Multiperiod Refinery Planning: Development and Applications

Multiperiod Refinery Planning: Development and Applications

by

Alexander Nguyen, B. Eng

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science

McMaster University

© Copyright by Alexander Nguyen, August 2018

MASTER OF APPLIED SCIENCE (2018) McMaster University

(Chemical Engineering) Hamilton, Ontario, Canada

TITLE:	Multiperiod Refinery Planning: Development and Applications
AUTHOR:	Alexander Nguyen, B.Eng. McMaster University, Hamilton, Ontario, Canada
SUPERVISOR:	Dr. Christopher L.E. Swartz
NUMBER OF PAGES:	xxii, 158

Lay Abstract

Petroleum refineries consist of complex units that serve a certain purpose, such as separating components of a mixed stream or blending intermediate products, in order to create final commercial products, e.g. gasoline and diesel. Due to the complexity and interconnectivity in a refinery, it is difficult to determine the optimal mode of operation. Thus, by formulating the refinery in mathematical form, optimization techniques may be used to find optimal operation. Furthermore, optimization problems can be formulated in a multiperiod fashion, where the problem is repeated over a set time horizon in partitions. The advantage is a higher detail in the operation of the refinery but this comes at a cost of higher computation time. In this work, a multiperiod refinery is formulated and studied by exploring model size, computation times, comparison of solvers, and solution strategies such as decomposition.

Abstract

The purpose of this work aims to develop and explore a nonlinear multiperiod petroleum refinery model based on a real-world model. Due to the inherent complexity and interconnected nature of petroleum refineries, various studies are implemented to describe the multiperiod model. The model is based around maximizing the profit of a petroleum refinery, starting from the crude inputs through the crude distillation unit, to the intermediate product processing through various unit operations, and finally to the blending of the final products. The model begins as a single period model, and is re-formulated as a multiperiod model by incorporating intermediate product tanks and dividing the model into partitions.

In solving the multiperiod model, the termination criteria for convergence was varied in order to investigate the effect on the solution; it was found that it is acceptable to terminate at a relaxed tolerance due to minimal differences in solution. Several case studies, defined as deviations from normal operation, are implemented in order to draw comparisons between the real-world model and the model studied in this thesis. The thesis model, solved by CONOPT and IPOPT, resulted in significant gains over the real-world model.

Next, a Lagrangean decomposition scheme was implemented in an attempt to decrease computation times. The decomposition was unable to find feasible

solutions for the subproblems, as the nonlinear and nonconvex nature of the problem posed difficulty in finding feasibilities. However, in the case of a failed decomposition, the point where the decomposition ends may be used as an initial guess to solve the full space problem, regardless of feasibility of the subproblems. It was found that running the decomposition fewer times provided better initial guesses due to lower constraint violations from the infeasibilities, and then combined with the shorter decomposition time resulted in faster computation times.

Acknowledgements

I would like to thank Dr. Christopher Swartz for his guidance, support, and patience throughout the entirety of my Master's degree and project.

To Eric Bouveresse, without his expertise, excellent mentoring, and continued support, I would not have learned as much about petroleum refining. I am extremely grateful for our detailed weekly meetings, his belief in me, and his continued vouching for me at TOTAL; I would not have the opportunities presented to me if not for him.

I would like to express my gratitude to Meriam Chebre and TOTAL for providing us with the project topic, petroleum refinery model, and the resources to succeed in this project. TOTAL has been a key contribution in the direction of the project and has allowed us to align industry goals with academia. TOTAL has been an invaluable resource to us throughout the project and their collaboration is greatly appreciated.

Thank you Dr. Jake Nease and Dr. David Latulippe for all of your advice, mentorship, and opportunities throughout both undergraduate and graduate school (also thanks to Jake for introducing me to curling and Tuesday drives).

To my best friend and partner in crime, Kerala Brendon. Thank you for being there for me throughout grad school, for all of the adventures, and for all of

the experiences - I couldn't have done it without you.

To all of my friends from undergrad and grad school, thank you for helping me through the struggles of heat transfer, Mac Eng Musical and McMaster Musical Theatre, and for all of the experiences. Special thanks to the boys at 102 Royal Avenue for getting me through the rest of grad school.

Last but not least, thank you to my parents, Tran Truong and Minh Nguyen, and my siblings, Kevin and Elizabeth. Thank you for the frequent visits down from Halifax and for your love and support.

Table of Contents

List of Figures

List of Tables

Declaration of Academic Achievement

All research presented in this thesis is original work completed and drafted by Alexander Nguyen, with editorial assistance and supervision under Dr. Christopher Swartz.

Chapter 1

Introduction

1.1 Motivation & Goals

Petroleum refineries often consist of complex units that serve a certain purpose, such as distillation of cuts in crude, separating components of a mixed stream, or upgrading the qualities of a product. These units are often intertwined and rely on each other in order to operate. From this connectivity between units, there are many scenarios that can be studied and modeled using multiperiod optimization. Multiperiod optimization involves formulation of a problem spanning over a specified length of time and partitioned into periods; there

must be constraints and variables that link time periods in order to create a multiperiod problem. This method takes into account dynamics of the system that may occur between time periods, as opposed to traditional single-period optimization. Using multiperiod optimization, we are able to explore the behaviour of the refinery when subject to anomalies in operation, such as the reduced throughput of a unit, and how intermediate stock tanks will react to the deviation in operation. Petroleum refineries provide a complex and interesting model to study, as there are many connections between units, strict operating bounds, and a large number of properties and compositions to maintain.

The objective of this thesis is to develop a multiperiod optimization planning solution in the context of a petroleum refinery using a real-world model, and to study various scenarios that may occur in a refinery. The thesis aims to improve current optimization practices at TOTAL by introducing multiperiod optimization over longer time horizons (e.g. 30 periods over 1 month), explore alternative nonlinear solvers and solution strategies, and to improve solution times. By improving solution times, larger problems (i.e. more time periods) may be studied. Furthermore, decomposition solution strategies can be implemented in the multiperiod refinery model in order to further decrease convergence time.

1.2 Thesis Contributions

The contributions that this thesis presents include:

- Review of published literature pertaining to petroleum refinery modeling (unit operation processes and plant configuration) and their various

applications in multiperiod and single period optimization.

- Development of a nonlinear multiperiod refinery model based on a real-world petroleum refinery model, including yield relationships, property calculations, use of plant data and correlations, and detailed information regarding feedstocks to the refinery.
- Exploration of various solution strategies to solve the multiperiod refinery model.
- Exploration on the effect of solver tolerance options on the solution of the multiperiod refinery model and effect on computation time; exploration of solution times for various sizes of the multiperiod problem.
- Comparison of performance from current practices of solving the optimization model in industry versus using IPOPT and CONOPT.
- Review of the Lagrangean decomposition algorithm and its application in multiperiod optimization. Formulation and implementation of the Lagrangean decomposition algorithm in the multiperiod refinery model.
- Investigating the selection of end points at various iterations of the decomposition as an initial guess for solving the full space model.

1.3 Thesis Overview

Chapter 2 - Literature Review

Chapter 2 presents a review of literature surrounding the work completed in this thesis. Background is provided on the fundamentals and configuration of petroleum refining, including descriptions of major unit operations and

their processes, utility generation in the refinery, blending operations, and possible configurations of the refinery process. This is followed by optimization approaches and application in petroleum refining with context in both single and multiperiod formulations; studies published in literature regarding refinery planning and scheduling in nonlinear formulations are discussed. Next, nonlinear programming in general is discussed, with descriptions of interior point methods, generalized reduced gradient approaches, IPOPT, and CONOPT. A description of the model formulation in AMPL is also discussed. Lastly, Lagrangean decomposition formulation and methods are described and discussed, as well as discussion on published literature that utilities these methods in multiperiod problems.

Chapter 3 - Multiperiod Refinery Optimization: Model Formulation

Chapter 3 discusses the formulation of the multiperiod petroleum refinery model that is studied in the thesis. First, a detailed description of the development of the model is discussed. In this section, the process in formulating the multiperiod model from the single model is described. Next, the mathematical formulation of the multiperiod refinery model, as well as equation transformations are presented. Following the mathematical formulation, the intermediate tanks required for multiperiod optimization are formulated and described.

Chapter 4 - Solution Strategies

Chapter 4 describes the solution strategies proposed in the study. The first strategy proposed involves taking information from the model development stage as initial guesses. The next strategy involves the implementation of Lagrangean decomposition on the multiperiod refinery model, and discusses the formulation of the decomposition in the study. Lastly, a hybrid method is proposed, where in the case of an unsuccessful decomposition, the end

point of the decomposition may pose to be a good initial guess to solve the remainder of the problem to optimality.

Chapter 5 - Case Studies

Chapter 5 presents various base case (nominal operation) and case study (deviations from nominal operation) applications and analysis on the multiperiod refinery model. First, the effect of termination tolerance of the IPOPT solver is compared for the refinery model when solving from the initial point to the nominal operation of the refinery (base case). The next two sections describe several case studies (deviations from nominal operation) and are compared to the original model developed by TOTAL. This is to compare the performance when solving with IPOPT and CONOPT. Next, the scalability of the multiperiod model is discussed, where the size of the problem is adjusted (i.e. changing number of time periods) in order to study the effect of larger problems on computational solve time. All case studies were compared for various sizes, and an analysis of the constraint violation and dual infeasibilities are discussed. Finally, the results of the Lagrangean decomposition are discussed. In this section, exploration of the performance, convergence, and feasibility of the decomposition method are discussed. Furthermore, the results of the hybrid method (of solving from the end of the decomposition) are discussed.

Chapter 6 - Conclusions and Recommendations

The thesis is then summarized in this chapter, highlighting the development and applications of the multiperiod petroleum refinery, the results obtained in the study, and interpretation of the results. Future developments and work are outlined and discussed.

Chapter 2

Literature Review

2.1 Refinery Process Description

Refineries can be operated in many different configurations depending on the plant conditions such as the crude being processed, types of final products, environmental regulations, or new technology [1]. However, most refineries share the same major unit operations — the atmospheric crude distillation unit (CDU), vacuum distillation unit (VDU), hydrocracker (HC), hydrotreater (HT), and the catalytic reformer and/or cracker. Figure 2.1 shows an example of a modern refinery [1].

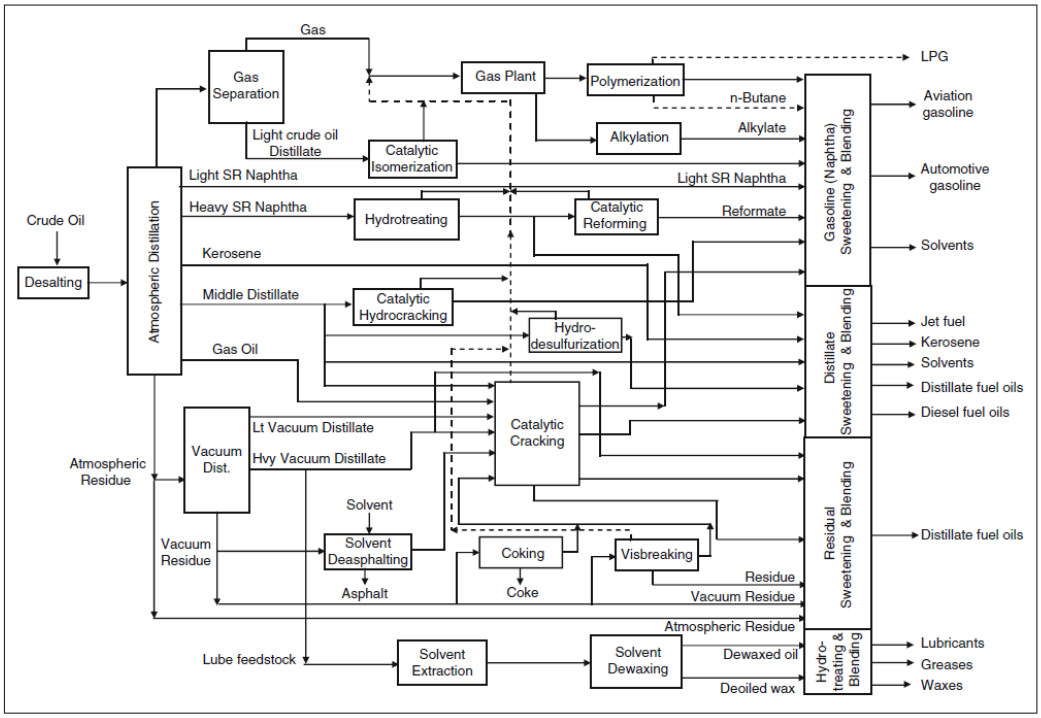

Figure 2.1: A modern petroleum refinery process scheme [1].

In petroleum refinery optimization, there are several examples from literature with varying levels of detail — detailed models of unit operations, refinery-focused models, blending models, and supply chain optimization are the various types of petroleum refinery models used in optimization. All of these

types of models are encompassed in Figure 2.2 and are often segregated into 3 subproblems: crude-oil scheduling with unloading, mixing, and tank control; optimization and scheduling of the processing units of the refinery; and final product blending, tank control, and exports [2].

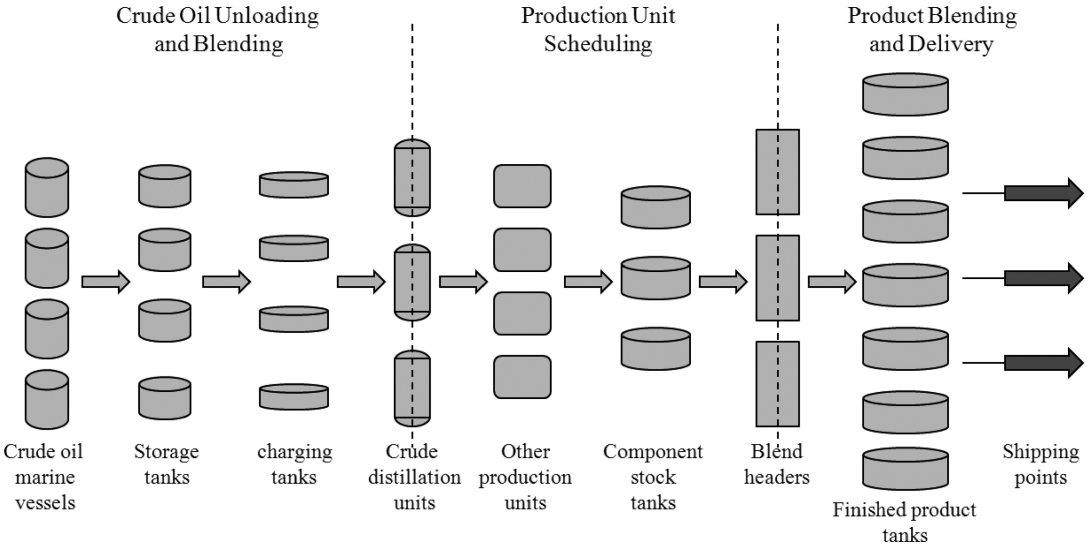

Figure 2.2: Subproblems of petroleum optimization - crude-oil scheduling, refinery optimization, and product blending [2].

In this work, the model used in building the optimization problem was based on an in-house refinery model built by TOTAL and coincides with the second type of sub-problem denoted by Shah et al. [2] where the optimization is focused around the operation of the production units in the refinery. This encompasses crude feed entering the atmospheric distillation unit, to the export of final products such as gasoline, diesel, fuel oils, and naphthas and the operation of the units in between.

With the type of refinery optimization model in mind, there are various configurations of a petroleum refinery that can be designed. Depending on the crude type, product demand, location, and various other end goals specific to the operator, the refinery may or may not require specific units. However, the majority of refineries include crude distillation (CDU), hydrotreating (HT),

and catalytic reforming (CCR) of naphtha based on various textbooks and literature [3], [1], [4]. A study by Abella and Bergerson [5] investigated the impact of greenhouse gases and energy use in a petroleum refinery subject to various crude quality and refinery configurations, and assumed that all configurations contain CDU, HT, and CCR units. Other units such as visbreakers, fluidized catalytic crackers (FCC), and hydrocrackers are then incorporated based on final product demands, crude inputs, and desired operation of the refinery. The intermediate streams of a petroleum refinery are subject to various transformations - direct to final product blending, upgrading (breaking high carbon/heavy streams down to higher value intermediates), impurity removal (e.g. sulfurs, metals, nitrogen), and utility separation (e.g. stripping hydrogen, refinery fuel gas from intermediate streams).

There are other refinery configurations used in published literature. Figure 2.3 shows the configuration used in a study by Wu et al. [6] which focuses on bridging the gap between theory and application in short-term scheduling by using control theory and safe states in order to generate feasible solutions. Note that the configuration also follows the idea of having a CDU, HT, and CCR. Figure 2.4 shows the optimized topology of a state-task network (STN)-based superstructure optimization problem from a study by Khor and Elkamel [7]. The STN representation contains all possible feasible configurations/topologies for a petroleum refinery and is broken down into four major processing sections: naphtha exiting the CDU; residue to the VDU from the CDU; vacuum gas oil from the VDU; and heavy oil upgrading. A mixed-integer linear problem that aims to maximize profit and minimize environmental impacts is formulated based on the superstructure and subsequently, is solved. A life cycle analysis is incorporated into the problem to account for the environmental impacts; further details can be found in the

published literature [7].

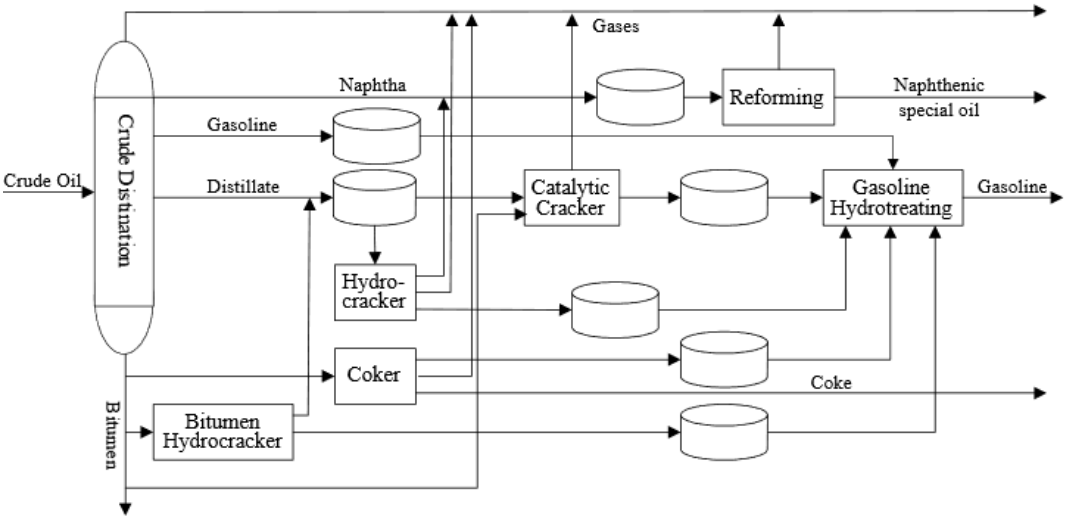

Figure 2.3: Refinery process model used by Wu et al. for short-term scheduling using control theory [6].

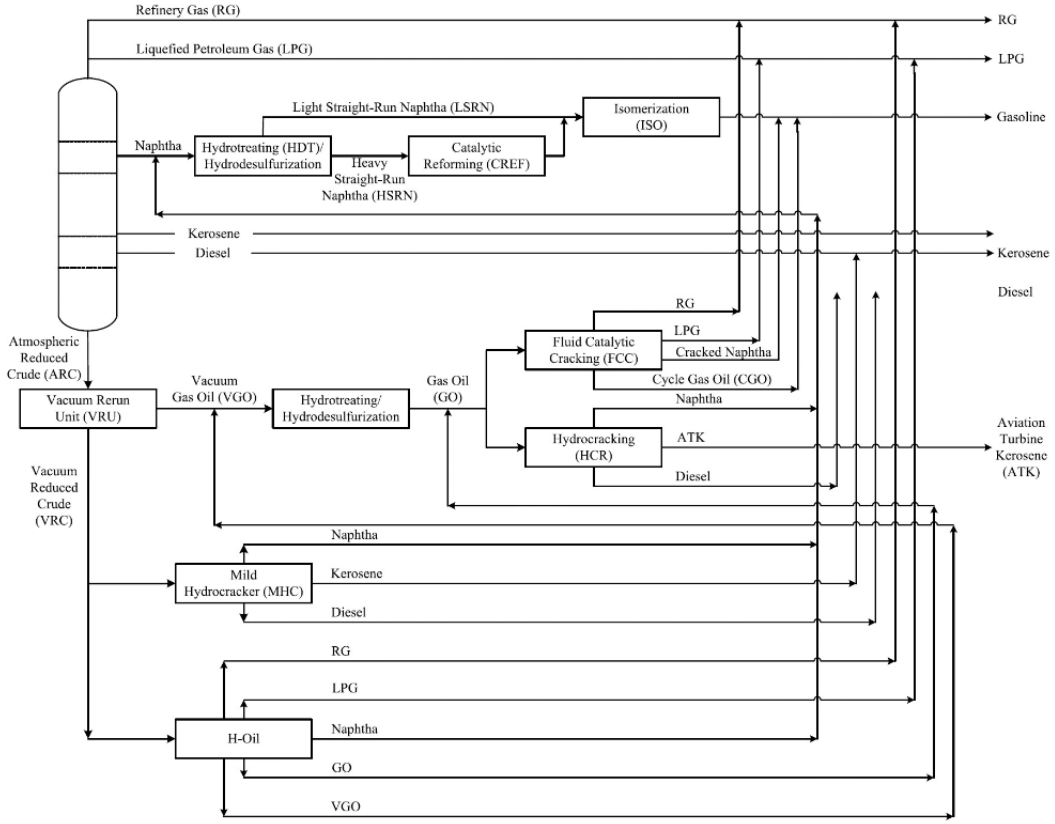

Figure 2.4: Optimized topology of a state-task network superstructure used in by Khor and Elkamel [7].

The following section will describe the possible major unit operations in a petroleum refinery in more detail. Again, the configuration of the refinery depends on the product desired, crude oil processed, and operational variables that may be desired or necessary. For example, a refinery may not utilize a visbreaker and/or delayed/fluid coker if the refinery plans on selling the vacuum residue as is and it is not beneficial to further upgrade the residue. These decisions are primarily based on operator and process designer expertise and knowledge, and is often kept proprietary to the company as to maintain an advantage over other producers [5].

2.1.1 Major Unit Operations

2.1.1.1 Crude Distillation Unit

The crude distillation unit (CDU) is located at the front of the refinery and is the first unit that processes the crude inputs to the plant. Before the crude enters the CDU, sediment and free water are separated out in storage tanks by gravity and it is then sent through the cold side of various heat exchangers (which use CDU products as hot side streams) and heated to 120 to 150°C. The stream heads into a desalting unit to remove dissolved salts, and is then sent to a furnace to be heated up to 330 to 385°C [1].

The separation of the crude occurs in the distillation column, often consisting of multiple columns for each product, as shown in Figure 2.5 [8]. The high temperature crude feed enters near the bottom of the column where it is flashed into higher ends and residue drops to the bottom of the column. Steam injection occurs at the bottom of the column, where the steam strips lighter hydrocarbons out of the residue and lowers the partial pressure and boiling

point of the hydrocarbons [1]. The column then extracts the products based on boiling point as side draws and generates products such as (in ascending order based on weight): light ends and naphtha (LN), heavy naphtha (HN), kerosene, and light and heavy gas oils (LGO and HGO, respectively). The products are then stripped in a column with steam injection to remove light ends from the heavier streams and reintroduced into the CDU.

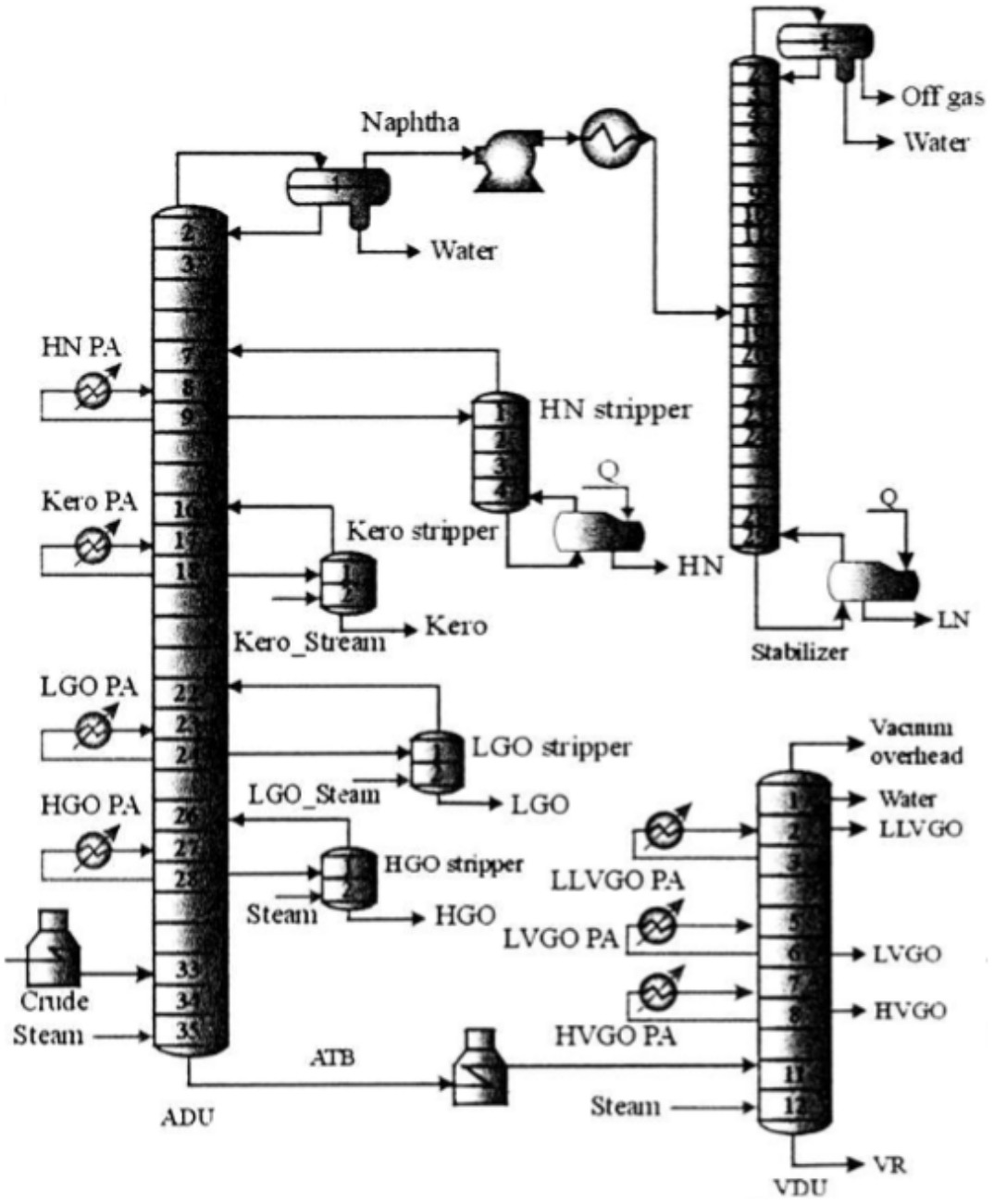

Figure 2.5: Crude distillation unit (CDU) model with stage numbers, product streams, and smaller fractionation units [8].

There have been various mathematical formulations of the crude distillation unit, such as fixed-yield equations, multicomponent (swing-cut) formulations, and nonlinear steady-state models. In fixed-yield equations, parameters are determined beforehand and the outlet products are linearly related for the

inlet flow rate, as shown below [9]:

$$ST^{out}_{j,k,f_s} = a_{k,i,j,f_s} * ST^{in}_{i,k,f_s} \tag{2.1}$$

where ST^{in}_{i,k,f_s} and ST^{out}_{j,k,f_s} are the inlet and outlet flow rates, respectively, and a_{k,i,j,f_s} is the yield parameter for unit $k \in K$ (i.e. the CDU), inlet and outlet streams to k for $i \in I$ and $j \in J$, respectively, and crude fed to unit k for $f_s \in F$

A swing cut formulation is built upon the fixed-yield relation by introducing swing cuts and cut points. Crude cut points can be used to determine how much of each product is produced; each product is produced within a range of temperatures, with cut points separating each the boiling point of each product. By shifting the cut point of a product, there will be an increase or decrease in the product yield with inverse effects on its adjacent product cuts. For example, if the cut point of kerosene is lowered, resulting in a decrease in kerosene yield, then the light gas oil can increase its yield by widening its operating range [1]. Swing cuts represent the allowable change in cut point of a product, and equation (2.1) can be re-formulated as [9]:

$$ST_{j,k,f_s} = a_{k,i,j,f_s} * ST_{i,k,f_s} + b_{k,j,f_s,front} + b_{k,j,f_s,back} \tag{2.2}$$

$$SwC = b_{k,j,f_s,front} + b_{k,j,f_s,back} \tag{2.3}$$

where $b_{k,j,f_s,front}$ and $b_{k,j,f_s,back}$ are variables that represent the front and back of the swing cut, respectively, for unit $k \in K$, outlet stream from k for $j \in J$, and crude feed $f_s \in F$, and SwC is a fixed value.

Finally, a fractionation-index model is used to model the CDU in a non-linear

fashion, where

$$\frac{x_{top,i}}{x_{bottom,i}} = \alpha_{io}^{FI} \frac{x_{top,o}}{x_{bottom,o}} \tag{2.4}$$

is used to calculate the composition of the product streams and cut point temperatures. x represents the mole fraction of component i to a reference component o and α_{io}^{FI} represents the relative volatility between the components. As well, the component equilibrium constant K can be used in place of α_{io}^{FI} as an approximation [10]. A full derivation of the fractionation-index model for CDU is described in Alattas et al. [9].

2.1.1.2 Vacuum Distillation Unit

The residue from the CDU must be further processed in the vacuum distillation unit (VDU) in order to extract distillates; the products that the VDU produces are the heavy, medium, light, and very light gas oils (HVGO, MVGO, LVGO, LLVGO), and vacuum residue. The defining characteristic of the VDU is that it operates at a lower pressure in order to process the residue at a lower temperature to prevent thermal cracking and the formation of petroleum coke. A general schematic of the VDU in relation to the CDU is shown in Figure 2.5 and a detailed diagram of the VDU is shown in Figure 2.6.

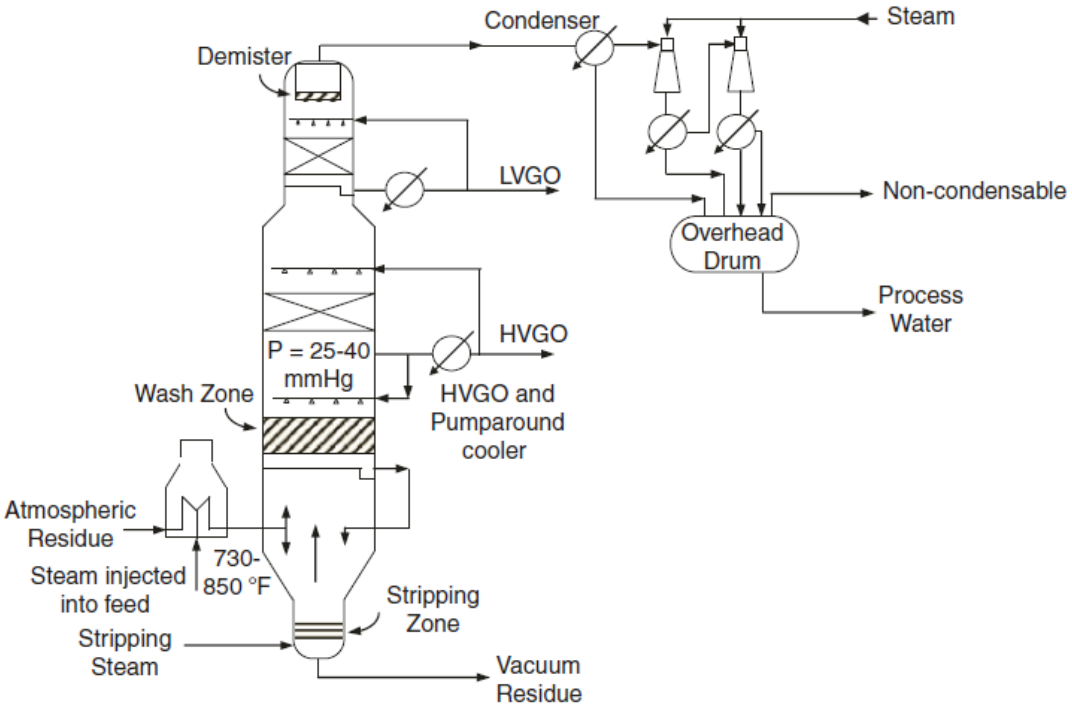

Figure 2.6: Detailed diagram of the vacuum distillation unit (VDU) process [1].

Before the atmospheric residue is fed into the VDU, it is heated up between 380 and 415°C with steam injection in the furnace to prevent thermal cracking. It then enters near the bottom of the VDU and operates similarly to the CDU and the vacuum gas oils are extracted as side streams. Steam is injected at the bottom of the unit to strip lighter ends from the residue, but is also maintained at a temperature of 365°C to prevent coking [11].

Defining characteristics of the VDU include design components that generate the vacuum pressure and components that reduce pressure drop throughout the column. Packing in the fractionation and heat exchange areas are an example of the latter. A vacuum is generated using either ejectors and liquid ring pumps *or* ejectors alone [1].

2.1.1.3 Visbreaker & Delayed/Fluid Coking

In order to make use of the residue from the CDU and VDU, the residue must be broken down or in other words, cracked, at high temperatures. This cracking reaction can be performed in three different types of units/processes — listed in order of increasing temperature tolerances: visbreaking, delayed coking, and fluid coking.

