Slope also increases. This is due to the fact that in this model some of the crude oils are partly interchangeable. This was also shown in Figure 5.1, where ARL seems to have a flexibility and a utility index larger than zero.

Besides the consequences on prices, the consequences on properties of a certain component can also be analysed. If once more the price of a crude oil changes, the consequences of e.g. the sulphur content of a certain stream can be measured. In Figure 5.5 the sulphur content of the input stream of a so-called Cat Cracker unit (Cat Feed pool) is analysed for price changes of Arabian Light.
According to Figure 5.5, a higher price of Arabian Light than in the base case (Table C.1), as well as a lower price causes an increase of the sulphur percentage in the Cat Feed pool. This is due to the fact that a change in the price of a crude oil could cause different compositions of components downstream the process and therefore including the composition of this Cat Feed pool. Such a different composition of a stream means different properties of that stream (e.g. the sulphur percentage).

Thus with this analysis tool a better view of the consequences of price changes can be investigated. Notice that these analyses can also be executed for the prices of end products, instead of crude oils. Then the user can analyse purchases, sells, and properties for changes of the price of this end product.

5.4 Goal Programming

5.4.1 Introduction

The Goal Programming feature can be used to examine secondary and tertiary goals, in addition to the primal goal of maximizing the objective function value. After a first run of the model has been executed, in which the goal is to maximize the objective function value, the user can identify other goals to improve the solution on other areas. In order to meet the additional goals, it may be necessary to make concessions to the other goals, i.e. an acceptable level of degradation of the preceding goal has to be defined. Thus for the secondary goal, the user can define a maximum allowable decrease in the objective function value and for the tertiary goal a maximum allowable change in the secondary goal. Examples of secondary (or tertiary) goals are the maximization of the throughput for a process unit or the minimization of product quality giveaway. After this secondary and tertiary goal (if specified) is optimized, the Goal Programming procedure finally runs for the second time a case in which the objective function will be maximized, since this is the most important goal of the optimization process. In this run it is allowed to change the preceding optimized goal (the secondary or tertiary) again with a user-specified percentage.

5.4.2 Goal Programming applied on a small demo model (Example 7)

We run two scenarios of Goal Programming to illustrate this tool. The first scenario (Scenario A) will only have a secondary goal of maximizing the purchase of Arabian Light with a maximum allowable decrease of the objective function value of 2%. Also the secondary goal is allowed to change 2% (in the last step of the procedure). The second scenario (Scenario B) will additionally have a tertiary goal of minimizing the sulphur content of the Cat Feed pool with a maximum allowable change in the secondary goal of 3%. Also the maximum allowable change in the tertiary goal equals 3%. The results of these Goal Programming scenarios, together with the base scenario, are shown in Table 5.1.

<table>
<thead>
<tr>
<th></th>
<th>base scenario</th>
<th>Scenario A</th>
<th>Scenario B</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>INPUT</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Secondary goal</td>
<td></td>
<td>Max. purchase of Arabian Light</td>
<td>Max. purchase of Arabian Light</td>
</tr>
<tr>
<td>Tertiary goal</td>
<td></td>
<td></td>
<td>Min. sulphur percentage</td>
</tr>
</tbody>
</table>
Table 5.1 Goal Programming (Example 7)

According to this figure, the objective function value in Scenario A has been decreased with 1.90% compared to the objective function value in the base scenario. As one can see, this is not equal to the maximum percentage change in the objective value. This is due to the fact that the Goal Programming procedure ends with the main goal (maximizing the objective function), as explained above. To clarify this, the three steps of the Goal Programming procedure in Scenario A are shown in Table 5.2.

Table 5.2 Steps of Goal Programming of Scenario A (Example 7)

Thus in Step 1 of the Goal Programming process the objective function is maximized. Its optimal value equals $540.02, which equals the base scenario of Table 5.1. After that, the purchase of the crude oil Arabian Light is maximized in Step 2, with a maximum decrease in the objective value of 2%. Thus now either the purchase of Arabian Light will reach its upperbound of 50 bbls/day (Table C.1), with a decrease in objective value less than 2%, or the objective value will decrease 2% with the corresponding purchase of Arabian Light. In this case the objective value decreases with 2% (to 529.22). The corresponding amount of Arabian Light that will be purchased equals 40.01 bbls/day (and is indeed less than 50 bbls/day). In Step 3 the objective function value will again be maximized, with a maximum percentage decrease in the amount of purchasing Arabian Light equal to 2%. This eventually results in the solution with an objective function value of $529.74 and a purchase amount of 39.21 bbls Arabian Light per day. Hence the base solution of the model has now slightly been adjusted to obtain a solution in which more goals are fulfilled.

Using these three additional features in PIMS, the optimal solution can be analysed and/or finetuned, which gives a better view of the solution and arises the opportunity to obtain a more appropriate solution than in first case. In the following chapters the practical part of the thesis will be elaborated.
6 Case comparison

6.1 Introduction

The two available solution methods in PIMS (Distributive Recursion and Extended Nonlinear Programming) are extensively outlined in the previous chapters of this thesis. According to Kunt (2003) and Kunt (2004), AspenTech (the developer of PIMS) claims that XNLP performs in general better than DR. In particular they claim less local optima in the XNLP mode, which implies higher objective values on average. They also claim that XNLP needs in general less time to obtain the final solution. These claims will be tested in this chapter, by using models of refineries owned by Shell. The real names of the refineries are not mentioned, but denoted by Refineries A, B, and so on. Note that the linear solver used for both methods is the Xpress solver, which is only used once in the XNLP mode (in the first recursion). In addition, the nonlinear solver in the XNLP mode is the XSLP solver (as explained in Chapter 3).

6.2 Case study

To perform a reliable study and to gain a good overview of the performances of both modes, ten refinery-planning problems will be used for this study. For these ten refineries, a run has been executed in both the Distributive Recursion and the Extended Nonlinear Programming mode. The results of this case study are shown in Appendix D, in which the objective function value, the computation time, and the number of nonzero elements in the matrix are shown. Note that this last statistic is a good indicator for the size of the model. The refineries are ordered from small to large using the number of nonzero elements in the matrix.

Before drawing conclusions about this case study, notice that three out of thirteen models, which gave feasible solutions in the Distributive Recursion mode, did not give feasible solutions or did not even start optimizing in the Extended Nonlinear Programming (XNLP) mode (before optimizing, an error message appears). The reason of this error message could be the fact that all models are build (and therefore the starting points are chosen) to be run in DR and therefore not appropriate to be run in XNLP. However, there is not enough evidence to draw any conclusions about these models. Besides this, AspenTech claims that the starting points (guesses) in the XNLP mode are less important to avoid local optima. This means that obtaining infeasibility in these three models could be in contradiction with the claims of AspenTech. These three models are not used in this case study and not shown in Appendix D.

Besides infeasible models, there are also models (Refineries C, D, G, I, and J), which had to be run twice in the XNLP mode to obtain a feasible solution. This happens when the starting point of the run ends up at an infeasible solution. We then use this infeasible solution as starting point in the new run and a feasible solution is obtained. These refinery
models are, in contrast with the infeasible ones, shown in Appendix D with a corresponding computation time of only the last run.

Also note that for some models (Refineries G, I, and J) the tolerances in the XNLP mode (as mentioned in Section 3.3) had to be relaxed to obtain a feasible solution. Remember that the tolerances are used to check whether the solution of the linear subproblem is a sufficiently good approximation of the original NLP problem. When using the default values of the tolerances, the solutions did not converge and hence the problems could not be solved. Relaxing these constraints results in ‘feasible’, but also less accurate solutions.

### 6.3 Conclusions

We first consider the objective function values of the ten models. To illustrate the tables in Appendix D, the objective function values of the different models are plotted in Figure 6.1.

![Figure 6.1 Objective function values of 10 refineries in DR and XNLP mode](image)

According to this figure, the objective values do not differ significantly for most of the models. Only the objective value of Refinery C is extremely different for both modes. It seems that the recursions converge to a (negative) local optimum in the XNLP mode, which does not seem to be the case in the DR mode. On the other hand, Refineries G, I, and J have higher objective values in the XNLP mode. Notice however that these three refineries had an infeasible solution in advance, but by relaxing the tolerances (as mentioned in Section 6.2) these problems became feasible. As described earlier, relaxing the user-specified tolerances results in less computation time and (often) a significantly different ‘feasible’ solution. The objective values of Refineries G, I, and J are higher in the XNLP mode, which may be due to this increase of these XNLP-typical tolerances. The
only refinery that gave a significantly (more than 12%) higher objective value in the XNLP mode without the obligation of running twice or relaxing the tolerances is Refinery B. To clarify this difference, the objective function values of Refinery B are enlarged in Figure 6.1. In this case, the objective function value in the DR mode seems to be a local optimum. Note that it is not guaranteed that the objective function value in the XNLP mode is indeed a global optimum, but it is at least a higher local optimum than the one in DR.

By considering the objective values, one can conclude that they are in general approximately equal for both modes. This is in contrast with the opinion of AspenTech, who claim that using XNLP results in less local optima and thus higher objective values on average. Considering the computation times of the different models may show us some more differences and could give us a better idea of the functionality of XNLP with respect to DR. A similar figure as Figure 6.1, but now with the computation times on the y-axis is shown below.

![Figure 6.2 Computation time of 10 refineries in DR and XNLP mode](image)

First note that the computation time depends on the capacities of the computer and the guesses at the first iteration. From this figure it can be concluded that when the model size increases, the computation time of XNLP increases on average more than the time of DR, up to a difference in computation times of 350% in Refinery J. For large refineries F – J the computation time is on average twice as large when using XNLP instead of DR. For the smaller models A – E these computation times are approximately equal.

After this small case study has been executed, one can conclude that on average Distributive Recursion performs better than Extended Nonlinear Programming at this moment. In this case study there are no refineries, which have a higher objective value plus less computation time in the XNLP mode. Taking into consideration that Distributive Recursion has significantly been improved since its release and Extended Nonlinear
Programming is a relative new method, the performance of XNLP may increase in the next years. But for now running all models of Shell in XNLP does not seem to increase the total performance of PlanStar/PIMS, by giving more accurate and higher quality solutions.

Although DR gives, according to this case study, higher quality solutions, XNLP could still be a useful mode to execute the runs of refinery-planning models, since it generates additional features that can be used to analyse the base solution (as explained in Chapter 5). In Chapter 7 one of these new features (Ranging Analysis) will be implemented in the Marginal Analysis Tool to test whether or not this Ranging Analysis feature could increase the functionality of this tool. Using this test, an answer to the following question will be provided: “How can the additional features be used to improve the functionality of the Marginal Analysis Tool?”. 
7 Ranging Analysis in MAT

7.1 Introduction

Shell uses an Excel template called ‘Marginal Analysis Tool’ (MAT), developed by ORTEC. This tool analyses the solution of the base case, which is optimized with PlanStar/PIMS. This base solution is called the monthly plan of the refinery. As explained in the introduction of this thesis, the user can run cases (prepared with MAT) in which one of the values of a parameter or variable is adapted. Cases can be defined with e.g. a different unit capacity, blend specification, or crude oil import. Thus using MAT, the solution space around the optimum can be explored to gain better understanding of the drivers impacting the margin. In this chapter the emphasis will be on the part of MAT that allows the user to run cases in which the crude oil purchase amounts can be adapted. As explained in the introduction, this part is called Cargo Analysis and in this chapter it is investigated whether the functionality of this analysis can be improved. For the other part of MAT, which is called Marginal Steering (as explained in the introduction), no study will be performed in this chapter to improve its functionality.

The input for Cargo Analysis is the monthly plan, updated with the latest information available. The set of crude oils inside this monthly plan is called the base crude diet. It is a feasible crude composition, which is accepted and usually purchased by the refinery. Outside this base crude diet there are a lot of other crude oils available on the market. Using Cargo Analysis it can be evaluated whether one of these crude oils could be a good displacer for one of the crude oils inside the base crude diet. It can also be evaluated whether a crude oil inside the base crude diet could replace a different one, due to delay or unavailability of that crude oil. In Section 7.2 this method will extensively be discussed.

According to AspenTech, the new feature Ranging Analysis (explained in Chapter 5), which is only available in the XNLP mode, could give a good overview of the exchangeability of all crude oils. AspenTech claims that Ranging Analysis is very powerful and applicable at two levels, based on Varvarezos (2003). On one hand, the crude oil range can be used as a way to address uncertainty in the availability of crude oils. On the other hand, it can be utilized as a strategic tool for trading and evaluation decisions. The flexibility indices, which are generated during the Ranging Analysis procedure, denote the fraction of its purchase amount that can be displaced by other crude oils, within a predefined decrease in the objective value. As a result, purchasing crude oils with a low flexibility index increases the overall risk. Therefore purchasing crude oils with higher flexibility indices can minimize the risk caused by delay and unavailability of other crude oils. The utility index denotes the fraction of the total feed that can be displaced by the corresponding crude oil, within a predefined decrease in the objective value. Figure 5.1 shows for instance that AHV could replace the highest fraction of the total feed, since it has the highest utility index. Therefore the user can also identify a crude oil with a higher utility index as the “back-up” feed, since that crude oil could replace most others if needed. Utilizing this information (by creating for instance inventory of this particular crude oil) will minimize the risk on operating sub-optimally in the case of an unforeseen
change in the monthly plan. Hence in this way the indices determined with Ranging Analysis can be useful for the traders.

Based on these statements of AspenTech, Shell is excited about this tool and wants to evaluate whether Ranging Analysis could improve the functionality of Cargo Analysis, by implementing the new feature within this old method. In that case Ranging Analysis is used for evaluating all crude oils on the market. If it does not prove to be useful in this case, Ranging Analysis could still be applicable for addressing uncertainty in the availability and as a strategic tool (as described above). This means that investigating the working of this method and comparing it to the current method is anyhow relevant for Shell and ORTEC.

Since AspenTech claims that Ranging Analysis gives a good overview of the exchangeability of all crude oils, Shell wants to investigate whether the utility indices of all crude oils outside the base crude diet could give a good representation of which crude oils are appropriate to displace the ones in the base crude diet. Using Ranging Analysis, the functionality of MAT could therefore be improved.

However, there are some significant differences between the Ranging Analysis procedure and the current method (Cargo Analysis), such that it probably causes different results in the case study. These differences will be considered in Section 7.3. Since ORTEC predicts that the differences will be of such a degree that Ranging Analysis will not be useful in combination with Cargo Analysis and since case studies are useful to convince Shell, the Ranging Analysis tool has been implemented in MAT (Section 7.4) in order to investigate whether it can improve the functionality of MAT. It is compared to Cargo Analysis, since it already has been proved that this method performs well.

### 7.2 Current method: Cargo Analysis

At this moment all crude traders at the refineries of Shell use the results of the Cargo Analysis method. On one hand, this analysis is used to investigate whether there are opportunities to increase the refinery margin by purchasing other crude oils that are available on the market, instead of a crude oil in the base crude diet (first goal). On the other hand, it can be investigated which crude oil in the base solution could be appropriate to be purchased more than the optimal value (as inventory), due to uncertainty in the delivery of a crude oil in the base solution (second goal). In the rest of this section the first goal will be explained based on the Cargo Analysis method.

The crude oils, which are frequently processed on the refinery and easily available, are usually included in the base crude diet. Most of the purchase amounts of these crude oils are fixed by contracts or already available at the refinery as inventory, when running the Cargo Analysis procedure. Nevertheless there are many other crude oils available on the market, which might add value to the refinery margin. To evaluate these other crude oils on the market, an additional amount of such a crude oil can be forced in a new case. Besides this, the user can specify which amounts of the crude oils purchased in the base crude diet can be replaced in this new case and which ones have to stay fixed. The purchases of the ones that can be replaced are usually easy to cancel. Usually it is always
the same set of crude oils, which are e.g. transported to the refinery through pipelines (and not by ship), such that they can easily be cancelled. In MAT they are free to move between the minimum and maximum purchase amounts. The other ones keep their actual value of the base solution as fixed value in the new case. In Table 7.1 the monthly plan of an arbitrary refinery is shown. The trader can distinguish the fixed with the free ones by putting crosses at the right side of the table in MAT (see Table 7.1), based on his experience and information.

In this table all the crude oils in the base crude diet are shown with corresponding purchase costs (input costs) in $, purchase constraints (MIN and MAX) in kton/month, and actual purchase amount in the base solution (ACTUALS) in kton/month. In Table 7.1 only the crude oil ARAL has not been fixed. This one is called the balancing crude oil. This means that the trader wants to evaluate whether a crude oil is available on the market, which is more appropriate than ARAL. Such a crude oil is more appropriate when it increases the objective value. Thus in the sensitivity cases, which will be prepared by MAT (to be run in PlanStar), the amount of ARAL is allowed to increase or decrease between the minimum and maximum purchase amounts of Table 7.2. Note that in practice this amount will decrease since the capacity to process all the crude oils is in general fully used and the evaluated crude oil is forced in. In Table 7.2 such a case is shown for the crude oil ABCD (which is available on the market but not in the base crude diet). Now a fixed amount of 80 barrels (bbls) will be forced to be in the new case.

Notice that the user can also specify a variable amount, which is added in the new case, by defining a minimum and maximum amount that will be forced (instead of a fixed amount). This occurs less often, since the available sizes are usually fixed and depending on the size of the crude oil tankers/boats.

Thus in the new case all the crude oil purchases are fixed to the amount of the base solution, except the one of ARAL (which is free to move) and ABCD (which is forced to be 80 bbls). This new case can be sent to PlanStar/PIMS and a new solution can be calculated. The corresponding objective value gives a good representation of the exchangeability of ARAL by ABCD. When one wants a good overview of the exchangeability of ARAL by all possible crude oils, a large number of cases has to be executed. For all these cases a ‘Crude Indifference Value’ (CIV) in $/bbl is calculated in

<table>
<thead>
<tr>
<th>Crude oil</th>
<th>MIN</th>
<th>MAX</th>
<th>FIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARAH</td>
<td>169.29</td>
<td>70.0</td>
<td>140.0</td>
</tr>
<tr>
<td>ARAL</td>
<td>181.36</td>
<td>0.0</td>
<td>280.0</td>
</tr>
<tr>
<td>BREB</td>
<td>198.16</td>
<td>0.0</td>
<td>160.0</td>
</tr>
<tr>
<td>FORC</td>
<td>192.76</td>
<td>0.0</td>
<td>160.0</td>
</tr>
<tr>
<td>OMAN</td>
<td>184.20</td>
<td>0.0</td>
<td>140.0</td>
</tr>
<tr>
<td>UPZA</td>
<td>187.67</td>
<td>0.0</td>
<td>240.0</td>
</tr>
</tbody>
</table>

Table 7.1 Base solution with corresponding free and fixed crude oils

<table>
<thead>
<tr>
<th>Crude oil</th>
<th>MIN</th>
<th>MAX</th>
<th>FIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABCD</td>
<td></td>
<td></td>
<td>80</td>
</tr>
</tbody>
</table>

Table 7.2 Additional crude oil that is forced in a new case
MAT, which is the additional (marginal) value in the objective function value per barrel of the corresponding case with respect to the base solution (Table 7.1). The larger this value, the more attractive the corresponding crude oil. Hence these values are calculated after the cases are run and could therefore give a good rank of the exchangeability of ARAL by all available crude oils outside the base crude diet.

This current method used within Shell is an accurate method, but extremely time-consuming since all possible cases have to be run. Using the utility indices of Ranging Analysis in MAT (see expression (5.2)) could reduce this number of cases, by running only cases for those crude oils, which have a relative high utility index. This is since these crude oils can displace the largest fraction of the total feed, with a predefined decrease in objective value. Note that in this case this user-specified decrease in the objective value will usually be 0%. Hence the time necessary to get a good overview of the exchangeability of for instance the crude oil ARAL could be reduced. Therefore it first has to be investigated whether Ranging Analysis could indeed be useful in MAT. Before that, the differences of both methods will first be discussed in the next section.

### 7.3 Differences between both methods

Although AspenTech claims that Ranging Analysis is a good way to test the exchangeability of balancing crude oils by other crude oils outside the base crude diet, there are some significant differences with the current method of Cargo Analysis.

In the current method every case that has been run forces only one crude oil in the base solution and allows only the balancing crude oil to decrease. This base solution contains only the crude oils inside the base crude diet. On top of that, in Ranging Analysis the crude oils that have to be evaluated and that are outside the base crude diet are also free to be purchased in the base solution. This means that this base solution will usually be significantly different than the base solution of Cargo Analysis (with only crude oils from the base crude diet). Since all crude oils are allowed to be purchased in the case of Ranging Analysis, a lot of small drops of different crude oils will be purchased. Since this does not reflect the reality, it can be seen as a shortcoming of the Ranging Analysis procedure. Besides in the base solution, also in the cases to calculate the utility and flexibility indices all purchase amounts can be adapted in Ranging Analysis, which is also not realistic. This is significantly different from allowing only two crude oil amounts to change in the case of Cargo Analysis. The different contents of the base case and other cases are of such a degree that it will be very likely that Ranging Analysis will not improve the functionality of the current Cargo Analysis method.

Another difference between both methods is the goal to optimize. In the current Cargo Analysis method the objective function of the model will be maximized, given the base crude diet with one balancing crude oil and one forced crude oil, which is outside the base crude diet. For each of these crude oils a given purchase amount will be forced into the model in separate cases. The only purchase variable in all these cases is the purchase amount of the balancing crude oil. In this method the objective values of these cases may therefore turn out higher than the objective value of the base case. In contrast with Cargo Analysis, Ranging Analysis optimizes the purchase amount, taking into account a
predefined decrease in the objective function value. This predefined decrease should be
specified by the user and can also be 0%. Using this tolerance, the objective value will
decrease to its lowerbound, such that the maximum purchase amount \(X_{\text{MAX}}\) can be
calculated. Note that when the trader wants to investigate whether there are opportunities
to increase the refinery margin by purchasing other crude oils that are available on the
market (the first goal, as explained in Section 7.2), the user-specified decrease in the
objective value will usually be 0%. Then only those crude oils, which do not decrease the
objective value, give positive values of \(X_{\text{MAX}}\). The purchase amount in Ranging Analysis
is therefore not fixed, but maximized, such that the objective value will not decrease more
than the user-specified percentage. This means that both methods have a different goal to
optimize. However when only considering this difference, we believe there will still be a
positive relation between both methods. If for instance a crude oil performs well in Cargo
Analysis, i.e. the corresponding case has a relative high CIV, it is attractive to purchase
this crude oil. When it has on the other hand a high utility index with a user-specified drop
of 0%, it means that purchasing a relative large amount of this crude oil does not decrease
the objective value. We believe these two scenarios are positively correlated. If a crude oil
has a negative CIV, it means that it is not attractive to force this crude oil into the base
solution, instead of the balancing crude oil. This is also usually positively correlated with a
utility index of e.g. zero, which means that the objective function value has to decrease to
force this crude oil into the base solution.

Considering the differences, it is not straightforward that Ranging Analysis can improve
the Cargo Analysis process. However, to advice Shell on this topic, Ranging Analysis will
be implemented in MAT, such that both methods can be compared. Though we have our
doubts it could not be used for this purpose, it could still be used to address the uncertainty
in the availability of crude oils in the base crude diet (the second goal, as explained in
Section 7.2).

7.4 Using Ranging Analysis to improve MAT

To reduce the number of cases and therefore the time to obtain a good overview of the
exchangeability of the crude oils, the results of the Ranging Analysis procedure can be
used in MAT. As explained in Section 5.2, Ranging Analysis could give a good overview
of the flexibility of crude oils. Running only those cases for which the forced crude oil has
a sufficiently high utility index could reduce the time of the complete process and may
therefore improve the functionality of MAT. To clarify this method and to test the
differences and similarities, the algorithm in the following subsection has been developed.

7.4.1 Algorithm

Step 1:
- Run PlanStar/PIMS to obtain a base solution of a given model and load it from
  PlanStar into MAT.

Step 2:
- Divide the crude oils of the base solution in crude oils with fixed and free purchase
  amounts. The user can do this by putting crosses in the ‘fixed in base case’-column
in MAT (see Table 7.1) at the row of the corresponding crude oil. To be able to run
the Cargo Analysis, the crude oils are divided into three disjoint sets:
  o Set 1 (fixed crude oils): Crude oils of the base crude diet that the user
defines as fixed for the cargo evaluation cases. These are chosen to ensure
that a good test can be executed to see what happens with the purchase of
other crude oils (ceteris paribus).
  o Set 2 (balancing crude oils): Crude oils of the base crude diet that the user
defines as free to move in the cargo evaluation cases with respect to their
purchase constraints. Usually the user will define only one or two balancing
crude oils. The balancing crude oil(s) will (partly) be pushed out in the
Cargo Analysis cases.
  o Set 3 (crude oils for cargo evaluation): Crude oils that the user wants to
evaluate. For each of these crude oils a (fixed) amount is added to the base
solution, to displace the balancing crude oils. These crude oils are not part
of the base crude diet, but can be forced in the new cases (see Table 7.2).

• When the user has specified these three sets, the minimum and maximum allowed
purchase amounts (purchase constraints) of all crude oils for the new cases have to
be adapted. This is automatically executed by the upgraded MAT version, which
results in the following minimum and maximum values (which are also shown in
Table 7.3):
  o Set 1: All the minimum and maximum allowed purchase amounts are set
equal to the actual values of the base solution for these crude oils and hence
these crude oils stay fixed.
  o Set 2: The minimum allowed purchase amounts of these crude oils are set
equal to their minimum value of the base solution and the maximum
purchases equal to the actual value of the base solution. Hence when a
certain crude oil in Set 3 will be forced in the base solution, the purchase of
the crude oils in this set can be reduced.
  o Set 3: The minimum allowed purchase amount of the crude oils are set
equal to zero and the maximum equal to a sufficiently large number. Hence
these crude oils are free to be purchased in the new cases, such that the
values of $X_{\text{MIN}}$, $X_{\text{MAX}}$, and $X_{\text{BASE}}$ can be calculated. With these values the
utility indices of these crude oils can be calculated.

<table>
<thead>
<tr>
<th></th>
<th>Name</th>
<th>Minimum value</th>
<th>Maximum value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set 1</td>
<td>Fixed crude oils</td>
<td>Actual</td>
<td>Actual</td>
</tr>
<tr>
<td>Set 2</td>
<td>Balancing crude oils</td>
<td>Minimum</td>
<td>Actual</td>
</tr>
<tr>
<td>Set 3</td>
<td>Crude oils for cargo evaluation</td>
<td>0</td>
<td>$\infty$</td>
</tr>
</tbody>
</table>

*Table 7.3 Summary of sets used in the new method*

**Step 3:**
• The user now has to specify the tolerance that distinguishes the crude oils in Set 3
that are appropriate enough to displace the crude oils of Set 2 with the ones that are
not, depending on their utility index. For the appropriate displacers, a new case will
be prepared in MAT, similar as the one in Table 7.2. Selection criteria can be of the following forms:

- Absolute criterion: When the utility index of the crude oil is above a certain fraction \( c \), a new case will be prepared. Note that using this criterion gives an unknown number of new cases to be run.
- Relative criterion: When the utility index is in the first \( d \% \)-quantile of all utility indices, with \( 0 \leq d \leq 100 \), a case will be prepared. This way, one can control the number of new cases that will be prepared.

**Step 4:**

- Run the Ranging Analysis procedure for Set 3 with the new minimum and maximum allowed purchase amounts of all crude oils. This results in values of \( X_{\text{MIN}} \), \( X_{\text{MAX}} \), and \( X_{\text{BASE}} \) for crude oils in Set 3, as explained in Section 5.2. Thus \( X_{\text{MIN}} \) is the minimum amount of the crude oil without a predefined drop in the objective value, and similarly \( X_{\text{MAX}} \) the maximum. \( X_{\text{BASE}} \) is the new actual purchase value of the corresponding crude oil, given the new minimum and maximum allowed purchase amounts of all crude oils.