The visbreaker unit employs a mild form of thermal cracking (i.e. the reactions are stopped before completion using quenching) and is fed both atmospheric residue and vacuum residue from the CDU and VDU, respectively, and produces the following products, in ascending yield [1]:

- gases (C4) at 2-4 total weight percent
- naphtha (C5) at 5-7 total weight percent
- gas oils at 10-15 total weight percent
- residue at 75-85 total weight percent

The visbreaker unit operates at a temperature range of 455 to 510°C at a pressure of 50 to 300 psi [12]; these conditions depend on the configuration of the visbreaker: coil visbreaker (cracking occurs in the furnace coil) or soak visbreaker (cracking occurs in a soak drum), shown in Figure 2.7. Panel A in Figure 2.7 shows the coil visbreaker configuration, where the feed is first heated then cracked in the furnace at a temperature of 450 to 480°C and 43 to 145 psi. The stream is then quenched (i.e. mixed with a cold stream often comprised of gas oil and visbreaker residue) to stop the reaction — this is to prevent coke from forming in the fractionation tower. In the fractionation

tower, the gases, naphtha (gasoline), gas oil, and residue are separated out and either sent to downstream blending, units, or as a quench stream [1]. Panel B shows the soak visbreaker configuration, similar to that of the coil visbreaker, with the difference being a soaker vessel placed between the furnace and the fractionation tower. With this configuration, the process has a longer residence time, as the product conversion is maintained at a reaction temperature in the soaker vessel instead of the furnace and thus, can be operated at a lower temperature. Soak visbreakers can be thought of as low temperature-high residence time, whereas coil visbreakers as high temperature-low residence time [12].

The advantage of the visbreaker unit is the low cost (low fuel requirements) and relatively low severity compared to other coking units for the upgrading of petroleum residue. However, drawbacks regarding the visbreaker are the potential for unstable products, as thermal cracking at lower pressures can yield olefins in the naphtha product which can undergo further reactions to produce tar and other heavy compounds. Furthermore, visbreaking leaves behind a large fraction of residue that is not converted into other products [12].

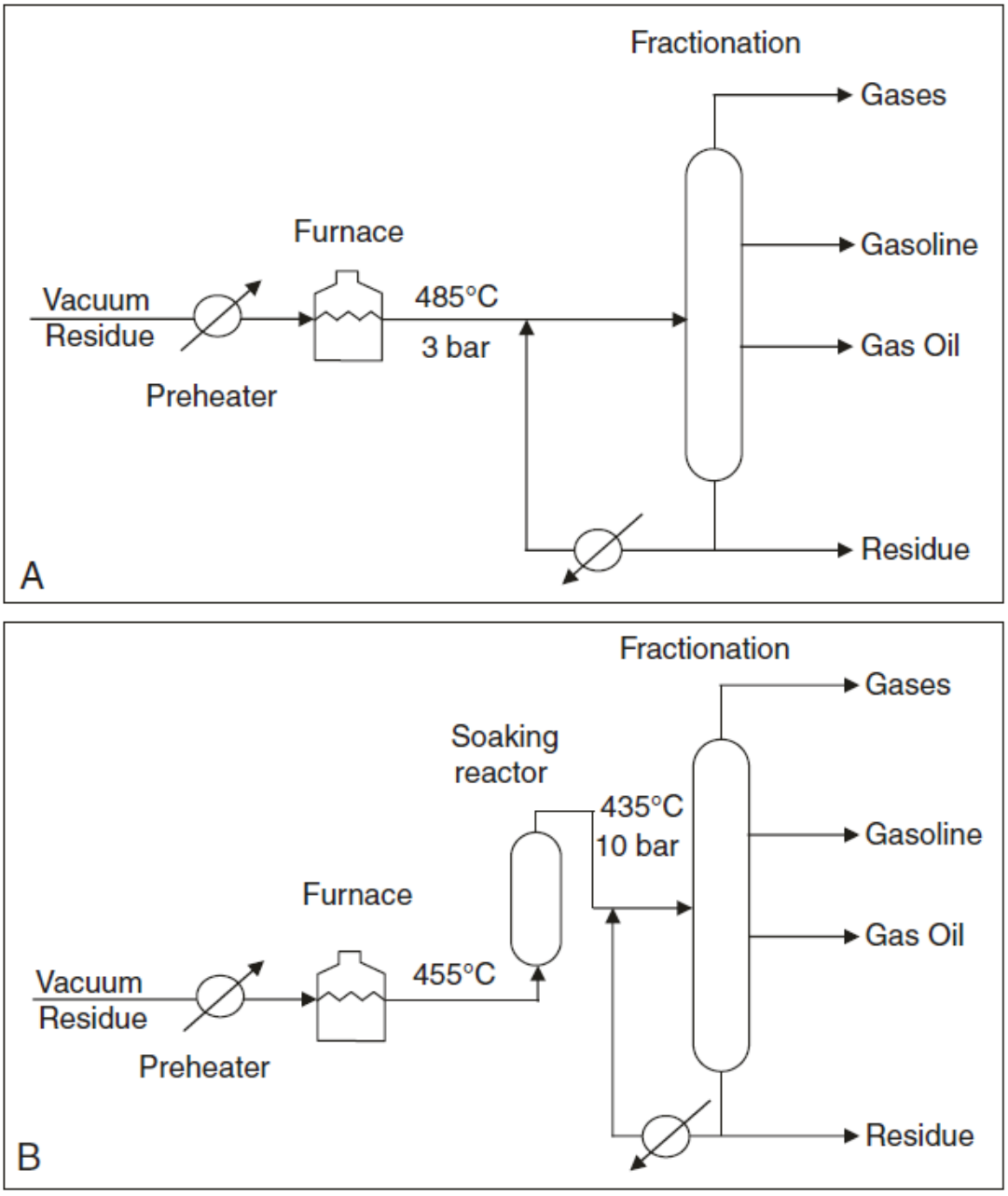

Figure 2.7: Two configurations of the visbreaker unit: Panel A shows a coil visbreaker configuration; Panel B shows a soak visbreaker configuration [1].

Similar to the visbreaker, the delayed coker unit can process high density streams but with the added benefit of being able to process streams with high metal content. The delayed coker operates in a two-stage fashion, where the thermal energy is supplied by the furnace, and then coking (the deposition of carbon solids) occurs in the drum. This two-stage process reduces the

residence time required in the furnace while allowing sufficient time in the drum [1]. This configuration is shown in Figure 2.8 where the feed stream enters the fractionation tower, the light ends are separated and the bottoms of the tower (heavy streams, residues) are sent to the furnace, where the cracking reaction occurs. The furnace operates between 480 to 515°C and the heated residue is sent to the coking drums, which operate between 415 to 465°C at 0.1 to 0.4 MPa. As the reaction proceeds, the overhead products (the cracked products) re-enter the fractionation tower and residue deposits in the interior of the coking drum. For continuous operation, two drums are used — one is used for the reaction while the other drum undergoes cleaning [13].

Vacuum residue is often the feed stream for the delayed coker unit, but can also process visbreaker residue and bottoms product (slurry) from the fluidized catalytic cracker (FCC). The product streams of the delayed coker also differ from the visbreaker, which are unsaturated gases (C1 to C4), olefins (C2 to C4), and isobutane (iC4).

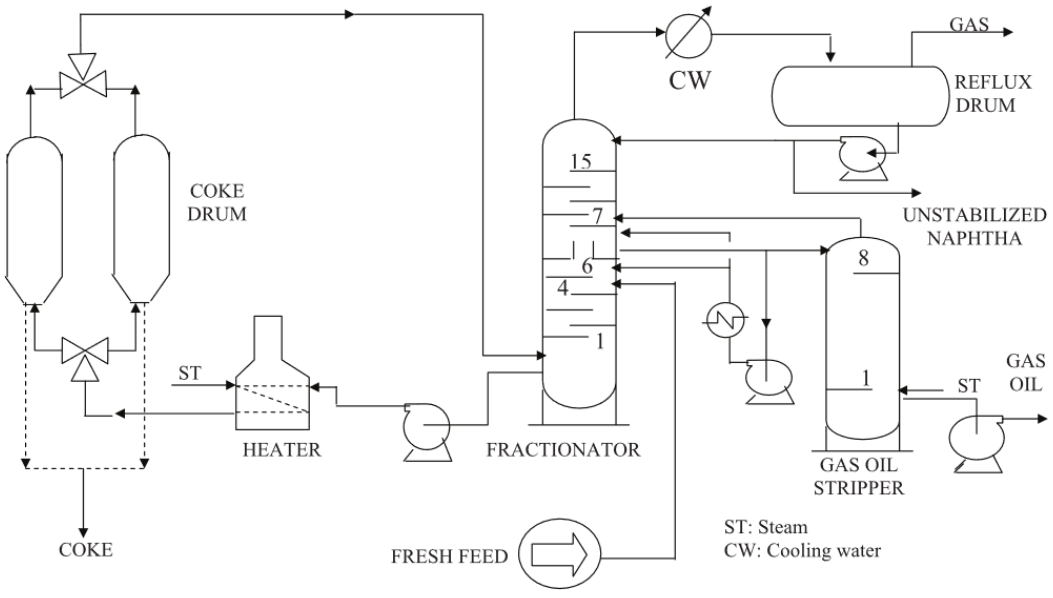

Figure 2.8: Two stage delayed coker process [13].

The third type of thermal cracking occurs in the fluid coking unit. The fluid coker consists of a fluidized bed reactor followed by a fluidized bed burner, where the reactor is used for the main cracking reactions, and the burner is used for taking coke from the reactor and sending it back to the reactor at a higher temperature to provide energy. The process starts with the residue feed at a temperature of 260°C that is injected into the fluidized section of the reactor operating at 500-600°C as shown in Figure 2.9 [14]. The feed is then cracked, depositing coke on the fluidized bed and sending lighter fractions into a scrubber and then onto a fractionation tower. The deposited coke is then fluidized by steam injection and then sinks through a stripper, where any interstitial lighter carbons are stripped out. The remainder of the coke is sent to the burner unit which operates at a temperature of 593 to 677°C [1]. In the burner, the fluidized coke is injected with steam where flue gas is produced and the hot coke is sent back to the reactor to provide heat.

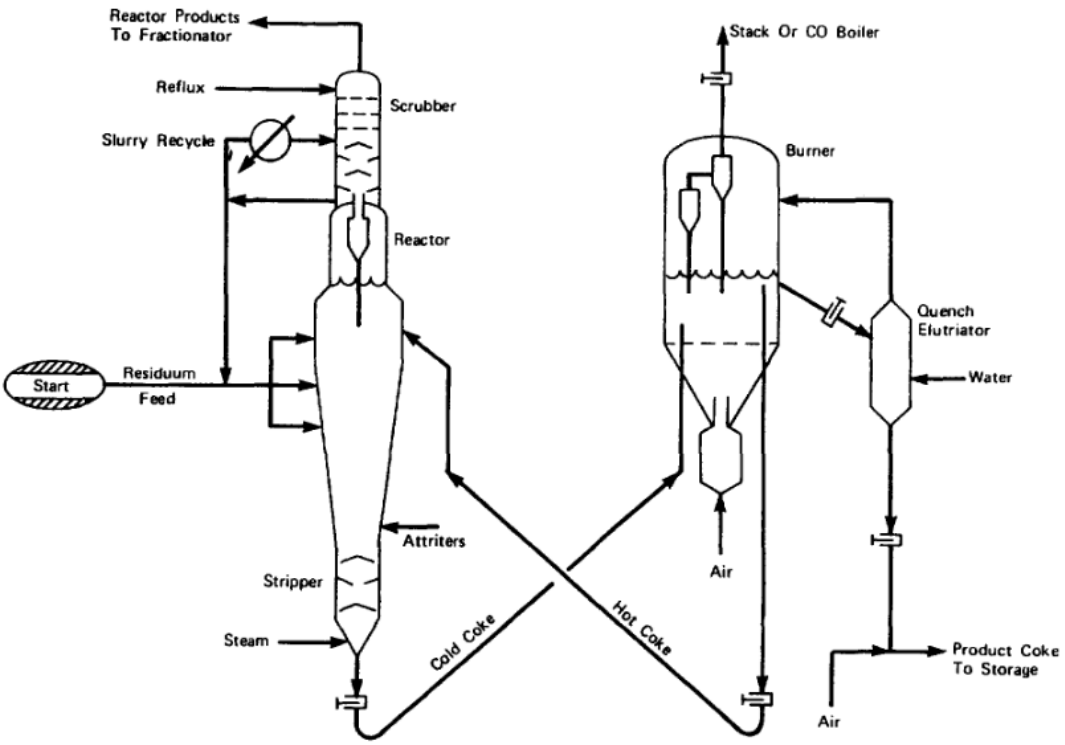

Figure 2.9: Fluid coking process [14].

2.1.1.4 Fluidized Catalytic Cracker

One of the most valuable unit operations in a refinery, the fluidized catalytic cracker (FCC), converts gas oils produced by the visbreaker/coking units into high value (high octane) products used for gasoline blending. The FCC also produces olefin feedstock for alkylation units, which in turn, produce more products for gasoline blending.

Figure 2.10 shows the FCC process from the feed through to the fractionation tower. First, the feed and steam heated to 316 to 427°C and are mixed with regenerated catalyst (649 to 760°C) and fed to the riser at the bottom of the reactor. Vapourization of the feed occurs in the riser as soon as the catalyst contacts the feed/steam stream and the endothermic cracking reaction begins as the vapourized stream rises. The residence time of the reaction in the riser is typically between 2-5 seconds [15]. Once the stream enters the reactor vessel, otherwise known as the disengagement zone, the cracked vapours enter the fractionation tower and the remaining product (heavy ends) and catalyst flow into the disengagement zone. The heavy ends are then stripped using steam and physically separated using cyclones in the reactor and the spent catalyst is then sent back to the regenerator. In the regenerator, excess air is introduced to burn off the residual coke formed on the catalyst [16].

The products that leave the fractionation tower consist of light and heavy cycle gas oils, heavy and light naphtha, decant slurry, and smaller carbon cuts such as C3 and C4 [1].

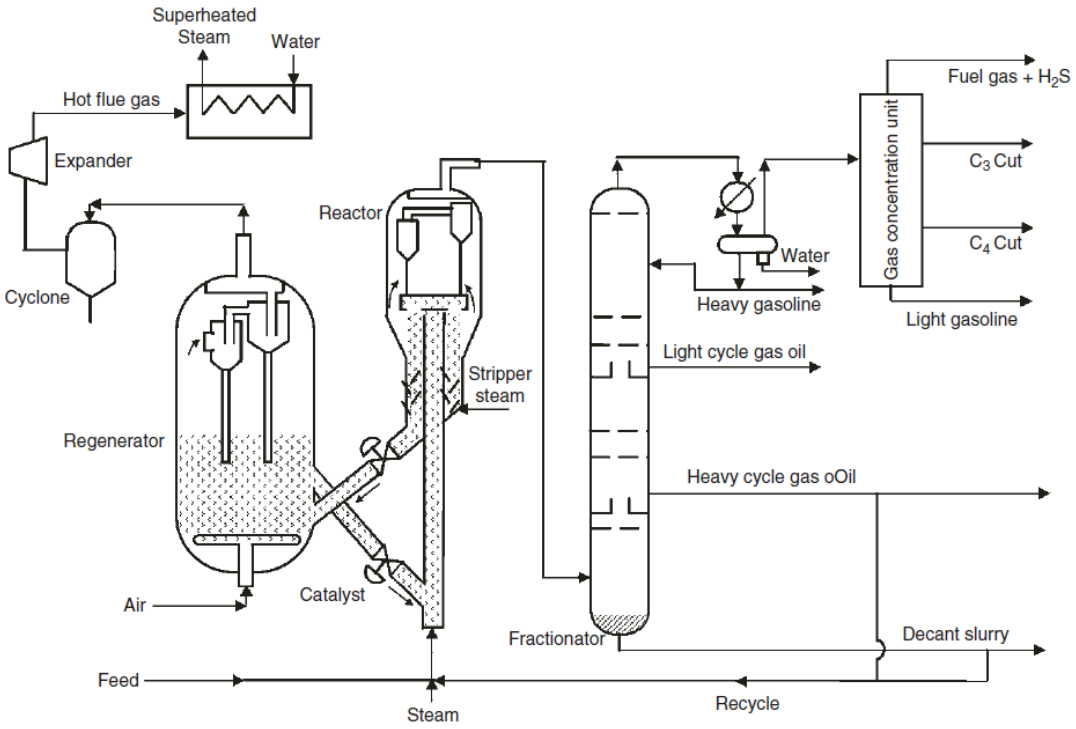

Figure 2.10: Fluidized catalytic cracker (FCC) unit and fractionation tower [1].

2.1.1.5 Alkylation

The alkylation process is used to convert olefins (C3, C4, C5, and iC4) from the refinery gas plant, coking units, and the FCC into alkylates for gasoline blending. There are several types of alkylation processes with various configurations depending on the refinery. The sulphuric acid alkylation process has two commonly used configurations — the auto-refrigeration process by Exxon and effluent refrigeration by Stratford. The hydrofluoric acid alkylation also has two available processes — the Phillip process and UOP process. Finally, solid catalyst alkylation techniques are developed but are not employed on an industrial scale [17].

In the aqueous alkylation process, the reaction is driven by a strong acid (sulphuric acid or hydrofluoric acid) due to the high temperatures and pressures

required in the absence of a catalyst (50°C and less than 3000 kPa versus 500°C and 20,000 to 40,000 kPa). Figure 2.11 shows the general alkylation scheme; once the reaction has completed, the products are sent to a settler where the acid is recycled back and the products continue on to be separated [1].

In the sulphuric acid processes, the Exxon and Stratford processes are similar except for differences in the reactor design. In the Exxon process, the reaction emulsion is cooled by the evaporation of isobutane and butylene, whereas in the Stratford process, a separate refrigeration unit provides the cooling. As well, the Exxon reactor operates at 5°C and 90 kPag whereas the Stratford reactor operates at 10°C and 420 kPag to prevent evaporation.

In the hydrofluoric acid processes, there are few differences between the Phillip and UOP processes. Isobutane is highly soluble in hydrofluoric acid and is injected through nozzles at the bottom of the reactor — due to the high solubility, no mechanical mixing is required. Reaction temperatures for the processes are 30 to 40°C and residence time ranges from 10 to 30 seconds; the higher temperature allows use of cooling water as coolant.

There are several advantages of using either acid catalyst. In using sulphuric acid, the production of dimethylhexane, an undesirable product, is lower than using hydrofluoric acid and thus, higher quality alkylates are produced. Additionally, propane and n-butane are not produced using sulphuric acid as the catalyst and thus, isobutane is not wasted in this reaction [18]. Therefore, less isobutane is required as feed and costs to recover and recycle the isobutane are lower. Most importantly, sulphuric acid is safer than hydrofluoric acid and is attractive from a safety standpoint [19]. In using hydrofluoric acid, required temperatures are higher (30 to 40°C versus 5 to 10°C) so less energy is needed to cool down the reactors (cooling water instead of refrigeration). Hydrofluoric

acid reactors are much smaller and have a lower residence time (compare 10 to 30 seconds versus 20 to 30 minutes [20]). As well, hydrofluoric acid has a lower viscosity than sulphuric acid which allows easier dispersion and mechanical stirring is not required (whereas sulphuric acid processes must use agitation) [17].

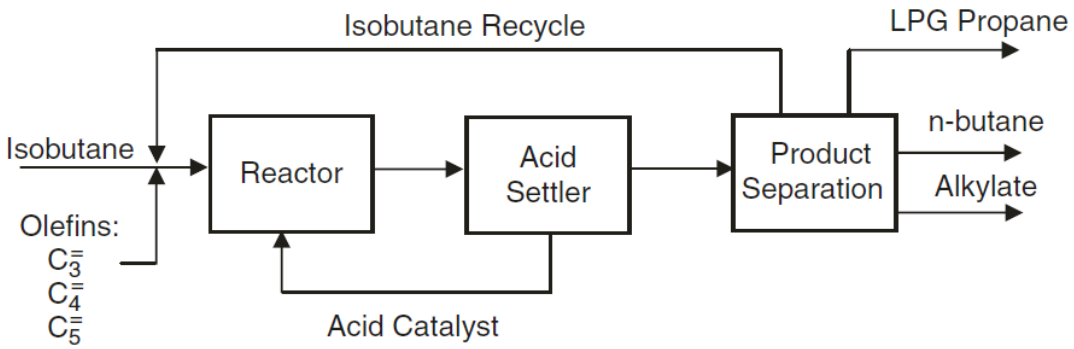

Figure 2.11: Alkylation process - products from various unit operations in the plant feed into this process. Sulphuric acid processes include the Exxon and Stratford processes; hydrofluoric acid processes include the Phillip and UOP processes [1].

2.1.1.6 Hydrotreater

In order to further break down product streams from the CDU, these intermediate products must be removed of impurities and other compounds that may inhibit effective conversion in other units (e.g. catalyst poisoning in the catalytic cracker or reformer). The hydrotreater has various objectives:

- removing impurities such as sulphur, nitrogen, aromatics, and metals for various streams such as naphtha, vacuum gas oil, and hydrocracker feed
- saturation of olefins for product stability
- removal of aromatics in kerosene within cetane number specifications

Due to the importance of the hydrotreater in conditioning feed streams to

other units, the hydrotreater is located before the reformer, hydrocracker, FCC, and various product streams as shown in Figure 2.12 [1]. Hydrotreaters are sometimes referred to by their specific function [3]:

- sulphur removal/hydrodesulphurization (HDS)
- nitrogen removal/hydrodenitrogenation (HDN)
- metal removal/hydrodemetallation (HDM)
- aromatic saturation/hydrodearomatization (HDA)

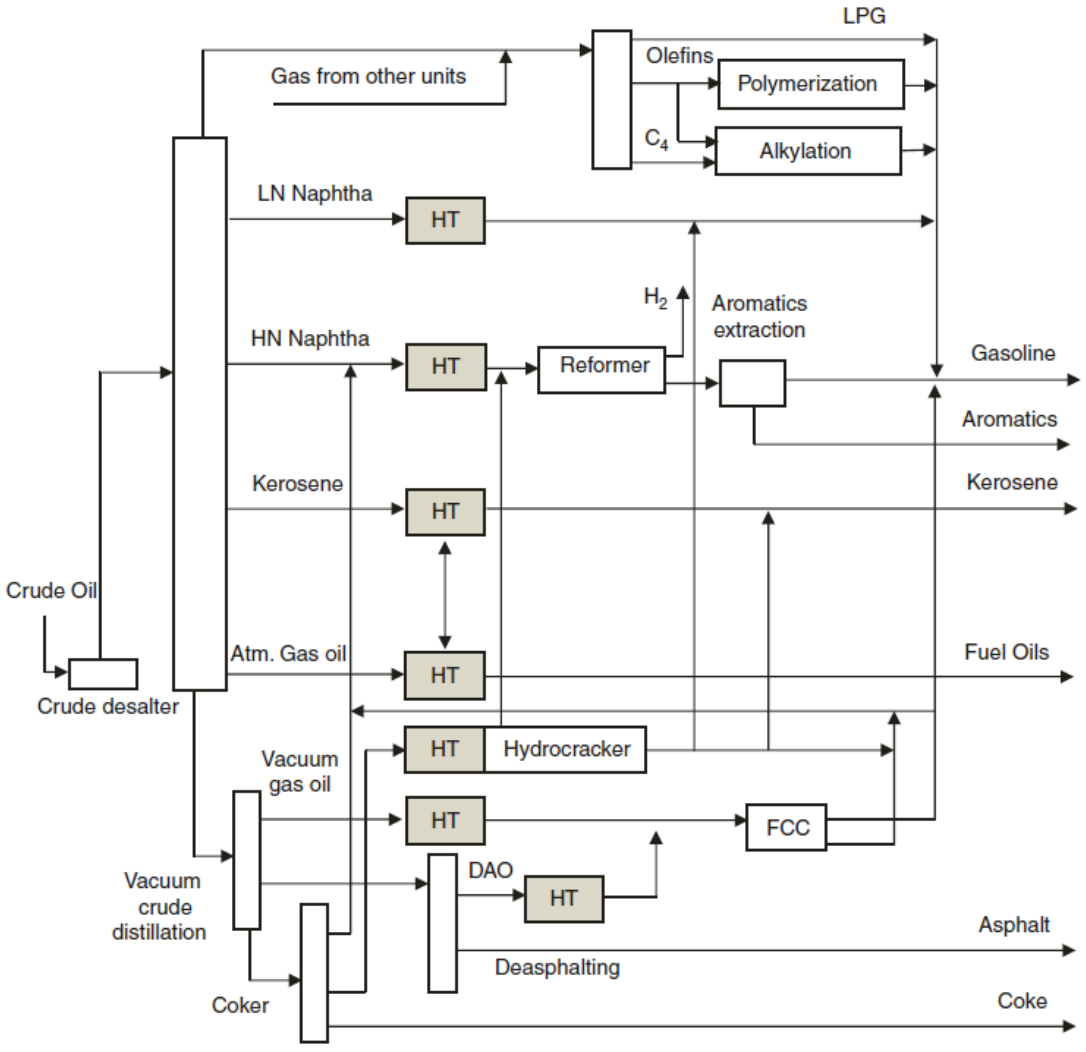

Figure 2.12: Various locations that the hydrotreater (HT) may be placed in a petroleum refinery process. The HT removes impurities (as well as other objectives), and is important in pre-processing streams before catalyst units [1].

In the hydrotreating process, the feed is mixed with hydrogen and enters the reactor at a high pressure of 300 to 1800 psig. The reactor contains fixed-bed catalysts, which are commonly porous alumina embedded with cobalt-molybdenum (Co-Mo), nickel-molybdenum (Ni-Mo), or nickel-tungsten (Ni-W) [1]. The feed/hydrogen mixture passes through the catalyst beds, where an exothermic reaction occurs and hydrogen sulphide (HDS), ammonia (HDN), or other impurities are produced. As the reaction is exothermic, hydrogen-rich quench gas streams are injected in the reactor in order to control the reaction temperatures. The products are then separated and the hydrogen gas is recycled back to the reactors [3].

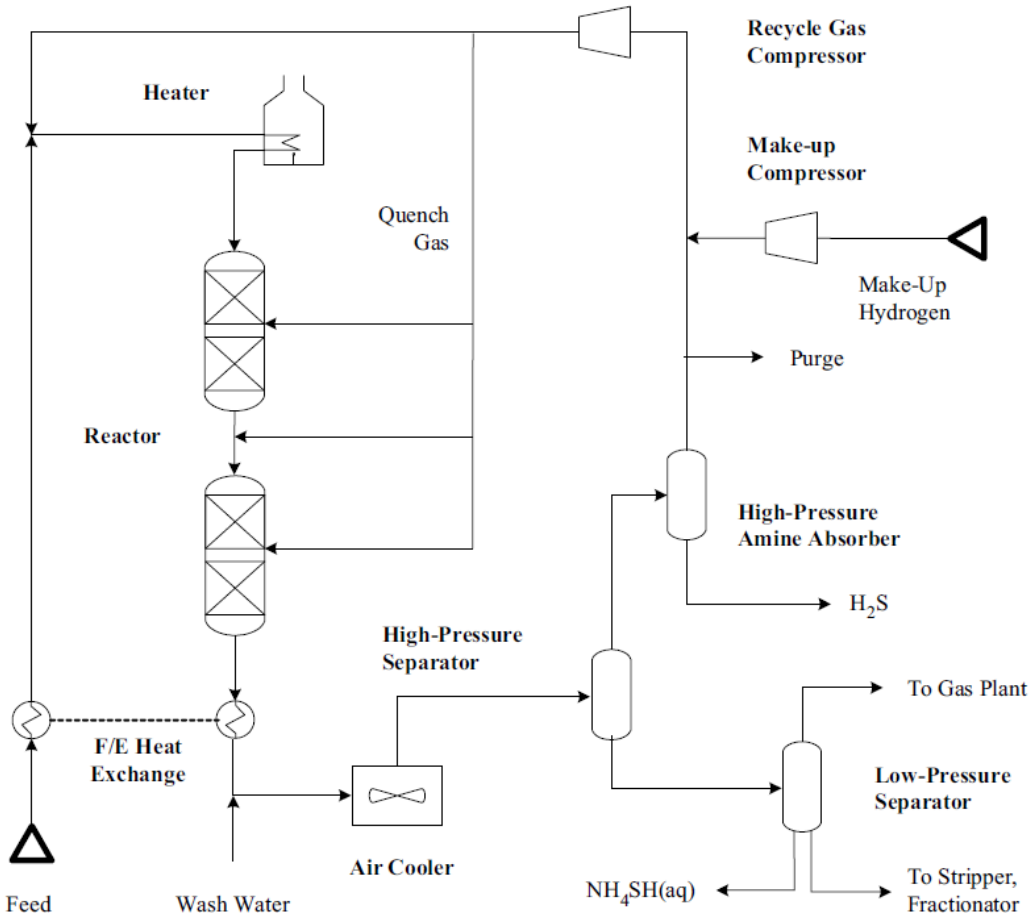

Figure 2.13: Hydrotreater (HT) process and configuration [3].

The hydrotreater must operate under strict conditions such as pressure, temper-

ature, catalyst loading, throughput, and hydrogen partial pressure to ensure correct operation. While higher temperatures increase reaction rates, thermal cracking and subsequently, coke formation will occur if temperatures are too high. Additionally, hydrogen partial pressure must be higher than the hydrocarbon partial pressure to ensure sufficient impurity removal and lower coke formation [1].

2.1.1.7 Hydrocracker

The hydrocracker is a crucial component of a petroleum refinery, as it is designed to take high molecular weight compounds and break them down into smaller products. Products that the hydrocracker yields, such as kerosene, diesel, and heavy naphtha, are sent directly to blending or for further processing in units such as the catalytic reformer. There are several configurations of hydrocrackers, mainly single-stage, once through, two-stage, and separate hydrotreat hydrocracking.

Figure 2.14 shows the configuration of a single-stage hydrocracker with and without a recycle (once through) [1]. In this configuration, the feed (often vacuum gas oil (VGO) and atmospheric gas oil) may be mixed with recycle oil, then is mixed with recycled and fresh hydrogen. The feed is hydrotreated if necessary, and is fed into the hydrocracking reactor, where the operation is similar to that of the hydrotreater — the mixture passes downward through the catalyst bed and the conversion rate is usually between 40 and 70 percent per volume [21]. The product stream is then sent to a fractionator where the lighter ends are sent to blending or further processing and the bottoms are sent to blends or units and/or recycled and mixed with fresh feed. The benefit to using once through configuration (no recycle), is that the unconverted oil

can be upgraded in another unit downstream such as the catalytic cracker; this is especially important when the recycle stream does not benefit from further hydrocracking and can occur in cases when the feed is heavily reliant on heavy VGO and gas oils from coking units [21].

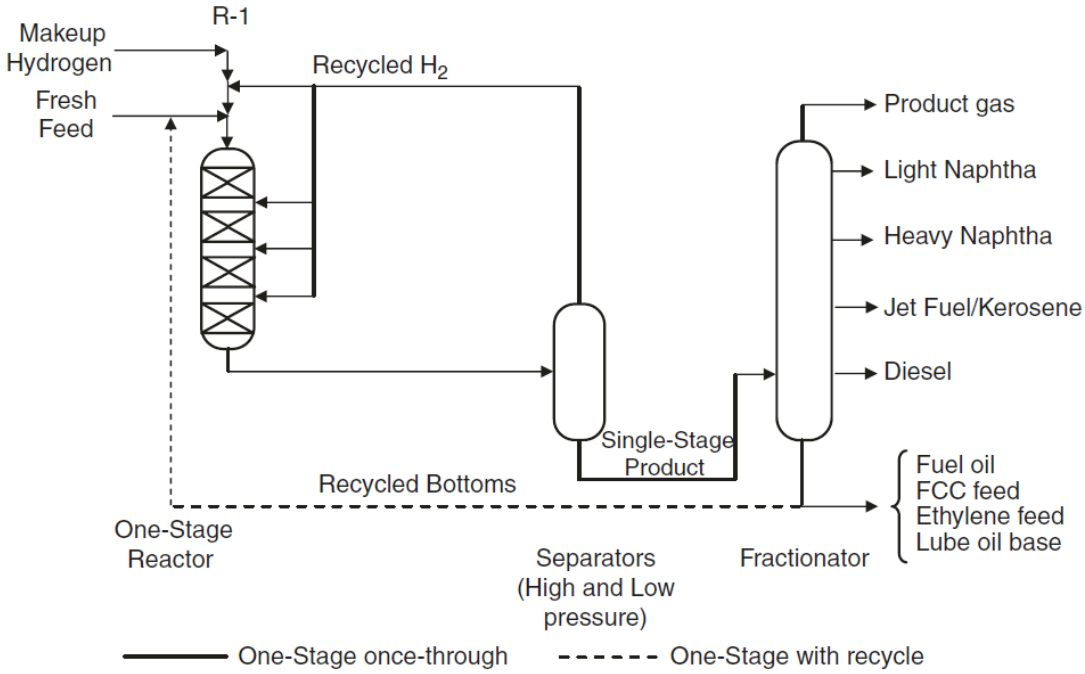

Figure 2.14: Single-stage hydrocracker (HC) process with an optional recycle (denoted by the dotted line) [1].

Two-stage hydrocracking processes are different than one-stage where the bottoms of the fractionator are first sent to a separator and the gas is recycled back into the hydrocracker and the bottoms are sent to another hydrocracker and separator (the lighter ends separated in the fractionator remain the same and are sent elsewhere in the refinery). In the second stage separator, the gas is recycled back into the second stage hydrocracker, and the bottoms are sent back into the fractionator. Figure 2.15 shows the two-stage configuration; it should be noted that the two-stage configuration is not common [21].

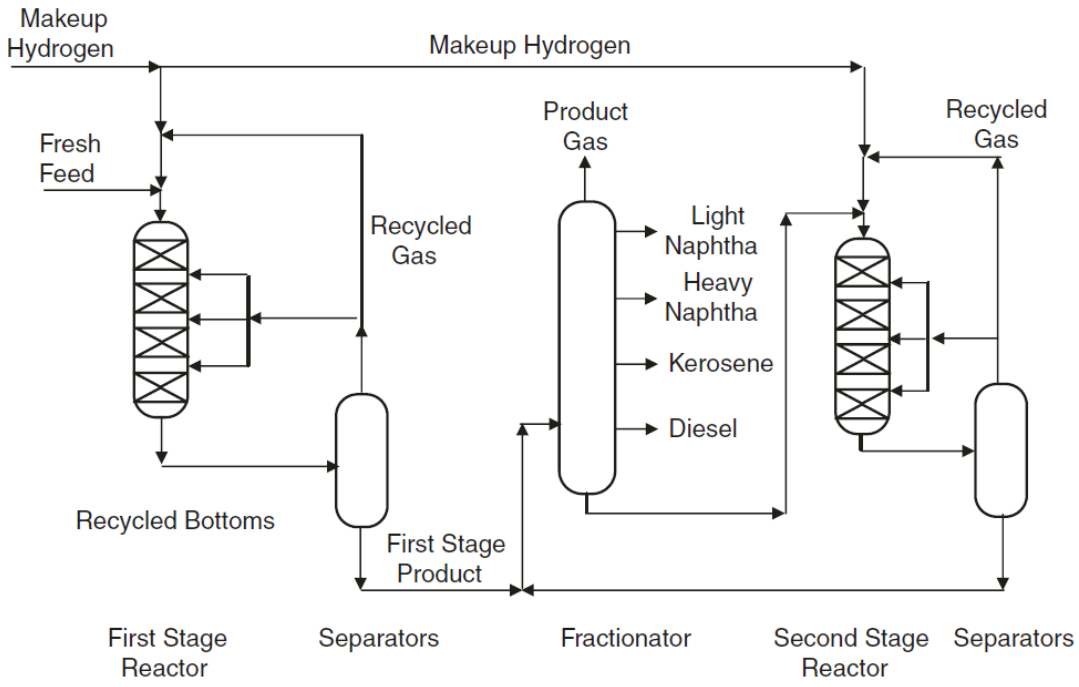

Figure 2.15: Two-stage (reactor) hydrocracker (HC) system with a fractionation tower [1].

Separate hydrotreat hydrocracking is also another uncommon configuration where the feed is processed by a hydrotreater, then is cooled and fractionated first. The bottoms of the fractionator is then sent to a hydrocracking unit.

There are several catalysts available for use in the hydrocracker, depending on if cracking function (acid support) or hydrogenation function (metals) is desired and can be used simultaneously or sequentially. Catalysts with a high cracking function tend to produce lighter ends such as gasoline/naphtha, while catalysts with a high hydrogenation function tend to produce heavier products such as middle distillates [22]. Cracking function catalysts include amorphous oxides (SiO_2, Al_2O_3, Hal-Al_2O_3), crystalline zeolite (modified Y zeolite + Al_2O_3), or a mixture of zeolite and amorphous oxide (modified Y/SiO_2-Al_2O_3). Hydrogenation function catalysts include nobel metals (platinum, palladium) or non-nobel metal sulphides (molybdenum, tungsten, cobalt, nickel) [1].

2.1.1.8 Catalytic Reformer

The catalytic reformer is one of the most valuable process units within a petroleum refinery, as it transforms low-octane heavy naphtha from other units, such as the CDU and hydrocracker, into high-octane products such as aromatics and iso-paraffins (which are subsequently sent to gasoline blends). There are three main configurations of the catalytic reformer: semi-regenerative (SRR), cyclic, and continuous-catalyst regeneration (CCR). SRR is the most common type of reformer used, while CCR is commonly used in modern refineries [23].

Figure 2.16 illustrates the SRR process and operates by periodic catalyst regeneration due to inactivity from coke deposits and as a result, lower conversion; catalyst cycles in SRR last from 6 to 12 months [3]. Several reforming units are placed in series with furnaces between each reactor and is operated continuously. The naphtha feed is mixed with a hydrogen-rich recycle stream and is heated to a temperature of 477 to 517°C before entering the first reactor which operates between 20 to 30 atm [24]. In each reactor, endothermic reactions occur such as:

- naphthene dehydrogenation and dehydrocyclization in the first reactor (fast)
- paraffin isomerization (fast) and naphthene dehydroisomerization (slow) in the second reactor
- dehydrocyclization and hydrocracking in the third reactor (slow)

Thus, heaters are placed in between the reactors with the most heat being provided after the first reactor [1]. The product from the last reactor is then

cooled and sent to a separator where the hydrogen-rich gas is recycled and the reformate product is sent for further distillation or blending.

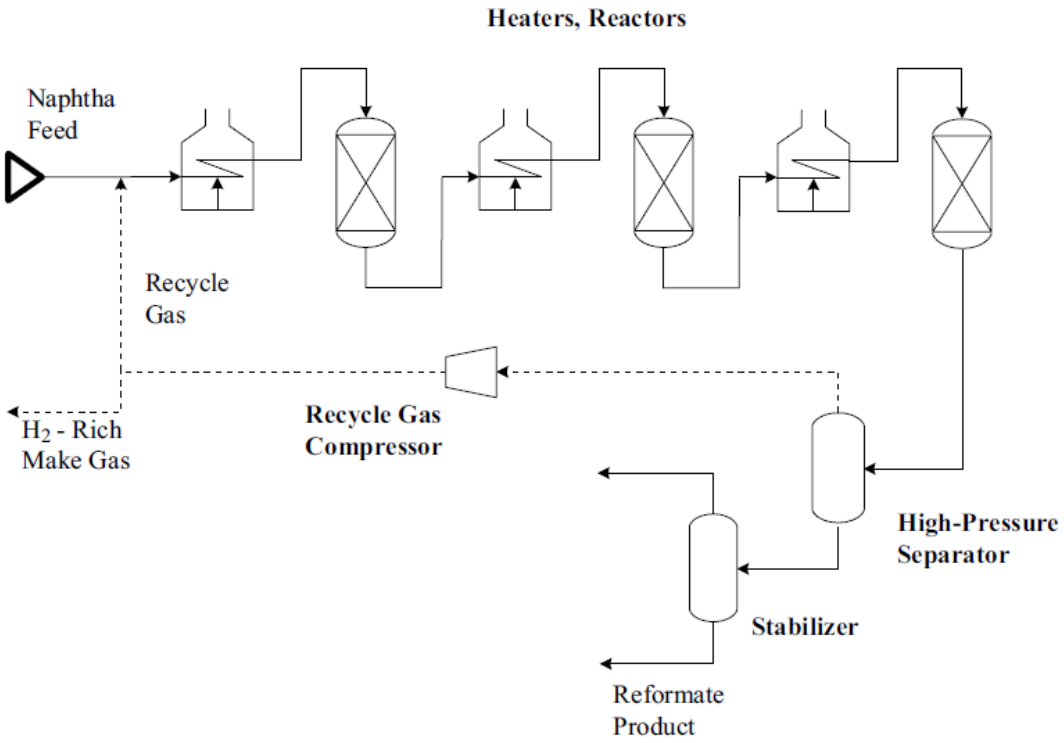

Figure 2.16: Process diagram of the catalytic reformer in a semi-regenerative (SRR) configuration [3].

In cyclic reforming, extra reactor(s) are placed in the process where one reactor can be taken out and be regenerated and operation can be switched over to the spare reactor. In this process, the catalyst is spent in a short amount of time (days up to a month) due to low operational pressure, low hydrogen content, and wide boiling range feed. This process provides advantages over SRR by using a lower operating pressure, lower variation in catalyst activity, and does not need to be shutdown in order to regenerate the catalyst, but has the disadvantages of operational complexity, constant alternation of operation and regeneration pressure, and fast catalyst cycles due to the lower pressures (whereas SRR units may run continuously between 6 to 24 months) [23].

In CCR processes, instead of having to shutdown and regenerate the catalyst periodically, a regeneration system is placed in-line with the process. The general configuration of the system is the same as the aforementioned processes — hydrotreated feed mixed with a hydrogen-rich stream which is then heated and sent to the first reactor. The regenerative cycle works by taking the catalyst from the last reactor, feeding it to the regeneration unit, and then burning off coke deposits on the catalyst in the regenerator. The clean catalyst is then recycled back to the first reactor. In turn, catalyst from the first reactor is sent to the second reactor at a lower activity and the same applies to the second and subsequent reactors. By the time the catalyst reaches the final reactor, it is de-activated to the point where it must be regenerated. Figure 2.17 shows a CCR configuration of reactors in series [25], but the process may also be configured in a "top-wise" fashion where the reactors are stacked on top of one another, and the catalyst trickles down. In this configuration, the effluent from the reactors are sent to be heated before entering the next reactor [1].

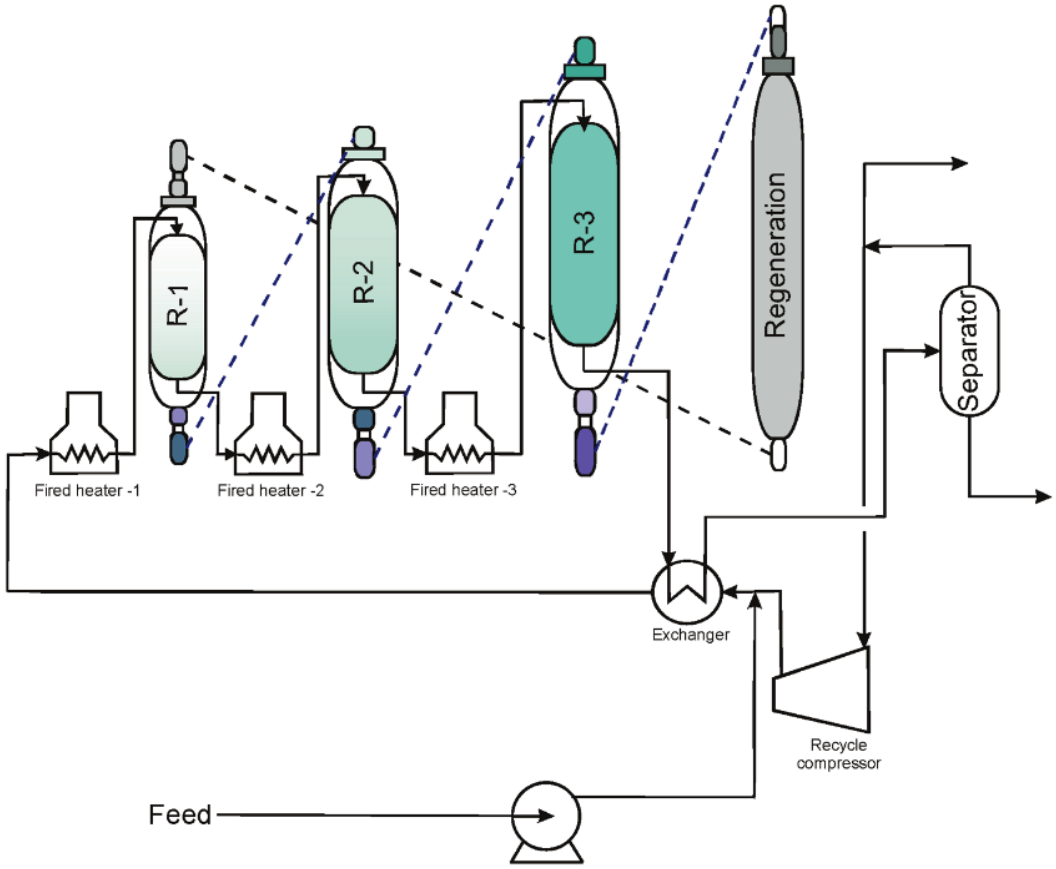

Figure 2.17: Continuous catalytic reformer (CCR) process. Catalyst activity decreases from left to right (i.e. R-1 contains the most active catalyst, followed by R-2 and R-3, then is regenerated after being spent in R-3). This configuration may also be in a vertical/trickle-down configuration, where R-1 would be at the top and the regenerator at the bottom [25].

2.1.1.9 Hydrogen Recovery & Production

Hydrogen is an extremely important component in a petroleum refinery and is required in several unit operations such as the hydrotreater, hydrocracker, and catalytic reformer. While fresh, high-purity hydrogen can be purchased or produced on-site, it is more economical to recover hydrogen from offgas streams and redistribute it throughout the refinery — however, this method is limited. In the refinery, hydrogen can be produced or recovered. The methods that produce hydrogen are steam-hydrocarbon reforming (natural

gas reforming) and partial oxidation of heavy feedstocks, whereas hydrogen recovery units (HRU) are used to recover hydrogen from offgases [26].

In the steam reforming process, there are older and modern procedures to produce hydrogen. The older procedure uses a combination of synthesis gas, high-temperature shift converters (HTSC), low-temperature shift converters (LTSC), and stripping solutions to produce hydrogen at 95 to 98% purity. The overall scheme is as follows [1]:

1. produce synthesis gas (mixture of H_2 and CO) via steam reforming — the feed to the steam reformer is natural gas (high methane content)
2. convert the CO to CO_2 using high-temperature shift converter (HTSC) and low-temperature shift converter (LTSC) — this is so the CO_2 can be stripped from the mixture
3. strip the CO_2 from the H_2/CO_2 mixture using an amine solution or potassium carbonate
4. convert the remaining CO and CO_2 into CH_4 and H_2O using a methanator

In the modern process, instead of a two-stage process to convert CO to CO_2, it is completed in a one-stage high temperature shift converter, and a pressure swing adsorption (PSA) unit is used to strip the CO_2 in place of the amine/potassium carbonate solution. In this process, the final purity is normally 99.9% purity. Figure 2.18 illustrates the improved hydrogen production process [1].

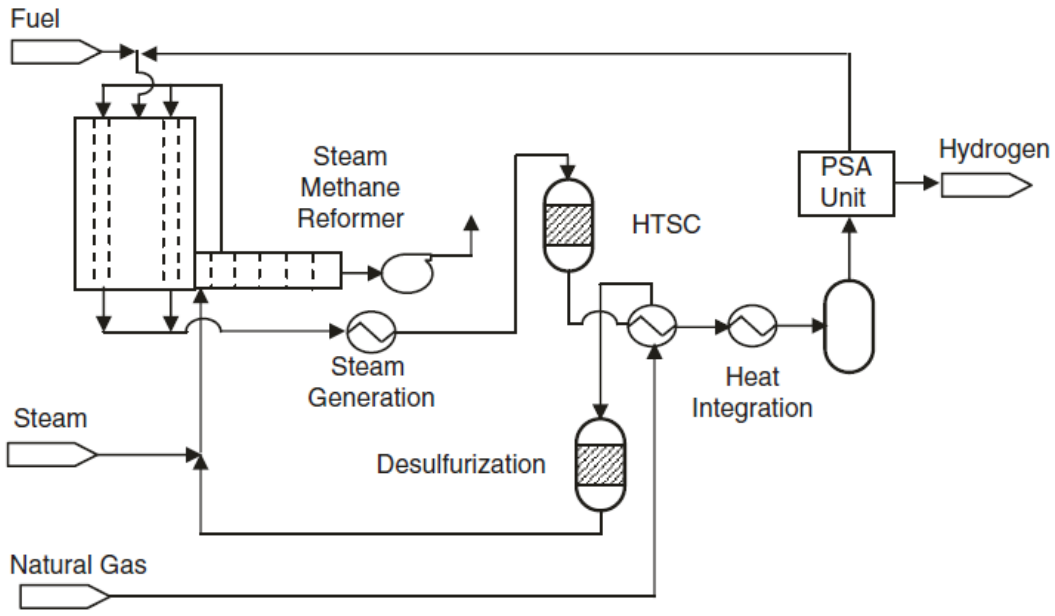

Figure 2.18: Process diagram of the improved hydrogen production process with one shift converter unit (HTSC), steam-reforming unit, and pressure swing adsorption (PSA) unit [1].

Steam reformer units can take various feed streams such as natural gas and refinery offgases [3][26], and/or lighter hydrocarbons (C_3 to C_7) [1]. Regardless, the streams must be pre-treated by removing catalyst poisons (e.g. sulphur, nitrogen, halogenated compounds, mercaptans) as the steam reformer uses a nickel-alumina catalyst. The hydrogenation of organic sulphur and chloride using a hydrotreater with a Co-Mo catalyst is often used to produce HCl and H_2S and then removed. The clean hydrocarbon stream is mixed with superheated steam and heated to 540 to 580°C. Then the stream passes through a furnace reactor with a series of tubes filled with nickel catalyst at a temperature of 820 to 880°C and 294 to 368 psig. The synthesis gas that leaves the reactor consists of H_2, CO, CO_2, CH_4, H_2O, and excess steam [1].

The stream then enters the HTSC, where the purpose of the unit is to convert the CO to CO_2, which is then stripped downstream. The H_2/CO mixture from the stream reformer is fed into the HTSC, where the CO reacts with steam at

300 to 560°C with a CuO/FeO or Fe_3O_4/Cr_2O_3 catalyst in a fixed-bed reactor to produce CO_2. The reaction associated with the conversion is at equilibrium, shown in equation (2.5), so high temperatures will convert less CO due to equilibrium, but lower temperatures will also convert less CO due to lower catalyst activity. The LTSC then converts the remainder of the CO into CO_2 using a CuO/ZnO catalyst at a lower temperature of 230°C. Modern hydrogen removal processes utilize only the HTSC due to improvements in technology and design [1].

$$CO + H_2O \rightleftharpoons CO_2 + H_2 \tag{2.5}$$

Finally, the PSA unit selectively adsorbs compounds that are less volatile and highly polar (i.e. most compounds in the stream except for hydrogen) on the internal surfaces of the adsorbent bed. This occurs at high pressures at approximately 275 psig. The desorption (regneration) of the PSA unit occurs at low pressures (2.8 to 5.6 psig) [26]. Thus, several PSA units are employed in this step so that some units may operate while others are being regenerated. This process results in a 99.9% purity hydrogen stream [1].

Heavy feedstocks such as vacuum residue and bitumen can be partially oxidized to produce synthesis gas, which can then be separated. In this process, the feed is partially burned with oxygen where thermal cracking occurs and synthesis gas (CO and H_2) is produced. These reactors operate between 1093 and 1538°C and 1200 to 2000 psig. The synthesis gas is then cooled and sent to a single-stage shift converter and the remainder of the process to strip the hydrogen is employed [26].

Hydrogen-rich offgases from other unit operations in the refinery are consolidated and sent to the HRU. Before being sent the HRU, the feed is conditioned

by knocking out liquids and stripping any H_2S from the stream. The stream is then sent to a PSA unit for removal of CO_2 and other impurities [26].

2.1.1.10 Product Blending

In petroleum refining, unit operation products are generally not ready to be sold and must undergo blending with other products in order to meet specifications for commercial use. Products that are produced in the refinery depend on the configuration of the refinery (e.g. one refinery may produce more diesel than another), but generally, the products generated in a refinery are gasoline, kerosene, jet fuel, diesel, fuel, residual fuel oil, lubricants, asphalt, petroleum coke, and liquefied petroleum gas (LPG). Some of these products can come in various grades (e.g. different grades of gasoline based on octane number). Depending on the final product, various specifications must be met, such as, Reid vapour pressure (RVP), flash point, viscosity, pour point, cloud point, aniline point, smoke point, research octane number (RON), and motor octane number (MON). Some examples include RON, MON, RVP, and volatility for gasoline; flash point and viscosity for kerosene; and diesel index, flash point, pour point, and viscosity for gas oils [1].

There are several formulations for calculating properties in blending, both linear and nonlinear. A common linear blending rule for properties in a product tank is shown in equation (2.6) [27]:

$$PR_{p,k,t} = \sum_{i} pr_{i,k} v^{I}_{i,p,t} \quad \forall\, p,k,t \tag{2.6}$$

with product specification bounds:

$$pr_{p,k}^{min} \leq PR_{p,k,t} \leq pr_{p,k}^{max} \tag{2.7}$$

where $PR_{p,k,t}$ is the value for the property k for product p in time period t, $pr_{i,k}$ is the property value for intermediate blend stream i, and $v_{i,p,t}^{I}$ is the volume fraction of inlet stream i. $pr_{p,k}^{min}$ and $pr_{p,k}^{max}$ are the minimum and maximum product specifications for $PR_{p,k,t}$. Equation 2.6 stays linear if $pr_{i,k}$ stays fixed using correlations or average values from plant data [27].

However, equation (2.6) may be nonlinear if $v_{i,p,t}^{I}$ is linked to volumetric flow rates with:

$$v_{i,p,t}^{I} F_{p,t}^{P} = F_{i,p,t}^{I} \quad \forall\, p, k, t \tag{2.8}$$

where $F_{i,p,t}^{I}$ are the mass flow rates being sent to the blender at time t with stream i for product p, and $F_{p,t}^{P}$ represents the outlet flow rate of the blender of the blended product p at time t [28]. The sum of the inlet flow rates are not necessarily equal to the outlet flow rate, due to the inventory in the blend tanks during the period.

Mendez et al. [28] linearize equation (2.6) by multiplying by $F_{p,t}^{P}$ to yield:

$$PR_{p,k,t} F_{p,t}^{P} = \sum_{i} pr_{i,k} v_{i,p,t}^{I} F_{p,t}^{P} \quad \forall\, p, k, t \tag{2.9}$$

then substituting using equation (2.8):

$$PR_{p,k,t} F_{p,t}^{P} = \sum_{i} pr_{i,k} F_{i,p,t}^{I} \quad \forall\, p, k, t \tag{2.10}$$

and by multiplying the terms in product specification bounds in equation 2.7

with $F^P_{p,t}$:

$$pr^{min}_{p,k} F^P_{p,t} \leq PR_{p,k,t} F^P_{p,t} \leq pr^{max}_{p,k} F^P_{p,t} \tag{2.11}$$

and finally substituting equation 2.10 in equation 2.11:

$$pr^{min}_{p,k} F^P_{p,t} \leq \sum_i pr_{i,k} F^I_{i,p,t} \leq pr^{max}_{p,k} F^P_{p,t} \quad \forall\, p,k,t \tag{2.12}$$

From this final relationship, the nonlinear term $v^I_{i,p,t}$ is not required and the relationship stays linear — $pr^{min}_{p,k}$ and $pr^{max}_{p,k}$ are constants, and $pr_{i,k}$ stays fixed using correlations or average values from plant data [27]. This is in the case if $v^I_{i,p,t}$ is nonlinear due to equation 2.8.

Along with the blending of intermediate product streams to obtain final products, several additives are blended into the final products for additional protection against corrosion, shelf-life, freezing, and other desirable properties. Common additives to gasoline include anti-oxidation, metal passivation (slow oxidation from trace metals), corrosion inhibitors, anti-icing, IVD (intake valve deposits) control, CCD (combustion chamber deposits) control, and anti-knock. Similarly for diesel fuel, additives include anti-oxidation, cetane improvement, dispersion, anti-icing, detergent, metal passivation, corrosion inhibitors, and cold-flow improvement [3].

2.2 Single Period Refinery Formulation

Single period refinery models may be used for a variety of studies, including modelling under uncertainty and incorporation of detailed models of particular unit operations. This is compared to a multiperiod problem, where the overall objective is to optimize operation over a planning horizon. Although

refinery planning and scheduling problems are generally multiperiod, there are formulations within refinery optimization that are single period such as stochastic optimization [2].

Pinto and Moro [29] developed a nonlinear single period planning model framework that includes crude processing and product blending. The model was developed in order to improve on current practices at the time that mainly involved linear models formulated in software such as Refinery and Petrochemical Modeling System (RPMS) and Process Industry Modeling System (PIMS). Furthermore, complex nonlinear models are often restricted to a subsection of the plant and do not represent the whole refinery. Two real world petroleum refineries were modelled using this framework and produced promising results from this application. However, the study is limited due to the linear models used for some of the major unit operations such as the fluidized catalytic cracker (FCC) [29].

Li et al. [30] developed a refinery planning model that uses nonlinear empirical models for the crude distillation unit (CDU), FCC, and blending process in order to improve on the common practice of linear process models. The model considers crude characteristics, unit yields, qualities, and so on. The CDU is modelled by determining the size of the swing cuts (which are the ranges for the cut points) using the weight transfer ratio (WTR) of the CDU fractions (calculated using an empirical procedure and ASTM boiling ranges) and then the planning model optimizes these cut points. The FCC is modelled using a regression model based on work by Gary et al. [4] [30].

Elkamel et al. [31] developed a detailed MINLP refinery optimization model that integrates planning with CO_2 emission reduction. The study includes property blending of common qualities that are measured in refineries, and

includes API gravity, sulfur, octane number, Reid vapour pressure, cetane number, diesel index, and smoke point. Most of the properties utilize a blending index relation where the sum of the property index multiplied by the mass or volume fraction for all streams equals the blending index [31]:

$$BI_p = \sum_{s=1}^{s} IN_{p,s} X_s$$

where BI_p is the blending index for the property p, $IN_{p,s}$ is the index for property p in stream s, and X_s is the mass or volume fraction (depending on the property). Following the property calculation, the remainder of the formulation contains unspecified relationships between unit yields, operational variables, and properties. However, a unique formulation of the study includes fuel switching for the furnaces in the refinery, where a binary variable is used to select the type of fuel used for the furnaces. As well, part of the objective includes reduction of CO_2 emissions, and the formulation includes the quantitative value of CO_2 emissions from the furnaces, as well as whether or not to employ CO_2 capture for a furnace and which type of capture system. Due to the complexity of the inclusion of CO_2 capture options and fuel switching, the model remains as a single period model [31].

2.3 Multiperiod Planning & Applications

Multiperiod optimization is a tool widely used in many processes to provide optimal states of operation over a specified time horizon. Multiperiod optimization appears in problems, such as chemical and oil refineries that process various types of crude and products, problems that have seasonal demand, financial risk in investments, and scheduling and planning scenarios. These

type of optimization formulations differ from traditional single-period optimization methods in that the problem is optimized for multiple time periods, each with its own characteristics and properties, while being optimal for the problem as a whole. Additionally, multiperiod optimization takes into account long-term dynamics of the system. There are two distinct classes of variables which are optimized – design variables and state variables. Design variables represent information that stays constant throughout the entire period (e.g. unit operation dimensions), while state variables represent operating conditions (e.g. flow rates, temperature, pressure) and will vary from each time period.

By incorporating a planning horizon, various problems may be studied such as shutting down and starting up a unit, scheduling of crude oils, uncertainty over time, and incorporating detailed descriptions of unit operations (e.g. catalyst activity). Benefits include a more detailed description of the refinery, the ability to predict how a unit will behave, and can provide a closer representation of the refinery operation. There have been several formulations of multiperiod planning and scheduling optimization of oil refineries [32], maintenance scheduling of refinery units [33], and optimization for risk management [27]. Some models are more rigorous than others in terms of unit operation modelling.

To formulate a multiperiod problem, an additional temporal index is added to all variables and most constraints, and in turn, generates a significantly larger problem based on the number of periods and the size of the original model. However, in order to exploit the properties of a multiperiod problem, there must be constraints that link the periods together using 'pass-on' or 'interconnection' variables which link time periods together. These interconnection variables can be considered as design variables (e.g. intermediate tank

inventories) [34]. Multiperiod problems have a 'sparse' structure that can be exploited - i.e. rows of the constraint matrices will only contain a few elements while the rest of the row contains zeros. In this structure, the rows and columns correspond to constraints and variables, respectively. Another way to visualize the multiperiod structure is by a block-type structure. Figure 2.19 by Neiro and Pinto [32] illustrates this concept, where each period is contained within a block and the rest of the matrix row contain zeros. This is repeated for each time period, and the linking constraints (in this case, inventory constraints) encompass multiple time periods and are shown at the bottom of the figure.

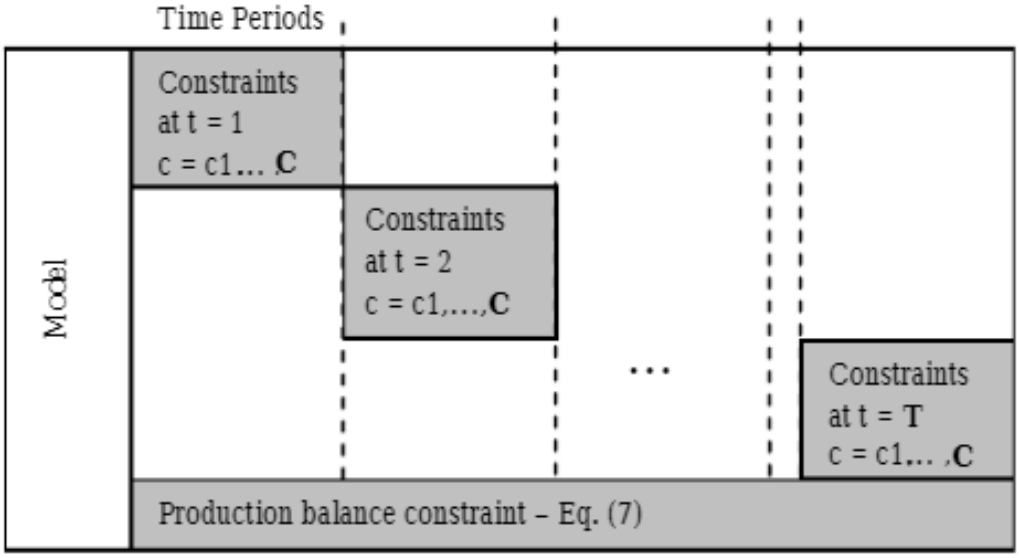

Figure 2.19: Block structure of the multiperiod problem constraints presented by Neiro and Pinto. The constraints of the multiperiod problem resemble a sparse structure, where a block of constraints pertain to one period and the rest of the row contain zeros. The production balance constraints (i.e. inventory/interconnectivity constraints) are segregated and gathered at the bottom of the structure to show their connectivity between units [32].

There have been multiple contributions within the past few decades in multi-period petroleum refinery planning. Neiro and Pinto [32] have formulated a multiperiod optimization for petroleum refinery production using a relatively rigorous model. The study employed a single period non-linear programming

(NLP) formulation for planning proposed by Pinto et al. [29], which generalized each unit operation with mixers before the unit, and splitters for the output stream. The planning model describes the refinery using mass balances at the mixers and splitters; demand, operating and quality constraints; relationships between unit feed and mixer feed properties; and relationships between outlet flow and properties with inlet flow and properties (dependent on the unit operation). The product flow rates were determined using mass balances and yield expressions, which are based on standard values that are found using average values from plant data. The majority of the physical properties are calculated using correlations (mixing indexes), with the rest being calculated using gains through correlations developed by Pinto et al. [29]. The focus of the study aimed to formulate a multiperiod optimization model using a NLP model, whereas the previous NLP model by Pinto et al. incorporated optimization over a single period. In the multiperiod formulation, each consecutive time period is linked by inventory variables. Neiro and Pinto [32] proposed three models: multiperiod production planning using inventory levels and crude selection; multiperiod with uncertainty in fluctuations of feed and product prices and demand, with inventory levels; and an extended model which incorporates the two models and includes constraints on petroleum availability based on brine separation times. The multiperiod solution spans up to 5 time periods for the first model, and 20 time periods for the second and third model, each period being 1 day each.

Zhao et al. [35] formulated a multiperiod planning model for an integrated refinery production and utility system, and formulates the problem as a mixed-integer NLP (MINLP). In the study, the production planning model that is studied is shown in Figure 2.20 and is described using demand constraints, material inventory balances (used to link periods), operation modes, and

blending constraints, which includes mass balances and property constraints. The property calculations include cetane number, API gravity, pour point, sulfur content, and carbon content, and are simplified using quality indexes for linearity. The objective of the study is to optimize the production planning of the refinery units, while also optimizing the operational planning of the utility equipment. However, due to the non-convexity and potential of non-convergence, the two problems are decomposed into MILP models and simplified to provide an initial estimate to the full problem. The final blend product properties are fixed to obtain an initial solution estimate for the production MILP model. The solution of the production model is passed onto the utility MILP model (utility demand and byproduct production). The solution of these two problems is then passed back into the original integrated model as an initial guess. The solution includes the following three scenarios, all with a planning horizon of 8 time periods: changing market demand of 6 products at each period, but within the production capacity based on maximum energy generation; changing demand of 6 products, but allowing demand to exceed production capacity (i.e. cannot fulfill the entire demand); and optimization of penalties on environmental regulations on energy utilization and production unit operation [35].

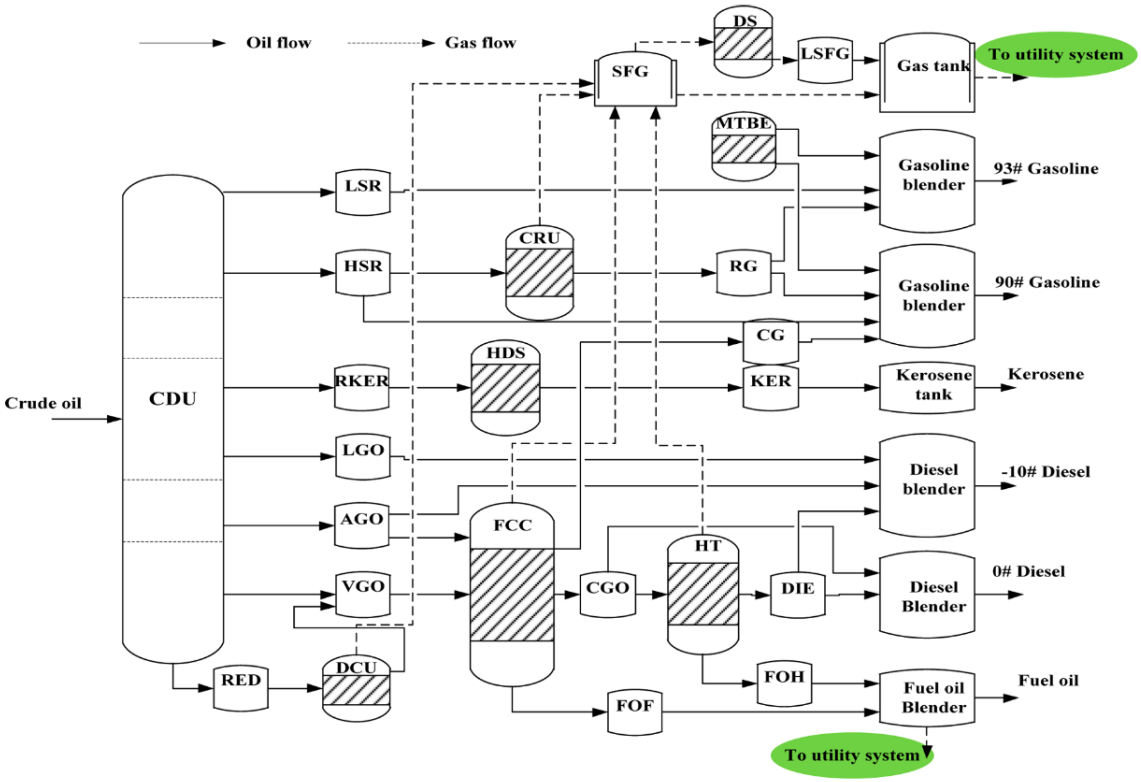

Figure 2.20: Process diagram of the multiperiod integrated refinery production and utility system MINLP model used in the study by Zhao et al. [35].