**Step 5:**

- Import these calculated values in the upgraded MAT version. The software automatically calculates the utility indices for the crude oils in Set 3, by using expression (5.2). These values give a good representation of the capability of displacing base case crude oils by the ones in Set 3. Hence these values are now used to represent the capability of displacing only the crude oil(s) in Set 2.

Due to the selection criterion described above, the number of cases that are necessary to get a good overview of the exchangeability of crude oils in Set 2 is reduced significantly, such that the complete procedure could be less time-consuming. To clarify this algorithm, an example of a large refinery model (Example 7) will be discussed in the next section.

### 7.4.2 Example of Ranging Analysis in MAT (Example 7)

A model of a large refinery is used to clarify the method above. This model consists all necessary PlanStar/PIMS data together plus the MAT version with the Ranging Analysis feature. The data contains for instance more than 100 crude oils that can be purchased. Ranging Analysis could give a good overview of the exchangeability of these crude oils, to avoid running more than 90 cases.

**Step 1:**

- The base solution of this example is shown in Table 7.4, which is obtained by running PlanStar/PIMS in the XNLP mode.
**BASE CASE CRUDE OILS**

<table>
<thead>
<tr>
<th>Crude oil</th>
<th>Input costs</th>
<th>MIN</th>
<th>MAX</th>
<th>ACTUALS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALSH</td>
<td>72.48</td>
<td>0.0</td>
<td>140.0</td>
<td>0.0</td>
</tr>
<tr>
<td>DUBA</td>
<td>73.86</td>
<td>0.0</td>
<td>537.3</td>
<td>537.3</td>
</tr>
<tr>
<td>ENFI</td>
<td>85.76</td>
<td>0.0</td>
<td>289.3</td>
<td>287.2</td>
</tr>
<tr>
<td>MIRI</td>
<td>81.49</td>
<td>10.3</td>
<td>206.7</td>
<td>206.7</td>
</tr>
<tr>
<td>MLOC</td>
<td>70.27</td>
<td>0.0</td>
<td>8.4</td>
<td>8.4</td>
</tr>
<tr>
<td>NIDO</td>
<td>68.21</td>
<td>0.0</td>
<td>15.5</td>
<td>15.5</td>
</tr>
<tr>
<td>QATL</td>
<td>77.99</td>
<td>0.0</td>
<td>673.7</td>
<td>673.7</td>
</tr>
<tr>
<td>QATM</td>
<td>74.88</td>
<td>0.0</td>
<td>103.3</td>
<td>34.8</td>
</tr>
<tr>
<td>UMMS</td>
<td>78.73</td>
<td>0.0</td>
<td>650.0</td>
<td>650.0</td>
</tr>
</tbody>
</table>

*Table 7.4 Base solution (Example 5)*

**Step 2:**
- According to the crosses in Table 7.4 (inserted by the user), the division of sets becomes:
  - Set 1: \{ALSH, DUBA, ENFI, MIRI, MLOC, NIDO, QATL, QATM\}.
  - Set 2: \{UMMS\}
  - Set 3: All other crude oils available in this model (outside the base crude diet, i.e. a maximum allowed purchase amount of zero in the base solution)
- The corresponding minimum and maximum allowed purchase amounts in the different sets are now calculated by MAT, as described in Section 7.3.1.

**Step 3:**
- The user has defined a relative selection criterion with \(d = 10\%\) in this case. This means that when the utility index of the crude oil is in the highest 10%-quantile, a case for the corresponding crude oil will be prepared and run with the Cargo Analysis method. Then, according to the user, this crude oil could be a good displacer for the balancing crude oil UMMS.

**Step 4:**
- In columns two and three of Table E.1 in Appendix E the values of \(X_{\text{MIN}}\) and \(X_{\text{MAX}}\) are shown for the crude oils in Set 3. These values are calculated by running Ranging Analysis in PIMS using the minimum and maximum values of Step 2. These calculated values are automatically sent to MAT.

**Step 5:**
- The corresponding utility indices for the crude oils in Set 3 are shown in column four of Table E.1. This table is also shown in MAT. In the last column of this table it is shown whether or not a case will be prepared for PlanStar/PIMS for the corresponding crude oil.

The new cases, which are ready to be transferred to PlanStar/PIMS, are shown in Table 7.5. Note that the additional purchase amounts of these crude oils are equal to 80 kiloton. These are standard amounts in the petroleum industry, but can be adjusted by the user. Also notice that the number of cases now equals 9, whereas when not using the Ranging
Analysis procedure, the number of cases is equal to 94. However, the question of whether or not these cases are indeed the most appropriate ones will be answered using a small case comparison in Section 7.5.

<table>
<thead>
<tr>
<th>Case name</th>
<th>Crude oil</th>
<th>MIN</th>
<th>MAX</th>
<th>FIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>T ANGS H801</td>
<td>ANGS</td>
<td>80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T BRNL H801</td>
<td>BRNL</td>
<td>80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T KATA H801</td>
<td>KATA</td>
<td>80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T KEKW H801</td>
<td>KEKW</td>
<td>80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T MADU H801</td>
<td>MADU</td>
<td>80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T SENI H801</td>
<td>SENI</td>
<td>80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T SLLG H801</td>
<td>SLLG</td>
<td>80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T TAPU H801</td>
<td>TAPU</td>
<td>80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T WSEN H801</td>
<td>WSEN</td>
<td>80</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7.5 New cases with additional crude oils in Set 3 (Example 5)

This whole procedure has been programmed in Excel, using Microsoft Visual Basic. Hence a new, upgraded version of MAT is available, with an implementation of the Ranging Analysis feature. The added module in MAT is shown in Appendix F, together with the summarized contents and points to improve.

7.5 Comparison of the current and new method

After the complete process has been implemented, it can be tested whether or not the idea of deciding which cases will be run by looking at their utility indices, results in running the most attractive and profitable cases (according to their Crude Indifference Values). In other words, do the current method Cargo Analysis and the method with Ranging Analysis show similar results, such that the functionality of the Marginal Analysis Tool could be improved, given the differences described in Section 7.3?

7.5.1 Comparison of both methods (Example 8)

Two refineries Y and Z are run with both methods to compare the rank of the utility indices with the rank of the CIV’s. Then one can see whether there is a positive correlation between both ranks. The results of these runs are shown in Appendix G. To clarify these tables, the crude oils are sorted by CIV rank and plotted in Figure 7.1 and 7.2.
When looking at both figures, it is clear that the blue and pink curves do not show similarities and do not have a positive correlation. This means that the ranks do not show similarities and hence the utility indices are, according to these two models, not appropriate to indicate for which crude oils a case has to be prepared and run. If, for instance, the ten ‘best’ crude oils have to be prepared, the ten crude oils with the lowest rank will be chosen. According to the CIV’s (blue curve), the ten crude oils at the left side of the figure gain the highest marginal profit, when buying 80 kiloton of this crude oil. On the other hand, according to the utility indices (pink curve), the ten lowest points in both
figures are scattered over the whole x-axis and hence it are mostly other crude oils. So, these crude oils do not give the highest marginal profit, although according to the new method they seem to do. In other words, the difference between the rank of cases that are chosen by utility indices and that are chosen by CIV’s are extremely different, such that the utility indices cannot be used for this purpose. The main reason of this dissimilarity is the first difference of Section 7.3. Thus the base solution of Ranging Analysis will usually be significantly different than the base solution of Cargo Analysis (with only crude oils from the base crude diet) and therefore the ranks for both methods do also not correlate.

7.5.2 Conclusions and motivation

One can conclude that, according to these two models, Ranging Analysis is indeed not useful to increase the functionality of the Marginal Analysis Tool, by choosing the most appropriate cases based on the highest utility indices. It means that the differences between the current method and the one with Ranging Analysis, which are explained in Section 7.3, are (as expected) of such a degree that different results are obtained. The results of Cargo Analysis are more accurate, since they are calculated afterwards. It is therefore not attractive to replace the current version of Cargo Analysis, since the results of the implemented version with Ranging Analysis are not good enough. Also notice that the time of the complete process is at this moment not significantly less than in the old method. Even though the number of cases to be run has been reduced, the time required for running Ranging Analysis to obtain the utility indices is large, such that the completion time is approximately equal. However, the case study is still very useful since it assists explaining the differences to Shell and the conclusion that Ranging Analysis cannot improve Cargo Analysis in this way.

According to the case study, Ranging Analysis is not suitable for the first goal of Cargo Analysis to be implemented within the current Cargo Analysis method. This is mainly due to the first difference explained in Section 7.3. Both calculation methods are namely based on a different base case: Ranging Analysis is based on all crude oils on the market, whereas Cargo Analysis is only based on the base crude diet. However, it could still be used for other purposes. The utility and flexibility indices of the base crude oils are still relevant for the trader at two levels. On one hand, it can still be used as a way to address uncertainty in the feedstock availability (second goal of Cargo Analysis), since for this purpose only the crude oils inside the base crude diet are analysed and then Ranging Analysis works well (see Section 5.2). On the other hand, Ranging Analysis can be utilized as a strategic tool for trading. The user could e.g. identify an additional amount of a base crude oil with a high utility index as inventory and minimize crude oils with low flexibility indices. Also for this analysis the base solution only contains crude oils of the base crude diet. Utilizing this information will minimize the risk of operating sub-optimally in the case of an unforeseen change in the monthly plan. This means that the implementation of Ranging Analysis in MAT could still be useful for other purposes.

7.5.3 Recommendations

In this chapter the Ranging Analysis procedure has been implemented in the Marginal Analysis Tool to improve its functionality. As explained in Section 7.1, Ranging Analysis has only been implemented in the Cargo Analysis part of MAT. For this part it has been shown that, by looking at the case study of Section 7.5, the utility indices do not help
filtering the most attractive cases. At the end of this chapter the following recommendations for further research of Ranging Analysis and MAT may be considered:

- The first difference from Section 7.3 is caused by the structure of the Ranging Analysis procedure. As explained earlier, it is not realistic when all crude oils can be purchased in the base solution. If AspenTech adjusts the Ranging Analysis procedure, this problem can be solved. When this difference disappears, Ranging Analysis could be more useful to improve the first goal of Cargo Analysis and therefore a new study should be executed. Since the second difference of Section 7.3 is less relevant, this new study will probably gain better results.

- When using Ranging Analysis for the first goal of Cargo Analysis (see Section 7.2), all purchase amounts are free to move to obtain a reliable value. This often results in purchasing a lot of small drops of different crude oils. In reality this is often not desirable and possible. Therefore dummy variables could be added to the process to ensure that the purchase amounts of the crude oils are at least one cargo (e.g. 80 kiloton), such that no small drops can be purchased. This is more realistic and the method shows more similarities with Cargo Analysis. When the user wants for instance a minimum purchase amount of $b$ kiloton (if it is purchased) and an allowable drop in the objective value of $\varepsilon$ (which will usually be zero in this case), the purchase constraints have to be defined as:

$$b \cdot I_i \leq X_i \leq C \cdot I_i ,$$

(7.1)

where $C$ is a sufficiently large number, $X_i$ the purchase amount for crude oil $i$, and

$$I_i = \begin{cases} 1 & \text{if objective function} \geq (1 - \varepsilon) \cdot \text{base objective function} \\ 0 & \text{else,} \end{cases}$$

for crude oil $i$. Besides this, the following constraint also has to be added to ensure that not more than one crude oil is purchased:

$$\sum_{i \in [1 \ldots M]} I_i \leq 1 ,$$

(7.2)

where $M$ is the number of crude oils available on the market. Now crude oil $i$ will only be purchased when an amount of at least $b$ kiloton does not result in a decrease of the objective value (assuming $\varepsilon = 0$). Thus if a certain crude oil would have an optimal purchase amount which is less than $b$, it will not be purchased in the adjusted version of Ranging Analysis, due to the additional constraint (7.1). On the other hand, the small amount will be purchased in the current version of Ranging Analysis. Hence adding constraint (7.1) would make it more realistic and more in line with Cargo Analysis. Since, according to (7.2), there could also not be purchased more than one crude oil each case, the procedure approaches the Cargo Analysis method. Hence using this idea could improve the functionality of Ranging Analysis in combination with Cargo Analysis, for evaluating all crude oils on the market.
• Also for the Marginal Steering part of MAT, Ranging Analysis could be implemented. With Ranging Analysis one also has the opportunity to analyse product sales, capacities, and other constraints. A small investigation has been executed to evaluate the performance of Ranging Analysis in Marginal Steering. When only running the Ranging Analysis procedure for capacities and products in PIMS (without implementing the utility indices into MAT), the characteristics seem to be the same as running Ranging Analysis for crude oils. This means that the same differences as explained in Section 7.3 may appear in the Marginal Steering part of MAT. However, since Marginal Steering works significantly different than Cargo Analysis and since the whole procedure has not been implemented in MAT, a case study as described in this chapter could be executed. To perform such a case study, a similar module can be added in MAT. This has not been done in this thesis, but could be a study to perform in the future by ORTEC and/or Shell.
8 Conclusions and recommendations

The objective of this thesis was to investigate the mathematics and the functionality of the new nonlinear solution method XNLP available in PIMS. To achieve this goal, four research questions had to be answered. In Section 8.1 the conclusions are shown, which can be drawn concerning these questions and main goal. Furthermore, in Section 8.2 the corresponding recommendations according to this thesis and for further research are discussed.

8.1 Conclusions

The first research question of this thesis is: “How do the two solution methods Distributive Recursion and Extended Nonlinear Programming work on mathematical level?” As explained in Chapter 1, the current mode Distributive Recursion is based on the Successive Linear Programming (SLP) method, in which the nonlinear problem is represented in PIMS as a linear problem, by presuming some of the variables as known. Since PIMS transforms the NLP problem into linear subproblems, only linear solvers are used to solve the problems. To compensate the error in the guesses of some of the variables (to transform the NLP into an LP problem), an error variable is introduced, which is also optimized in every LP subproblem. Using user-specified tolerances it is decided whether the solution of the LP problem is a sufficiently good approximation of the optimal solution of the NLP problem or a new guess has to be made to start a new iteration with a new LP problem.