Other examples in multiperiod refinery planning include Zhang and Hua [36] who incorporate a utility system into a petroleum refinery by splitting the system into parts - first, the overall planning problem is solved and includes crude selection, products, and unit interactions; second, the utility system and process system are solved separately and reconciled to check for a feasible solution. Leiras et al. [37] reformulated a linear multiperiod model that was adapted from Allen [38] and studied the problem under uncertainty in costs, product price and demand, and yields using robust optimization. However, this was only on a small case study and not representative of a full real-world refinery. Alattas et al. [9][39] have contributed several papers in nonlinear multiperiod refineries with a nonlinear crude distillation unit (CDU) model. The first paper involved developing a nonlinear fractionation index for the

CDU for a single-period planning model. The second paper extends this work into a MINLP over multiple time periods in order to model the sequencing, changeovers, and processing of the crude oils.

2.4 Nonlinear Programming

In optimization theory, mathematical programming is used to find the optimal solution of an objective function, either minimized or maximized, subject to a set of constraints which can be inequalities or equalities. A mathematical program can be defined generally as:

$$\min_{x} f(x) \tag{2.13}$$

$$\text{Subject to:} \quad g_i(x) \geq 0, \quad i = 1, ..., m \tag{2.14}$$

$$h_j(x) = 0, \quad j = 1, ..., p \tag{2.15}$$

where the objective is to find a vector x that minimizes a function $f(x)$ subject to m inequality constraints $g_i(x)$, and p equality constraints $h_j(x)$. The problem is then considered **nonlinear** if any equations in the problem are nonlinear in x [40].

In both linear programming (LP) and nonlinear programming (NLP) problems, there can be multiple solutions. However, they differ in that the solution to an LP is a global solution, but in an NLP problem there can be multiple solutions in the form of global or local optimums as shown in Figure 2.21. This poses as problem, as most nonlinear solvers rely on gradients and thus, will find the optimum nearest to its starting point [41]. An exception to this is in a convex NLP, where the local optimum will be the global optimum for the problem. Two popular methods that are widely used in academia to solve nonlinear

optimization problems are interior point algorithms and generalized reduced gradient (GRG) methods.

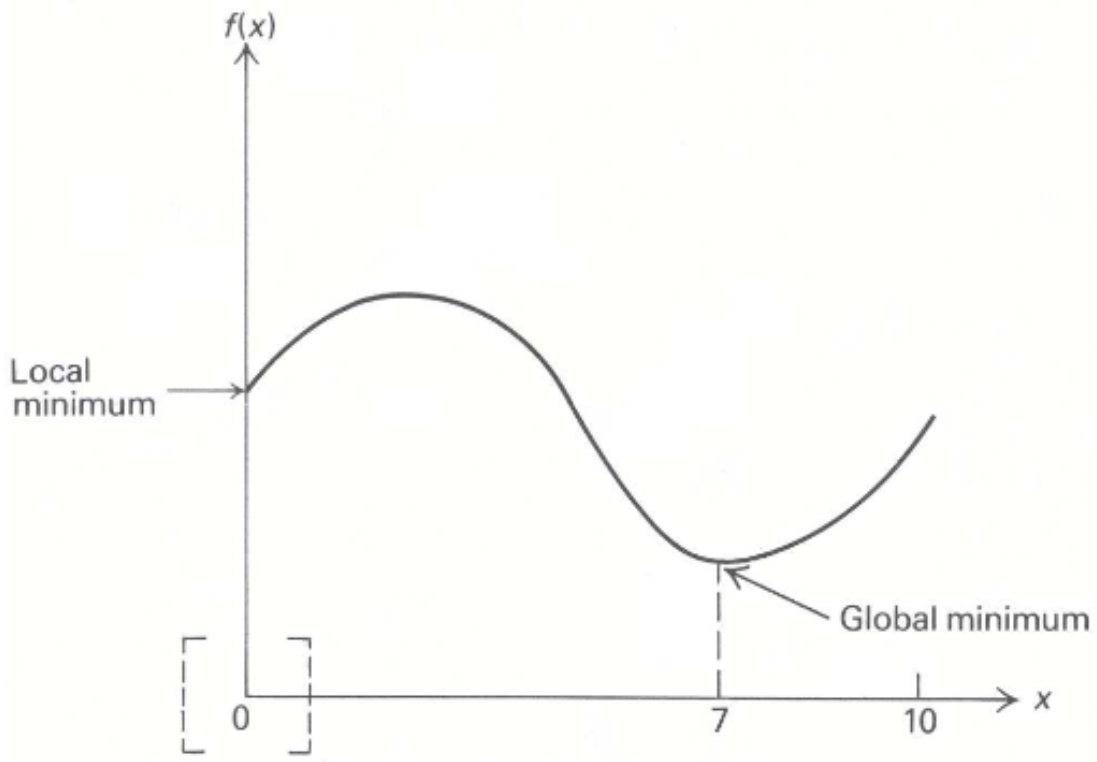

Figure 2.21: Example of a nonlinear function and the two minima (local and global) within the bounds of the function [41].

2.4.1 Interior Point Approaches to NLP

Interior point algorithms do not make a priori assumptions as to which constraints are active. They apply Newton-type iteration on a large-scale system corresponding to a modified version of the first-order optimality conditions (KKT conditions). Upon convergence, the active inequality constraints are identified. There are two major types of interior point methods, which are the primal Newton barrier method and the primal-dual barrier method, and in both, the algorithm begins in the interior of the feasible region and proceeds

through a path using Newton's method.

2.4.1.1 Primal Newton Barrier Method

Starting from the NLP formulation from equations 2.13 to 2.15 with the addition of $x \geq 0$ for non-negativity, slacks are introduced to transform the inequalities into equalities.

$$\min_x \ f(x) \tag{2.16}$$

$$\text{s.t.} \quad h(x) = 0 \tag{2.17}$$

$$x \geq 0 \tag{2.18}$$

From this new NLP formulation (with equalities only), the logarithmic barrier function $B(x, \mu)$ is associated with the original objective function [42]. The new formulation is as follows, with the non-negativity constraint accounted for in the the barrier function argument:

$$\min_x \ B(x, \mu) = f(x) - \mu \sum_{i=1}^{n} \ln(x^i) \tag{2.19}$$

$$\text{s.t.} \quad h(x) = 0 \tag{2.20}$$

Note that as x moves toward the boundary, the barrier function becomes larger and due to the minimization of the problem, the solution should tend away from the boundary and remain within the interior (hence the name of the method). This logarithmic effect can be controlled by μ [43]. The Lagrange function is then shown as the following:

$$\mathcal{L}_\mu(x, \lambda) = f(x) - \mu \sum_{i=1}^{n} \ln(x_i) - h(x)^T \lambda \tag{2.21}$$

Thus, the necessary first-order optimality conditions (KKT) (stationary) for the NLP formulation (i.e. Lagrange parameter λ_μ and x that satisfies the following) are as follows [40]:

$$\nabla_x \mathcal{L}_\mu(x, \lambda) = 0 \tag{2.22}$$

$$\nabla_\lambda \mathcal{L}_\mu(x, \lambda) = 0 \tag{2.23}$$

which can also be equivalently written as:

$$\nabla f(x) - \mu X^{-} 1e - \nabla h(x)^T \lambda = 0 \tag{2.24}$$

$$-h(x) = 0 \tag{2.25}$$

where μ is the barrier parameter, X is a diagonal matrix of the decision variables, e is a column of ones, and λ and z are the Lagrangean multipliers for equations 2.17 and 2.18, respectively.

The solution to the barrier problem can be solved using a Newton step direction $-J_\mu d_k = 0 = \nabla \mathcal{L}$, where:

$$J_\mu = \begin{bmatrix} \frac{d}{dx}\nabla_x \mathcal{L} & \frac{d}{d\lambda}\nabla_x \mathcal{L} \\ \frac{d}{dx}\nabla_\lambda \mathcal{L} & \frac{d}{d\lambda}\nabla_\lambda \mathcal{L} \end{bmatrix} = \begin{bmatrix} \nabla_x^2 f(x) + \mu X^{-2} - \nabla_x^2 h(x)\lambda & -\nabla_x h(x) \\ -\nabla_x h(x) & 0 \end{bmatrix} \tag{2.26}$$

$$d_k = \begin{bmatrix} \Delta x_k \\ \Delta \lambda_k \end{bmatrix} \tag{2.27}$$

$$\Delta \mathcal{L} = \begin{bmatrix} \nabla f(x) - \mu \sum_{i=1}^{n} \frac{1}{x_i} - \nabla h(x)^T \lambda \\ -h(x) \end{bmatrix} \tag{2.28}$$

Putting it all together:

$$\begin{bmatrix} \nabla_x^2 h(x)\lambda - \nabla_x^2 f(x) - \mu X^{-2} & \nabla_x h(x) \\ \nabla_x h(x) & 0 \end{bmatrix} \begin{bmatrix} \Delta x_k \\ \Delta \lambda_k \end{bmatrix} = \begin{bmatrix} \nabla f(x) - \mu \sum_{i=1}^{n} \frac{1}{x_i} - \nabla h(x)^T \lambda \\ -h(x) \end{bmatrix} \tag{2.29}$$

Once the step direction is solved, a Newton step is taken and the barrier parameter μ is reduced in the next iteration until convergence [44]. The following algorithm describes the primal Newton barrier method [44]:

Algorithm 1 Primal Newton Barrier Method

Choose $x_0 \in \mathcal{F}^0$ and $\mu_0 > 0$

k = 0

while $||x_k - x_{k-1}|| \geq \epsilon$ or stop criterion not satisfied **do**

Compute constrained Newton direction d_k using equation 2.29

$x_{k+1} = x_k + \alpha \Delta x_k$

$\lambda_{k+1} = \lambda_k + \alpha \Delta \lambda_k$

Choose $\mu_{k+1} \in (0, \mu_k)$

For short-step, $\mu_{k+1} = \mu_k \Big/ \left(1 + \frac{1}{8\sqrt{n}}\right)$

$k \leftarrow k + 1$

end while

2.4.1.2 Primal-Dual Barrier Method

The primal-dual barrier method is the underlying algorithm used by modern large-scale solvers, such as IPOPT, and is a generalized version of the primal Newton barrier method as described previously. In the Primal-Dual method, the Newton step updates both the primal and dual variables.

Starting with equations 2.16 to 2.18 with z as slack variables for the inequalities

constraints, the primal-dual KKT optimality conditions are as follows [42]:

$$F_\mu(x_\mu, \lambda_\mu, z_\mu) = \begin{bmatrix} \nabla f(x) + \nabla h(x)\lambda - \mu X^{-1} e \\ h(x) \\ x_i z_i \quad \forall i = 1, 2, ..., n \end{bmatrix} = 0 \tag{2.30}$$

The substitution $Ze = \mu X^{-1} e$ can be used [45], where Z is a diagonal matrix of the Lagrange multipliers for the non-negativity constraint and thus, the KKT conditions can be expressed as [44]:

$$F_\mu(x_\mu, \lambda_\mu, z_\mu) = \begin{bmatrix} \nabla f(x) + \nabla h(x)\lambda - Ze \\ h(x) \\ XZe - \mu e \end{bmatrix} = 0 \tag{2.31}$$

In order to solve this system, first, a search direction for $(x_\mu, \lambda_\mu, z_\mu)$ must be found by solving $J_\mu(x, \lambda, z)d = -F_\mu(x, \lambda, z)$, where J_μ is the Jacobian and $d = (\Delta x, \Delta\lambda, \Delta z)$:

$$\begin{bmatrix} H(x) & \nabla h(x)^T & -I \\ \nabla h(x) & 0 & 0 \\ Z & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta z \end{bmatrix} = - \begin{bmatrix} \nabla f(x) + \nabla h(x)^T \lambda - z \\ h(x) \\ XZe - \mu e \end{bmatrix} \tag{2.32}$$

where the Hessian $H(x) = \nabla^2_{xx} \mathcal{L}_\mu(x, \lambda, z)$ where $\mathcal{L}_\mu(x, \lambda, z) = f(x) - h(x)^T\lambda - z$ [42]. The solution yields $d = (\Delta x, \Delta\lambda, \Delta z)$ and can be used as a step change for the next iteration of $(x_\mu, \lambda_\mu, z_\mu)$ [42][45][46]:

$$(x_{k+1}, \lambda_{k+1}, z_{k+1}) = (x_k, \lambda_k, z_k) + \alpha(\Delta x, \Delta\lambda, \Delta z) \tag{2.33}$$

In order to calculate the step size α, there are several strategies. One strategy

outlined by Wright and Nocedal [47] is shown below:

$$\alpha_{max,x} = \min\left\{1, -\frac{x_{k,i}}{\Delta x_{k,i}} \mid \Delta x_{k,i} < 0\right\} \tag{2.34}$$

$$\alpha_{max,z} = \min\left\{1, -\frac{z_{k,i}}{\Delta z_{k,i}} \mid \Delta z_{k,i} < 0\right\} \tag{2.35}$$

$$\overline{\alpha}_{k,x} = \eta_k \alpha_{max,x}$$

$$\overline{\alpha}_{k,z} = \eta_k \alpha_{max,z}$$

where $\eta_k \in [0.9, 1]$. Additionally, the *fraction to the boundary* rule may be used, where $(\alpha_{k,x}, \alpha_{k,z}) < (\overline{\alpha}_{k,x}, \overline{\alpha}_{k,z})$, which leads to a guaranteed decrease in primal and dual infeasibilities, as shown by Curtis and Nocedal [48]. Thus, the primal-dual algorithm is as follows [44]:

Algorithm 2 Primal-Dual Barrier Method

Choose (x_0, λ_0, z_0) that $(x_0, z_0) > 0$

$k = 0$ and $\mu_0 = (x^0)^T s^0 / n$

while Termination criteria not met **do**

$\mu_k = \rho \mu_{k-1}$

Compute the primal-dual Newton direction $(\Delta x, \Delta\lambda, \Delta z)$ using equation 2.32

Calculate step-size α_k using equations 2.34 and 2.35

$x_k = x_{k-1} + \alpha_{k,x} \Delta x$

$(\lambda_k, z_k) = (\lambda_{k-1}, z_{k-1}) + \alpha_{k,z}(\Delta\lambda, \Delta z)$

$\mu_{k+1} = (x^{k+1})^T s^{k+1} / n$

$k \leftarrow k + 1$

end while

2.4.1.3 Interior Point Optimization (IPOPT)

Coined IPOPT, a full space interior point solver using the primal-dual barrier method was developed by Wachter and Biegler in 2006, and can solve large-scale NLPs efficiently [42]. IPOPT employs a logarithmic barrier method to the equality constraints in the nonlinear model and iterates through solving the barrier problem with changing barrier parameter (μ), as outlined in the previous section. However, Wachter and Biegler modify the algorithm in order to improve efficiency and avoid potential pit-falls. Some of these modifications include a filter line-search method, feasibility restoration phase, second-order corrections, and inertia correction.

The termination criteria for the barrier problem are as follows:

$$E_0(x^*, \lambda^*, z^*) \leq \epsilon_{tol} \tag{2.36}$$

$$E_\mu(x, \lambda, z) = \max \left\{ \frac{\|\nabla f(x) + \nabla h(x)\lambda - z\|_\infty}{s_d}, \|h(x)\|_\infty, \frac{\|XZe - \mu e\|_\infty}{s_c} \right\} \tag{2.37}$$

where equation 2.36 is the termination when $\mu = 0$ in equation 2.37, and measures the original problem optimality error [42]. $\epsilon_{tol} > 0$ is user-defined, and $s_d, s_c \geq 1$ are scaling factors for λ and z. (s_d, s_c) may become large when, for example, the gradients of the active constraints are close to being linearly dependent and thus are calculated using:

$$s_d = \max \left\{ s_{max}, \frac{\|\lambda\|_1 + \|z\|_1}{m+n} \right\} / s_{max} \tag{2.38}$$

$$s_d = \max \left\{ s_{max}, \frac{\|z\|_1}{n} \right\} / s_{max} \tag{2.39}$$

where n and m are the number of variables and equality constraints, respectively, and $s_{max} \geq 1$ and is user-defined.

While the aforementioned termination criteria is for $\mu = 0$, Wachter and Biegler [42] utilize an approach proposed by Byrd et al. [49] to achieve fast convergence for other values of μ. In this approach, it is proven that superlinear convergence occurs under standard second-order sufficient conditions and thus, an 'outer-loop' is used to find an approximate solution to the barrier problem. The tolerance for this approximation, for a given μ_j is:

$$E_{\mu j}(x^*_{j+1}, \lambda^*_{j+1}, z^*_{j+1}) \leq \kappa_\epsilon \mu_j \tag{2.40}$$

where $\kappa_\epsilon > 0$, and j is the iteration for the 'outer-loop'. Thus, the barrier parameter for the next iteration μ_{j+1} is calculated:

$$\mu_{j+1} = \max\left\{\frac{\epsilon_{tol}}{10}, \min\left\{\kappa_\mu \mu_j, \mu_j^{\theta_\mu}\right\}\right\} \tag{2.41}$$

where $\kappa_\epsilon \in (0,1)$ and $\theta_\mu \in (1,2)$.

IPOPT then solves the barrier problem as per equation 2.32 with all elements subscripted with iteration k, representing the 'inner-loop' with the exception of μ which is subscripted with iteration j for the 'outer-loop'. However, a slight modification is applied in order to solve an easier, symmetric linear system where the last block row is eliminated:

$$\begin{bmatrix} H(x_k) + \sum_k & \nabla h(x_k) \\ \nabla h(x_k)^T & 0 \end{bmatrix} \begin{bmatrix} \Delta x_k \\ \Delta \lambda_k \end{bmatrix} = - \begin{bmatrix} \nabla B_{\mu_j}(x_k) + \nabla h(x_k)\lambda_k \\ h(x_k) \end{bmatrix} \tag{2.42}$$

where $\sum_k = X_k^{-1} Z_k$ and then $\Delta z_k = \mu_j X_k^{-1} e - z_k - \sum_k \Delta x_k$. However, in order to ensure certain descent properties and prevent non-existent solutions due to

singularity, the expression 2.42 is further modified:

$$\begin{bmatrix} H(x_k) + \sum_k + \delta_w I & \nabla h(x_k) \\ \nabla h(x_k)^T & \delta_c I \end{bmatrix} \begin{bmatrix} \Delta x_k \\ \Delta \lambda_k \end{bmatrix} = - \begin{bmatrix} \nabla B_{\mu_j}(x_k) + \nabla h(x_k)\lambda_k \\ h(x_k) \end{bmatrix} \quad (2.43)$$

where $\delta_w, \delta_c \geq 0$ and the selection of these parameters (inertia correction) are described in Wachter and Biegler [42].

IPOPT then follows equation 2.33 in order to make the step change. IPOPT will take a different approach in selecting α than described in the previous section, and uses the *fraction-to-the-boundary* rule:

$$\alpha_k^{max} = \max \left\{ \alpha \in (0,1] \, : \, x_k + \alpha \Delta x_k \geq (1 - \tau_j) x_k \right\} \quad (2.44)$$

$$\alpha_k^{z} = \max \left\{ \alpha \in (0,1] \, : \, z_k + \alpha \Delta z_k \geq (1 - \tau_j) z_k \right\} \quad (2.45)$$

where $\tau_j = \max \{ \tau_{min}, 1 - \mu_j \}$, $\tau_{min} \in (0,1)$.

Furthermore, Wachter and Biegler prove that using a backtracking line-search for step size $\alpha_k \in \left(0, \alpha_k^{max}\right]$ using a decreasing sequence of trial step sizes $\alpha_{k,l} = 2^{-l} \alpha_k^{max}$, $l = 0, 1, 2, ...$ ensures global convergence under certain assumptions. In this procedure, the barrier problem is solved as a bi-objective problem - minimizing the objective function, and minimizing the constraint violation. A trial step point $x_k(\alpha_{k,l}) = x_k \alpha_{k,l} \Delta x_k$ is taken and checked if the barrier objective function is sufficiently improved. The method also uses a filter, where prohibited combinations of the constraint violations and objective function values for a successful trial step point are contained in the filter. The filter then rejects the trial point if the trial point exists in the filter. The purpose of the overall method is to ensure that the algorithm does not cycle between two points that decrease the constraint violation and barrier objective function [50].

Another fail-safe that is necessary is the failure to find a new trial step size where $\alpha_{k,l} \leq \alpha_k^{min}$. In this case, the algorithm switches to a *feasibility restoration phase*. In this phase, the algorithm attempts to find a new point $x_{k+1} > 0$ that is acceptable to the filter and holds for the following condition:

$$\theta(x_k(\alpha_{k,l})) \leq (1 - \gamma_\theta)\theta(x_k) \tag{2.46}$$

$$\varphi_{\mu_j}(x_k(\alpha_{k,l})) \leq \varphi_{\mu_j}(x_k) - \gamma_\varphi \theta(x_k) \tag{2.47}$$

where fixed constants γ_θ and $\gamma_\varphi \in (0, 1)$ are user-defined, θ is the constraint violation, and φ is the barrier objective function value. This is accomplished by reducing the constraint violation iteratively. The restoration phase will fail if the problem is infeasible, and in which case, should converge to a local minimizer or feasible point. [43] and [42] describe the restoration phase in further detail, as well as additional modifications to the primal-dual barrier method that characterize the IPOPT algorithm.

2.4.2 Generalized Reduced Gradient Approaches to NLP

The Generalized Reduced Gradient (GRG) method is an active set strategy, where the active inequality constraints are treated as equality constraints (as the constraint is active at its bounds), resulting in a reduced search space. However, the true active set is unknown prior to convergence, thus the postulated active set is updated as iterations progress, based on current information. Furthermore, the GRG method is an non-linear application of the Reduced Gradient method (used for linear constraints).

The overall idea of the GRG method is to take the nonlinear equality con-

straints and replace them by their linear Taylor approximation equivalent, then to apply the reduced gradient algorithm in the transformed problem. The following derivation is described in notes by Dr. Klerk at the University of Waterloo [51]. First, starting from the general NLP formulation:

$$\begin{aligned} & \min_x f(x) \\ \text{Subject to:} \quad & h_j(x) = 0, \quad j = 1, ..., m \\ & x \geq 0 \end{aligned}$$

the Jacobian of the equality constraints J_k is calculated for x_k at iteration k and then separated into basic and non-basic variables, $J_k(x_b), J_k(x_n)$. From there, the reduced gradient search direction for the linearized constraints is:

$$h(x_k) + J_k(x - x_k) = 0$$

where all values of $h(x_k)$ are the right hand side values of the constraints (i.e. 0). Then, when splitting the variables into basic and non-basic variables, we can rearrange the basic variables in order to eliminate them from the linearization of the NLP problem:

$$\begin{aligned} J_k(x_b)x_b + J_k(x_n)x_n &= J_k(x_k)x_k \\ \text{let} J_k(x_k)x_k &= c \quad \therefore \\ x_b(x_n) = \left[J_k(x_b)\right]^{-1}c - \Big(\left[J_k(x_b)\right]^{-1}J_k(x_n)\Big)x_n & \end{aligned}$$

The linearized problem, in terms of the non-basic variables, becomes:

$$\min_{x_n} F(x_n) = f(x_b, x_n) = f(\left[J_k(x_b)\right]^{-1}c - \Big(\left[J_k(x_b)\right]^{-1}J_k(x_n)\Big)x_n, x_n) \quad (2.48)$$

$$\text{Subject to:} \quad x_b \geq 0 \rightarrow \left[J_k(x_b)\right]^{-1}c - \Big(\left[J_k(x_b)\right]^{-1}J_k(x_n)\Big)x_n \geq 0 \quad (2.49)$$

$$x_n \geq 0 \quad (2.50)$$

The reduced gradient can then be written as:

$$\nabla F(x_n) = -\nabla_b f(x) \left[J_k(x_b)\right]^{-1} J_k(x_n) + \nabla_n f(x) \quad (2.51)$$

The following steps are common to that for the linear case. The non-basic variables are further split into superbasic variables x_{s1} and x_{s2}, where x_{s1} are between their bounds, and x_{s2} are at a bound. Then, the reduced gradient with respect to x_{s2} is used to determine if any of the superbasic variables at the bound, x_{s2}, should be released to join x_{s1}. The reduced gradient with respect to x_{s1} is used to form the search direction.

The search direction can be found either with the conjugate gradient methods or variable metric methods [52] [53] [54]. Conjugate gradient methods include algorithms by Abadie [55] or Fletcher and Reeves [56] and have the advantage in large problems as it only requires a few vectors and no matrices [57]. On the other hand, variable metric methods such as the Goldfarb algorithm [58] or by Davidon [59] are better equipped to handle the sparse properties of the Hessian [57].

An adaptation of the Goldfarb algorithm to find the search direction is presented by Lasdon et al. using the following:

$$d_i = -H_i \nabla F(x_i) \quad (2.52)$$

where d_i is the search direction, $\nabla F(x_i)$ is the gradient of the objective function, and H_i is a symmetric positive semi-definite matrix that projects any vector onto the bounds - in other words, for any vector v, $Hv = 0$ in the i^th position

for x_i that is at a bound. H_i is decided by an algorithm described by Lasdon et al. [52].

Once the search direction is computed, a one-dimensional search is started in order to solve the following optimization problem [60][52]:

$$\min_{\alpha>0} F(x_{nb} + \alpha d)$$

where α is iterated using positive values to find the best solution. At each iteration of α, $F(x_{nb} + \alpha d)$ is evaluated and is equivalent to:

$$f(x_b(x_{nb} + \alpha d), x_{nb} + \alpha d)$$

The basic variables x_b must satisfy the equality constraints:

$$h(x_b, x_{nb} + \alpha d) = 0$$

where x_b is to be found. However, if any of the basic variables appear in a nonlinear constraint, then it must be solved iteratively - a variant of Newton's method is often used. In this case of nonlinear constraints, the one-dimensional search may terminate through one of the following ways, defined by Lasdon et al. [52]. The first is failure of convergence of Newton's method; bound violation of a converged Newton Step (basic variables may violate bounds); or the search can continue until a worse solution is found. The algorithm will terminate if a value of α cannot be found.

In summary, the reduced gradient algorithm is as follows:

Algorithm 3 Generalized Reduced Gradient Method

Select a feasible solution x_0

for $k = 0, 1, 2, ...$ **do**

Compute Jacobian of the equality constraints $J_k(x_k) = \partial h(x_k)/\partial x_k$

Separate variables into basic (b) and non-basic (n) variables

Ensure $J_k(x_b) = \partial h(x_b)/\partial x_b^k$ is nonsingular

Calculate Kuhn-Tucker multiplier vector $u^T = \nabla_b f(x)^T \left[J_k(x_b)\right]^{-1}$

Compute reduced gradient as per equation 2.51

Check for tolerance (satisfies KKT conditions or other condition)

Separate the non-basic variables into superbasic variables x_{s1} between their bounds and x_{s2} at their bounds

Calculate the reduced gradient WRT to x_{s2} to determine if any superbasics at the bound should join x_{s1}

Calculate the reduced gradient WRT to x_{s1} to form the search direction

Calculate the search direction d

Solve the one-dimensional optimization subproblem along the search direction d with iterations of α

Save the solution if it improves on the current solution

end for

2.4.2.1 CONOPT

CONOPT is an optimization solver based on the GRG algorithm, and is designed to solve large scale problems including sparse problems. While the GRG algorithm helps with reliability and speed for solving highly nonlinear models, CONOPT is well suited for nonlinear problems where feasibility may be difficult to achieve [61].

When calculating the constraint Jacobian, sparse-matrix algorithms taken from linear programming techniques have been used and modified — when selecting the set of basic variables in the Jacobian and factorizing this sub-matrix, sparse LU factorization is used on the Jacobian, similar to work described by Suhl and Suhl [62]. Furthermore, CONOPT was designed on the assumption of a sparse problem in mind, and uses the Jacobian to identify the sparsity of the problem. Subsequently, this assumption allows CONOPT to deal with the sparse structure encountered in dynamic and multiperiod models [63].

CONOPT is well suited for models with relatively few degrees of freedom as it contains a fast method. The phase-0 and phase-1 algorithm outlined by Drud [63][64] can find a first feasible solution after a few iterations if a good initial starting point is given as only a few basis changes would be required. For problems with a large number of variables compared to constraints (i.e. a large number of degrees of freedom), CONOPT is able to use second derivatives to improve solution times over MINOS and SNOPT. IPOPT is also able to use second derivatives but both solvers calculate this information differently [65].

In addition to the base GRG algorithm, CONOPT contains several extensions such as a preprocessing step (reduction of model space), feasibility restoration, dynamic tolerance selection of the Newton method, bound handling, and finding a first feasible solution. As well, CONOPT assumes nonlinearity and applies to various parts of the algorithm, such as using the nonlinearities instead of bounds to search for the optimal step length. These extensions are described in detail in the CONOPT literature of Drud [63].

2.5 AMPL Description

The optimization software that was chosen for the project is AMPL (A Mathematical Programming Language). AMPL is an optimization modelling language used to represent problems at a high-level and is written similar to the mathematical representation of the problem (using variables, parameters, sets, and constraints) [66]. The structure of the language is similar to that of other high-level programming languages, where equations and sets can be entered directly. By working in the AMPL environment, the model becomes more flexible as it is coded in general terms and thus, easier to add or remove unit operations or change plant configuration. It becomes much easier to implement case studies, as it involves only changing a few constraints or parameter values. Furthermore, AMPL has the built-in ability to read and write data tables from Microsoft Excel workbooks which is beneficial to the TOTAL model, as many of the model parameters are stored in spreadsheet applications. As well, AMPL can write to a spreadsheet file and thus, the results can be displayed in a format that the user desires [67].

In the AMPL environment, there are many available solvers for use. In particular, two highly regarded nonlinear solvers, CONOPT and IPOPT, are available for use in the AMPL environment. Along with the effectiveness of these solvers, AMPL itself includes an effective preprocessing phase which reduces and simplifies the optimization problem before passing onto the solver. With the combination of the solvers and preprocessing, AMPL is able to solve comparable sized Excel problems significantly faster. With the ability to solve problems faster, there is an opportunity to solve larger problems. The solver options can also be changed, such as warm start capabilities for IPOPT, convergence tolerance limits, and solve phases.

Another benefit to the AMPL environment is the ability to implement solutions and solution strategies. Within the structure of the language there is the capability to use loops, and thus, decomposition strategies such as Lagrangean Decomposition may be used, where two problems are iterated back and forth. Additionally, there is the capability of running consecutive optimization problems, where a parameter may be varied.

2.6 Lagrangean Decomposition

Lagrangean decomposition is a special case of Lagrangean relaxation, where a problem may have several sets of constraints such that a relaxed problem containing a subset of these constraints is easier to solve than the full problem. Thus, the overall problem can be broken down into subproblems and solved iteratively. The main driver of the method is to make copies of the variables that are common between each constraint set (i.e. variables that appear in multiple constraints) and equality constraints are added to equate the original variables to its copies. These equality constraints are then dualized and the overall model is decomposed into subproblems for each constraint set and solved individually [68] [69]. Lagrangean decomposition is suitable for mixed-integer problems, as well as nonlinear problems that have linking constraints, such as inventory levels.

An example is presented by van den Heever et al. [69] to illustrate the transformation of a problem using Lagrangean decomposition. A mixed-integer optimization problem is shown as follows, with y^1 and y^2 as binary variables:

$$\max Z = c^T x + d^T (y^1 + y^2)$$

$$s.t. \quad A^1 x + B^1 y^1 \leq b^1$$

$$
\begin{aligned}
A^2x + B^2y^2 &\leq b^2 \\
h_1(x) &\leq 0 \\
h_2(x) &\leq 0 \\
x \geq 0,\, y^1,\, y^2 &\in \{0,1\}
\end{aligned}
$$

From this problem, x is the common variable between all constraints. In order to apply Lagrangean decomposition, x is duplicated as variable z:

$$
\begin{aligned}
\max Z = c^T x + d^T(y^1 + y^2) & \\
s.t. \quad A^1x + B^1y^1 &\leq b^1 \\
A^2z + B^2y^2 &\leq b^2 \\
h_1(x) &\leq 0 \\
h_2(z) &\leq 0 \\
x &= z \\
x,\, z \geq 0,\, y^1,\, y^2 &\in \{0,1\}
\end{aligned}
$$

The equality constraint $x = z$ is then dualized in order to relax the problem and to separate the problem into two subproblems:

$$
\begin{aligned}
\max Z^{LD} = c^T x + d^T(y^1 + y^2) + \lambda^T(z - x) & \\
s.t. \quad A^1x + B^1y^1 &\leq b^1 \\
A^2z + B^2y^2 &\leq b^2 \\
h_1(x) &\leq 0 \\
h_2(z) &\leq 0 \\
x,\, z \geq 0,\, y^1,\, y^2 &\in \{0,1\}
\end{aligned}
$$

Finally, the problem is decomposed into two subproblems (P1 and P2):

$$\begin{aligned}
\text{(P1):} \quad \max Z^{P1} &= c^T x + d^T(y^1) - \lambda^T x \\
s.t. \quad A^1 x + B^1 y^1 &\leq b^1 \\
h_1(x) &\leq 0 \\
x \geq 0,\ y^1 &\in \{0,1\} \\
\text{(P2):} \quad \max Z^{P2} &= d^T(y^2) + \lambda^T z \\
s.t. \quad A^2 z + B^2 y^2 &\leq b^2 \\
h_2(z) &\leq 0 \\
z \geq 0,\ y^2 &\in \{0,1\}
\end{aligned}$$

Since the problem is now relaxed, assuming convex constraints, the objective function of P1 + P2 should be an upper bound to the original problem [70]. In theory, solving the following minimization problem (where Z is the objective function) should produce the tightest upper bound:

$$\min_{\lambda} Z^{P1} + Z^{P2} \tag{2.53}$$

However, solving 2.53 can be difficult and time-consuming, and although there are algorithms that can solve this problem [71], it is not available for use. Heuristics are used instead to solve 2.53 and to solve the overall problem. Fisher suggests using an iterative subgradient method to find λ as it is proven to work well and is easy to implement [72]:

$$\lambda^{k+1} = \lambda^k + t^k(z^k - x^k) \tag{2.54}$$

where t^k is a scalar step size and z^k and x^k are the solutions to the Lagrangean

problem at λ^k. t^k ideally should converge to zero and can be calculated using the following from Fisher [73]:

$$t^k = \frac{\alpha_k\left(\left[Z^{P1}(\lambda^k) + Z^{P2}(\lambda^k)\right] - Z^*\right)}{\left\|z^k - x^k\right\|^2} \tag{2.55}$$

where $(Z^{P1}(\lambda^k) + Z^{P2}(\lambda^k))$ are the combined objective values for P1 and P2 for a λ_k, Z^* is the best current objective value for the overall problem, and $\alpha_k \in \{0, 2\}$ which is halved each time $(Z^{P1}(\lambda^k) + Z^{P2}(\lambda^k))$ does not improve within a set number of iterations. Held et al. [74] have shown the theoretical convergence properties of the subgradient method, which is support for utilizing the relationship for the step update. λ can be initialized at a value of 0, as described by Escobar et al. [75]. Figure 2.22 shows the general algorithm for using the Lagrangean decomposition method, adapted from [69] [76] [77] [78].

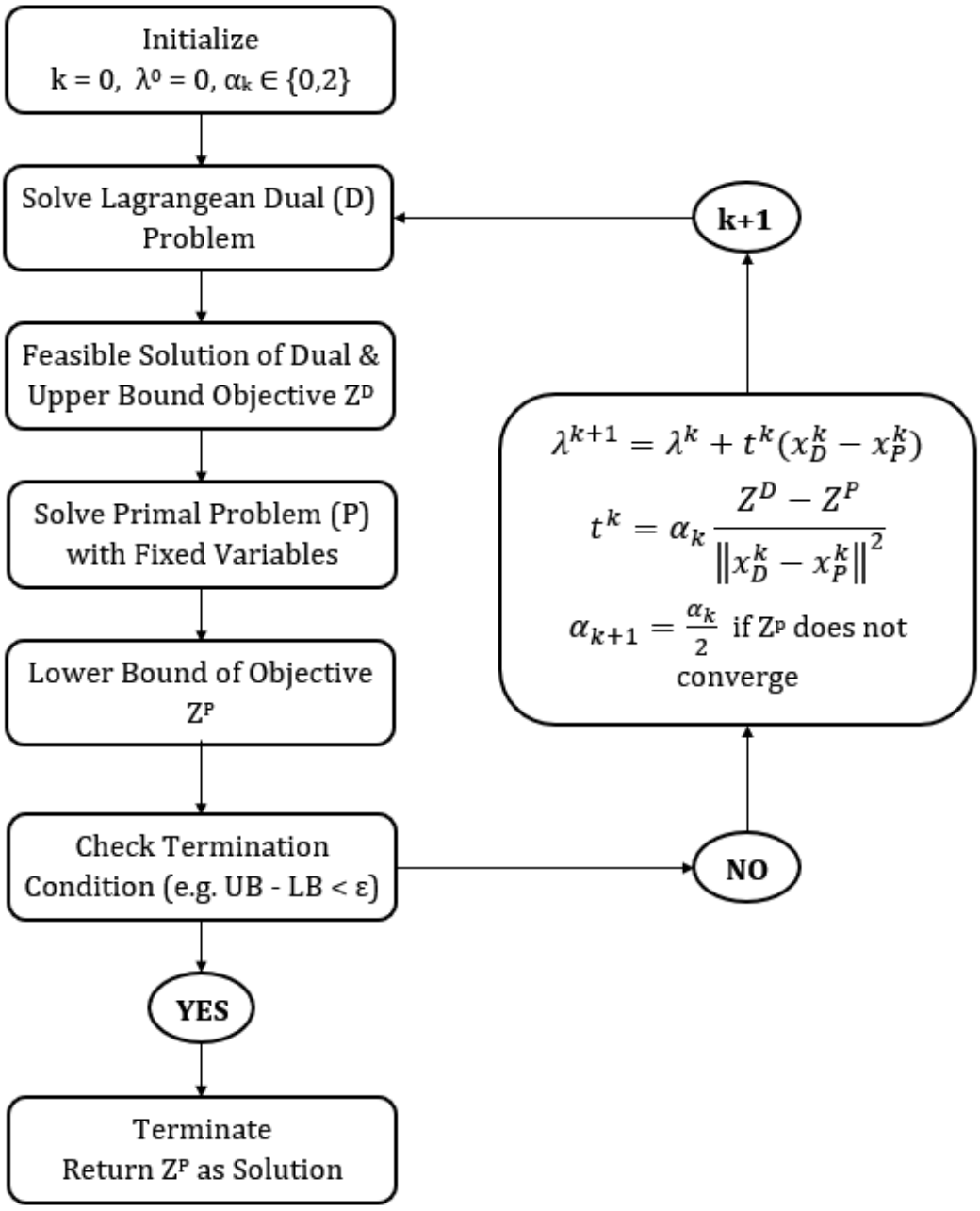

Figure 2.22: Lagrangean decomposition scheme adapted from [69] [76] [77] [78].

Several applications of Lagrangean decomposition are published in mixed integer, multiperiod, and/or stochastic optimization problems. In the case of mixed integer programming problems, one example includes a study by Karuppiah and Grossmann [79] who study the mixed integer nonlinear optimization of a integrated water network under uncertainty. In this study,

the objective is to minimize annual capital costs and operating costs that are incurred during each scenario in an integrated water network. The network includes process units (e.g. reactors and washers), treatment units (e.g. membranes), mixers, and splitters, which all operate under uncertain operating conditions (contaminant loads and removal). A spatial approach is taken (i.e. where the subproblems are each uncertainty scenario) where the constraints that link each scenario with the design variables state that the pipe flow rate in each scenario must be lower than the maximum allowable flow rate. Tied to the flow rate are binary variables that are linked to the bounds of the flow rate, where the value of the binary variable is 0 if there is no flow through the pipes. Thus, these binary variables are also duplicated and dualized. The Lagrangean decomposition was combined with a branch and cut algorithm and convex relaxations, and resulted in a reduction of more than an order of magnitude in solution time [79].

In the study of multiperiod problems, Neiro and Pinto [77] have proposed two formulations of the Lagrangean decomposition applied to a multiperiod petroleum refinery planning NLP under uncertainty. In this study, a real-world production planning model from Petrobras is considered, where the model incorporates the selection and processing of petroleum crude oils, unit operation, and the inventory management and revenue from final products over a planning horizon of 10 periods. The uncertainty in this problem stems from the probability of 2 possible crude slates - the base case considers a 100% probability of crude slate 1, whereas the second case study provides 40% and 60% as the probability of crude slate 1 and 2, respectively, and the third case study swaps the probability to 60% and 40% to slates 1 and 2, respectively. Thus, in this problem there are two types of linking constraints which give rise to two types of decomposition - spatial and temporal decomposition. The

temporal linking constraints (and subsequently, the temporal decomposition problem) correspond to the product inventory constraint:

$$Vol_{u,t,c} = Vol_{u,t-1,c} + QF_{u,t,c} - Dem_{u,t,c}$$

where u, t, and c are the unit, time period, and scenario respectively; Vol, QF, and Dem are the inventory volume, inlet flow rate, and demand (outlet) for the tank, respectively. The spatial linking constraint binds the petroleum supply tanks:

$$y_{u,t,c}QS_u^L \leq QS_{u,s,t,c} \leq y_{u,t,c}QS_u^U$$

where s is the stream; y is the binary variable that corresponds to the selected crude type, and QS is the outlet flow rate of the crude tanks. From these two types of linking constraints, there are two Lagrangean decomposition strategies: dualizing the inventory variables (temporal) or dualizing the binary variables (spatial). In the temporal case, the inventory tank inventories are the connection between time periods and are severed to create independent problems per time period. In the spatial case, the scenarios correspond to changing flow rate bounds in each scenario using the binary variable y. Thus, by severing y, each scenario becomes its own independent subproblem. The results show that both strategies exhibit lower solution times by an order of magnitude, and is exaggerated when the number of incorporated scenarios increases (e.g. there is a large improvement for 3 or 5 scenarios versus only 1 scenario). Between the two strategies, the dualization of the inventory constraints (temporal decomposition) consistently shows lower solution times [77].

Another study in multiperiod optimization using Lagrangean decomposition is published by Jackson and Grossmann [76]. In this study, Lagrangean de-

composition is applied to a multiperiod nonlinear production planning and distribution model. In this model, several production plants located in different areas produce products that are distributed to various global markets, where each plant produces various products. From this problem, there are two methods of Lagrangean decomposition that are applied - spatial and temporal decomposition. In the spatial case, ties are severed between the product sites (plants) and markets, as shown in figure 2.23.

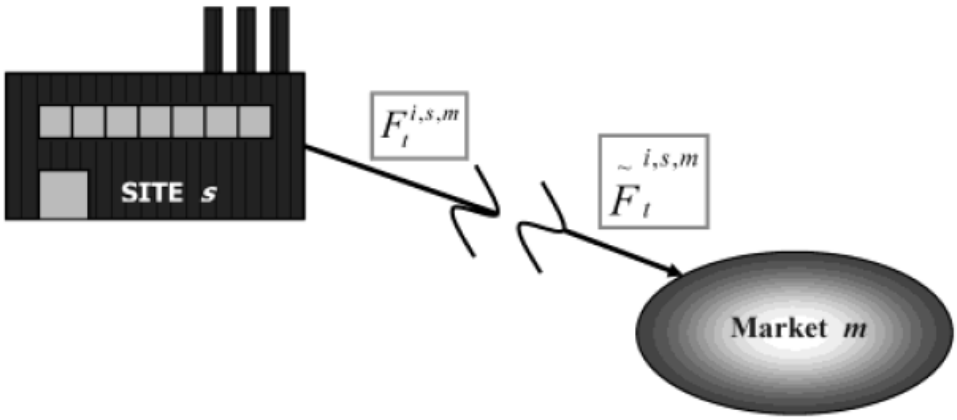

Figure 2.23: Spatial decomposition of the multisite multiproduct production planning model by Jackson and Grossmann. In the spatial case, the interconnection variables that link the sites to the markets (flow rates that connect sites and markets) are severed [76].

For the spatial case, the flows of product $i \in PR$ from the sites $s \in SITES$ to market $m \in MAR$ in time period $t \in T$ are severed. These flows are separated into a flow *from* the site, and a flow *to* the market. Thus, the problem is separated into two subproblems - one that only concerns itself with production from the sites $s \in SITES$, and the other that concerns itself with the markets $m \in MAR$.

In the temporal case, the ties between time periods are severed instead of the markets/sites, and are illustrated in figure 2.24.

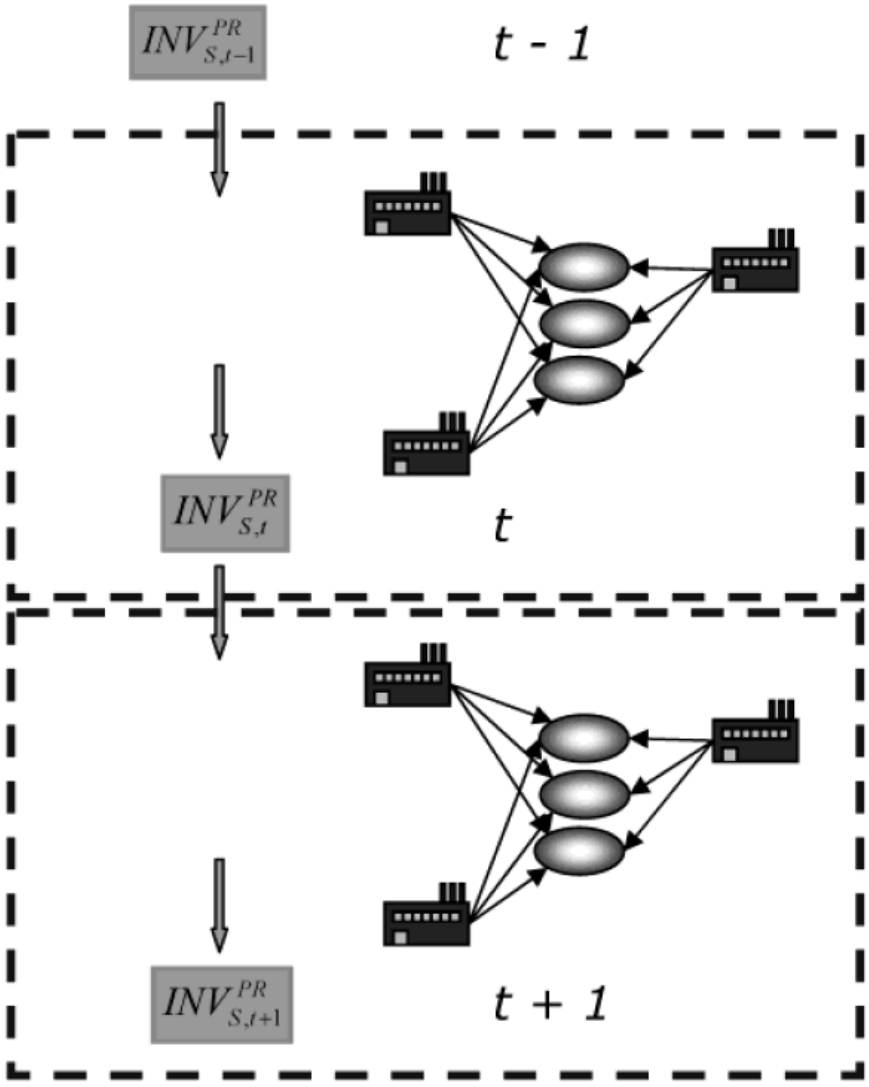

Figure 2.24: Temporal decomposition of the multisite multiproduct production planning model by Jackson and Grossmann. In the temporal case, the interconnection variables that link the time periods (inventories between periods) are severed [76].

For the temporal case, the inventories are disconnected between time periods t and $t-1$, and the problem is formulated similarly to the study by Neiro and Pinto [77].

The study found that the temporal decomposition method was more robust and effective than the spatial decomposition - i.e. the converged solution was closer than in the spatial case. Depending on the size and type of the problem, either method was faster than the other. In the first example of a

linear network, the temporal formulation was faster and provided a closer solution. However, both solutions were slower than the simultaneous solution.

In the second example of a nonlinear industrial network, the spatial formulation was faster for a problem with 3 time periods but temporal decomposition was faster for the 6 and 12 time period problems and all temporal solutions were closer to the true optimum than the spatial solutions. In the third example of the large nonlinear industrial network, spatial decomposition was faster for all problem sizes, but the temporal solution was significantly closer to the solution than the spatial case. For both examples two and three, the decomposition methods were significantly faster than the simultaneous solution.

It is also noted that for all examples, there are difficulties in finding feasible solutions in the spatial case and thus, some constraints must be relaxed. In the temporal solution, feasible solutions were found without relaxation.

References

[1] M. Fahim, T. Alsahhaf, and A. Elkilani. *Fundamentals of Petroleum Refining*. Elsevier Science, Nov. 2009, p. 516.

[2] N. K. Shah, Z. Li, and M. G. Ierapetritou. "Petroleum refining operations: key issues, advances, and opportunities". In: *Industrial & Engineering Chemistry Research* 50.3 (2010), pp. 1161–1170.

[3] C. Hsu and P. Robinson. *Practical Advances in Petroleum Processing*. Vol. 1. Springer, Jan. 2006.

[4] J. H. Gary, G. E. Handwerk, and M. J. Kaiser. *Petroleum Refining: Technology and Economics*. Ed. by C. Press. 5th ed. CRC Press, Mar. 2007. ISBN: 9780849370380.

[5] J. P. Abella and J. A. Bergerson. "Model to investigate energy and greenhouse gas emissions implications of refining petroleum: Impacts of crude quality and refinery configuration". In: *Environmental science & technology* 46.24 (2012), pp. 13037–13047.

[6] N. Wu, M. Zhou, and F. Chu. "Short-term scheduling for refinery process: Bridging the gap between theory and applications". In: *International Journal of Intelligent Control and Systems* 10.2 (2005), pp. 162–174.

[7] C. Khor and A. Elkamel. "Superstructure optimization for oil refinery design". In: *Petroleum Science and Technology* 28.14 (2010), pp. 1457–1465.

[8] J. W. Seo, M. Oh, and T. H. Lee. "Design optimization of a crude oil distillation process". In: *Chemical Engineering & Technology: Industrial Chemistry-Plant Equipment-Process Engineering-Biotechnology* 23.2 (2000), pp. 157–164.

[9] A. M. Alattas, I. E. Grossmann, and I. Palou-Rivera. "Integration of nonlinear crude distillation unit models in refinery planning optimization". In: *Industrial & Engineering Chemistry Research* 50.11 (2011), pp. 6860–6870.

[10] R. Jakob. "Estimates Number of Crude Trays". In: *Hydrocarbon Process* 50 (1971), pp. 149–152.

[11] S. Ji and M. Bagajewicz. "Design of crude distillation plants with vacuum units. I. Targeting". In: *Industrial & engineering chemistry research* 41.24 (2002), pp. 6094–6099.

[12] J. Speight. "Visbreaking: A technology of the past and the future". In: *Scientia Iranica* 19.3 (2012), pp. 569–573. ISSN: 1026-3098. DOI: https://doi.org/10.1016/j.scient.2011.12.014. URL: http://www.sciencedirect.com/science/article/pii/S1026309812000612.

[13] A. N. Sawarkar, A. B. Pandit, S. D. Samant, and J. B. Joshi. "Petroleum residue upgrading via delayed coking: A review". In: *The Canadian Journal of Chemical Engineering* 85.1 (2007), pp. 1–24.

[14] J. M. Matsen. "Scale-up of fluidized bed processes: principle and practice". In: *Powder Technology* 88.3 (1996), pp. 237–244.

[15] I.-S. Han, C.-B. Chung, and J. B. Riggs. "Modeling of a fluidized catalytic cracking process". In: *Computers & Chemical Engineering* 24.2-7 (2000), pp. 1681–1687.

[16] R. Sadeghbeigi. *Fluid Catalytic Cracking Handbook: An Expert Guide to the Practical Operation, Design, and Optimization of FCC Units*. 3rd ed. Elsevier, Jan. 2012. ISBN: 9780123869654.

[17] L. F. Albright. "Present and future alkylation processes in refineries". In: *Industrial & Engineering Chemistry Research* 48.3 (2009), pp. 1409–1413.

[18] L. F. Albright and D. J. Am Ende. *Encyclopedia of Catalysis: Alkylation Homogeneous*. Vol. 1. John Wiley and Sons, 2003, pp. 191–210.

[19] S. I. Hommeltoft. "Isobutane alkylation: Recent developments and future perspectives". In: *Applied Catalysis A: General* 221.1-2 (2001), pp. 421–428.

[20] L. F. Albright and D. J. Am Ende. *Encyclopedia of Catalysis: Alkylation Industrial*. Vol. 1. John Wiley and Sons, 2003, pp. 226–281.

[21] J. W. Ward. "Hydrocracking processes and catalysts". In: *Fuel Processing Technology* 35.1-2 (1993), pp. 55–85.

[22] J. Scott and A. Bridge. "The continuing development of hydrocracking". In: *Origin and Refining of Petroleum*. ACS Publications, 1971. Chap. 6, pp. 113–129.

[23] M. R. Rahimpour, M. Jafari, and D. Iranshahi. "Progress in catalytic naphtha reforming process: A review". In: *Applied energy* 109 (2013), pp. 79–93.

[24] U. Taskar and J. B. Riggs. "Modeling and optimization of a semiregenerative catalytic naphtha reformer". In: *AIChE Journal* 43.3 (1997), pp. 740–753.

[25] M. Z. Stijepovic, P. Linke, and M. Kijevcanin. "Optimization approach for continuous catalytic regenerative reformer processes". In: *Energy & Fuels* 24.3 (2010), pp. 1908–1916.

[26] S. Parkash. *Refining Processes Handbook*. Ed. by Elsevier. Gulf Professional Publishing, 2003.

[27] A. Pongsakdi, P. Rangsunvigit, K. Siemanond, and M. J. Bagajewicz. "Financial risk management in the planning of refinery operations". In: *International Journal of Production Economics* 103.1 (2006), pp. 64–86.

[28] C. A. Mendez, I. E. Grossmann, I. Harjunkoski, and P. Kabore. "A simultaneous optimization approach for off-line blending and scheduling of oil-refinery operations". In: *Computers & chemical engineering* 30.4 (2006), pp. 614–634.

[29] J. Pinto, M. Joly, and L. Moro. "Planning and scheduling models for refinery operations". In: *Computers & Chemical Engineering* 24.9-10 (2000), pp. 2259–2276.

[30] W. Li, C.-W. Hui, and A. Li. "Integrating CDU, FCC and product blending models into refinery planning". In: *Computers & chemical engineering* 29.9 (2005), pp. 2010–2028.

[31] A. Elkamel, M. Ba-Shammakh, P. Douglas, and E. Croiset. "An optimization approach for integrating planning and CO2 emission reduction in the petroleum refining industry". In: *Industrial & Engineering Chemistry Research* 47.3 (2008), pp. 760–776.

[32] S. M. Neiro and J. M. Pinto. "Multiperiod optimization for production planning of petroleum refineries". In: *Chem. Eng. Comm.* 192.1 (2005), pp. 62–88.

[33] T. M. Alkhamis and J. Yellen. "Refinery units maintenance scheduling using integer programming". In: *Applied Mathematical Modelling* 19.9 (1995), pp. 543–549.

[34] D. K. Varvarezos, L. T. Biegler, and I. E. Grossmann. "Multiperiod design optimization with SQP decomposition". In: *Computers & chemical engineering* 18.7 (1994), pp. 579–595.

[35] H. Zhao, G. Rong, and Y. Feng. "Multiperiod planning model for integrated optimization of a refinery production and utility system". In: *Industrial & Engineering Chemistry Research* 53.41 (2014), pp. 16107–16122.

[36] B. Zhang and B. Hua. "Effective MILP model for oil refinery-wide production planning and better energy utilization". In: *Journal of Cleaner Production* 15.5 (2007), pp. 439–448.

[37] A. Leiras, S. Hamacher, and A. Elkamel. "Petroleum refinery operational planning using robust optimization". In: *Engineering Optimization* 42.12 (2010), pp. 1119–1131.

[38] D. Allen. "Linear programming models for plant operations planning". In: *British Chemical Engineering* 16.8 (1971), p. 685.

[39] A. M. Alattas, I. E. Grossmann, and I. Palou-Rivera. "Refinery production planning: multiperiod MINLP with nonlinear CDU model". In: *Industrial & Engineering Chemistry Research* 51.39 (2012), pp. 12852–12861.

[40] M. Avriel. *Nonlinear programming: analysis and methods*. Courier Corporation, 2003.

[41] J. B. Orlin. *Optimization Methods in Business Analytics: Nonlinear Programming*. Lecture. 2018. URL: http://web.mit.edu/15.053/www/AMP-Chapter-13.pdf.

[42] A. Wachter and L. T. Biegler. "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming". In: *Mathematical programming* 106.1 (2006), pp. 25–57.

[43] A. Wachter. "An Interior Point Algorithm for Large-Scale Nonlinear Optimization with Applications in Process Engineering". PhD thesis. Carnegie Mellon University, Jan. 2002. URL: http://researcher.watson.ibm.com/researcher/files/us-andreasw/thesis.pdf.

[44] F. A. Potra and S. J. Wright. "Interior-point methods". In: *Journal of Computational and Applied Mathematics* 124.1-2 (2000), pp. 281–302.

[45] S. J. Wright. *Primal-Dual Interior-Point Methods*. Ed. by S. for Industrial and A. Mathematics. Vol. 1. Society for Industrial and Applied Mathematics, 1997.

[46] R. J. Vanderbei and D. F. Shanno. "An interior-point algorithm for nonconvex nonlinear programming". In: *Computational Optimization and Applications* 13.1-3 (1999), pp. 231–252.

[47] S. Wright and J. Nocedal. "Numerical optimization". In: *Springer Science* 35.67-68 (1999), p. 7.

[48] F. Curtis and J. Nocedal. "Steplength selection in interior-point methods for quadratic programming". In: *Applied Mathematics Letters* 20.5 (2007), pp. 516–523.

[49] R. H. Byrd, G. Liu, and J. Nocedal. "On the local behavior of an interior point method for nonlinear programming". In: *Numerical analysis* 1997 (1997), pp. 37–56.

[50] A. Wachter and L. T. Biegler. "Line search filter methods for nonlinear programming: Motivation and global convergence". In: *SIAM Journal on Optimization* 16.1 (2005), pp. 1–31.

[51] E. d. Klerk, C. Roos, and T. Terlaky. *Nonlinear Optimization (CO 367) Notes*. University of Waterloo, Lecture Notes. Aug. 2004. URL: `http://www.cas.mcmaster.ca/~deza/nonlinopt.pdf`.

[52] L. S. Lasdon, R. L. Fox, and M. W. Ratner. "Nonlinear optimization using the generalized reduced gradient method". In: *Revue francaise d'automatique, informatique, recherche operationnelle. Recherche operationnelle* 8.V3 (1974), pp. 73–103.

[53] L. S. Lasdon and A. Waren. "Large scale nonlinear programming". In: *Computers & Chemical Engineering* 7.5 (1983), pp. 595–604.

[54] G. Gabriele and K. Ragsdell. "The generalized reduced gradient method: A reliable tool for optimal design". In: *Journal of Engineering for Industry* 99.2 (1977), pp. 394–400.

[55] J. Abadie. *Application of the GRG algorithm to optimal control problems*. North-Holland Publishing Company, 1970.

[56] R. Fletcher and C. M. Reeves. "Function minimization by conjugate gradients". In: *The computer journal* 7.2 (1964), pp. 149–154.

[57] G. A. Gabriele and K. Ragsdell. "Large scale nonlinear programming using the generalized reduced gradient method". In: *Journal of Mechanical Design* 102.3 (1980), pp. 566–573.

[58] D. Goldfarb. "Extension of Davidons variable metric method to maximization under linear inequality and equality constraints". In: *SIAM Journal on Applied Mathematics* 17.4 (1969), pp. 739–764.

[59] W. C. Davidon. "Variable metric method for minimization". In: *SIAM Journal on Optimization* 1.1 (1991), pp. 1–17.

[60] R. B. S. Paul T. Boggs Richard H. Byrd. *Numerical Optimization 1984: Proceedings of the SIAM Conference on Numerical Optimization*. Ed. by SIAM. Boulder, Colorado4: SIAM, June 1984.

[61] A. Drud. *The CONOPT Algorithm*. Website. URL: `http://www.conopt.com/Algorithm.htm`.

[62] U. H. Suhl and L. M. Suhl. "Computing sparse LU factorizations for large-scale linear programming bases". In: *ORSA Journal on Computing* 2.4 (1990), pp. 325–335.

[63] A. Drud. "CONOPT: A GRG code for large sparse dynamic nonlinear optimization problems". In: *Mathematical Programming* 31.2 (1985), pp. 153–191.

[64] A. Drud. *CONOPT Technical Manual*. ARKI Consulting and Development A/S. Bagsvaerd, Denmark. URL: https://www.pik-potsdam.de/research/sustainable-solutions/models/remind/conopt.pdf.

[65] A. Drud. *GAMS Documentation - CONOPT*. 25.1.1. ARKI Consulting and Development AS. Bagsvaerd, Denmark, May 2018. URL: https://www.gams.com/latest/docs/S_CONOPT.html.

[66] W. Wells, R. Fourer, D. Gay, and B. Kernighan. *AMPL TABLE HANDLERS*. May 2013. URL: https://ampl.com.

[67] W. Wells, R. Fourer, D. Gay, and B. Kernighan. *AMPL*. May 2018. URL: https://ampl.com/NEW/TABLES/.

[68] M. Guignard and S. Kim. "Lagrangean decomposition: A model yielding stronger Lagrangean bounds". In: *Mathematical programming* 39.2 (1987), pp. 215–228.

[69] S. A. van den Heever, I. E. Grossmann, S. Vasantharajan, and K. Edwards. "A Lagrangean decomposition heuristic for the design and planning of offshore hydrocarbon field infrastructures with complex economic objectives". In: *Industrial & engineering chemistry research* 40.13 (2001), pp. 2857–2875.

[70] M. L. Fisher. "The Lagrangian relaxation method for solving integer programming problems". In: *Management science* 27.1 (1981), pp. 1–18.

[71] K. Kiwiel. "User's Guide for NOA 2.0/3.0: A Fortran package for convex nondifferentiable optimization". In: *Systems Research Institute, Polish Academy of Sciences, Warsaw* (1994).

[72] M. L. Fisher. "The Lagrangian relaxation method for solving integer programming problems". In: *Management science* 50.12_supplement (2004), pp. 1861–1871.

[73] M. L. Fisher. "An applications oriented guide to Lagrangian relaxation". In: *Interfaces* 15.2 (1985), pp. 10–21.

[74] M. Held, P. Wolfe, and H. P. Crowder. "Validation of subgradient optimization". In: *Mathematical programming* 6.1 (1974), pp. 62–88.

[75] M. Escobar, J. O. Trierweiler, and I. E. Grossmann. "A heuristic Lagrangean approach for the synthesis of multiperiod heat exchanger networks". In: *Applied Thermal Engineering* 63.1 (2014), pp. 177–191.

[76] J. R. Jackson and I. E. Grossmann. "Temporal decomposition scheme for nonlinear multisite production planning and distribution models". In: *Industrial & engineering chemistry research* 42.13 (2003), pp. 3045–3055.

[77] S. M. Neiro and J. M. Pinto. "Langrangean decomposition applied to multiperiod planning of petroleum refineries under uncertainty". In: *Latin American applied research* 36.4 (2006), pp. 213–220.

[78] F. Oliveira, V. Gupta, S. Hamacher, and I. E. Grossmann. "A Lagrangean decomposition approach for oil supply chain investment planning under uncertainty with risk considerations". In: *Computers & Chemical Engineering* 50 (2013), pp. 184–195.

[79] R. Karuppiah and I. E. Grossmann. "Global optimization of multiscenario mixed integer nonlinear programming models arising in the synthesis of integrated water networks under uncertainty". In: *Computers & Chemical Engineering* 32.1-2 (2008), pp. 145–160.

Chapter 3

Multiperiod Refinery Optimization: Model Formulation

3.1 Model Building Process and Procedure

The petroleum refinery optimization model was developed in stages. The first stage of development involved formulation and implementation of the single period refinery optimization model, which is also divided in sub-stages. Once the single period optimization was successful and various verification steps were taken to ensure correct model behaviour, the multiperiod optimization was formulated and implemented.

When developing the single period optimization model, the TOTAL refinery optimization model was translated into mathematical form, including indices, sets, mass and yield equations, property relationships, and constraints. Once this step was completed, a process flowsheet was created in order to visualize the interconnectivity of the unit operations, as well as track all blending operations and pathways of intermediate products. The model was then programmed into sub-stages.

The first sub-stage involved setting up the model parameter tables that are internally generated by TOTAL through various simulations of unit operations. AMPL features connectivity between its interface and spreadsheet applications in order to link parameter tables.

The next step involves programming the refinery in sections. First, the mass balance and yield relationships are programmed and includes all unit operations, crude inputs to the plant, intermediate pools, and split points, blenders, and final products. Within this first step, compositions and densities are implemented, as some product streams require the molecular composition to calculate the yield, and some streams require conversion between volume and mass. Next, the unit operating conditions and critical properties are calcu-

lated. Some of the operating conditions include reaction temperatures, and catalyst properties. Since some of the operating conditions require properties such as sulfur, octane number, and Reid vapour pressure, these properties are implemented. Next, the remainder of the properties are calculated such as aromatic content, benzene, flash point, and cloud point. Then, the blending relationships are implemented into the model and include blending properties. Finally, the model specific constraints such as final product specifications, reaction temperatures, etc. are implemented.

Once the single period model is completed and verified, the multiperiod model is developed. The progression of creating the multiperiod model is illustrated in Figure 3.1. First, the single period model is used as the starting point, then is duplicated to create the basis of the multiperiod problem. The periods are then linked together using linking variables and constraints (in this case, tanks are incorporated to link the periods together) and the crude slate and imports are partitioned appropriately between the periods.

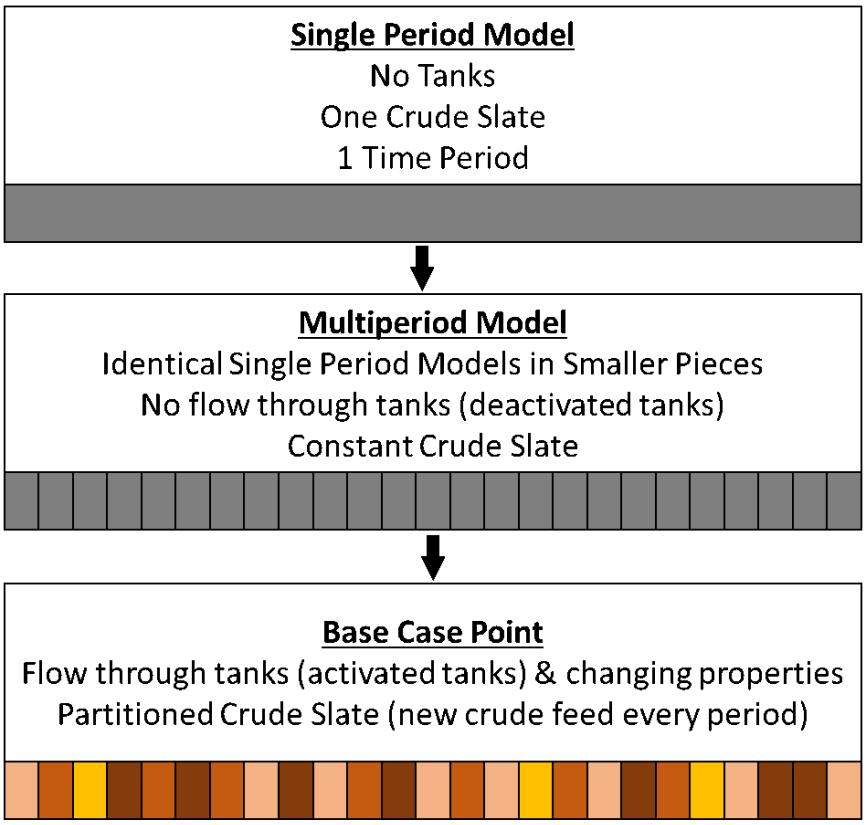

Figure 3.1: Development model for the petroleum refinery model. A single period model is used as a initial point, and is then partitioned into multiple periods. In this first iteration of the multiperiod model, all tanks are fixed with zero flow through, and the crude slate and imports are divided equally across all periods. From this point, the model is solved and the solution to this problem is the initial guess for the nominal operation problem. In the nominal operation (base case) problem, flow is allowed through the tanks and the crude slate and imports are partitioned as defined by the user.