On the other hand, the new mode Extended Nonlinear Programming does not send an LP to a linear solver. Instead, an NLP is sent to a nonlinear solver. In PIMS there are two nonlinear solvers available, XSLP and CONOPT, which can be used in this mode. The XSLP solver is also based on a Successive Linear Programming method, but significantly differs from Distributive Recursion, since the linearization happens inside the solver and uses first order Taylor expansions. The other nonlinear solver CONOPT is based on a Generalized Reduced Gradient method. The idea of this method is that under certain assumptions all variables can be expressed in terms of an independent subset of the variables, such that the number of variables is reduced.

The second research question is: “How appropriate are the nonlinear solvers, which are available in PIMS, for solving refinery-planning problems, compared to other major nonlinear solvers?” To provide a reliable answer, two other prominent nonlinear solvers (SNOPT and MINOS) have been discussed in Chapter 4. After investigating these other two solvers and comparing them to the two available in PIMS, it can be concluded that XSLP and CONOPT are in general appropriate solvers for refinery-planning problems.

The third question of this thesis was: “To what extent does XNLP provide more accurate and higher quality solutions in the refinery planning area?” Therefore a case study has been executed, which shows us that for ten refinery models of Shell Extended Nonlinear Programming does not perform on average better than Distributive Recursion. Though
AspenTech claims less local optima when using XNLP, the objective function values are usually not higher than when using DR. Besides that, the computation times are, especially for the larger models, significantly larger in the XNLP mode.

Although the performance of XNLP may not be better than DR at this moment, there are some additional features available in the XNLP mode, which may be very useful for the user. The first feature Ranging Analysis gives the user the opportunity to analyse the range under optimality conditions, for each feedstock, product, or capacity. A second additional feature, called Parametric Analysis, can be used to evaluate cases in which variables are analysed when prices of components or products have changed. The last additional feature is Goal Programming. Using this feature, the model can be optimized with a secondary (and tertiary) goal in addition to the primal goal of maximizing the profit.

The additional feature Ranging Analysis can be used as a way to address uncertainty in the feedstock availability, which is one of the goals of Cargo Analysis in the Marginal Analysis Tool. It can also be utilized as a strategic tool for trading, such that the risk of operating sub-optimally in the case of an unforeseen change in the monthly plan has been minimized. In this way one of the additional features could therefore be used to improve the Marginal Analysis Tool (fourth research question).

It is also shown in this thesis that Ranging Analysis could not improve the functionality of Cargo Analysis, when the goal is to evaluate whether there are opportunities to increase the refinery margin. This is done by purchasing other crude oils in the base solution, instead of a crude oil in the base crude diet. The main reason that that Ranging Analysis does not improve the functionality of Cargo Analysis is that the base solution of Cargo Analysis is only based on the crude oils inside the base solution. Besides that, the base solution in Ranging Analysis could also contain the other crude oils that are available on the market. This means that this base solution will usually be significantly different than the base solution of Cargo Analysis (with only crude oils from the base crude diet), which results in different outcomes. This result has been obtained by investigating both methods and implementing the Ranging Analysis procedure in MAT, such that it could be compared with the current Cargo Analysis method.

Hence by answering these four research questions, the mathematics and the functionality of the new nonlinear solution method XNLP available in PIMS are investigated and a good overview of the solution methods in PIMS has been gained.

8.2 Recommendations & further research

8.2.1 Recommendations

According to the conclusions above, the recommendation is to keep using Distributive Recursion to solve the refinery-planning problems at the refineries of Shell. At this moment Extended Nonlinear Programming does not outperform Distributive Recursion, by looking at the objective function values and computation times of the case study in Chapter 6.
Using Chapter 7 it is also shown that Ranging Analysis does not improve the Cargo Analysis procedure in MAT, when the goal is to evaluate opportunities to increase the refinery margin. Hence running all possible cases is at this moment still the best manner. However, the trader at the refinery could still use Ranging Analysis in MAT, as explained in the previous section.

8.2.2 Further Research

In this thesis Distributive Recursion has been compared to Extended Nonlinear Programming. In practice, only the XSLP solver of XNLP has been used, due to the absence of a CONOPT license. Using the CONOPT solver may give other results than using the XSLP solver and therefore evaluating this solver is a good subject for further research.

Besides Cargo Analysis, the other part of MAT, which is called Marginal Steering, could also be evaluated in combination with Ranging Analysis. The principle of Marginal Steering is slightly different than Cargo Analysis and it may therefore give other results than the research of Cargo Analysis in Chapter 7.

In this thesis only the Ranging Analysis feature has been implemented in the Marginal Analysis Tool. Further research about the possibility to implement the other additional features (Parametric Analysis and Goal Programming) in MAT may give interesting results and might increase the functionality of MAT.

It may also be considered to discuss the bottlenecks and pitfalls, which have been found during the whole process, with AspenTech. AspenTech could also explain why they believe optimizing with XNLP results in less local optima. Discussing the first two recommendations of Subsection 7.5.3 could assist improving Ranging Analysis, by making it more compatible for Shell. Then it could also be discussed why they think that the additional output, generated with XNLP, could be useful for Shell. Note that XNLP therefore first has to perform better than DR before considering using the additional output.
References


Appendix A – Continuation of Example 2

Given the parameter values shown in Table 2.6, the second recursion of the Distributive Recursion method results in activities that differ from those of the first recursion and are shown in Table A.1.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Activity (barrels per day)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>40.62</td>
</tr>
<tr>
<td>$x_2$</td>
<td>197</td>
</tr>
<tr>
<td>$x_3$</td>
<td>200</td>
</tr>
<tr>
<td>$x_4$</td>
<td>59.38</td>
</tr>
<tr>
<td>$x_5$</td>
<td>337.56</td>
</tr>
<tr>
<td>$x_ε$</td>
<td>71.54</td>
</tr>
</tbody>
</table>

**Optimal Value (dollar)**

| Objective | 515.75 |

Table A.1 Optimal solution of recursion two (Example 2)

The most significant change, compared with recursion one, is the increase of the purchase of FTV ($x_2$). This additional amount moves almost entirely to blend $b$. This is due to the adjustment of the guess of the direction proportion $s$ of the pool. Also notice that the absolute value of this error term is smaller than the error term of recursion one. Now this error term is not equally divided between the two blends, but 13.2% to blend $a$ and 86.8% to blend $b$.

Once more it can be tested whether the corresponding sulphur percentages of the two blends are at their upperbound after this recursion:

\[
\begin{align*}
\text{Blend } a: & \quad \frac{q_1x_1 + q_4x_4 + d_εx_ε}{x_1 + x_4} = \frac{3.5 \times 40.62 + 0.816 \times 59.38 - 0.132 \times 71.54}{100} = 1.81\% S \\
\text{Blend } b: & \quad \frac{q_5x_5 + d_εx_ε}{x_5} = \frac{0.816 \times 337.56 - 0.868 \times 71.54}{337.56} = 0.632\% S.
\end{align*}
\]

Thus, it can be concluded that both blends do not contain their maximum allowable sulphur percentage in this recursion.

With the composition of POOL from 197 barrels/day FTV ($= x_2$) and 200 barrels/day TVG ($= x_3$), the calculated sulphur percentage of the pool equals

\[
\frac{q_2x_2 + q_3x_3}{x_2 + x_3} = \frac{1.5 \times 197 + 0.5 \times 200}{197 + 200} = 0.996\% S.
\]
Now we see that the sulphur percentage is almost equal to the maximum allowable percentage, since the sulphur percentage of blend b can be at most 1 %S.

To check whether these outcomes are appropriate for the original model, ATol and RTol have to be calculated once more:

\[
ATol = |0.996 - 0.816| = 0.180 > 0.001 \\
RTol = \frac{|0.996 - 0.816|}{0.816} = 0.221 > 0.001.
\]

Thus one can conclude that a new recursion has to be done, to obtain a sufficiently good solution. Therefore, the new \( d_\varepsilon \) has to be calculated again:

\[
d_\varepsilon = \frac{59.38}{197 + 200} = 0.150.
\]

Hence, the presumed parameter values for recursion three become:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_b )</td>
<td>0.996</td>
</tr>
<tr>
<td>( d_\varepsilon )</td>
<td>0.150</td>
</tr>
</tbody>
</table>

*Table A.2 Presumed parameter values for recursion three (Example 2)*

These recursions are automatically repeated until the tolerances are small enough and an appropriate, optimal solution has been obtained. For this example, it takes four recursions to obtain a solution with tolerances smaller than 0.001. It results in the following activities and objective value shown in Table A.3.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Activity (barrels per day)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>40</td>
</tr>
<tr>
<td>( x_2 )</td>
<td>200</td>
</tr>
<tr>
<td>( x_3 )</td>
<td>200</td>
</tr>
<tr>
<td>( x_4 )</td>
<td>60</td>
</tr>
<tr>
<td>( x_5 )</td>
<td>340</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Optimal Value (dollar)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objective</td>
</tr>
</tbody>
</table>

*Table A.3 Optimal solution determined with distributive recursion (Example 2)*

As shown in Table A.3, the components \( FTV \) and \( TVG \) are purchased maximally, in contrast to the amount of \( SRS \). Also the optimal distribution term can now be calculated:

\[
d_\varepsilon = \frac{60}{200 + 200} = 0.150.
\]
Thus in the optimal solution, 15% of the petroleum from the pool is directed to blend $a$ and 85% to blend $b$. The sulphur percentages of the pool and of both blends can also be determined again and are shown in Table A.4.

<table>
<thead>
<tr>
<th>Component</th>
<th>Sulphur percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>POOL</td>
<td>1 %S</td>
</tr>
<tr>
<td>blend $a$</td>
<td>2 %S</td>
</tr>
<tr>
<td>blend $b$</td>
<td>1 %S</td>
</tr>
</tbody>
</table>

*Table A.4 Sulphur percentages in optimal solution (Example 2)*

Thus we can conclude that in the optimal solution both blends contain the maximum allowable percentage of sulphur.
Appendix B – Continuation of Example 4

**Iteration 2**

Step 1: The Jacobian $A$ and the corresponding vector $b$ can now be calculated, using the new values of $x$ at $t=1$.

$$A = JH(x_{t=1}) = \begin{pmatrix} N & B \end{pmatrix} = \begin{pmatrix} \frac{\partial h_i}{\partial x_1} & \frac{\partial h_i}{\partial x_2} \end{pmatrix} = \begin{pmatrix} 2x_{1|1} & 4x_{2|1} \end{pmatrix} = \begin{pmatrix} 4.888 & 15.500 \end{pmatrix},$$

and

$$b = Ax_{t=1} = \begin{pmatrix} 4.888 & 15.500 \end{pmatrix} \begin{pmatrix} 2.444 \\ 3.875 \end{pmatrix} = 72.000.$$  

Step 2: Now the basic variable $x_{2|1}$ can once more be written as a function of the non-basic variable $x_{1|1}$:

$$x_2 = B^{-1}b - B^{-1}Nx_1 = 4.645 - 3.154 \cdot 10^{-1} x_1.$$  

Step 3: Using this, model (3.7) can be written as a model with only the variable $x_{1|1}$. For simplicity, the term $t=1$ has been omitted.

$$\begin{align*}
\text{min} & \quad 2.398 x_1^2 - 13.720 x_1 + 86.304 \\
\text{s.t.} & \quad 4.645 - 3.154 \cdot 10^{-1} x_1 \geq 0 \\
& \quad x_1 \geq 0.
\end{align*} \tag{B.1}$$

Step 4: Now the search directions at $t=1$ of the basic and the non-basic part have to be determined:

$$s_{N|1} = s_{1|1} = -\delta \hat{f}_N(x_{1|1}) = -4.796 x_{1|1} + 13.720 = 2.000,$$

and

$$s_{B|1} = s_{2|1} = -B^{-1}Ns_{N|1} = -\frac{1}{15.500} 4.889 \cdot 2.000 = -6.308 \cdot 10^{-1}.$$  

Step 5: Again the expressions for the new variables can be determined by using the search directions determined above.

$$x_{1|2} = x_{1|1} + \lambda s_{N|1} = 2.444 + 2.000 \lambda \geq 0 \tag{B.2a}$$
\[ x_{2r=2} = x_{2r=1} + \lambda s_{Bt=1} = 3.875 - 6.308 \cdot 10^{-1} \lambda \geq 0. \] (B.2b)

**Step 6:** At this iteration, \( \lambda \) cannot exceed the value \( \lambda_{\text{max}} = 6.143 \), according to formula (B.2b).

Now filling in the new value of \( x_1 \) (by using formula (B.2a)), the objective function of model (B.1) can be minimized with respect to \( \lambda \). This results in the value \( \lambda^* = 2.084 \cdot 10^{-1} \).

Since \( \lambda^* < \lambda_{\text{max}} \) the optimal value of \( \lambda \) equals

\[ \lambda_2 = 2.084 \cdot 10^{-1}. \]

**Step 7:** To obtain the new values of \( x_1 \) and \( x_2 \), the value of \( \lambda_2 \) has to be inserted into expressions (B.2a) and (B.2b). This results in

\[ x_{r=2} = (x_{lr=2}, x_{2r=2})^T = (2.861, 3.744)^T. \]

**Step 8:** Since \(|H(x_1)| = 0.220 > \varepsilon\), a next iteration has to be executed. Once more, \( x_N = x_1 \) will be fixed and \( x_B = x_2 \) will be chosen such that the constraint of the original model (3.7) holds, thus \( x_{r=2} = (2.861, 3.729)^T \) with objective value 66.270. Thus these values are the new starting values for the next iteration, where \( x_2 \) will again be used as the basic variable, since \( x_{lr=2} \) stayed positive.