To begin this multiperiod model building procedure, the overall time horizon and the number of periods are decided by the user - for example, the time horizon was selected to be 30 days divided in 5 periods, therefore 6 days per period. The single period model is then modified to represent the overall time horizon (in this case, 30 days) - this involves changing the operating hours of each unit and modifying the crude slate and imports for 30 days. Note that the single period can take on different lengths of time (e.g. the single period model can span 30 days, 365 days, 200 days, etc.), as long as the crude slate, imports, and operating time of the units are adjusted.

Once the 30 day single period model is successfully solved, a time index was added to all constraints and variables and the number of time periods is changed to the desired partition (5 periods in this case) and the number of days per period is modified as well (modified from 30 days to 6 days). The tanks and their locations are then decided and implemented as units in the model. The tanks include property mixing, inventory levels, and mass balances. However, once the tanks are implemented, zero flow is fixed through the tanks in order to keep the periods disconnected and the flow is then bypassed around the tank, directly to the unit.

Next, the crude slate and imports should be adjusted to represent the new model where the crudes and imports used in the single period model are divided between the periods (e.g. if 100 units of import A are used in the 30 day single period model, for a 5 period model, each period will use 20 units of import A). Thus, this new 5 period, 6 day representation is the 30 day single period model divided into 5 periods, where each period is identical. This now becomes the basis of the multiperiod model. Once this model is completed and solved, the final stage of the model building can be implemented.

The final stage of the implementation is to allow flow through the intermediate product tanks and to change the crude slates and imports to represent the desired inputs to the plant per period (e.g. imports may only be fed to the refinery every odd period and the crude slate may be different for each period). The solution to this final step becomes the initial starting point for any case studies, and concludes the multiperiod refinery optimization model building procedure.

The single period model totals over 4000 variables and constraints, and as the number of periods increases, so does the number of variables and constraints.

Table 3.1 summarizes the number of variables and constraints before pre-processing for a range of periods. The problem essentially scales linearly with number of time periods, with differences due to the interconnection variables and constraints. However, it should be noted that the solution time does not scale linearly along with the problem size, and is discussed in the next chapter.

Table 3.1: Petroleum refinery model size for different number of periods. The number of variables and constraints are listed for various model sizes from 1 to 30 periods.

No. of Periods	Variables	Constraints
1	4,230	4,447
3	12,807	13,280
5	21,345	22,132
10	42,690	44,262
30	128,070	132,782

3.2 Mathematical Formulation

The multiperiod refinery optimization model is formulated as follows:

$$\max PROFIT = \sum_{t \in T} \Bigg[\sum_{u \in U_b} \sum_{j \in J_u} c_j F_{j,u,t} - \sum_{u \in U_d} \sum_{i \in I_u} c_i F_{i,u,t} - \sum_{u \in U_{NG}} c_{NG} F_{NG,u,t} \Bigg] \tag{3.1}$$

$$\text{subject to:} \quad \sum_{i \in I_u} F_{i,u,t} = \sum_{j \in J_u} F_{j,u,t} \quad \forall\, u \in U, t \in T \tag{3.2}$$

$$F_{j,u,t} = f(F_{i,u,t}, \theta_{j,u}, CP_{j,u,t}, Z_{z,u,t}) \quad \forall\, i \in I_u, j \in J_u, u \in U, t \in T, z \in Z_u \tag{3.3}$$

$$F_{i,u,t} = f(V_{i,u,t}, \rho_{i,u,t}, H_u) \quad \forall\, i \in I_u, u \in U, t \in T \tag{3.4}$$

$$P_{p,j,u,t} = f(F_{i,u,t}, V_{i,u,t}, \rho_{i,u,t}, \theta_{p,i,u}, CP_{j,u,t}, X_{x,i,u,t}) \quad \forall\, p \in P_u, i \in I_u, j \in J_u, u \in U, x \in X, t \in T \tag{3.5}$$

$$P^{LB}_{p,j,u,t} \leq P_{p,j,u,t} \leq P^{UB}_{p,j,u,t} \quad \forall\, p \in P_u, j \in J_u, u \in U, t \in T \tag{3.6}$$

$$Z_{z,u,t} = f(F_{i,u,t}, V_{i,u,t}, P_{p,i,u,t}, F_{j,u,t}, V_{j,u,t}, P_{p,j,u,t}, \theta_{z,u}) \quad \forall\, p \in P_z, i \in I_u, j \in J_u, u \in U, t \in T, z \in Z_u \tag{3.7}$$

$$Z^{LB}_{z,u,t} \leq Z_{z,u,t} \leq Z^{UB}_{z,u,t} \quad \forall\, z \in Z_u, u \in U, t \in T \tag{3.8}$$

$$X_{x,j,u,t} = f(F_{i,u,t}, \theta_{x,j,u}, F_{j,u,t}) \quad \forall\, x \in X, i \in I_u, j \in J_u, u \in U, t \in T \tag{3.9}$$

$$\sum_{u \in U^{in}_{H2}} F_{H_2,u,t} = \sum_{u \in U^{out}_{H2}} F_{H_2,u,t} \quad \forall\, t \in T \tag{3.10}$$

$$P^{LB}_{p,j,u,t} \leq P_{p,j,u,t} \leq P^{UB}_{p,j,u,t} \quad \forall\, p \in P_b, j \in J_u, u \in U_b, t \in T \tag{3.11}$$

$$F, V, CP, \rho, P, X, Z \in \Re \tag{3.12}$$

Equation 3.1 describes the objective function being maximized in the optimization problem, which is the total profit of the plant over the time horizon $t \in T$, defined as the revenue subtract the cost of crude/imports subtract the cost of natural gas. The revenue is defined as the product mass flow rates $F_{j,u,t}$ in outlet streams $j \in J_u$ from unit $u \in U_b$ times their cost c_j, where U_b are the blending units and J_u are the streams leaving from these units u. The expenses are defined as the import/crude mass flow rates $F_{i,u,t}$ in inlet streams $i \in I_u$ from unit $u \in U_d$ times their cost c_i, where U_d are the units that use the imports/crude and I_u are the streams entering from these units u. The natural gas (NG) utility usage is defined as the usage in mass flow rate $F_{NG,u,t}$ from unit $u \in U_{NG}$ times the natural gas cost c_{NG}.

Equation 3.2 then describes the mass balance of each unit operation $u \in U$ for each time period $t \in T$ in the refinery, where the sum of the mass flows $F_{i,u,t}$ in the set $i \in I_u$ entering unit u is equal to the sum of the flows $F_{j,u,t}$ leaving unit

u in the set $j \in J_u$; this equation includes blend tanks in the set U.

Equation 3.3 denotes the relationship between the product mass flow rate $F_{j,u,t}$ in set $j \in J_u$ of a unit $u \in U$ as a function of the unit's inlet flow rates in set I_u, output model parameters $\theta_{j,u}$, cut points for specific product j $CP_{j,u,t}$, and unit operating conditions $Z_{z,u,t}$ in set Z_u where the operating conditions are specific to the unit.

Equation 3.4 denotes the conversion relationship from volume to mass (and vice versa) of inlet stream $F_{i,u,t}$ in $i \in I_u$ entering unit $u \in U$ for all time periods $t \in T$ and is related to the inlet volume $V_{i,u,t}$, density $\rho_{i,u,t}$, and uptime (in days) of the unit H_u which are parameters.

Equation 3.5 denotes the relationship for all time periods $t \in T$ between the property $P_{p,j,u,t}$ in the set of properties P_u for the unit u of the outlet stream $j \in J_u$ for all units $u \in U$ as a function of inlet mass flow rates of the unit $F_{i,u,t}$, each inlet's corresponding volume $V_{i,u,t}$, inlet stream densities $\rho_{i,u,t}$, model parameters $\theta_{p,i,u}$ for each property p for the inlet stream, cut point $CP_{j,u,t}$ and composition $X_{x,i,u,t}$ for certain properties. Examples of properties include density, sulphur, octane number, and Reid vapour pressure. Equation 3.6 describes the bounds of the properties described in equation 3.5, and differs depending on the property.

Equation 3.7 denotes the relationship between operating conditions $Z_{z,u,t}$ in the set $z \in Z_u$ of unit $u \in U$ for all time periods $t \in T$ where the operating condition is specific to the unit. The operating conditions for the unit are calculated using: unit inlet and outlet mass flow rates $F_{i,u,t}$ and $F_{j,u,t}$, respectively; inlet and outlet volumetric flow rates $V_{i,u,t}$ and $V_{j,u,t}$, respectively; inlet and outlet properties $P_{p,i,u,t}$ and $P_{p,j,u,t}$, respectively, in $p \in P_z$ where the property depends on the operating condition z (e.g. reaction temperature

may require properties A, B, and C, whereas catalyst conditions uses properties D and E); and model parameters $\theta_{z,u}$ specific to the unit and operating condition. Some examples of operating conditions would include reaction temperatures, changes in temperature between time periods, and heat duties. Equation 3.8 then describes the bounds for the operating conditions described in equation 3.7.

Equation 3.9 denotes the relationship for molecular composition of component X_x, j, u, t for component $x \in X$ where X is a fixed list of molecules (e.g. C2, C3, benzene, etc.) of the outlet stream $j \in J_u$ of unit $u \in U$ as a function of inlet and outlet mass flow rates $F_{i,u,t}$ and $F_{j,u,t}$, respectively, and model parameters $\theta_{x,j,u}$ for the component x for the stream $j \in J_u$ from unit $u \in U$.

Equation 3.10 describes the hydrogen balance for the entire refinery, where the sum of all hydrogen flow rates $F_{H_2,u,t}$ entering hydrogen-fed units u in set U_{H2}^{in} is equal to the sum of hydrogen produced by units in set U_{H2}^{out}. In other words, all hydrogen produced in the refinery must be consumed by other units in the refinery.

Equation 3.11 describes the specifications for the final blend properties $p \in P_b$ for the product streams $j \in J_u$ leaving the blend tank units $u \in U_b$. In other words, specific properties of product streams must be met before being sold to market. For example, gasoline must meet a specific octane number depending on the type of gasoline, while fuel oil is not bound by octane levels but may be bound by sulphur.

Equation 3.12 denotes the domain of the variables in the aforementioned equations, reiterated as: mass flow rate (F), volumetric flow rate (V), cutpoint (CP), density (ρ), property (P), composition (X), and operating condition (Z). All variables are in the real domain.

3.2.1 Equation Transformations

With the presence of conditional statements in the original optimization model, the discontinuity of these statements poses a problem due to the gradient-based nature of nonlinear optimization solvers. If-else statements can be written as max or min functions - for example:

$$\text{if } 5x < 0 \text{ then } y = 0$$
$$\text{else } y = 5x$$

can be written equivalently as: $y = \max\left(0,\, 5x\right)$. Gopal and Biegler [1] have devised smoothing methods for complementarity problems in process engineering using the sigmoidal distribution function, which is commonly used in neural networks. Using the sigmoidal function, the above example can be approximated as:

$$y \approx 5x + \frac{1}{\alpha}\ln(1 + e^{-\alpha 5x})$$

where α is a user-defined parameter. There are various other variables, such as the max of two functions (as opposed to one function and a constant), min functions, and nested min/max functions. Applications and derivations can be found in the published literature by Gopal and Biegler [1].

We then apply this smoothing method to the conditional statements in the optimization model, mainly in the hydrotreater operating conditions and other unit operating conditions.

3.3 Tank Formulation

In order to fully formulate the multiperiod model, there must be a connection between the periods. To connect the periods together, intermediate product tanks are used; the model contains 4 tanks:

- Heavy gas oil (HGO) preceding the distillate hydrocracker (DHC)
- Vacuum gas oil (VGO) preceding the DHC
- Atmospheric residue preceding the vacuum distillation unit (VDU)
- Atmospheric residue imports preceding the VDU

Using these tanks, the volume levels and stock are used to link the periods together where the current level of the tank is dependent on the previous time period and the material entering and leaving the tank at that current period, as shown:

$$EOD_{u,t} = EOD_{u,t-1} + \sum_{i \in I_u} F_{i,u,t} - \sum_{j \in J_u} F_{j,u,t} \quad \forall\, u \in U_k, t \in T$$

where $EOD_{u,t}$, $F_{i,u,t}$, and $F_{j,u,t}$ represent the tank inventory level, mass flow rate entering, and mass flow rate leaving the tank, respectively, for time period $t \in T$. The tank units are denoted by $u \in U_k$ where k is the list of tanks. This same formulation of stock levels at the end of the period is employed by several researchers such as Kolodziej et al. [2] and Castro [3]. This same concept is applied to all properties of the tanks, where the property in the current period is calculated using the inlet and outlet streams, as well as the property in the tank in the previous period. By using these relationships, the periods are now linked together. With the inclusion of the intermediate

inventory tanks, the problem is reformulated as follows:

$$\max PROFIT = \sum_{t\in T}\Bigg[\sum_{u\in U_b}\sum_{j\in J_p} c_j F_{j,u,t} - \sum_{u\in U_d}\sum_{i\in I_d} c_i F_{i,u,t} - \sum_{u\in U_{NG}} c_{NG} F_{NG,u,t}\Bigg] \tag{3.13}$$

subject to: Equations 3.2 to 3.12

$$EOD_{u,t} = EOD_{u,t-1} + \sum_{i\in I_u} F_{i,u,t} - \sum_{j\in J_u} F_{j,u,t} \quad \forall\, u \in U_k, t \in T \tag{3.14}$$

$$EOD_{u,T} = EOD_{u,0} \quad \forall\, u \in U_k \tag{3.15}$$

$$EOD_{u,t}^{LB} \leq EOD_{u,t} \leq EOD_{u,t}^{UB} \quad \forall\, u \in U_k, t \in T \tag{3.16}$$

$$P_{p,u,t} = f(F_{i,u,t}, V_{i,u,t}, \rho_{i,u,t}, P_{p,u,t-1}, EOD_{u,t}) \quad \forall\, i \in I_u, p \in P_u, u \in U_k, t \in T \tag{3.17}$$

$$EOD \in \Re \tag{3.18}$$

Equation 3.15 states that the inventory level in the tank must be replenished to its original stock level for use in future operation after the optimization. Equation 3.16 states the bounds for the inventory level in the tank, with the lower bound usually set as 0. Equation 3.17 shows the relationships between the properties of the tank content, and is related by the mass and volume flow rate, $F_{i,u,t}$ and $V_{i,u,t}$, entering the tank at the current period, the density $\rho_{i,u,t}$ of the inlet streams, the property in the tank from the previous period $P_{p,u,t-1}$, and the stock in the tank from the previous period t-1. Equation 3.18 states the real domain of the stock level.

References

[1] V. Gopal and L. T. Biegler. "Smoothing methods for complementarity problems in process engineering". In: *AIChE journal* 45.7 (1999), pp. 1535–1547.

[2] S. P. Kolodziej, I. E. Grossmann, K. C. Furman, and N. W. Sawaya. "A discretization-based approach for the optimization of the multiperiod blend scheduling problem". In: *Computers & Chemical Engineering* 53 (2013), pp. 122–142.

[3] P. M. Castro. "New MINLP formulation for the multiperiod pooling problem". In: *AIChE Journal* 61.11 (2015), pp. 3728–3738.

Chapter 4

Solution Strategies

4.1 Initialization Strategies

In order to solve the refinery optimization model, the following solution strategy is proposed, as shown in Figure 4.1. The reasoning for this solution strategy is that the initial point is a relatively simple point to start from, and the base case (i.e. nominal, or normal, operation of the refinery) is a good feasible point to start from when solving a case study, as the overall operation of the refinery will not drastically change due to constraint bounds.

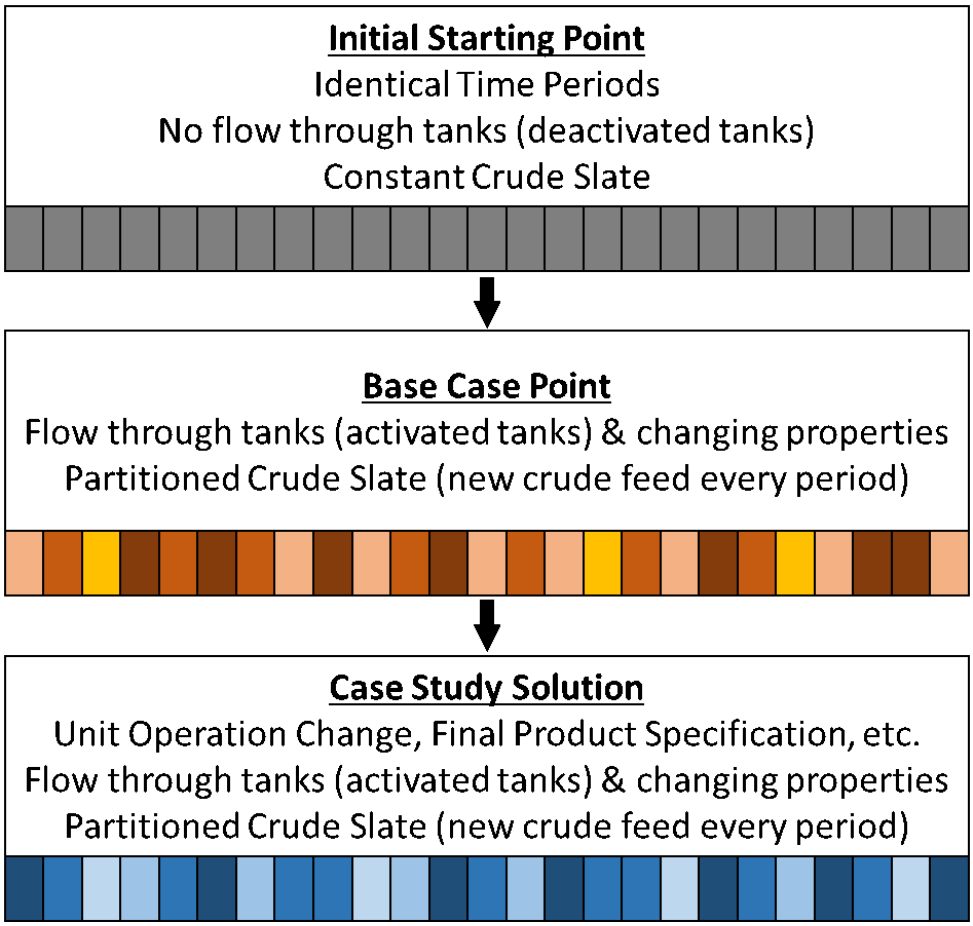

Figure 4.1: Initialization strategy for the multiperiod problem. The initial point is used as the solution to the equally partitioned multiperiod problem with zero flow though intermediate tanks. The solution of this problem is used as the initial point to solve to the base case (i.e. nominal/normal operation of the refinery), where flow is allowed through the tanks and the crude slate and imports are partitioned as per user definition. The solution to this problem is the nominal operation of the plant and is used as the initial point to solve to a case study (i.e. deviations from nominal operation of the refinery.)

First, the optimization problem initializes at a point where every time period is identical by restricting flow through the intermediate product tanks, and by taking the proposed crude slate and averaging it over all periods. For example, if the proposed final crude slate is 100% crude A in period 1, 100% crude B in period 2, 100% crude C in period 3, and 100% crude D in period 4, then the initial starting point would be 25% crude A/B/C/D for all 4 periods. This also applies to the imports to the plant, where if the imports are divided

unevenly throughout the time horizon, they are also averaged out for the time horizon (e.g. if imports only arrive during odd periods, the total sum of these imports are averaged out for the entire horizon). Next, the problem is then solved to a 'base case' point, or nominal/normal operation using the solution of the previous problem (duplicated periods). Flow is allowed through the intermediate tanks and the crude slate and imports become fully partitioned (changed from having the crude slate/imports averaged over the time horizon from the previous step). This point is a suitable starting point, as it is within the normal operating range of the petroleum refinery. Most studies that deviate from this point will not drastically change the operation of the refinery. In the case that there is a failure to converge to the case study, then it is possible that the new point is outside of the operating range of the refinery.

Another benefit to solving from the duplicated periods to the nominal operation point is that it is easier to change time period length and time horizon. For example, it is easy to change the problem from a 5 period problem to a 30 period problem, as solving from the initial point to nominal operation is often feasible. The two problems essentially start at the same point, but the nominal operation of the two problems will be different. Therefore, the nominal operation of the 5 period model cannot be used as the nominal operation of the 30 period model. By allowing the solver to move from the duplicated case to nominal operation, there are fewer problems with trying to find a feasible starting point, as opposed to manually attempting to fix variables to find a feasible starting point.

By attempting to solve from the initial point (of duplicated periods) directly to the case study without first solving to nominal operation, feasibility issues arise and the problem is less likely to converge than trying to solve from the nominal operating point. This is due to the fact that solutions to the case

studies and the duplicated model are relatively far away from each other and are more likely to encounter infeasibilities and difficulty in moving back to a feasible region.

This is compared to solving the problem incrementally (i.e. duplicated model to nominal operation to case study). Furthermore, IPOPT is used to solve from the initial point to nominal operation. Both IPOPT and CONOPT were tested to solve to the nominal point, but only IPOPT was successful. IPOPT does not require a feasible point to start and can update the objective function without satisfying the constraints and identifies the active-set constraints at the solution. On the other hand, CONOPT must identify and calculate the active-set constraints at every iteration and tries to maintain feasibility while improving the objective [1]. Thus, IPOPT is overall more robust and can find the feasible solution to the base case even from a difficult starting point. However, both IPOPT and CONOPT are used to find solutions to the case studies. In solving from nominal operation to deviations in operation, CONOPT has a feasible starting point and can find the new operating point, provided the new operating point is within the same feasible region.

4.2 Lagrangean Decomposition and Formulation

Lagrangean decomposition is a suitable choice for multiperiod problems, as the problem is formulated in a block-structure where each time period is essentially its own problem but with variables and constraints that link the periods together (in this case, the inventory variables and constraints). This block structure is illustrated in Figure 2.19 by Neiro and Pinto.

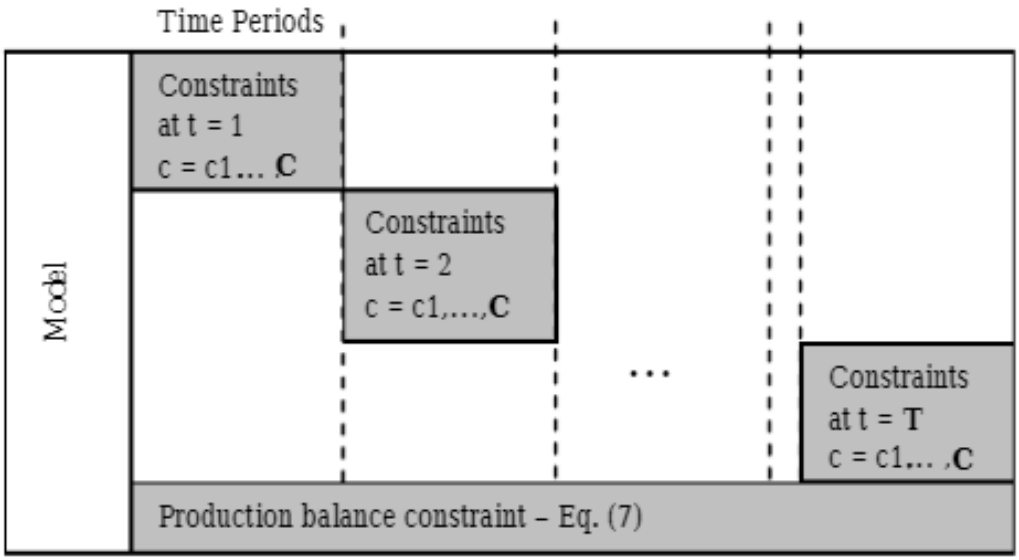

Figure 2.19: Block structure of the multiperiod problem constraints presented by Neiro and Pinto. The constraints of the multiperiod problem resemble a sparse structure, where a block of constraints pertain to one period and the rest of the row contain zeros. The production balance constraints (i.e. inventory/interconnectivity constraints) are segregated and gathered at the bottom of the structure to show their connectivity between units [2].

This block structure can be exploited by separating the interconnection equations into independent equations and solving each time period independently; this is repeated in an iterative procedure. The benefit to this strategy is that solving 30 single period problems is significantly faster than solving a single 30 period optimization model.

The formulation of the Lagrangean Decomposition method, including reformulation of dual and primal subproblems and the algorithm itself, is as follows. To separate the interconnection constraints, the mathematical reformulation is as follows.

Replace equation 3.14 with:

$$EOD^{A}_{u,t} = EOD^{B}_{u,t-1} + \sum_{i \in I_u} F_{i,u,t} - \sum_{j \in J_u} F_{j,u,t} \quad \forall\, u \in U_k, t \in T$$

$$EOD^{A}_{u,t} = EOD^{B}_{u,t} \quad \forall\, u \in U_k, t \in T$$

$\therefore$ the problem becomes:

$$\max PROFIT = \sum_{t \in T} \left[\sum_{u \in U_b} \sum_{j \in J_p} c_j F_{j,u,t} - \sum_{u \in U_d} \sum_{i \in I_d} c_i F_{i,u,t} - \sum_{u \in U_{NG}} c_{NG} F_{NG,u,t} \right]$$

subject to: Equations 3.2 to 3.12, and equations 3.15 to 3.18

$$EOD^{A}_{u,t} = EOD^{B}_{u,t-1} + \sum_{i \in I_u} F_{i,u,t} - \sum_{j \in J_u} F_{j,u,t} \quad \forall\, u \in U_k, t \in T$$

$$EOD^{A}_{u,t} = EOD^{B}_{u,t} \quad \forall\, u \in U_k, t \in T$$

In this reformulation, the linking variables, $EOD_{u,t}$ and $EOD_{u,t-1}$, are redefined as two new variables, $EOD^{A}_{u,t}$ and $EOD^{B}_{u,t-1}$, respectively. However, this reformulation still does not change the problem, as the equality $EOD^{A}_{u,t} = EOD^{B}_{u,t}$ still links the two variables together. The next step is to relax the problem by dualizing the equality constraint (i.e. moving it to the objective function while adding a penalty to the constraint) and separate each time period as independent problems as follows:

$$\max PROFIT_t = \left[\sum_{u \in U_b} \sum_{j \in J_p} c_j F_{j,u,t} - \sum_{u \in U_d} \sum_{i \in I_d} c_i F_{i,u,t} - \sum_{u \in U_{NG}} c_{NG} F_{NG,u,t} \right] + \sum_{u \in U_k} \lambda_{u,t} (EOD^{B}_{u,t} - EOD^{A}_{u,t})$$

subject to: Equations 3.2 to 3.12, and equations 3.15 to 3.18

$$EOD^{A}_{u,t} = EOD^{B}_{u,t-1} + \sum_{i \in I_u} F_{i,u,t} - \sum_{j \in J_u} F_{j,u,t} \quad \forall\, u \in U_k, t \in T$$

However, each time period is still not independent as the duplicated variable $EOD^{B}_{u,t-1}$ in the inventory constraint and $EOD^{B}_{u,t}$ in the objective function are mismatched in time. Thus, the duplicated inventory variable in the objective function is shifted back one period:

$$\max PROFIT_t = \left[\sum_{u \in U_b} \sum_{j \in J_p} c_j F_{j,u,t} - \sum_{u \in U_d} \sum_{i \in I_d} c_i F_{i,u,t} - \sum_{u \in U_{NG}} c_{NG} F_{NG,u,t} \right] + \sum_{u \in U_k} (\lambda_{u,t-1} EOD^{B}_{u,t-1} - \lambda_{u,t} EOD^{A}_{u,t})$$

This new problem is now able to be solved for each time period independently. As this problem is now relaxed and is not the same as the original problem, it is denoted as the 'dual subproblem' and the original problem is denoted as the 'primal subproblem'. The Lagrangean decomposition algorithm may now be used, and is as follows:

1. Initialize values of λ_t.
2. The dual subproblem is solved for each time period and the profit from each period are added together to obtain the overall profit - this generates an upper bound on profit.
3. The interconnected (inventory) variables are passed onto the primal subproblem as fixed variables.
4. The primal subproblem is now solved for each time period independently - this should generate a feasible solution with respect to the full problem (e.g. inventory at the end of the time horizon should be at the same level as the start of the optimization). This generates a lower bound

on profit.

5. The profit from the two problems are compared.
6. If the gap between profits is small enough, then exit the algorithm.
7. If the gap is not sufficiently small, then modify λ_t using a lambda step and go back to step 2.

The Lagrangean decomposition algorithm is also illustrated in the chapter 2 in Figure2.22.

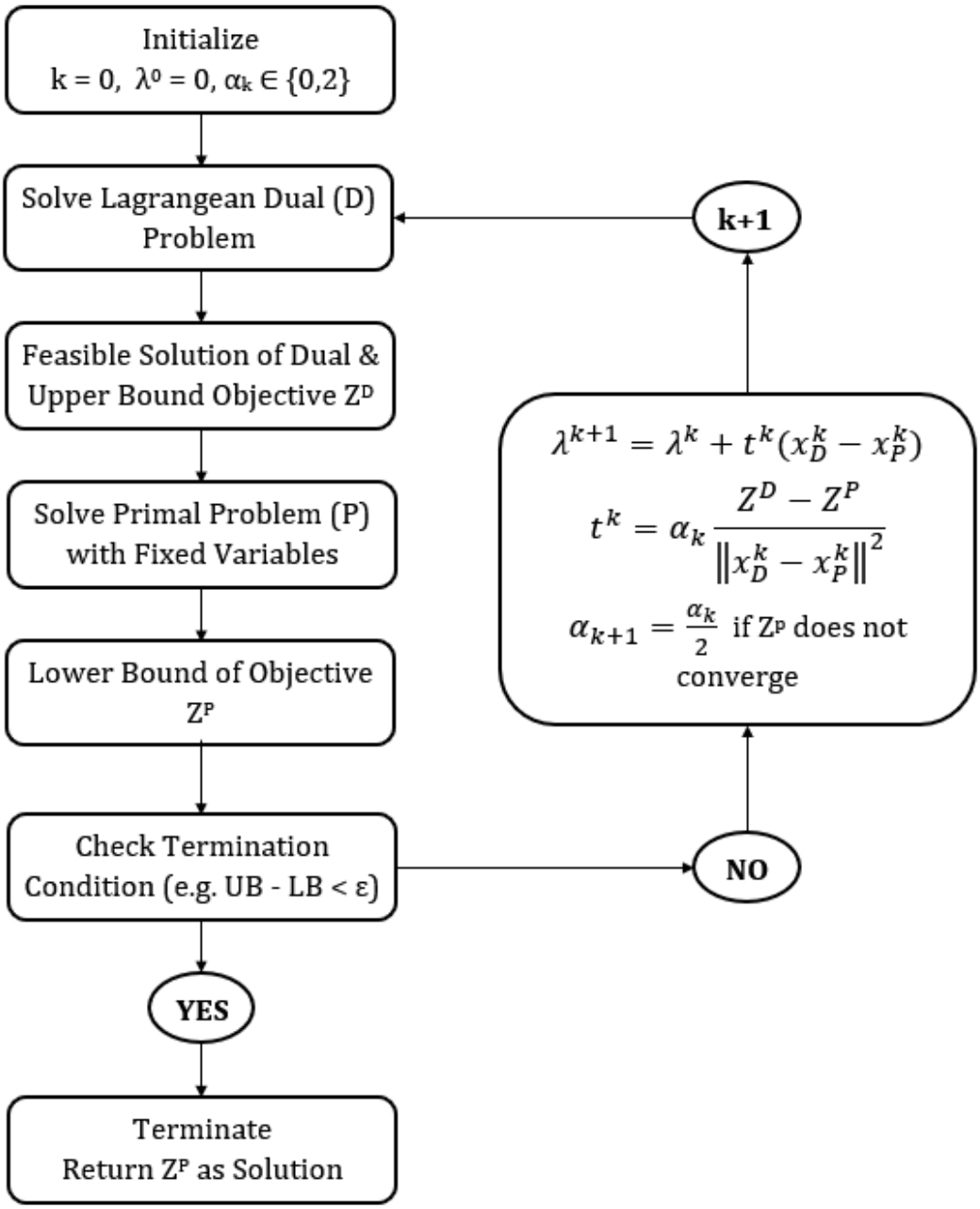

Figure 2.22: Lagrangean decomposition scheme adapted from [3] [4] [5] [6].

4.3 Hybrid Method

The hybrid method is proposed, as it takes a significant amount of time to solve to the base case point from the initial point of duplicated periods. Using the method of directly solving from the initial point to the base case using IPOPT, Table 4.1 lists the solution times for a range of time periods. For smaller problems, the solution times to obtain the base case are relatively small but for larger problems, such as problems with 10 to 30 periods, these times become significantly large (close to 1 hour for the 30 period problem, to solve to the base case).

Table 4.1: Computation times for solving the multiperiod refinery problem using the conventional method of solving directly from duplicated periods as the initial point to the nominal operation of the refinery.