**Iteration 3**

**Step 1:** At this iteration, the Jacobian \( A \) and the corresponding vector \( b \) equal

\[
A = (N \quad B) = \begin{pmatrix} 2x_{lr=2} & 4x_{2r=2} \end{pmatrix} = \begin{pmatrix} 5.722 & 14.916 \end{pmatrix},
\]

and

\[
b = Ax_{r=2} = \begin{pmatrix} 5.722 & 14.916 \end{pmatrix} \begin{pmatrix} 2.861 \\ 3.729 \end{pmatrix} = 72.000.
\]

**Step 2 & Step 3:** Using these values, model (3.7) can again be written as a model with only one variable, namely the basic variable \( x_{lr=2} \), in a similar way as in the previous iterations. For simplicity, the term \( t=2 \) has been omitted.

\[
\min \quad 2.588x_1^2 - 16.812x_1 + 93.200
\]

\[
s.t. \quad 4.827 - 3.836 \cdot 10^{-1} \geq 0 \quad \quad x_1 \geq 0. \] (B.3)

**Step 4:** Consequently, the search directions in this iteration become:
\[ s_{Nt=2} = s_{Bt=2} = -5.176 x_{fr=2} + 16.812 = 2.003, \]

and
\[ s_{Bt=2} = s_{2t=2} = -7.684 \cdot 10^{-1}. \]

**Step 5:** Now the variables at \( t=3 \) as a function of \( \lambda \) will equal:

\[ x_{1t=3} = 2.861 + 2.003 \lambda \geq 0 \]  \hspace{1cm} (B.4a)

\[ x_{2t=3} = 3.729 - 7.684 \cdot 10^{-1} \lambda \geq 0. \]  \hspace{1cm} (B.4b)

**Step 6:** From (B.3) and (B.4) one obtains \( \lambda_{\text{max}} = 4.853 \) and \( \lambda^* = 1.932 \cdot 10^{-1}. \)

Since \( \lambda^* < \lambda_{\text{max}} \) the optimal value of \( \lambda \) in this iteration equals

\[ \lambda_3 = 1.932 \cdot 10^{-1}. \]

**Step 7:** To obtain the new values of \( x_1 \) and \( x_2 \), the value of \( \lambda_3 \) has to be inserted into expressions (B.4a) and (B.4b). This results in

\[ x_{r=3} = (x_{fr=3} \quad x_{2fr=3})^T = (3.248 \quad 3.581)^T. \]

**Step 8:** Since \( |H(x_{r=3})| = 0.197 \geq \varepsilon \), a fourth iteration has to be executed. Notice that the objective value has been improved and the left hand sides of the constraints are closer to zero. To omit the violation of the constraint, \( x_{fr=3} \) will again be adjusted. Thus for the next iteration (where \( i=3 \)), \( x_{r=3} = (x_{fr=3} \quad x_{2fr=3})^T = (3.248 \quad 3.567)^T \) with an objective value of 65.497. These values are the new starting values for the next iteration, where \( x_2 \) will again be used as the basic variable, since \( x_{fr=2} \) stayed positive.

**Iteration 4**

This iteration results in \( x_{r=4} = (x_{fr=4} \quad x_{2fr=4})^T = (3.602 \quad 3.406)^T \), where \( |H(x_{r=4})| = 0.176 \). Correcting the value of \( x_{fr=4} \) gives us

\[ x_{r=4} = (x_{fr=4} \quad x_{2fr=4})^T = (3.602 \quad 3.393)^T, \]

with an objective value of 64.795. Since \( |H(x_{r=4})| > \varepsilon \), a fifth iteration has to be executed.

**Iteration 5**
In this iteration, $x_{tr5} = (3.922 \ 3.223)^T$ and $|H(x_{tr4})| = 0.158$, which is smaller than $\varepsilon$. That means that the algorithm stops after this iteration. The calculated value for $x_{pr5} = x_{tr5}$ has to be adjusted, to obtain the right solution. This eventually gives us the optimal solution $x_{tr5} = (3.922 \ 3.211)^T$ with an objective value 64.162.
## Appendix C – Summary statistics of Example 5

<table>
<thead>
<tr>
<th>Feedstocks</th>
<th>Amount (bbls/day)</th>
<th>Marginal value</th>
<th>MIN (bbls/day)</th>
<th>MAX (bbls/day)</th>
<th>Price ($/day)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabian Heavy</td>
<td>25.196</td>
<td>0.000</td>
<td>0.00</td>
<td>50.00</td>
<td>15.00</td>
</tr>
<tr>
<td>Alaskan N Slope</td>
<td>5.722</td>
<td>0.000</td>
<td>0.00</td>
<td>33.00</td>
<td>17.50</td>
</tr>
<tr>
<td>Arabian Light</td>
<td>23.804</td>
<td>0.000</td>
<td>20.00</td>
<td>50.00</td>
<td>17.00</td>
</tr>
<tr>
<td>Bachequero</td>
<td>1.000</td>
<td>-0.754</td>
<td>1.00</td>
<td>8.00</td>
<td>14.75</td>
</tr>
<tr>
<td>Kuwait Export</td>
<td>0.000</td>
<td>0.000</td>
<td>0.00</td>
<td>20.00</td>
<td>16.25</td>
</tr>
<tr>
<td>North Sea Forties</td>
<td>20.000</td>
<td>0.594</td>
<td>0.00</td>
<td>20.00</td>
<td>18.30</td>
</tr>
<tr>
<td>Tiajuana Light</td>
<td>10.000</td>
<td>0.000</td>
<td>0.00</td>
<td>25.00</td>
<td>17.70</td>
</tr>
</tbody>
</table>

*Table C.1 Summary statistics (Example 5)*
Appendix D – Refineries for case study

Refinery A:

<table>
<thead>
<tr>
<th></th>
<th>DR</th>
<th>XNLP</th>
<th>Relative changes of XNLP w.r.t. DR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objective value ($)</td>
<td>58115</td>
<td>58082</td>
<td>-0.0568%</td>
</tr>
<tr>
<td>Computation time (min)</td>
<td>00:55</td>
<td>00:51</td>
<td>-0.0727%</td>
</tr>
<tr>
<td>Nr. of nonzero matrix elements</td>
<td>10053</td>
<td>10053</td>
<td>0%</td>
</tr>
</tbody>
</table>

*Table D.1 Results of Refinery A*

Refinery B:

<table>
<thead>
<tr>
<th></th>
<th>DR</th>
<th>XNLP</th>
<th>Relative changes of XNLP w.r.t. DR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objective value ($)</td>
<td>8682</td>
<td>9784</td>
<td>+12.7%</td>
</tr>
<tr>
<td>Computation time (min)</td>
<td>01:23</td>
<td>01:46</td>
<td>+18.7%</td>
</tr>
<tr>
<td>Nr. of nonzero matrix elements</td>
<td>31551</td>
<td>32050</td>
<td>+15.8%</td>
</tr>
</tbody>
</table>

*Table D.2 Results of Refinery B*

Refinery C:

<table>
<thead>
<tr>
<th></th>
<th>DR</th>
<th>XNLP</th>
<th>Relative changes of XNLP w.r.t. DR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objective value ($)</td>
<td>10980</td>
<td>-12935</td>
<td>-217%</td>
</tr>
<tr>
<td>Computation time (min)</td>
<td>1:24</td>
<td>2:02</td>
<td>+62.9%</td>
</tr>
<tr>
<td>Nr. of nonzero matrix elements</td>
<td>32254</td>
<td>32254</td>
<td>0%</td>
</tr>
</tbody>
</table>

*Table D.3 Results of Refinery C*

Refinery D:

<table>
<thead>
<tr>
<th></th>
<th>DR</th>
<th>XNLP</th>
<th>Relative changes of XNLP w.r.t. DR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objective value ($)</td>
<td>13312</td>
<td>13322</td>
<td>+0.0751%</td>
</tr>
<tr>
<td>Computation time (min)</td>
<td>1:31</td>
<td>2:03</td>
<td>+55.0%</td>
</tr>
<tr>
<td>Nr. of nonzero matrix elements</td>
<td>74226</td>
<td>74226</td>
<td>0%</td>
</tr>
</tbody>
</table>

*Table D.4 Results of Refinery D*

Refinery E:

<table>
<thead>
<tr>
<th></th>
<th>DR</th>
<th>XNLP</th>
<th>Relative changes of XNLP w.r.t. DR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objective value ($)</td>
<td>11201</td>
<td>11113</td>
<td>-0.786%</td>
</tr>
<tr>
<td>Computation time (min)</td>
<td>01:09</td>
<td>01:29</td>
<td>+18.3%</td>
</tr>
<tr>
<td>Nr. of nonzero matrix elements</td>
<td>78792</td>
<td>75624</td>
<td>-4.02%</td>
</tr>
</tbody>
</table>

*Table D.5 Results of Refinery E*
### Refinery F:

<table>
<thead>
<tr>
<th></th>
<th>DR</th>
<th>XNLP</th>
<th>Relative changes of XNLP w.r.t. DR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objective value ($)</td>
<td>92584</td>
<td>92011</td>
<td>-0.619%</td>
</tr>
<tr>
<td>Computation time (min)</td>
<td>06:54</td>
<td>19:02</td>
<td>+191%</td>
</tr>
<tr>
<td>Nr. of nonzero matrix elements</td>
<td>81598</td>
<td>82438</td>
<td>+1.03%</td>
</tr>
</tbody>
</table>

Table D.6 Results of Refinery F

### Refinery G:

<table>
<thead>
<tr>
<th></th>
<th>DR</th>
<th>XNLP</th>
<th>Relative changes of XNLP w.r.t. DR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objective value ($)</td>
<td>28749</td>
<td>29557</td>
<td>+2.81%</td>
</tr>
<tr>
<td>Computation time (min)</td>
<td>05:28</td>
<td>09:52</td>
<td>+80.3%</td>
</tr>
<tr>
<td>Nr. of nonzero matrix elements</td>
<td>147979</td>
<td>137239</td>
<td>-7.26%</td>
</tr>
</tbody>
</table>

Table D.7 Results of Refinery G

### Refinery H:

<table>
<thead>
<tr>
<th></th>
<th>DR</th>
<th>XNLP</th>
<th>Relative changes of XNLP w.r.t. DR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objective value ($)</td>
<td>24403</td>
<td>24422</td>
<td>+0.0779%</td>
</tr>
<tr>
<td>Computation time (min)</td>
<td>5:31</td>
<td>18:53</td>
<td>+249%</td>
</tr>
<tr>
<td>Nr. of nonzero matrix elements</td>
<td>161638</td>
<td>139361</td>
<td>-13.8%</td>
</tr>
</tbody>
</table>

Table D.8 Results of Refinery H

### Refinery I:

<table>
<thead>
<tr>
<th></th>
<th>DR</th>
<th>XNLP</th>
<th>Relative changes of XNLP w.r.t. DR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objective value ($)</td>
<td>41599</td>
<td>46541</td>
<td>+11.9%</td>
</tr>
<tr>
<td>Computation time (min)</td>
<td>04:15</td>
<td>13:39</td>
<td>+223%</td>
</tr>
<tr>
<td>Nr. of nonzero matrix elements</td>
<td>181639</td>
<td>163403</td>
<td>-10.0%</td>
</tr>
</tbody>
</table>

Table D.9 Results of Refinery I

### Refinery J:

<table>
<thead>
<tr>
<th></th>
<th>DR</th>
<th>XNLP</th>
<th>Relative changes of XNLP w.r.t. DR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objective value ($)</td>
<td>72759</td>
<td>79165</td>
<td>+8.80%</td>
</tr>
<tr>
<td>Computation time (min)</td>
<td>11:42</td>
<td>51:34</td>
<td>+350%</td>
</tr>
<tr>
<td>Nr. of nonzero matrix elements</td>
<td>365926</td>
<td>349426</td>
<td>-4.51%</td>
</tr>
</tbody>
</table>