Number of Periods	Solve Time (seconds)
3	21
5	67 (1 min 7 sec)
10	285 (4 min 45 sec)
30	2947 (49 min 7 sec)

Therefore it is attractive to reduce these solution times using Lagrangean decomposition or a hybrid approach. With the hybrid approach, in the case that the gap between the two solutions does not become sufficiently small and the algorithm reaches its maximum number of iterations (which is decided by the user), then the current solution may be used as an initial guess to solve to the base case. Thus, the current solution at the end of the decomposition may be closer to optimality than the original starting point and it would provide the solver a closer point to the base case point. It may also allow other solvers

such as CONOPT to able to solve to the base case instead of only IPOPT, as the new point may provide a feasible point for CONOPT to start from.

References

[1] B. Baumrucker, J. Renfro, and L. T. Biegler. "MPEC problem formulations and solution strategies with chemical engineering applications". In: *Computers & Chemical Engineering* 32.12 (2008), pp. 2903–2913.

[2] S. M. Neiro and J. M. Pinto. "Multiperiod optimization for production planning of petroleum refineries". In: *Chem. Eng. Comm.* 192.1 (2005), pp. 62–88.

[3] S. A. van den Heever, I. E. Grossmann, S. Vasantharajan, and K. Edwards. "A Lagrangean decomposition heuristic for the design and planning of offshore hydrocarbon field infrastructures with complex economic objectives". In: *Industrial & engineering chemistry research* 40.13 (2001), pp. 2857–2875.

[4] J. R. Jackson and I. E. Grossmann. "Temporal decomposition scheme for nonlinear multisite production planning and distribution models". In: *Industrial & engineering chemistry research* 42.13 (2003), pp. 3045–3055.

[5] S. M. Neiro and J. M. Pinto. "Langrangean decomposition applied to multiperiod planning of petroleum refineries under uncertainty". In: *Latin American applied research* 36.4 (2006), pp. 213–220.

[6] F. Oliveira, V. Gupta, S. Hamacher, and I. E. Grossmann. "A Lagrangean decomposition approach for oil supply chain investment planning under uncertainty with risk considerations". In: *Computers & Chemical Engineering* 50 (2013), pp. 184–195.

Chapter 5

Case Studies

5.1 Comparison of Termination Tolerance for Base Case Study

The petroleum refinery optimization model that is used to study deviations from normal operation of the plant is a complex, nonlinear, and nonconvex model. The refinery consists of 21 unit operations, 16 blending units/products, 8 pooling units, 4 intermediate product tanks, 6 types of crude oil, and 3 types of imports. From these unit operations, streams, products, properties, and operating conditions, the model totals up to 4230 variables and 4447 constraints for the single-period problem (which does not include tanks). Table 3.1, as shown in chapter 3 and re-iterated below, shows the model size as the number of time periods increases. For the multiperiod problems, the tanks and other interconnection variables are included in these models.

Table 3.1: Petroleum refinery model size for different number of periods. The number of variables and constraints are listed for various model sizes from 1 to 30 periods.

No. of Periods	Variables	Constraints
1	4,230	4,447
3	12,807	13,280
5	21,345	22,132
10	42,690	44,262
30	128,070	132,782

In order to study deviations from normal operation of the plant, the problem must first be solved to the state of normal operation. Before solving to the base case point, or nominal operation, the problem is first initialized by averaging the crude slate and imports over the time horizon, and preventing flow to enter

through the tanks (by fixing the flow rates to 0), effectively creating replicas of the time periods, as discussed in chapter 4. Once the model solves this problem, the solution is used as an initial guess. The flow is then unfixed and flow is allowed through the tanks, the crude slate and imports are partitioned throughout the time periods, and then the problem is solved to the base case/nominal operation of the plant.

By reformulating the refinery model into a multiperiod model (i.e. higher resolution solution), there is a small benefit in regards to objective function. By taking a 30 day single period model and formulating it into a 5-period 6-day multiperiod model that incorporates tanks and partitioned crude slate, the profit value increases by 0.097065% (with the single period profit as the reference point). Thus, the refinery is able to shift to a slightly better mode of operation when formulated as a multiperiod problem.

When solving to this base case point (normal operation of the refinery) where the crude slate and imports are partitioned and there is flow through the intermediate tanks, the termination tolerance is relevant in determining the time required for convergence to this point. Additionally, it is necessary to confirm that the convergence of this point, regardless of tolerance, does not affect the objective value significantly. The benefit to using a relaxed convergence tolerance is that the computational time may decrease significantly without adversely affecting the solution, and thus a lower tolerance may be used.

Table 5.2 summarizes the computation (CPU) time in seconds for various tolerances for different sizes of problem using the nonlinear solver IPOPT. The tolerance values shown, as defined in IPOPT, are the scaled NLP error described in equations 2.36 and 2.37 [1]. If the NLP error becomes smaller than the defined value, then the optimization will terminate.

Table 5.2: Computation time for solving the multiperiod problem using the conventional method of solving directly from the initial point of duplicated periods to nominal operation. The problem was solved for various sized problems, from 3 to 30 periods, at various convergence tolerances. The computational time (in seconds) and number of iterations required for convergence are listed.

No. of Periods	Tolerance	CPU Time (s)	Iterations
3	Default (1e-8)	21	106
	10	21	103
	1e-5	21	104
5	Default (1e-8)	106	213
	10	67	150
	1e-5	74	156
	1e-6	162	313
	1e-7	103	222
	1e-8	106	213
	1e-9	176	325
10	10	285	248
	1e-5	346	289
30	10	2947	674

For problems with 3 periods, varying the termination criteria does not have an appreciable effect on computation time and number of iterations. This indicates that the problem is small enough that the final iterations of the solution do not take an extended period of time to settle on a final value.

For problems with 5 periods, however, the variance in tolerance causes the solution time to change significantly due to the increased complexity of the problem. Utilization of a large tolerance of 10 or 1e-5 shows significantly

decreased solution times, while tighter tolerances of 1e-8 or 1e-9 display higher CPU times. It is noted that the CPU times are vastly different between tighter and relaxed tolerances and is also indicated by the variation in the number of iterations required for solution. This is an indication that with tighter tolerances, it takes longer for the solver to settle on a solution and in particular, near the end of the optimization. Figures 5.1 and 5.2 show the evolution of the solution for the 5 period case for the default tolerance (1e-8) and a tolerance of 1e-5, respectively. The figures show the iteration number on the x-axis, while showing the normalized profit on the y-axis - the normalized profit is calculated as a fraction of the starting profit:

$$\text{Profit}_i^{Normalized} = \frac{\text{Profit}_i - \text{Profit}_0}{\text{Profit}_0} \tag{5.1}$$

where i is the iteration number. It is shown that the evolution of the solution is exactly the same for both, including the final objective value, but the final termination of the optimization is significantly shorter for the tolerance of 1e-5. The difference between the two optimization runs is 47 iterations, and translates to a reduction in computation time of 33 seconds. The difference in final objective value is 0.000606 and thus, is acceptable in this problem as the profit value is several orders of magnitude larger than this difference.

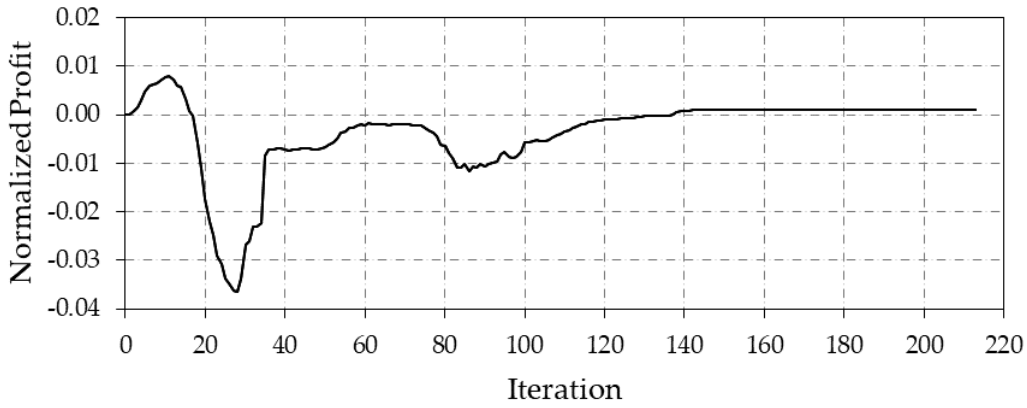

Figure 5.1: Objective function (profit) evolution versus iteration number for the 5 period model, solving from the initial point of duplicated periods to nominal operation. Solved using IPOPT at a default tolerance of 1e-8. The profit is normalized relative to the profit at iteration 0.

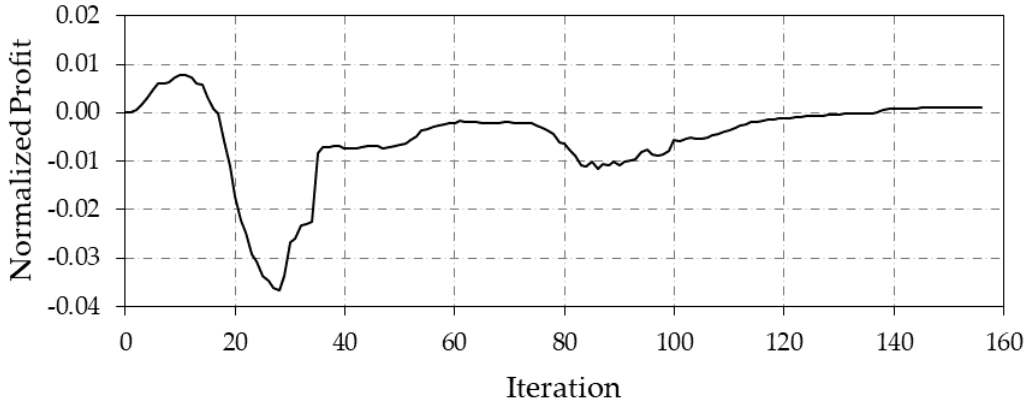

Figure 5.2: Objective function (profit) evolution versus iteration number for the 5 period model, solving from the initial point of duplicated periods to nominal operation. Solved using IPOPT at a tolerance of 1e-5. The profit is normalized relative to the profit at iteration 0.

An anomaly to the general trend of lower computational times for relaxed tolerances is the case of 1e-6. For all cases with 156 iterations or more, the solution evolution up to that point is exactly the same. However, once the optimization passes iteration 156, the objective function stops changing (to a precision of one thousandth); the $\log_{10}$ of the barrier parameter also changes, and is different for each tolerance case (for 1e-6, 1e-7, and 1e-8, the logarithmic

value of the barrier parameters change to -7, -8, and -8.6, respectively). This change in value has a different effect on each case, and in the case of 1e-6, the dual infeasibility starts to cycle. In the other cases, the dual infeasibility resolves relatively quickly. Table 5.3 shows this cycling, and summarizes information from the iteration log for the tolerance cases of 1e-6 and 1e-7, and lists the constraint violation, dual infeasibility, and logarithmic value of the barrier parameter. The table shows the first few iterates after 156 to show the change in infeasibilities and barrier parameter, then shows the cycling in iterates 174 to 180. The dual infeasibility cycling for the case of 1e-6 continues.

Table 5.3: Constraint violation and dual infeasibility for the solving the 5 period model with a tolerance of 1e-6. The iterations shown are when IPOPT stops adjusting the objective function and tries to satisfy the constraints in the model.

Iteration	Constraint Violation	Dual Infeasibility	$\log_{10} \mu$
Tolerance 1e-6			
156	0.00000608	0.00000139	-5.7
157	0.00005570	0.00000723	-7
158	0.00052600	0.00005270	-7
...			
174	0.0000599	0.0000044	-7
175	0.0000584	12.7000000	-7
176	0.0000768	0.0000056	-7
177	0.0000748	12.7000000	-7
178	0.0000985	0.0000071	-7
179	0.0000960	12.7000000	-7
180	0.0001270	0.0000090	-7
Tolerance 1e-7			
156	0.00000608	0.00000139	-5.7
157	0.00005500	0.09100000	-8
158	0.00051700	0.00005210	-8
...			
174	0.0004540	22.6000000	-8
175	0.0006080	0.0000421	-8
176	0.0005940	22.6000000	-8
177	0.0007990	0.0000548	-8
178	0.0001120	0.0000078	-8
179	0.0010600	0.0000718	-8
180	0.0001490	0.0000102	-8

For the 10 period system, the smallest tolerance that will allow convergence is 1e-5, and thus, only a tolerance of 10 and 1e-5 are tested. It is observed that the CPU time gap between these two tolerances is significantly higher than the gap shown for the smaller problems (i.e. over 1 minute gap for the 10 period problem versus only a 7 second gap for the 5 period problem), and is explained due to the larger problem and complexity due to the interconnectivity of all periods. Figures 5.3 and 5.4 show the evolution of the normalized objective function for the 10 period model for tolerances of 10 and 1e-5, respectively. The x-axis shows the iteration number, and the y-axis shows the normalized profit value. Again, similar to the 5 period case, the evolution of the profit is exactly the same for the two cases except for the termination at the end of the optimization, where the larger tolerance of 10 terminates earlier. The difference between these two problems in iteration number is 41, and the computation time difference is 61 seconds. Comparing final profit values, the difference is 0.004201 and again, is insignificant in the context of the refinery optimization problem.

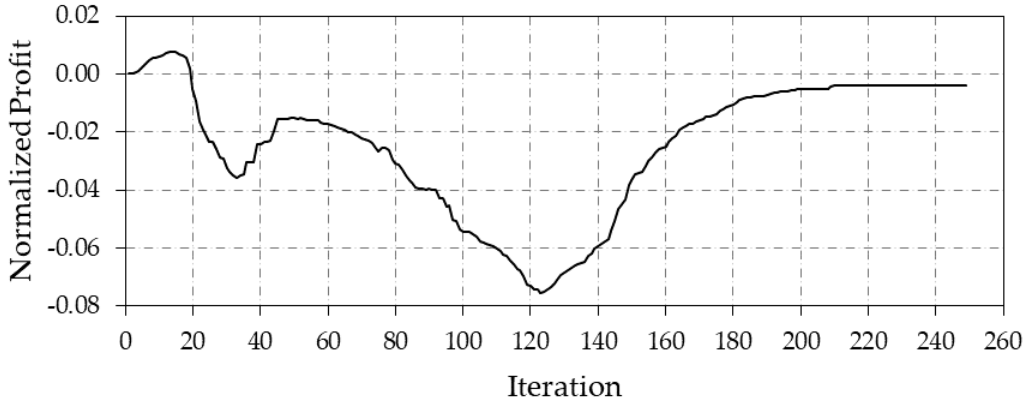

Figure 5.3: Objective function (profit) evolution versus iteration number for the 10 period model, solving from the initial point of duplicated periods to nominal operation. Solved using IPOPT at a default tolerance of 10. The profit is normalized relative to the profit at iteration 0.

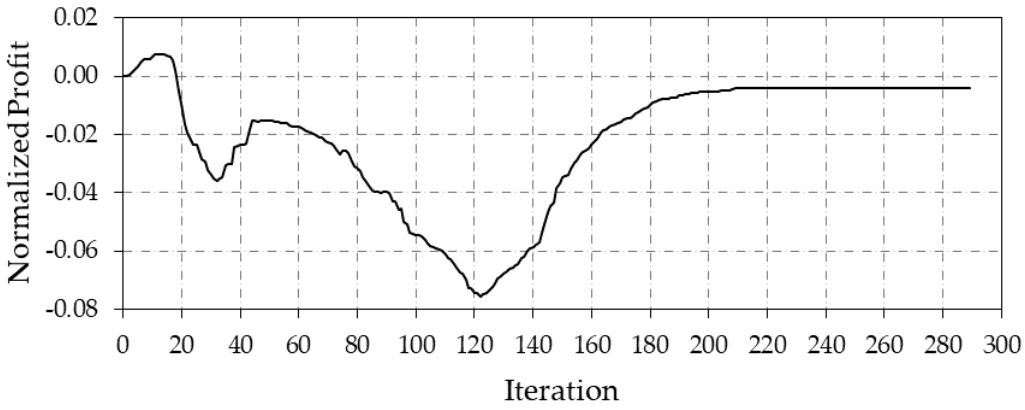

Figure 5.4: Objective function (profit) evolution versus iteration number for the 10 period model, solving from the initial point of duplicated periods to nominal operation. Solved using IPOPT at a default tolerance of 1e-5. The profit is normalized relative to the profit at iteration 0.

For the 30 period model, a tolerance of 10 requires close to 50 minutes to solve and other tolerances smaller than 10 did not solve in under an hour and were terminated. This large computational time displays the rapidly increasing complexity for larger problems, and that the problem does not scale linearly with time periods.

In the context of a full scale petroleum refinery schedule optimization model, the tolerance has a significant impact in the computational time, as well as feasibility. For the model studied, the tolerance required for termination was varied for the optimization from the initial point (of duplicated time periods) to the base case point (normal operation of the refinery). For smaller problems, the impact of the tolerance is less significant on the solution time and number of iterations required for termination and is evident for the 3 period model. The 5 period model solution times vary more than the 3 period model, but even with a tighter tolerance than default, solution times are reasonable. For larger problems, the tolerance termination criteria is significant in finding feasible solutions, as demonstrated by the 10 and 30 period models. In these

models, the default tolerance does not allow the problems to resolve in a reasonable time frame, and thus, the tolerance is lowered to find a solution. It is shown, however, that decreasing the tolerance does not affect the solution significantly, as shown in figures 5.3 and 5.4, and the same line of reasoning can be extended to the 30 period model, and other larger models.

5.2 Refinery Case Studies - Deviations from Nominal Operation

Various case studies were investigated in order to compare the effectiveness of IPOPT and CONOPT against the original model, as well as to compare the performance between the IPOPT and CONOPT solvers. The case studies that were proposed reflect deviations from nominal operation of the refinery, and illustrate potential issues that may arise. The cases studied were as follows:

- Decreasing the maximum available throughput of the distillate hydrocracker (DHC) by 5%
- Decreasing the upper limit of the vacuum gasoil (VGO) cut point by 2%
- Increasing the 95-octane gasoline Reid Vapour Pressure (RVP) allowable limit by 51%
- Decreasing the maximum available throughput of the catalytic reformer (CCR) by 6%

In the first case where the throughput to the DHC is decreased, this causes a disruption in the conversion of gas oils into high value products such as naphtha and kerosene. This can cause further problems downstream to other

units such as the catalytic reformer, which converts the naphtha into high-octane products for blending. Due to this throughput reduction, the heavy gas oil and vacuum gas oil tanks must compensate for the lower flow rate, and also adjust for the imports incoming to the refinery.

In the second case study, by decreasing the allowable range for the VGO cutpoint, the production of VGO is decreased. Again, this causes a cascading effect to the DHC and subsequent unit operations by decreasing one of the flow streams to the DHC. This may be compensated by adjusting the HGO cutpoint (and thus, the other cutpoints from the CDU), or the inventory of the VGO tank may be adjusted to maintain VGO feed into the DHC.

The third case study illustrates a relaxation in the specifications of a high valued final product, 95-octane gasoline. By increasing the allowable limit for Reid vapour pressure, the blending feeds of the gasoline may be optimized further by re-routing products from other blends to the gasoline.

The fourth case study decreases the throughput of the CCR. This can cause a similar situation as the decrease in DHC throughput but with more severe consequences. The products of the CCR are highly valued, as many of the products contain a high octane number and are critical for blending and meeting gasoline specification. Therefore, the CCR is always at maximum capacity and if throughput is decreased, then the amount of product is decreased as well. This may cause some final products to not meet specification, and it may be difficult for other units to compensate.

With the interconnectivity between unit operations, variables, properties, and streams in the refinery, even the smallest change from nominal operation, as illustrated in these case studies, may lead to cascading effects throughout the plant. Having a multiperiod formulation allows insight in the operation of the

refinery over a specified time horizon where the intermediate product tanks play a crucial role in allowing feasible solutions.

While the plant operation was varied for each case, the crude oil slate and imports to the refinery were kept consistent. Furthermore, they were repeated every 5 time periods for the crude slate and every 2 time periods for the imports. Table 5.4 shows the crude oil and import shipment schedule for a 5 period time horizon. Each column represents a time period, with each crude making up a percentage of the crude slate for that period (e.g. in period 1, the crude slate consists of 50% crude A and 50% crude F); the imports to the plant are both denoted as 100% for every other time period, as they are not considered within the crude slate, are independent from one another, are only imported to the plant every other period. This schedule is repeated for problems with larger time horizons (e.g. for a 10 period problem, this schedule is repeated once).

Table 5.4: Crude slate and import partition for the nominal operation of the refinery. For example, in period 1, the crude slate consists of 50% of crude A and 50% of crude F, and has maximum imports of import A and B.

Time Period	1	2	3	4	5
% Crude A	50	0	0	50	0
% Crude B	0	50	30	0	20
% Crude C	0	0	0	0	0
% Crude D	0	0	50	0	0
% Crude E	0	0	0	0	0
% Crude F	50	50	20	50	80
% Import A	100	0	100	0	100
% Import B	100	0	100	0	100

5.3 Performance of CONOPT and IPOPT Versus Original Model Solution

The four case studies outlined in the previous section, plus an additional case study where the crude shipment in periods 2 and 3 are forced to be 100% Crude D, were solved using CONOPT and IPOPT in order to compare the performance with the refinery model developed and solved by TOTAL. This is done in order to gauge the effectiveness of regarded solvers in academia on a real world problem and scale.

Table 5.5 shows the comparison of the aforementioned case studies between solutions of the original TOTAL model solved in a spreadsheet application and solving the problem using IPOPT/CONOPT for a 5 period multiperiod refinery model. The computational (CPU) time is shown for solving from the base case (nominal operation) to the case study using IPOPT, CONOPT, and the TOTAL model.

Table 5.5: Comparison of the computation time for the TOTAL model solved in a spreadsheet application versus solving using IPOPT and CONOPT for the 5 period model for the case studies (deviation from nominal operation). In all cases, the initial point used is the solution to the nominal operation case.

	CPU Time (s)		
Case Study	IPOPT	CONOPT	Original Model
DHC Throughput	74	26	606
VGO Cut Point	37	26	1,680
Gasoline Vapour Pressure	70	20	486
CCR Throughput	73	32	960
100% Crude D in Periods 2 & 3	208	43	850

From these comparisons, there is an immediate benefit to using IPOPT or CONOPT, with solution times significantly lower than the original solver and model by an order of magnitude. However, due to a different configuration of inventory tanks, as well as reformulation of several equations, the models cannot be directly compared side by side. In the TOTAL model, the configuration is different due to four additional small tanks used for other intermediate products. Another difference is that the if and else statements in the original model are represented using smoothing functions as described in the literature [2]. Thus, there are no results directly comparing profit objective values from the original model to the reformulated model in AMPL with IPOPT and CONOPT.

Nonetheless, all objective function values and model variable results are within the same order of magnitude and range, and both models display similar behaviour, such as the decisions made in the intermediate product tanks, where the contents are depleted and stocked in the same fashion. The results provide context and estimation of the required solve times.

Furthermore, the single period results can be used as a direct comparison between CONOPT/IPOPT and the TOTAL model, as no tanks are used in the formulation of the single period model and everything remains the same, with the exception of the smoothing functions. Table 5.6 shows the difference in the single-period model objective function between the TOTAL model and the model solved with CONOPT for various deviations from nominal operation (case studies), using the TOTAL model as the reference point. The results confirm that the model formulation and implementation in AMPL is correct for the single period model and display the same behaviour as the original TOTAL model. Thus, the use of the CONOPT and IPOPT solvers for the refinery optimization problem posed by TOTAL and other similar problems

proves to be an attractive choice for this application.

Table 5.6: Difference in objective function of the single period model between the TOTAL model solved in a spreadsheet application and solved using CONOPT. This is to verify that the refinery models behaves similarly and results may be compared directly for the single period case.

Case Study	% Difference in Profit Between Models
Base Case	0.000039%
Unrestrict Crude B	0.000017%
Increase VGO Cutpoint	0.003956%
Increase CDU Throughput	0.005660%
Increase VDU Throughput	0.000109%
Reduce CCR Throughput	0.136519%
Increase DHC Throughput	0.000128%
Increase Imports A and B	0.000101%
Zero Gasoil Import	0.000051%
Limited Jet Exports	0.000089%
Limited DHC Bleed	0.007239%
Increase Gasoline RVP	0.000054%
Increase ULSD Cloud Point	0.000071%
Increase % Kerosene in Fuel Oil	0.000059%

5.4 Problem Scalability — Solution of Larger Problems

When studying a multiperiod optimization problem, the size and complexity of the problem is directly related to the number of partitions defined by the user. By increasing the number of partitions, the time horizon can be increased

by maintaining the length of each period, or the resolution of the problem may be increased by decreasing the length of each period. For example, a 5 period model at 6 days per period equates to a time horizon of 30 days. If the model is extended to 30 periods, if the length is maintained at 6 days per period, the time horizon increases to 180 days; or if the length is decreased to 1 day per period, the overall problem becomes more detailed. As a result, operational decisions can be made at shorter time intervals or for problems that model long term dynamics, the overall operation and reaction of the process may be observed. However, as the problem size increases, the problem becomes larger and more complex due to the interconnection constraints and variables, and the resulting solution times become too large. As the computational time increases, it may be unattractive to solve high resolution problems as the time required to solve may not be fast enough to respond to process changes. Thus, the solution times for various sized problems for various case studies (deviations from normal operation) are presented and discussed.

Table 5.7 summarizes the comparison of the aforementioned case studies by solving with the IPOPT and CONOPT solvers for various model sizes, from a single period model up to 30 periods. The first column describes the 4 case studies that were outlined at the beginning of the chapter: reducing the throughput of the distillate hydrocracker unit; tightening the bounds of the VGO cutpoint; relaxing the allowable Reid Vapour Pressure limit of the gasoline product; and reducing the throughput of the catalytic reformer. Each case study was then run for a range of time horizons of 1, 3, 5, 10, and 30 periods for both the IPOPT and CONOPT solvers. From each run, the computational (CPU) time and number of iterations were recorded. In Table 5.7, the non-entries, denoted by a dash (-), represent a failure to convergence. It is noted that the tolerance for both solvers was set to 1e-5.

Table 5.7: Computation times and iterations for the case studies (deviation from nominal operation) for various sizes of the multiperiod model (from 1 to 30 periods) using IPOPT and CONOPT. Dashes (-) represent failure to converge.

		IPOPT		CONOPT	
Case Study	Periods	CPU Time (s)	Iter.	CPU Time (s)	Iter.
DHC	1	-	-	0.233	12
Throughput	3	-	-	-	-
	5	74	173	26	162
	10	281	312	117	212
	30	502	134	-	-
VGO Cutpoint	1	54	1135	0.344	44
	3	7.5	39	5.2	119
	5	37	80	26	224
	10	229	198	114	617
	30	251	72	-	-
Gasoline Reid	1	20	373	0.095	9
Vapour Pressure	3	8.5	43	10.5	680
	5	80	145	20	104
	10	108	101	136	1432
	30	658	165	-	-
CCR	1	-	-	0.457	16
Throughput	3	7.5	42	8.4	70
	5	73	158	32	166
	10	134	114	216	387
	30	1367	291	-	-

It is observed that in general, IPOPT is able to handle larger problems, and

CONOPT is the faster solver for solving smaller problems [3]. By the nature of the solvers, IPOPT is able to find solutions in difficult regions, as IPOPT does not require a feasible point to start and can update the objective function without satisfying the constraints. At the end of the optimization, IPOPT will identify the active-set constraints. CONOPT instead will find solutions quickly if the feasible starting point is close to the new optimum, as CONOPT requires constraint feasibility before proceeding with objective function improvement using the reduced gradient. In other words, CONOPT must deal with identifying the active-set constraints at each iteration [3]. If the new optimal solution is relatively close to the initial point without complex infeasible regions in between the points, then CONOPT is an effective solver. Thus, IPOPT is used to solve from the initial point to the base case to build the base multiperiod model, then either CONOPT or IPOPT is used to solve the case study to the new optimum. If CONOPT can find a solution to the case study, it will often solve significantly faster than IPOPT. However, if the case study optimum is in a difficult region, then CONOPT may fail and IPOPT may succeed.

Both methods return similar results, as shown in the profit values shown in Table 5.8. Table 5.8 lists the case study, the number of periods the model is partitioned into, and the deviation of profit from nominal operation for both IPOPT and CONOPT. The deviation is calculated as follows:

$$\Delta\text{Profit}^{deviation} = 100\,\frac{||\,\text{Profit}^{nominal} - \text{Profit}||}{\text{Profit}^{nominal}}$$

Furthermore, the mass balances for both methods were compared side-by-side to ensure robustness between the two solutions. Using the results in Tables5.7 and 5.8, the behaviour of the model can be explained based on the computational time of the solvers used.

Table 5.8: Comparison of solution for the case studies (deviation from nominal operation) for various sizes of the multiperiod model (from 1 to 30 periods) using IPOPT and CONOPT. Used to verify the solutions obtained from the two solvers. Dashes (-) represent failure to converge.

		IPOPT	CONOPT
Case Study	No. of Periods	Δ% From Nominal Profit	
DHC Throughput	1	-	0.96
	3	-	-
	5	5.24	5.23
	10	5.30	5.04
	30	22.00	-
VGO Cutpoint	1	4.31	4.31
	3	2.17	2.17
	5	3.30	3.30
	10	3.14	3.14
	30	24.64	-
Gasoline Reid Vapour Pressure	1	11.37	11.37
	3	11.07	11.07
	5	11.05	11.05
	10	11.14	10.96
	30	42.91	-
CCR Throughput	1	-	1.76
	3	2.47	2.47
	5	2.26	2.28
	10	2.39	2.39
	30	25.40	-

In the single period problem, CONOPT is superior to IPOPT and indicates

that the initial starting point is close to the optimal solution as the reduced gradient can quickly find the new optimal point. Additionally, with a small model such as the single period model, there is a small number of degrees of freedom and CONOPT is well suited to finding a first feasible solution for these types of problems [4]. However, IPOPT may try to improve the objective function and may encounter a new region which is infeasible and has difficulty in successfully backtracking in the restoration phase (as shown in the high number of iterations of the VGO cutpoint and Gasoline RVP cases).

For the 3 period model, the two solvers are comparable in successful case studies, where CONOPT may not fall into an infeasible region and the gradient is easily found at the initial point. The reduced distillate hydrocracker (DHC) throughput case shows that the problem is infeasible for both solvers and indicates infeasibilities within the model. This is attributed to the final product qualities and the missing crude slate from periods 4 and 5 as outlined in table 5.4. The properties from these missing crude slates help allow the final products to reach the desired final product qualities, and thus the lack of these crudes proves detrimental to the feasibility of the solution. In particular, the absence of Crude F and B causes the overall production of vacuum gas oil to decrease, which is an essential feed to the DHC, and compounded by the fact that the overall DHC throughput is restricted will not allow some of the final products to meet their specifications. For the remainder of the case studies, the computational time is close for both solvers.

For the 5 period model, the results from the solvers IPOPT and CONOPT start to deviate. In terms of objective value, both solvers produce the same results and is an indication that the initial starting point is in a region that both solvers can find a solution. In all case studies, however, IPOPT is shown to be significantly slower than CONOPT. In the first case study, IPOPT produces a

solution almost 3 times slower than CONOPT; in the second case study, the gap is close, with IPOPT slower by 11 seconds; in the third case study, IPOPT is 4 times slower than CONOPT; and finally, in the last case study, IPOPT is more than 2 times slower than CONOPT. This gives an indication of the characteristics of the 5 period model. The initial point (nominal operation) is at a feasible point, but more importantly, the path between this point and the case study is a suitable feasible region for CONOPT, where the solver does not encounter difficult regions and can go directly to the new solution. The fourth case study, reducing the CCR throughput, using CONOPT and IPOPT are illustrated by figures 5.5 and 5.6, respectively, where the x-axis shows the iteration number, and the y-axis shows the normalized profit, calculated using equation 5.1.

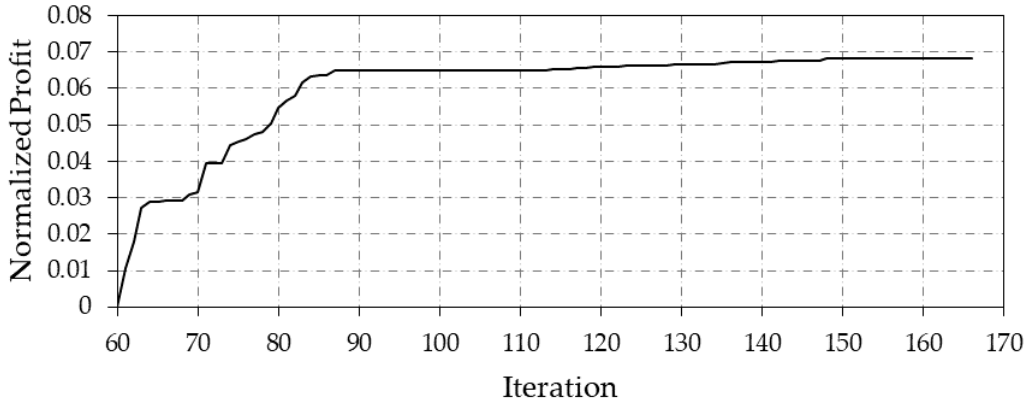

Figure 5.5: Objective function (profit) evolution versus iteration number for solving the 5 period model, solving from nominal operation to the case study of reduced CCR throughput. Solved using CONOPT. Profit is normalized from the starting profit value (at iteration 60, which is not the same starting point as the IPOPT case due to the phase 0 and 1 solve of CONOPT).