Table D.10 Results of Refinery J

84
### Appendix E – Outcomes of Example 7

<table>
<thead>
<tr>
<th>Crude oil</th>
<th>Xbase</th>
<th>Xmax</th>
<th>Utility index</th>
<th>Appropriate to run case</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMNA</td>
<td>0.00</td>
<td>8.13</td>
<td>0.18</td>
<td>NO</td>
</tr>
<tr>
<td>ANGS</td>
<td>0.00</td>
<td>16.71</td>
<td>0.37</td>
<td>YES</td>
</tr>
<tr>
<td>ARAH</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>NO</td>
</tr>
<tr>
<td>ARAL</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>NO</td>
</tr>
<tr>
<td>ARAM</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>NO</td>
</tr>
<tr>
<td>ARJU</td>
<td>0.00</td>
<td>13.37</td>
<td>0.30</td>
<td>NO</td>
</tr>
<tr>
<td>ARSL</td>
<td>0.00</td>
<td>7.34</td>
<td>0.16</td>
<td>NO</td>
</tr>
<tr>
<td>ARUN</td>
<td>0.00</td>
<td>10.50</td>
<td>0.24</td>
<td>NO</td>
</tr>
<tr>
<td>ARXL</td>
<td>0.00</td>
<td>8.63</td>
<td>0.19</td>
<td>NO</td>
</tr>
<tr>
<td>BAHO</td>
<td>0.00</td>
<td>8.52</td>
<td>0.19</td>
<td>NO</td>
</tr>
<tr>
<td>BASL</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>NO</td>
</tr>
<tr>
<td>BENC</td>
<td>0.00</td>
<td>8.59</td>
<td>0.19</td>
<td>NO</td>
</tr>
<tr>
<td>BINC</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>NO</td>
</tr>
<tr>
<td>BONL</td>
<td>0.00</td>
<td>8.04</td>
<td>0.18</td>
<td>NO</td>
</tr>
<tr>
<td>BREB</td>
<td>0.00</td>
<td>7.54</td>
<td>0.17</td>
<td>NO</td>
</tr>
<tr>
<td>BRNL</td>
<td>0.00</td>
<td>13.90</td>
<td>0.31</td>
<td>YES</td>
</tr>
<tr>
<td>BUKH</td>
<td>13.68</td>
<td>16.22</td>
<td>0.08</td>
<td>NO</td>
</tr>
<tr>
<td>BUKN</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>NO</td>
</tr>
<tr>
<td>CABD</td>
<td>0.00</td>
<td>7.12</td>
<td>0.16</td>
<td>NO</td>
</tr>
<tr>
<td>CAKE</td>
<td>16.11</td>
<td>19.47</td>
<td>0.12</td>
<td>NO</td>
</tr>
<tr>
<td>CANU</td>
<td>0.00</td>
<td>2.75</td>
<td>0.06</td>
<td>NO</td>
</tr>
<tr>
<td>CHEX</td>
<td>0.00</td>
<td>4.20</td>
<td>0.09</td>
<td>NO</td>
</tr>
<tr>
<td>COSB</td>
<td>0.00</td>
<td>9.20</td>
<td>0.21</td>
<td>NO</td>
</tr>
<tr>
<td>CYRU</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>NO</td>
</tr>
<tr>
<td>DOBA</td>
<td>0.00</td>
<td>4.03</td>
<td>0.09</td>
<td>NO</td>
</tr>
<tr>
<td>DURI</td>
<td>0.00</td>
<td>2.24</td>
<td>0.05</td>
<td>NO</td>
</tr>
<tr>
<td>ERHA</td>
<td>0.00</td>
<td>7.01</td>
<td>0.16</td>
<td>NO</td>
</tr>
<tr>
<td>ESPA</td>
<td>0.00</td>
<td>2.46</td>
<td>0.06</td>
<td>NO</td>
</tr>
<tr>
<td>FORC</td>
<td>0.00</td>
<td>5.77</td>
<td>0.13</td>
<td>NO</td>
</tr>
<tr>
<td>GIRA</td>
<td>0.00</td>
<td>6.75</td>
<td>0.15</td>
<td>NO</td>
</tr>
<tr>
<td>GULS</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>NO</td>
</tr>
<tr>
<td>HAND</td>
<td>0.00</td>
<td>10.30</td>
<td>0.23</td>
<td>NO</td>
</tr>
<tr>
<td>HOUT</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>NO</td>
</tr>
<tr>
<td>IBMS</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>NO</td>
</tr>
<tr>
<td>IRNH</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>NO</td>
</tr>
<tr>
<td>IRNL</td>
<td>0.00</td>
<td>1.76</td>
<td>0.04</td>
<td>NO</td>
</tr>
<tr>
<td>JATI</td>
<td>0.00</td>
<td>2.69</td>
<td>0.06</td>
<td>NO</td>
</tr>
<tr>
<td>KAJI</td>
<td>0.00</td>
<td>9.48</td>
<td>0.21</td>
<td>NO</td>
</tr>
<tr>
<td>KATA</td>
<td>0.00</td>
<td>16.75</td>
<td>0.38</td>
<td>YES</td>
</tr>
<tr>
<td>KEKW</td>
<td>0.00</td>
<td>14.15</td>
<td>0.32</td>
<td>YES</td>
</tr>
<tr>
<td>KHAF</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>NO</td>
</tr>
<tr>
<td>KHUF</td>
<td>0.00</td>
<td>12.24</td>
<td>0.27</td>
<td>NO</td>
</tr>
<tr>
<td>KIDU</td>
<td>0.00</td>
<td>9.17</td>
<td>0.21</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td></td>
</tr>
<tr>
<td>KIKE</td>
<td>0.00</td>
<td>9.67</td>
<td>0.22</td>
<td></td>
</tr>
<tr>
<td>KIRK</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>KUWA</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>LABU</td>
<td>0.00</td>
<td>6.25</td>
<td>0.14</td>
<td></td>
</tr>
<tr>
<td>LALA</td>
<td>0.00</td>
<td>8.80</td>
<td>0.20</td>
<td></td>
</tr>
<tr>
<td>LAVA</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>MADU</td>
<td>0.00</td>
<td>15.19</td>
<td>0.34</td>
<td></td>
</tr>
<tr>
<td>MALA</td>
<td>0.00</td>
<td>7.75</td>
<td>0.17</td>
<td></td>
</tr>
<tr>
<td>MALC</td>
<td>0.00</td>
<td>2.40</td>
<td>0.05</td>
<td></td>
</tr>
<tr>
<td>MALT</td>
<td>0.00</td>
<td>3.48</td>
<td>0.08</td>
<td></td>
</tr>
<tr>
<td>MARM</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>MASI</td>
<td>0.00</td>
<td>5.55</td>
<td>0.12</td>
<td></td>
</tr>
<tr>
<td>MESL</td>
<td>0.00</td>
<td>5.77</td>
<td>0.13</td>
<td></td>
</tr>
<tr>
<td>MUDI</td>
<td>0.00</td>
<td>13.14</td>
<td>0.29</td>
<td></td>
</tr>
<tr>
<td>MURB</td>
<td>0.00</td>
<td>11.08</td>
<td>0.25</td>
<td></td>
</tr>
<tr>
<td>NILE</td>
<td>0.48</td>
<td>5.71</td>
<td>0.12</td>
<td></td>
</tr>
<tr>
<td>NOWR</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>NWSC</td>
<td>0.00</td>
<td>6.13</td>
<td>0.14</td>
<td></td>
</tr>
<tr>
<td>OMAN</td>
<td>0.00</td>
<td>6.81</td>
<td>0.15</td>
<td></td>
</tr>
<tr>
<td>ORIE</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>QATC</td>
<td>0.00</td>
<td>9.93</td>
<td>0.22</td>
<td></td>
</tr>
<tr>
<td>RABL</td>
<td>0.00</td>
<td>7.87</td>
<td>0.18</td>
<td></td>
</tr>
<tr>
<td>RATA</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>RGAS</td>
<td>0.00</td>
<td>8.01</td>
<td>0.18</td>
<td></td>
</tr>
<tr>
<td>SAHB</td>
<td>0.00</td>
<td>11.62</td>
<td>0.26</td>
<td></td>
</tr>
<tr>
<td>SENI</td>
<td>0.00</td>
<td>17.70</td>
<td>0.40</td>
<td></td>
</tr>
<tr>
<td>SHAC</td>
<td>0.00</td>
<td>1.42</td>
<td>0.03</td>
<td></td>
</tr>
<tr>
<td>SIRR</td>
<td>0.00</td>
<td>1.96</td>
<td>0.04</td>
<td></td>
</tr>
<tr>
<td>SLEB</td>
<td>0.00</td>
<td>9.24</td>
<td>0.21</td>
<td></td>
</tr>
<tr>
<td>SLFN</td>
<td>0.00</td>
<td>1.03</td>
<td>0.02</td>
<td></td>
</tr>
<tr>
<td>SLHG</td>
<td>0.00</td>
<td>13.14</td>
<td>0.29</td>
<td></td>
</tr>
<tr>
<td>SLLG</td>
<td>0.00</td>
<td>17.05</td>
<td>0.38</td>
<td></td>
</tr>
<tr>
<td>SLNA</td>
<td>0.00</td>
<td>9.93</td>
<td>0.22</td>
<td></td>
</tr>
<tr>
<td>SLPG</td>
<td>0.00</td>
<td>13.84</td>
<td>0.31</td>
<td></td>
</tr>
<tr>
<td>SLPK</td>
<td>0.00</td>
<td>6.45</td>
<td>0.14</td>
<td></td>
</tr>
<tr>
<td>SLRS</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>SLTP</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>SOKL</td>
<td>0.00</td>
<td>11.95</td>
<td>0.27</td>
<td></td>
</tr>
<tr>
<td>SPAR</td>
<td>0.00</td>
<td>6.07</td>
<td>0.14</td>
<td></td>
</tr>
<tr>
<td>STAG</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>TAPU</td>
<td>0.00</td>
<td>19.46</td>
<td>0.44</td>
<td></td>
</tr>
<tr>
<td>TIAK</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>UDAN</td>
<td>0.00</td>
<td>8.48</td>
<td>0.19</td>
<td></td>
</tr>
<tr>
<td>UMMS</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>UPZA</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>URAL</td>
<td>0.00</td>
<td>2.15</td>
<td>0.05</td>
<td></td>
</tr>
<tr>
<td>VITY</td>
<td>0.00</td>
<td>7.18</td>
<td>0.16</td>
<td></td>
</tr>
<tr>
<td>WSEN</td>
<td>0.00</td>
<td>15.73</td>
<td>0.35</td>
<td></td>
</tr>
<tr>
<td>XICO</td>
<td>0.00</td>
<td>4.95</td>
<td>0.11</td>
<td></td>
</tr>
<tr>
<td>ZAKU</td>
<td>0.00</td>
<td>7.28</td>
<td>0.16</td>
<td>NO</td>
</tr>
</tbody>
</table>

*Table E.1 Utility indices shown in MAT*
Appendix F – Current implementation

The algorithm described in Section 7.3.2 has been implemented in the Marginal Analysis Tool, to simplify the procedure for the user. The current contents of the upgraded MAT are:

- The minimum and maximum allowed purchase amounts are automatically adjusted for the new cases by pressing the ‘Prepare Feedstock Analysis’ button.
- The values of $X_{\text{BASE}}$ and $X_{\text{MAX}}$, obtained by Ranging Analysis in PIMS, are shown in the sheet ‘Feedstock Analysis’ in MAT. Also the utility indices are automatically calculated for all crude oils in Set 3.
- The user can specify an absolute or relative selection criterion for deciding which crude oils in Set 3 seem appropriate enough to displace the crude oils in Set 2 and thus for which a new case has to be prepared.
- Cases are automatically prepared (as in Table 7.2) for those crude oils that are, according to the user, appropriate (i.e. flexible) enough.

Points to improve of the Marginal Analysis Tool with Ranging Analysis:

- The user has to start Ranging Analysis in PIMS manually at this moment. Starting this procedure automatically would be an improvement for the user.
- At this moment, the time of running the complete procedure has not been reduced when using the new method (Ranging Analysis in combination with MAT), instead of running the current method (running all possible cases). Although the number of cases to be run has been reduced, the time required for running Ranging Analysis to obtain the utility indices is large, such that the complete time is approximately equal. Note that version 15.28 of PIMS is used in this thesis and Ranging Analysis may become less time-consuming in newer versions.
Outline:

Sub PrepareFACase_click()

    Dim li_row As Long
    Dim li_Col As Integer
    Dim li_count As Integer
    Dim li_BaseTier As Integer
    Dim li_BaseLocation As Integer
    Dim li_BaseAct As Integer
    Dim li_BasePrice As Integer
    Dim li_BaseMaterType As Integer
    Dim li_BaseMaterLabel As Integer
    Dim li_i As Integer
    Dim li_Row As Integer
    Dim lb_notinbasecase As Boolean
    Dim lv_basecrudes() As Variant
    Dim li_count2 As Integer

    Call ProcPreparePlanStar
    Application.ScreenUpdating = False
    Call ReadControlParameters
    Call ReadModeInformation
    Application.ScreenUpdating = False

    Application.StatusBar = "Creating Commtrns Cargo Analysis..."
    gs_LastCrudeCombination = ""
    gd_Offset = Range("Offset_Cargo").value
    ThisWorkbook.Worksheets("CA Crude").Activate

    Application.StatusBar = "Creating Commtrns Cargo Analysis..."
    gs_LastCrudeCombination = ""
    gd_Offset = Range("Offset_Cargo").value
    ThisWorkbook.Worksheets(gs_CargoInput_Crude).Activate

    gi_col_commtrns_RHS = Application.WorksheetFunction.Match("RHS", Rows(1), 0)
    gi_col_commtrns_Location = Application.WorksheetFunction.Match("LOCATION", Rows(1), 0)
    gi_col_commtrns_MaterLabel = Application.WorksheetFunction.Match("CRUDE", Rows(1), 0)
    gi_col_commtrns_Tier = Application.WorksheetFunction.Match("TIER", Rows(1), 0)
    gi_col_commtrns_Price = Application.WorksheetFunction.Match("PRICE", Rows(1), 0)
    gi_col_commtrns_BaseCaseCrude = Application.WorksheetFunction.Match("BASE_MATERLABEL", Rows(1), 0)
    gi_col_commtrns_BaseCaseLocation = Application.WorksheetFunction.Match("BASELOCATION", Rows(1), 0)
    gi_col_commtrns_BaseCrude = Application.WorksheetFunction.Match("BASE_CRUDE", Rows(1), 0)
    gi_col_commtrns_MIN = Application.WorksheetFunction.Match("MIN_BASE", Rows(1), 0)
    gi_col_commtrns_MAX = Application.WorksheetFunction.Match("MAX_BASE", Rows(1), 0)
    gi_col_commtrns_Actual = Application.WorksheetFunction.Match("ACT_BASE", Rows(1), 0)
    gi_col_commtrns_BaseCase = Application.WorksheetFunction.Match("BASECASE", Rows(1), 0)
gi_col_commtrns_MINcorr = Application.WorksheetFunction.Match("MIN_CORR", Rows(1), 0)

Call Empty_commtrns

' Set variables that do not change
  c_commtrns_Version = Range("modelid1")
c_commtrns_Currency = Range("currency").value
  c_commtrns_FOBPrice = "N"
c_commtrns_OrderNr = "1"
c_commtrns_TransType = "IMP"
c_commtrns_MaterType = "CRUDE"
c_commtrns_QtyUOM = Range("Uom_Quant").value
  c_commtrns_PriceUOM = c_commtrns_QtyUOM

gd_first_drop_quantity = Range("first_drop_quantity")
c_commtrns_Period = Range("PeriodOutput")

' Check all base_cases that are fixed in a mode and fix them in the base cases

'this is for both crudes and feedstocks!
ls_sheet = "CA Crude"

ll_row = gl_row_input_start
ThisWorkbook.Worksheets(ls_sheet).Activate
li_BaseTier = Application.WorksheetFunction.Match("BASETIER", Rows(1), 0)
li_BaseLocation = Application.WorksheetFunction.Match("BASELOCATION", Rows(1), 0)
li_BaseAct = Application.WorksheetFunction.Match("BASE_ACT", Rows(1), 0)
li_BasePrice = Application.WorksheetFunction.Match("BASEPRICE", Rows(1), 0)
li_BaseMaterType = Application.WorksheetFunction.Match("BASE_MATERTYPE", Rows(1), 0)
li_BaseMaterLabel = Application.WorksheetFunction.Match("BASE_MATERLABEL", Rows(1), 0)

With Sheets(ls_sheet)
  Do While .Cells(ll_row, gi_col_commtrns_BaseCaseCrude) <> ""
    For li_i = 1 To gl_nrModes
      If gc_ModesCollection(li_i).tb_crude Then
        ReDim Preserve lv_basecrudes(ll_row - 13)
        lv_basecrudes(ll_row - 13) = .Cells(ll_row, li_BaseMaterLabel)
      End If
      If gc_ModesCollection(li_i).tb_active Then 'only export if mode is active
        If .Cells(ll_row, gc_ModesCollection(li_i).tl_column) <> "" Then 'crude is fixed for this mode
          c_commtrns_Location = .Cells(ll_row, li_BaseLocation)
          c_commtrns_MaterType = .Cells(ll_row, li_BaseMaterType)
          c_commtrns_MaterLabel = .Cells(ll_row, li_BaseMaterLabel)
          c_commtrns_Qty_Min = .Cells(ll_row, li_BaseAct)
          c_commtrns_Tier = .Cells(ll_row, li_BaseTier)
          c_commtrns_Price = .Cells(ll_row, li_BasePrice)
        End If
    Next li_i
  End If
  ll_row = ll_row + 1
With EndWith
c_commtrns_Qty_Max = .Cells(ll_row, li_BaseAct)
c_commtrns_Rhs = gc_ModesCollection(li_i).ts_rhs
c_commtrns_Dmn = "BASIC"
Call AddTo_commtrns_List
If gb_MPmodel Then
    c_commtrns_Dmn = "GLOBAL"
    Call AddTo_commtrns_List
End If
Else ' crude is not fixed for this mode
    c_commtrns_Location = .Cells(ll_row, li_BaseLocation)
c_commtrns_MaterType = .Cells(ll_row, li_BaseMaterType)
c_commtrns_MaterLabel = .Cells(ll_row, li_BaseMaterLabel)
c_commtrns_Qty_Min = 0 'min is always zero when crude is not fixed
c_commtrns_Tier = .Cells(ll_row, li_BaseTier)
c_commtrns_Price = .Cells(ll_row, li_BasePrice)
If .Cells(ll_row, 41) = 0 Then
    c_commtrns_Qty_Max = 9999 'if crude is not fixed and max=0
Else
    c_commtrns_Qty_Max = .Cells(ll_row, 41)
End If
c_commtrns_Rhs = gc_ModesCollection(li_i).ts_rhs
c_commtrns_Dmn = "BASIC"
Call AddTo_commtrns_List
If gb_MPmodel Then
    c_commtrns_Dmn = "GLOBAL"
    Call AddTo_commtrns_List
End If
End If
Next li_i
ll_row = ll_row + 1
Loop
End With
li_Row = 6
While Sheets("comp_label_crudes").Cells(li_Row, 3) <> ""
    lb_notinbasecase = True
    li_count2 = 0
    While (li_count2 <> ll_row - 13) And (lb_notinbasecase = True)
        If Sheets("comp_label_crudes").Cells(li_Row, 3) = lv_basecrudes(li_count2) Then 'is the crude in comp_label_crudes in the base case??
            lb_notinbasecase = False
        End If
        li_count2 = li_count2 + 1
    Wend
    If lb_notinbasecase Then 'if not in base case, min=0, max=9999
        c_commtrns_Location = Range("location")
c_commtrns_MaterType = "CRUDE"
Sub FeedstockAnalysis()

Dim as_planstar_basecase_outputdir As String
Dim as_runid_dir_manual As String
Dim ADOConnection As ADODB.Connection
Dim ADORS As ADODB.Recordset
Dim ADOConnectionString As String
Dim RowCounter As Long
Dim lws_FASheet As Worksheet
Dim lws_GeneralSheet As Worksheet
Dim ls_VBName As String
Dim li_count As Integer
Dim li_count2 As Integer
Dim ls_crudes() As Variant
Dim ls_crudes2 As String
Dim ls_crudes2cut() As String
Dim lv_query As Variant
Dim ls_sheet As String
Dim li_length As Integer
Dim ls_RangeCase As String
Dim ls_VariableName As String
Dim ll_VariableValue As Double
Dim ls_name As String
Dim li_NumberCrudes As Integer
Dim li_loop As Integer
Dim li_CrudeNumber As Integer
Dim li_cases As Integer
Dim gs_connection As String
Dim ld_totalfeed As Double
On Error GoTo errorhandler

Option Explicit

On Error GoTo errorhandler

as_planstar_basecase_outputdir = Left(Range("planstar_basecase_outputdir"), Len(Range("planstar_basecase_outputdir"))) & "Run_10" & Range("location") & "\"

as_runid_dir_manual = Left(Range("planstar_basecase_outputdir"), Len(Range("planstar_basecase_outputdir"))) & "Run_10" & Range("location") & "\Manual\"

gs_connection = "ODBC;DSN=" + Range("dsn") + ";UID=hia;PWD=hia"

'-------AB: Pick up the FA values from results.mdb -------
Sheets("CA Crude").Activate

'Build an ADO connection string for the database to be queried
ADOConnectionString = BuildADOConnectionString(as_planstar_basecase_outputdir & "Results.mdb")
If ADOConnectionString = "" Then
    Exit Sub
End If

'Open the ADO connection
If OpenADODatabaseConnection(ADOConnection, ADOConnectionString, False) = False Then
    GoTo SubExit
End If

'Create a new workbook with the right name: FeedstockAnalysis.xls (ipv BaseExpertBounds.xls)
Call DeleteBook(as_runid_dir_manual & "FeedstockAnalysis.xls")
Workbooks.Add
ActiveSheet.name = "FEEDSTOCKANALYSIS"
Set lws_FASheet = ActiveWorkbook.Sheets("FEEDSTOCKANALYSIS")
ls_VBName = ActiveWorkbook.name

'Insert the first row with the column headers
Range("A1").Select
Cells(1, 1) = "Range Case"
Cells(1, 2) = "Variable Name"
Cells(1, 3) = "Value"

'Open a recordset for the table
If OpenADORecordSet(ADOConnection, ADORS, "SELECT PrFeedstockRange.RangeCase, PrFeedstockRange.VariableName, PrFeedstockRange.VariableValue FROM PrFeedstockRange INNER JOIN PRSolutionID on PRFeedstockRange.SolutionID=PRSolutionID.SolutionID order by PrFeedstockRange.RangeCase, PrFeedstockRange.VariableName") = False Then
GoTo SubExit
End If

'Loop through the returned records one row at a time to return the data
li_NumberCrudes = 0
RowCounter = 2
li_count = 0
ReDim ls_crudes(0)

With ADORS
    While Not .EOF
        ls_RangeCase = .Fields(0).value
        ls_VariableName = .Fields(1).value
        ll_VariableValue = .Fields(2).value
        lws_FASheet.Cells(RowCounter, 1).value = ls_RangeCase
        lws_FASheet.Cells(RowCounter, 2).value = ls_VariableName
        lws_FASheet.Cells(RowCounter, 3).value = ll_VariableValue
        If RowCounter = 2 Then
            ls_crudes(0) = ls_VariableName
            ls_crudes2 = "('" & ls_VariableName & ","
            li_CrudeNumber = 1
            'AB: adding PS name
            lws_FASheet.Cells(1, 7) = ls_VariableName
            lv_query = Array( _
                "select hcl_complabel from complabel_mapping where mapping_id='east' and naming_convention='PIMS' and complabel=" & ls_VariableName & ","

        Else
            For li_loop = 0 To li_NumberCrudes
                If (ls_VariableName = ls_crudes(li_loop)) Then
                    li_CrudeNumber = li_loop + 1
                    GoTo AlreadyAddedbefore
                End If
            Next
End With

Else
    For li_loop = 0 To li_NumberCrudes
        If (ls_VariableName = ls_crudes(li_loop)) Then
            li_CrudeNumber = li_loop + 1
            GoTo AlreadyAddedbefore
        End If
    Next
End If
ReDim Preserve ls_crudes(li_NumberCrudes + 1)
ls_crudes(li_NumberCrudes + 1) = ls_VariableName
lws_FASheet.Cells(li_NumberCrudes + 2, 7) = ls_VariableName
li_NumberCrudes = li_NumberCrudes + 1
li_CrudeNumber = li_NumberCrudes + 1
ls_crudes2 = ls_crudes2 & "" & ls_VariableName & "" & ", "
lv_query = Array( _
"select hcl_complabel from complabel_mapping where mapping_id='east' and naming_convention='PIMS' and complabel=" & ls_VariableName & "" & ls_VariableName & "")

With ActiveSheet.QueryTables.Add(Connection:=gs_connection, Destination:=Cells(li_NumberCrudes + 1, 8))
  .CommandText = lv_query
  .FieldNames = False
  .name = "query_table_FeedstockAnalysis"
  ls_name = .name
End With
With ActiveSheet.QueryTables(ls_name)
  .PreserveFormatting = True
  .RefreshStyle = xlOverwriteCells
  .Refresh BackgroundQuery:=False
End With
Call SetQueryTableOptions("FEEDSTOCKANALYSIS", ls_name)
ThisWorkbook.Names.Add name:="Fixed_Crudes", RefersTo:=Names("FEEDSTOCKANALYSIS" + "]" + ls_name).RefersTo
End If
AlreadyAddedbefore:
If (lws_FASheet.Cells(RowCounter, 1) = "!Base") Then
  ld_totalfeed = ld_totalfeed + ll_VariableValue
  lws_FASheet.Cells(li_CrudeNumber, 9) = ll_VariableValue
ElseIf (lws_FASheet.Cells(RowCounter, 1) = ls_VariableName & "MAX") Then
  lws_FASheet.Cells(li_CrudeNumber, 10) = ll_VariableValue
End If
.MoveNext
RowCounter = RowCounter + 1
End With
li_length = Len(ls_crudes2)
Mid(ls_crudes2, li_length - 1) = ")"
For i = 1 To (li_NumberCrudes + 1)
  If Cells(i, 9) = "" Then
    Cells(i, 9) = Application.WorksheetFunction.VLookup(Cells(i, 7), Range("B1:C" & li_NumberCrudes + 2), 2, False)
  End If
  If Cells(i, 10) = "" Then
    Cells(i, 10) = Application.WorksheetFunction.VLookup(Cells(i, 7), Range("B1:C" & li_NumberCrudes + 2), 2, False)
  End If
Next i
'--------AB: Calculate utility indices for the crudes--------
For li_count2 = 0 To li_NumberCrudes
    If lws_FASheet.Cells(li_count2 + 1, 10) = "" Then
        lws_FASheet.Cells(li_count2 + 1, 11) = 0
        lws_FASheet.Cells(li_count2 + 1, 10) = lws_FASheet.Cells(li_count2 + 1, 9) / ld_totalfeed
    Else
        lws_FASheet.Cells(li_count2 + 1, 11) = (lws_FASheet.Cells(li_count2 + 1, 10) / ld_totalfeed - lws_FASheet.Cells(li_count2 + 1, 9) / ld_totalfeed) / (1 - lws_FASheet.Cells(li_count2 + 1, 9) / ld_totalfeed)
    End If
Next

lws_FASheet.Columns("G:K").Select
Selection.Copy
Application.DisplayAlerts = False
ActiveWorkbook.Close
Application.DisplayAlerts = True

'--------AB: Write Xmax-Xmin with corresponding crudes in FA Sheet------
Sheets("Feedstock Analysis").Activate
ActiveSheet.Range("A2:E300").ClearContents

'--------AB: Prepare appropriate cases in CA Crude sheet--------
Range("A2").Select
Selection.PasteSpecial
li_cases = 13
Sheets("CA Crude").Range("N13:O300").ClearContents
Sheets("CA Crude").Range("Q13:Q300").ClearContents
Sheets("CA Crude").Range("S13:S300").ClearContents
Sheets("CA Crude").Range("U13:X300").ClearContents
Sheets("CA Crude").Range("AA13:AA300").ClearContents

For li_count = 2 To (li_NumberCrudes + 2)
    If Sheets("Feedstock Analysis").Cells(li_count, 6) = "YES" Then
        Sheets("CA Crude").Cells(li_cases, 14) = Range("location")
        Sheets("CA Crude").Cells(li_cases, 15) = Sheets("Feedstock Analysis").Cells(li_count, 2)
        Sheets("CA Crude").Cells(li_cases, 17) = 1
        Sheets("CA Crude").Cells(li_cases, 19) = "Y"
        Sheets("CA Crude").Cells(li_cases, 24) = 80
        Sheets("CA Crude").Cells(li_cases, 27) = "H"
        li_cases = li_cases + 1
    End If
Next
Exit Sub
SubExit:
'Make sure the ADO recordset is closed
If Not ADORS Is Nothing Then
    ADORS.Close
End If
'Make sure the ADO connection is closed
If Not ADOConnection Is Nothing Then
    ADOConnection.Close
End If
'Clear object memory
Set ADORS = Nothing
Set ADOConnection = Nothing
Set lws_BoundsSheet = Nothing
Set lws_CapsSheet = Nothing
Set lws_GeneralSheet = Nothing

errorhandler:
    Application.ScreenUpdating = True
    MsgBox "Error " & Err & ": " & Error$ & " occurred."
    Application.DisplayAlerts = False
    ActiveWorkbook.Close
    Application.DisplayAlerts = True
End Sub

Sub Abs_criterion()
    Dim ls_value As String
    Dim i As Integer
    If Worksheets("General").OptionButton1.value = True Then
        ls_value = Worksheets("General").TextBox1.value
        If ls_value = "" Then
            Exit Sub
        End If
        Sheets("Feedstock Analysis").Range("F2:F300").FormulaR1C1 = "=IF(RC5=""="",""YES"",""NO""))"
    End If
End Sub

Sub Rel_criterion()
    Dim ls_value As String
    Dim i As Integer
    If Worksheets("General").OptionButton2.value = True Then
        ls_value = Worksheets("General").TextBox2.value
        If ls_value = "" Then


Exit Sub
End If

Sheets("Feedstock Analysis").Range("F2:F300").FormulaR1C1 =
"=IF(RC5=""",""",IF(RANK(RC5,C5)/(COUNT(C5)-1) <= 0.01 * ls_value & "",""YES",""NO""))"
End If

End Sub
## Appendix G – Outcomes of Example 8

### Refinery Y

<table>
<thead>
<tr>
<th>Crude oil</th>
<th>Rank CI's</th>
<th>Rank utility indices</th>
</tr>
</thead>
<tbody>
<tr>
<td>STAG</td>
<td>1</td>
<td>73</td>
</tr>
<tr>
<td>SLPG</td>
<td>2</td>
<td>71</td>
</tr>
<tr>
<td>MIRI</td>
<td>3</td>
<td>64</td>
</tr>
<tr>
<td>AGTL</td>
<td>4</td>
<td>34</td>
</tr>
<tr>
<td>KUIT</td>
<td>5</td>
<td>103</td>
</tr>
<tr>
<td>ZUAS</td>
<td>6</td>
<td>28</td>
</tr>
<tr>
<td>QINH</td>
<td>7</td>
<td>114</td>
</tr>
<tr>
<td>THEV</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>ANGS</td>
<td>9</td>
<td>14</td>
</tr>
<tr>
<td>ALSH</td>
<td>10</td>
<td>38</td>
</tr>
<tr>
<td>SLEB</td>
<td>11</td>
<td>66</td>
</tr>
<tr>
<td>BARR</td>
<td>12</td>
<td>41</td>
</tr>
<tr>
<td>XIJI</td>
<td>13</td>
<td>85</td>
</tr>
<tr>
<td>YOMB</td>
<td>14</td>
<td>112</td>
</tr>
<tr>
<td>VITY</td>
<td>15</td>
<td>50</td>
</tr>
<tr>
<td>EACR</td>
<td>16</td>
<td>45</td>
</tr>
<tr>
<td>RABL</td>
<td>17</td>
<td>76</td>
</tr>
<tr>
<td>SOKL</td>
<td>18</td>
<td>75</td>
</tr>
<tr>
<td>WALI</td>
<td>19</td>
<td>106</td>
</tr>
<tr>
<td>FORC</td>
<td>20</td>
<td>94</td>
</tr>
<tr>
<td>DOBA</td>
<td>21</td>
<td>108</td>
</tr>
<tr>
<td>MARM</td>
<td>22</td>
<td>9</td>
</tr>
<tr>
<td>ESPO</td>
<td>23</td>
<td>67</td>
</tr>
<tr>
<td>BINT</td>
<td>24</td>
<td>60</td>
</tr>
<tr>
<td>NALT</td>
<td>25</td>
<td>53</td>
</tr>
<tr>
<td>OMAN</td>
<td>26</td>
<td>35</td>
</tr>
<tr>
<td>KEKW</td>
<td>27</td>
<td>32</td>
</tr>
<tr>
<td>ZHAD</td>
<td>28</td>
<td>113</td>
</tr>
<tr>
<td>RASB</td>
<td>29</td>
<td>39</td>
</tr>
<tr>
<td>CEND</td>
<td>30</td>
<td>17</td>
</tr>
<tr>
<td>ENFI</td>
<td>31</td>
<td>95</td>
</tr>
<tr>
<td>DURI</td>
<td>32</td>
<td>111</td>
</tr>
<tr>
<td>QARU</td>
<td>33</td>
<td>116</td>
</tr>
<tr>
<td>RABB</td>
<td>34</td>
<td>37</td>
</tr>
<tr>
<td>NILE</td>
<td>35</td>
<td>78</td>
</tr>
<tr>
<td>MUTI</td>
<td>36</td>
<td>2</td>
</tr>
<tr>
<td>PUF7</td>
<td>37</td>
<td>31</td>
</tr>
<tr>
<td>RATA</td>
<td>38</td>
<td>18</td>
</tr>
<tr>
<td>ARAH</td>
<td>39</td>
<td>5</td>
</tr>
<tr>
<td>BNGA</td>
<td>40</td>
<td>59</td>
</tr>
</tbody>
</table>

### Refinery Z

<table>
<thead>
<tr>
<th>Crude oil</th>
<th>Rank CI’s</th>
<th>Rank utility indices</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOBA</td>
<td>1</td>
<td>56</td>
</tr>
<tr>
<td>DURI</td>
<td>2</td>
<td>63</td>
</tr>
<tr>
<td>NILE</td>
<td>3</td>
<td>52</td>
</tr>
<tr>
<td>STAG</td>
<td>4</td>
<td>69</td>
</tr>
<tr>
<td>UDAN</td>
<td>5</td>
<td>31</td>
</tr>
<tr>
<td>CABD</td>
<td>6</td>
<td>41</td>
</tr>
<tr>
<td>AMNA</td>
<td>7</td>
<td>32</td>
</tr>
<tr>
<td>LALA</td>
<td>8</td>
<td>27</td>
</tr>
<tr>
<td>BAHO</td>
<td>9</td>
<td>30</td>
</tr>
<tr>
<td>RABL</td>
<td>10</td>
<td>35</td>
</tr>
<tr>
<td>GIRA</td>
<td>11</td>
<td>44</td>
</tr>
<tr>
<td>KIKE</td>
<td>12</td>
<td>22</td>
</tr>
<tr>
<td>FORC</td>
<td>13</td>
<td>49</td>
</tr>
<tr>
<td>KEKW</td>
<td>14</td>
<td>8</td>
</tr>
<tr>
<td>CANU</td>
<td>15</td>
<td>59</td>
</tr>
<tr>
<td>ERHA</td>
<td>16</td>
<td>42</td>
</tr>
<tr>
<td>LABU</td>
<td>17</td>
<td>46</td>
</tr>
<tr>
<td>BENC</td>
<td>18</td>
<td>29</td>
</tr>
<tr>
<td>MARM</td>
<td>19</td>
<td>69</td>
</tr>
<tr>
<td>BONL</td>
<td>20</td>
<td>33</td>
</tr>
<tr>
<td>XICO</td>
<td>22</td>
<td>54</td>
</tr>
<tr>
<td>CHEX</td>
<td>23</td>
<td>55</td>
</tr>
<tr>
<td>SLEB</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>MUDI</td>
<td>25</td>
<td>13</td>
</tr>
<tr>
<td>JATI</td>
<td>26</td>
<td>60</td>
</tr>
<tr>
<td>KAJI</td>
<td>27</td>
<td>23</td>
</tr>
<tr>
<td>MASI</td>
<td>28</td>
<td>51</td>
</tr>
<tr>
<td>ANGS</td>
<td>29</td>
<td>5</td>
</tr>
<tr>
<td>SOKL</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>VITY</td>
<td>31</td>
<td>40</td>
</tr>
<tr>
<td>MALA</td>
<td>32</td>
<td>36</td>
</tr>
<tr>
<td>WSEN</td>
<td>33</td>
<td>6</td>
</tr>
<tr>
<td>TAPU</td>
<td>34</td>
<td>1</td>
</tr>
<tr>
<td>BREB</td>
<td>35</td>
<td>37</td>
</tr>
<tr>
<td>KIDU</td>
<td>36</td>
<td>26</td>
</tr>
<tr>
<td>ARJU</td>
<td>37</td>
<td>11</td>
</tr>
<tr>
<td>SAHB</td>
<td>38</td>
<td>16</td>
</tr>
<tr>
<td>BRNL</td>
<td>39</td>
<td>9</td>
</tr>
<tr>
<td>MURB</td>
<td>40</td>
<td>17</td>
</tr>
<tr>
<td>HAND</td>
<td>41</td>
<td>19</td>
</tr>
<tr>
<td>MINA</td>
<td>41</td>
<td>98</td>
</tr>
<tr>
<td>-------</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>PENL</td>
<td>42</td>
<td>84</td>
</tr>
<tr>
<td>KIKE</td>
<td>43</td>
<td>24</td>
</tr>
<tr>
<td>UDAN</td>
<td>44</td>
<td>69</td>
</tr>
<tr>
<td>RANG</td>
<td>45</td>
<td>44</td>
</tr>
<tr>
<td>JATI</td>
<td>46</td>
<td>80</td>
</tr>
<tr>
<td>KUWA</td>
<td>47</td>
<td>13</td>
</tr>
<tr>
<td>LEGE</td>
<td>48</td>
<td>25</td>
</tr>
<tr>
<td>ARJU</td>
<td>49</td>
<td>57</td>
</tr>
<tr>
<td>QATL</td>
<td>50</td>
<td>42</td>
</tr>
<tr>
<td>URAL</td>
<td>51</td>
<td>97</td>
</tr>
<tr>
<td>ZAKU</td>
<td>52</td>
<td>62</td>
</tr>
<tr>
<td>SUTU</td>
<td>53</td>
<td>82</td>
</tr>
<tr>
<td>CHEX</td>
<td>54</td>
<td>52</td>
</tr>
<tr>
<td>DAIH</td>
<td>55</td>
<td>20</td>
</tr>
<tr>
<td>LUFE</td>
<td>56</td>
<td>87</td>
</tr>
<tr>
<td>GERA</td>
<td>57</td>
<td>26</td>
</tr>
<tr>
<td>SLPH</td>
<td>58</td>
<td>115</td>
</tr>
<tr>
<td>LANG</td>
<td>59</td>
<td>19</td>
</tr>
<tr>
<td>MASI</td>
<td>60</td>
<td>40</td>
</tr>
<tr>
<td>UMMS</td>
<td>61</td>
<td>70</td>
</tr>
<tr>
<td>NKOS</td>
<td>62</td>
<td>91</td>
</tr>
<tr>
<td>BONL</td>
<td>63</td>
<td>88</td>
</tr>
<tr>
<td>KIDU</td>
<td>64</td>
<td>65</td>
</tr>
<tr>
<td>IRNL</td>
<td>65</td>
<td>79</td>
</tr>
<tr>
<td>M100</td>
<td>66</td>
<td>36</td>
</tr>
<tr>
<td>DUBA</td>
<td>67</td>
<td>8</td>
</tr>
<tr>
<td>CYRU</td>
<td>68</td>
<td>1</td>
</tr>
<tr>
<td>KUMK</td>
<td>69</td>
<td>54</td>
</tr>
<tr>
<td>HAND</td>
<td>70</td>
<td>83</td>
</tr>
<tr>
<td>CINT</td>
<td>71</td>
<td>96</td>
</tr>
<tr>
<td>BASK</td>
<td>72</td>
<td>43</td>
</tr>
<tr>
<td>ORIE</td>
<td>73</td>
<td>4</td>
</tr>
<tr>
<td>BUAT</td>
<td>74</td>
<td>29</td>
</tr>
<tr>
<td>IRNH</td>
<td>75</td>
<td>48</td>
</tr>
<tr>
<td>TAPU</td>
<td>76</td>
<td>58</td>
</tr>
<tr>
<td>QATM</td>
<td>77</td>
<td>47</td>
</tr>
<tr>
<td>KERA</td>
<td>78</td>
<td>110</td>
</tr>
<tr>
<td>BEKA</td>
<td>79</td>
<td>107</td>
</tr>
<tr>
<td>SARI</td>
<td>80</td>
<td>100</td>
</tr>
<tr>
<td>KAJI</td>
<td>81</td>
<td>86</td>
</tr>
<tr>
<td>SLPK</td>
<td>82</td>
<td>77</td>
</tr>
<tr>
<td>MUDI</td>
<td>83</td>
<td>68</td>
</tr>
<tr>
<td>KHAF</td>
<td>84</td>
<td>3</td>
</tr>
<tr>
<td>TUJO</td>
<td>85</td>
<td>55</td>
</tr>
<tr>
<td>ANOA</td>
<td>86</td>
<td>104</td>
</tr>
<tr>
<td>SENI</td>
<td>87</td>
<td>10</td>
</tr>
<tr>
<td>EOCE</td>
<td>88</td>
<td>102</td>
</tr>
<tr>
<td>LSFO</td>
<td>89</td>
<td>117</td>
</tr>
<tr>
<td>Positive</td>
<td>Neutral</td>
<td>Negative</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
<td>----------</td>
</tr>
<tr>
<td>SEPA 90</td>
<td>BUKH 92</td>
<td>101</td>
</tr>
<tr>
<td>ARAM 91</td>
<td>SLLG 93</td>
<td>11</td>
</tr>
<tr>
<td>MESL 92</td>
<td>DOBA 94</td>
<td>74</td>
</tr>
<tr>
<td>WIDU 93</td>
<td></td>
<td>89</td>
</tr>
<tr>
<td>MALT 94</td>
<td></td>
<td>33</td>
</tr>
<tr>
<td>SNOB 95</td>
<td></td>
<td>22</td>
</tr>
<tr>
<td>ARXL 96</td>
<td></td>
<td>21</td>
</tr>
<tr>
<td>RASG 97</td>
<td></td>
<td>30</td>
</tr>
<tr>
<td>BENC 98</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>GRIF 99</td>
<td></td>
<td>27</td>
</tr>
<tr>
<td>UPZA 100</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>MARG 101</td>
<td></td>
<td>90</td>
</tr>
<tr>
<td>BELI 102</td>
<td></td>
<td>56</td>
</tr>
<tr>
<td>BELK 103</td>
<td></td>
<td>23</td>
</tr>
<tr>
<td>COSB 104</td>
<td></td>
<td>72</td>
</tr>
<tr>
<td>ELAN 105</td>
<td></td>
<td>46</td>
</tr>
<tr>
<td>RGAS 106</td>
<td></td>
<td>49</td>
</tr>
<tr>
<td>ESPA 107</td>
<td></td>
<td>61</td>
</tr>
<tr>
<td>SLPL 108</td>
<td></td>
<td>99</td>
</tr>
<tr>
<td>BRGC 109</td>
<td></td>
<td>63</td>
</tr>
<tr>
<td>NAMC 110</td>
<td></td>
<td>92</td>
</tr>
<tr>
<td>UNDA 111</td>
<td></td>
<td>81</td>
</tr>
<tr>
<td>GERC 112</td>
<td></td>
<td>101</td>
</tr>
<tr>
<td>BINB 113</td>
<td></td>
<td>93</td>
</tr>
<tr>
<td>LAVA 114</td>
<td></td>
<td>120</td>
</tr>
<tr>
<td>SHAC 115</td>
<td></td>
<td>121</td>
</tr>
<tr>
<td>BINN 116</td>
<td></td>
<td>118</td>
</tr>
<tr>
<td>BINP 117</td>
<td></td>
<td>119</td>
</tr>
<tr>
<td>BANA 118</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>ARAL 119</td>
<td></td>
<td>12</td>
</tr>
<tr>
<td>**LOKE 120</td>
<td></td>
<td>109</td>
</tr>
<tr>
<td>BEAT 121</td>
<td></td>
<td>51</td>
</tr>
</tbody>
</table>

*Table G.1 Outcomes of Example 8*