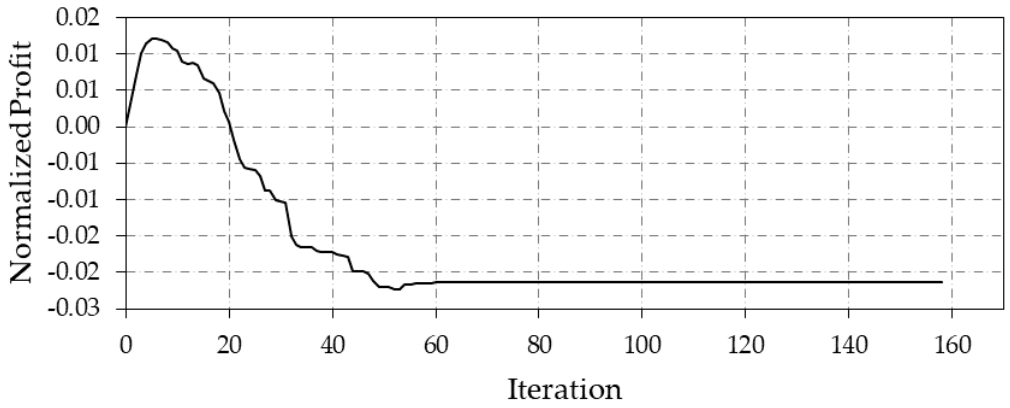

Figure 5.6: Objective function (profit) evolution versus iteration number for solving the 5 period model, solving from nominal operation to the case study of reduced CCR throughput. Solved using IPOPT. Profit is normalized from the starting profit value (at iteration 0).

For CONOPT, the iterations do not start at 0, as the first phase of the CONOPT algorithm attempts to find the first feasible solution, then will start the reduced gradient calculation. On the other hand, IPOPT does not have trouble finding a path to the solution, but takes longer to settle on a solution, as shown in the tail of the solution in figure 5.6. This long tail is attributed to the cycling of the dual infeasibilities shown in figure 5.8. Figures 5.7 and 5.8 show the constraint violation and dual infeasibilities corresponding to figure 5.6 (solving the CCR case study using IPOPT), respectively. IPOPT defines the constraint violation as the max-norm of h(x) in the reformulated NLP shown as the second term in equation 2.37 (IPOPT internally replaces inequality constraints with equality constraints with slack variables and then the bound constraints are the only inequality constraints remaining). IPOPT then defines the dual infeasibility as the max-norm of the first KKT condition of the barrier problem, shown as the first term in equation 2.37. In both figures, the constraint violation/infeasibility reported by IPOPT is shown on the y-axis, while the iteration count is displayed on the x-axis and starts at 65 as the objective value stops changing at iteration 65 and IPOPT tries to reduce the infeasibilities at

the optimum.

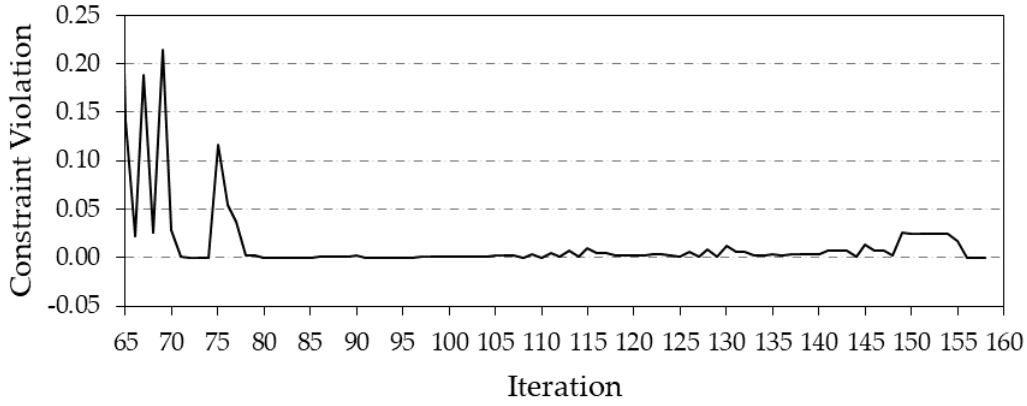

Figure 5.7: Constraint violation versus iteration (starting at 65, where the objective stops changing) for the IPOPT algorithm for the 5 period model of solving from nominal operation (base case) to the case study of reduced CCR throughput.

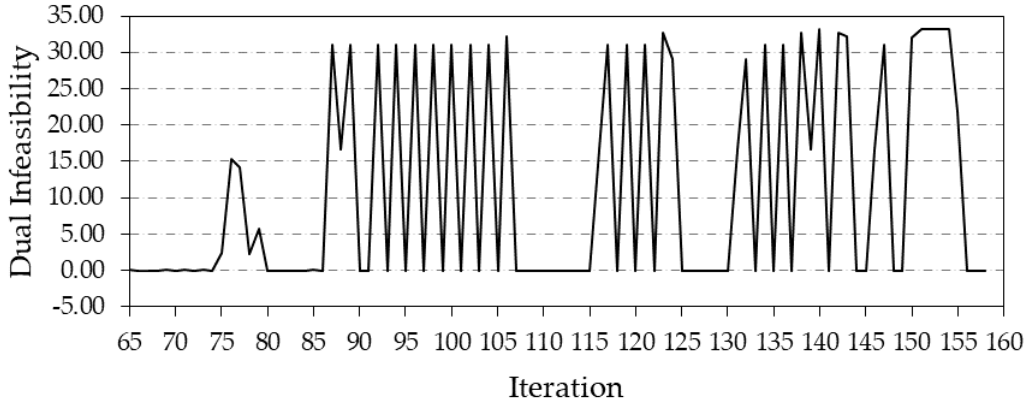

Figure 5.8: Dual infeasibility versus iteration (starting at 65, where the objective stops changing) for the IPOPT algorithm for the 5 period model of solving from nominal operation (base case) to the case study of reduced CCR throughput.

From figure 5.7, the constraint violation decreases rapidly and remains at a low value at an order of magnitude of 10^{-3} to 10^{-8}. However, the scaled dual infeasibility is cyclic, jumping to values between 10^{-4} and 30. While there are several potential reasons why this may occur by looking at the first term in equation 2.37, we narrow this down to one likely cause. First, the gradient of the objective function may be eliminated as a cause since at this

point, the objective is no longer changing significantly. Second, the gradient of the equality constraints h(x) may be eliminated as well because the constraint violation remains small and thus the gradient must be small as well. Thus, the only remaining terms that may affect the dual infeasibility are the multipliers (z) for the bound variables or the scaling factor. Therefore, this shows that the bound constraints of the problem have difficulty in stabilizing under the infeasibility tolerance (by default, the non-scaled dual infeasibility tolerance is 1). The long tail in the IPOPT CCR case study can be attributed to the cyclic dual infeasibility and the inability to stabilize, and thus, resulting in longer computational times.

For the 10 period model, both CONOPT and IPOPT are faster than the other in half of the case studies tested, where IPOPT is faster in the third and fourth case study, and CONOPT is faster in the first and second. Both solvers return similar results, with IPOPT obtaining a slightly higher value for cases one and three. For the 30 period model, only IPOPT is able to solve the model and CONOPT fails to converge to a solution. This shows that IPOPT is more suited to solve large problems, and can start from a far point, such as the base case point (which is more than 20% from the optimal solution for all case studies, based on profit, as shown in table 5.8).

In general, based on the results shown, CONOPT is more suited toward smaller problems, whereas IPOPT can handle larger problems. The benefit to using CONOPT for smaller problems is that many case studies can be run consecutively in a short amount of time in order to study many variations in operation, or to run a sensitivity analysis where one or several parameters can be varied incrementally. However, for medium sized problems, both IPOPT and CONOPT converge to the same or very similar optima and can be used for these problems.

5.5 Lagrangean Decomposition Results

Due to the complexity of the petroleum refinery model, it becomes increasingly difficult to solve the model as the time horizon increases. For example, as outlined in table 5.2, in order to solve to the nominal operation of the plant from the original starting point, it takes close to 50 minutes to solve the 30 period model, and then to solve it further to a deviation in operation will push the total solution time to over 1 hour. Lagrangean decomposition was proposed as a solution strategy for this problem, as it is relatively fast to solve a single period model, and if the multiperiod model could be broken down into smaller parts, then there are potentially significant gains in computational times.

First, the Lagrangean decomposition formulation is modified slightly to reflect the interconnectivity of the refinery model. Instead of only duplicating the inventory, as outlined in Chapter 4, all properties associated with the intermediate tanks are duplicated. The tank properties, such as sulfur, density, and nitrogen, are calculated using the inventory and properties from the previous time period, and thus must be separated in order to create independent time periods. The result of the new formulation changes the way that the multiplier step is calculated. Equation 2.55 was modified to incorporate all of the interconnected variables. Instead of only using the difference in inventory in the denominator, the differences of all of the interconnected variables and their duplicates was used and summed to give the denominator. The multiplier step is modified as so:

$$t^k = \frac{\alpha_k \left(Z^D - Z^P\right)}{\sum_{t \in T} \sum_{y \in Y_L} \left\| x^A_{y,t,k} - x^B_{y,t,k} \right\|^2} \tag{5.2}$$

where t_k is the step size, α_k is the reduction factor, Z^D is the objective value for the dual, Z^P is the objective value for the primal, $x^A_{y,t,k}$ and $x^B_{y,t,k}$ are the original and duplicated variables, respectively, k is the iteration number of the decomposition, y is the separation variable in the set Y_L. The multipliers are then calculated for all interconnected variables using the common calculated step. Additionally, the multipliers are initialized at 0 as suggested by Jackson and Grossmann [5] and Escobar [6].

Next, the duplicated variables are initialized to the values from the solution of duplicated time periods of the original problem. In other words, the original problem of no flow through tanks was solved first, and the duplicated variables were initialized at this point. This is done in order to prevent undefined calculations, such as division by 0 in equations in the dual subproblem that use the duplicated variables in denominators. This is not an issue for the primal problem, as that problem only uses the original variables and no division by 0 would occur. The algorithm is then followed, as outlined in figure 2.22. Once the dual subproblem is solved, all interconnected variables are fixed as the non-duplicated variables (the 'A' variables), including properties, and passed onto the primal subproblem. The primal and dual objectives are compared and if the gap between them is not sufficiently small, then the multipliers are updated and the algorithm repeats until convergence. All problems are solved with CONOPT. Figures 5.9 to 5.11 show the evolution of the dual and primal objectives, solving to the *nominal operation* point, where the dual is the upper bound and primal as the lower bound.

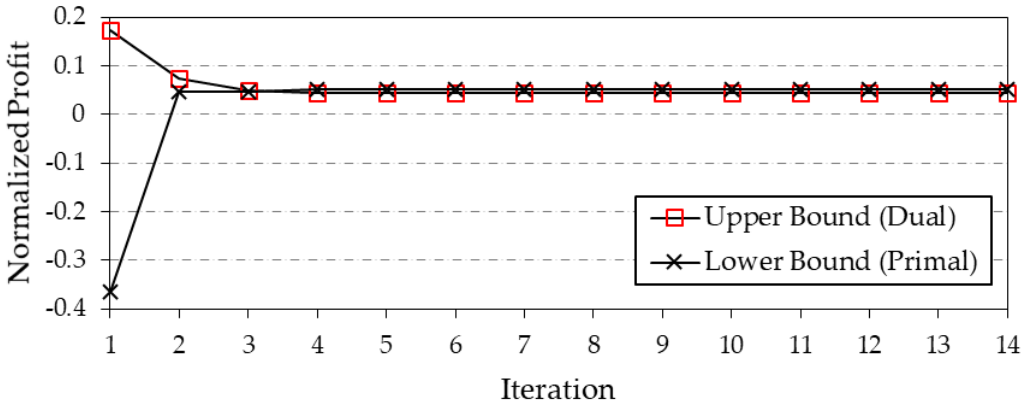

Figure 5.9: Objective value (profit) evolution of the dual (□) and primal (X) solutions versus the number of decomposition iterations (full runs; a run is defined as solving the dual and primal subproblems) for the 5 period model. Solving from the initial point of duplicated periods to nominal operation, using CONOPT for both the dual and primal subproblems. Profit is normalized from the nominal profit value from the original full space model.

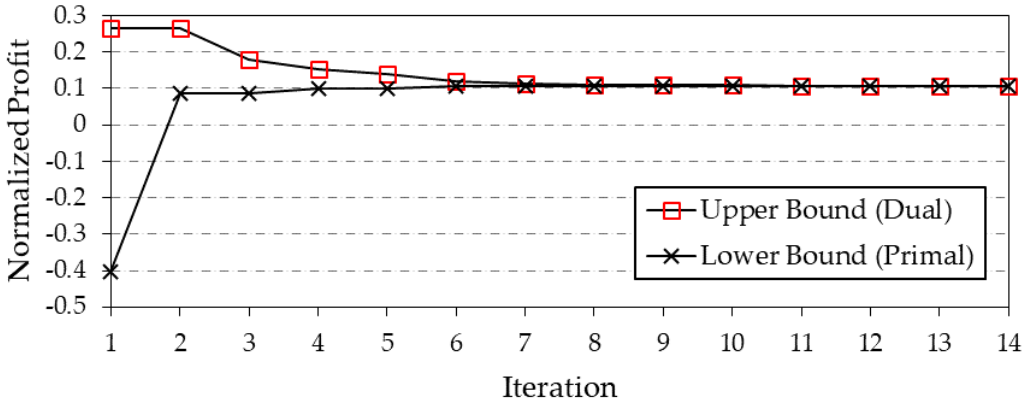

Figure 5.10: Objective value (profit) evolution of the dual (□) and primal (X) solutions versus the number of decomposition iterations (full runs; a run is defined as solving the dual and primal subproblems) for the 10 period model. Solving from the initial point of duplicated periods to nominal operation, using CONOPT for both the dual and primal subproblems. Profit is normalized from the nominal profit value from the original full space model.

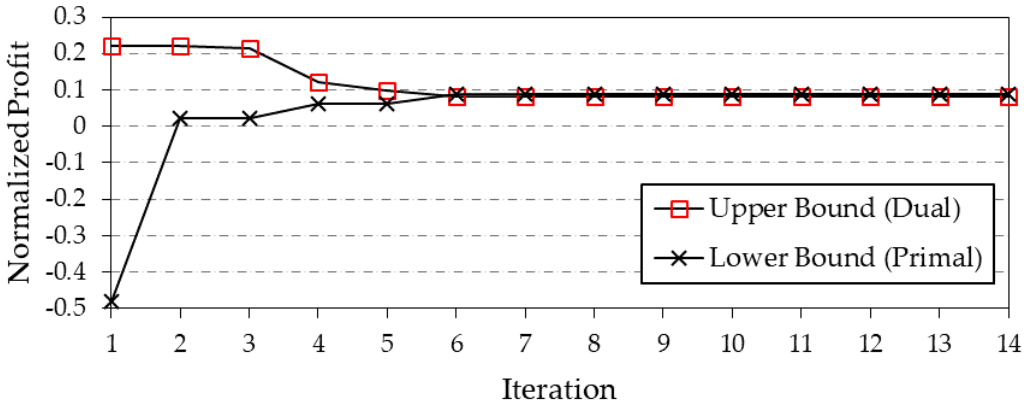

Figure 5.11: Objective value (profit) evolution of the dual (□) and primal (X) solutions versus the number of decomposition iterations (full runs; a run is defined as solving the dual and primal subproblems) for the 30 period model. Solving from the initial point of duplicated periods to nominal operation, using CONOPT for both the dual and primal subproblems. Profit is normalized from the nominal profit value from the original full space model.

The y-axis shows the normalized profit relative to the nominal base case solution (i.e. the difference between the solutions relative to the nominal base case). The x-axis shows the decomposition iteration, defined as a full solve of the dual subproblem and the primal subproblem. The 5 period model, shown in figure 5.9, shows a fast approach to a plateau for both problems within 3 iterations of the decomposition algorithm, while the 10 and 30 period problems, illustrated by figures 5.10 and 5.11, respectively, the plateau occurs at iteration 6 and 7, respectively. The termination criteria for all cases was set to a value of 1 in order to observe the plateaus of the two problems.

However, it is noted that the upper bound in the 5 and 30 period cases plateaus slightly below the plateau of the lower bound. This is due to the infeasibility of the lower bound primal solutions once the interconnected variables were fixed from the dual problem solution. However, the refinery model considered in this study incorporates the properties of the tanks and streams in addition to the tank inventories in regards to interconnection variables, while the stud-

ies published by Jackson and Grossmann [5], and Neiro and Pinto [7] only consider tank inventories. This results in a more complex model and thus, it is difficult to achieve feasibility for the primal problem. Jackson and Grossmann [5] also state difficulty in achieving feasibility in their primal problem and result in relaxing the demand constraints in their problem, and thus, in practice, it is difficult to achieve feasibility without relaxing constraints.

In regards to the dual problem, almost all periods were feasible - the exception being the final period. This may be attributed to the constraint that comes into effect only in the final period, where the final inventory must return to the same level as the initial stock. This constraint is implemented in order to foresee use of the inventory tank past the scope of the optimization (e.g. if a unit fails some time after the optimization ends, the inventory tank would be at its original level to deal with the failure). Several checks were performed in order to ensure that the problem was model complexity related and not formulation (or human error) related.

The first check was to eliminate the constraint entirely - this yielded a feasible solution for the final period without affecting the other periods. The second check was to force a feasible solution in the final period to ensure that the dual problem was able to find a feasible solution. As the dual subproblem is a relaxation of the primal problem, if a feasible solution is possible for the primal problem, the dual subproblem should be feasible as well. In this check, the final period variables were fixed to the original problem's solution and in order to deal with the interconnection variables, the duplicated variables were fixed as well. Since all time periods are severed using the duplicated variables, these duplicated variables were fixed at the corresponding original model solution from the previous time period (e.g. the inventory of the duplicated variable for period *T-1* was fixed at the original problem's solution in period

T-1). By doing so, the previous time periods were not affected by the forced solution in the final time period; if the period T-1 variables were fixed, then the solution in period *T-1* would be affected. As a result of this check, a feasible solution was able to be found for all dual subproblem periods. Thus, there is an opportunity to explore future work in initialization strategies to find feasible solutions for all dual subproblem periods. It is unreasonable to initialize the dual subproblem with the full solution in practice, as the solution would not be known ahead of time.

The sub-gradient optimization used by Jackson and Grossmann [5], and Neiro and Pinto [7] to calculate the Lagrange multipliers requires feasible dual and primal solutions in the calculation of the step size. In spite of this method, the decision was made to use the infeasible solutions generated by the dual and primal problems in the multiplier step calculation in order to generate better initial guesses for the full problem, despite the infeasibility. The calculation of the multiplier step size is one method (along with multiplier adjustment methods and column generation techniques), as described by Fisher [8] and is the most widely used. Using the step size calculation closely matched to that of the aforementioned studies, the difference being the additional incorporation of the interconnected properties, as per equation 5.2.

Despite the infeasibility of the primal problem and parts of the dual problem, we explore the potential benefit of using the results obtained as an initial starting point to solve the problem to optimality. Table 5.9 lists the times required to run the Lagrangean decomposition algorithm 14 times for various model sizes. The times shown are cumulative - for example, the first iteration of the 5 period problem (full solve of both the dual and primal subproblems) completes in 14 seconds, but consecutive solves after the first iteration are 50% of that time. For smaller problems, it is faster due to the smaller model size and

fewer periods required to solve. It is noted that although the decomposition was run for 14 iterations, the number of iterations may be decreased based on the plateaus shown in figures 5.9 to 5.11.

Table 5.9: Cumulative time (seconds) required to run the decomposition algorithm for the multiperiod model for 5, 10, and 30 time periods. An iteration is defined as the combined solve of the dual and primal subproblem. For example, 1 solve of the decomposition algorithm for the 5 period model takes 14 seconds.

	Total CPU Time (s)		
Iteration	5 Period	10 Period	30 Period
1	14	27	84
2	20	39	125
3	26	53	176
4	33	69	232
5	39	84	279
6	44	99	325
7	50	113	364
8	56	127	403
9	61	141	444
10	67	155	482
11	73	169	521
12	79	182	559
13	84	196	596
14	90	209	634

After the decomposition phase, the problem is solved to the nominal operation point, as the decomposition did not yet solve to this point. IPOPT is used, as CONOPT fails in this phase. Table 5.10 lists the computation time to optimality

from the end of 14 Lagrangean decomposition iterations. The solution was obtained using tolerances of default (1e-8) and 10, with the exception of the 30 period model, as the 5 and 10 period models did not show a significant difference between the two tolerance options.

For the 5 period model, the solution from the end of the decomposition to optimality shows an improvement in computational time from the conventional method of solving (previously shown in Table 5.2) for the default tolerance, and no improvement from the tolerance of 10. However, when the decomposition times from Table 5.10 are factored in, the solution time shows no improvement over the conventional method of solving to the nominal case. Similar results show for the 10 period case with a tolerance of 10 - the combined decomposition time with the final solve phase result in a slower time. However, gains start to be made for the default tolerance with a 32 second reduction. The 30 period case using the decomposition time plus the solve time (from the end of the decomposition) reduce the solution time (from conventional solving) by 412 seconds.

These results show that for smaller models, such as the 5 and 10 period case, the solution time does not improve by using the end point of the Lagrangean decomposition as an initial point. This is due to the nonlinearity and nonconvexity of the problem where different initial guesses may or may not have an impact on the convergence to optimality. Combined with the time required to run the decomposition, it may not be an attractive option. However, for larger problems such as the 30 period model, the decomposition end point provides a benefit to convergence time and shows that it is a better initial guess than the conventional method.

Table 5.10: Computation time for solving from the end of the decomposition after 14 iterations to the nominal operation point for various sizes of the multiperiod model. IPOPT is used to solve the problem, and default tolerance of 1e-8 and 10 are used. Final profit deviation from nominal profit (from the original full space model) is also listed.

No. of Periods	Tolerance	CPU Time (s)	Deviation from Nominal Profit (%)
5	Default	65.984	0.457
5	10	65.008	0.457
10	Default	172.52	6.299
10	10	171.54	6.299
30	10	2090.88	1.072

It is noted that there is a deviation from the nominal profit. Ideally, the solution should be the same as the nominal profit (i.e. a deviation of 0). However, the end of the decomposition puts the problem at a different starting point than the conventional method. From this different starting point, due to the nonlinear and nonconvex nature of the problem, a different optimum is found where it differs from the nominal operation of the plant. For the 5 period problem, the new initial guess gives close to the same results as the conventional method of solving, but for the 10 period problem, the new initial starting point results in an optimum that is 6.3% from the solution of the conventional solve.

To further analyze the effect of the end point of the decomposition, the 5 and 10 period models were solved using various decomposition end points. That is, the decomposition was run for 1 to 8 iterations and the model was solved from those points. Table 5.11 lists the computational time required to solve the refinery model after running the decomposition algorithm a set number of times, the difference in solution times between the computational time plus the decomposition time (from table 5.9) versus the original solve time (from

table 5.2), and the deviation from the nominal profit. The difference in solution times is calculated as the original solution time minus the (decomposition + new initial point) solve time - therefore, positive numbers in the table represent improvements (i.e. using the new initial guess solves the model faster than the old initial guess), and negative numbers represent slower solve times (i.e. the conventional way/old initial guess is faster). It is noted that the solve time from the new initial point to optimality can be found by taking the 'Combined CPU Time' in table 5.11 and subtracting the corresponding value found in table 5.9 - for example, for 1 'iteration' of the decomposition, the combined CPU time is 35.469 seconds. The decomposition time (listed in table 5.9) is 14 seconds, and thus, the time to solve from the end of the decomposition to optimality is 21.469 seconds.

Table 5.11: Computation (CPU) time for solving the 5 and 10 period models from the end of the decomposition to the nominal operation point. CPU time shown is for the decomposition time *plus* the solve time from the end of the decomposition to optimality. Difference between this combined time and the solve time of the conventional direct solving method (from duplicated periods to nominal operation) is listed, and the percent deviation from the profit obtained in that problem. Dashes (-) denote a failure to converge.

No. of Decomp. Iterations	Combined CPU Time (s)	Δ Combined Time From Original (s)	Δ Nominal Profit (%)
5 Period Model			
1	35.469	70.5	0.509
2	45.644	60.4	0.510
3	74.358	31.6	0.462
4	78.884	27.1	0.462
5	116.339	-10.3	0.462
6	-	-	-
7	119.304	-13.3	0.394
8	108.735	-2.7	0.457
10 Period Model			
1	153.188	192.8	6.323
2	251.462	94.5	6.115
3	229.546	116.5	6.078
4	262.587	83.4	6.295
5	430.222	-84.2	6.244

For the 5 period model, there is a significant improvement compared to the full-space solution in starting from the end point after 1 to 4 iterations of the decomposition. Combined with the time required for decomposition, there is a 70.5 second reduction in computational time from the original computational

time (106 seconds from table 5.2) for using 1 iteration of the decomposition. Similarly for 2 iterations of decomposition, there is an improvement of 60.4 seconds from the original solve. There is a reduction up until 5 iterations of decomposition, where the total solution time becomes slower than the original CPU time. This indicates that the end point of the decomposition becomes a poorer initial guess as iterations of the algorithm increases due to longer times in solving from the end point to optimality. The longer times required to complete the decomposition phase (listed in table 5.9) also contribute to the increased CPU times. These results are also shown for the 10 period model, where 1 iteration of the decomposition provides an excellent starting point to provide more than a 3 minute improvement in solution time (this is compared to a solve time of 346 seconds). As well, the initial guess deteriorates significantly once the algorithm passes 4 iterations. Thus, in order to reap the benefits of the decomposition scheme, it is better to stop the algorithm early and use the early termination as a good starting point.

In order to further investigate the effect of using the end point of the decomposition as an initial guess, figures 5.12 to 5.14 show the evolution of objective function starting from the end of the decomposition phase up to optimality of nominal operation. The y-axis shows the normalized profit, relative to the final objective profit, which is calculated as:

$$\text{Profit}_i^{Normalized} = \frac{\text{Profit}_f - \text{Profit}_i}{\text{Profit}_f}$$

where Profit_f is the final profit value, and Profit_i is the profit at the current iteration i. Figure 5.12 shows the evolution in objective function for 1 iteration of the decomposition algorithm, while figures 5.13 and 5.14 shows the results for 4 and 7 iterations of the decomposition algorithm, respectively.

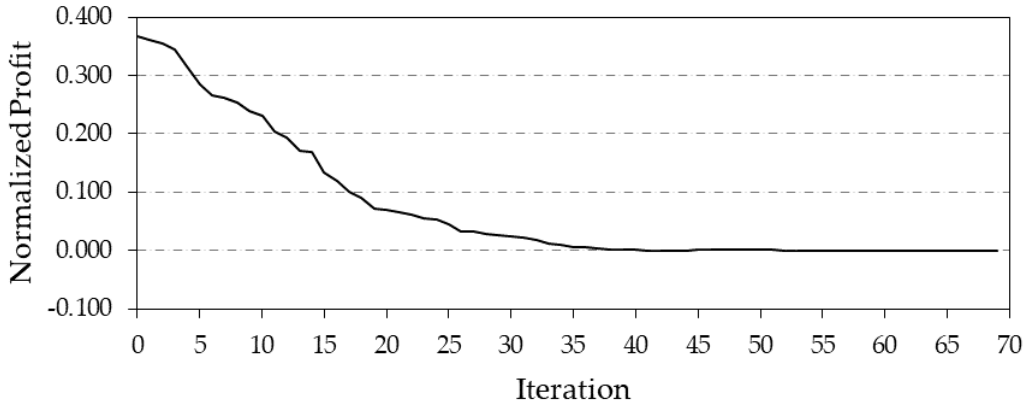

Figure 5.12: Objective value (profit) versus iteration number for solving the 5 period model using the end of the decomposition (after 1 run of the decomposition) as the initial guess to nominal operation optimality. Solved using IPOPT at the default tolerance. Profit is normalized based on the final profit value obtained from the optimization.

Figure 5.13: Objective value (profit) versus iteration number for solving the 5 period model using the end of the decomposition (after 4 runs of the decomposition) as the initial guess to nominal operation optimality. Solved using IPOPT at the default tolerance. Profit is normalized based on the final profit value obtained from the optimization.

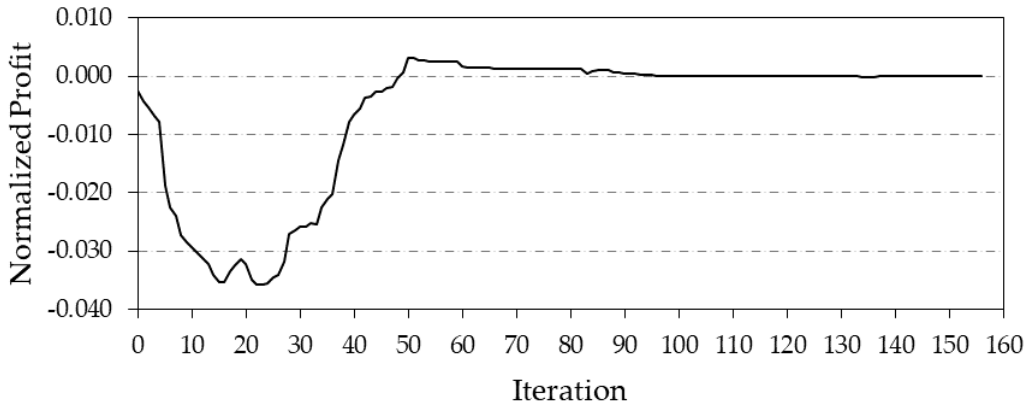

Figure 5.14: Objective value (profit) versus iteration number for solving the 5 period model using the end of the decomposition (after 7 runs of the decomposition) as the initial guess to nominal operation optimality. Solved using IPOPT at the default tolerance. Profit is normalized based on the final profit value obtained from the optimization.

The trend in figure 5.12 shows that the solution moves directly to the final objective, while figures 5.13 and 5.14 both start closer to the final objective, but move away from the optimum before finding the correct search direction to move closer to the optimum. This is an indication that although the initial objective is close to the solution, the constraints affect the search direction at that point and drive the search direction away from the optimum. It takes time for CONOPT to determine the next search direction, as CONOPT must identify the active-set constraints at every iteration and may elect to move to a worse objective while reducing constraint infeasibility. This is supported by figures 5.15 to 5.17, where the constraint violation evolution is shown on the y-axis, and the iteration number is shown on the x-axis for the 5 period model for 1, 4, and 7 runs/iterations of the decomposition algorithm.

Figure 5.15: Constraint violation versus iteration number for solving the 5 period model using the end of the decomposition (after 1 run of the decomposition) as the initial guess to nominal operation optimality. Solved using IPOPT at the default tolerance.

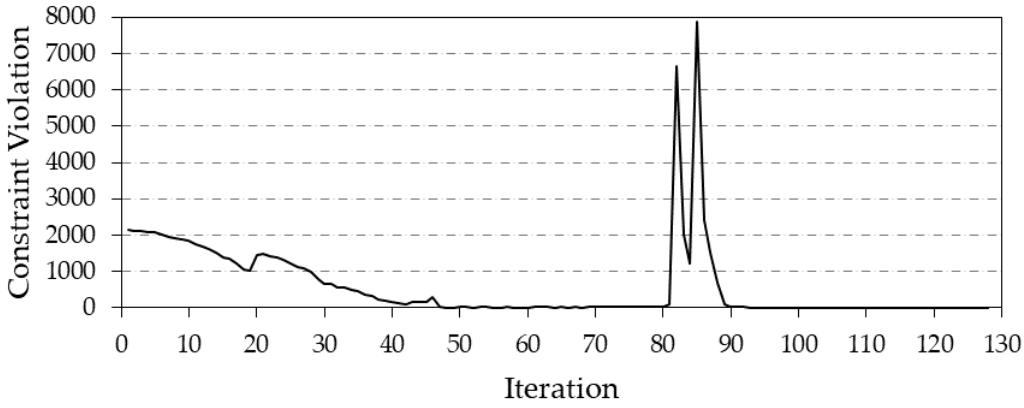

Figure 5.16: Constraint violation versus iteration number for solving the 5 period model using the end of the decomposition (after 4 runs of the decomposition) as the initial guess to nominal operation optimality. Solved using IPOPT at the default tolerance.

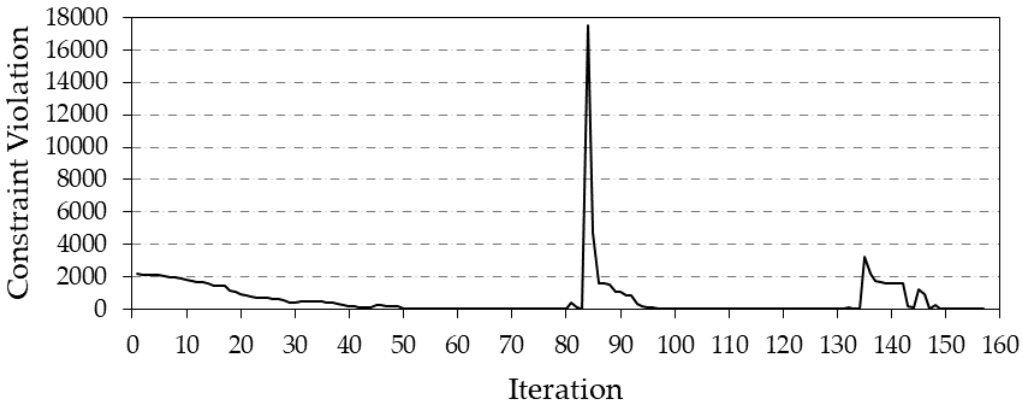

Figure 5.17: Constraint violation versus iteration number for solving the 5 period model using the end of the decomposition (after 7 runs of the decomposition) as the initial guess to nominal operation optimality. Solved using IPOPT at the default tolerance.

Figures 5.15, 5.16, and 5.17 show the constraint violation for the 5 period model for 1, 4, and 7 runs/iterations of the decomposition algorithm. This supports the claim that for the 4 and 7 iteration cases, the objective function moves away from the optimum because CONOPT chooses to reduce the constraint violation before improving the objective.

Additionally, figure 5.18 shows the difference between the optimum and the final objective at iteration 0 for the 5 period model for varying runs/iterations of the decomposition algorithm. The x-axis shows the number of times that the decomposition algorithm was run, while the y-axis shows the normalized profit at iteration 0, relative to the final objective value. Figure 5.18 shows that running the decomposition for more iterations moves the initial starting point closer to the optimal value. Again, the starting objective value cannot be used as an indication of how fast the solver will find the optimum; rather, it also depends on the constraint infeasibility and the search direction that the solver may decide to go. As well, running the decomposition longer may not provide a better starting point, and for this model in particular, running the

decomposition longer actually provides worse starting points.

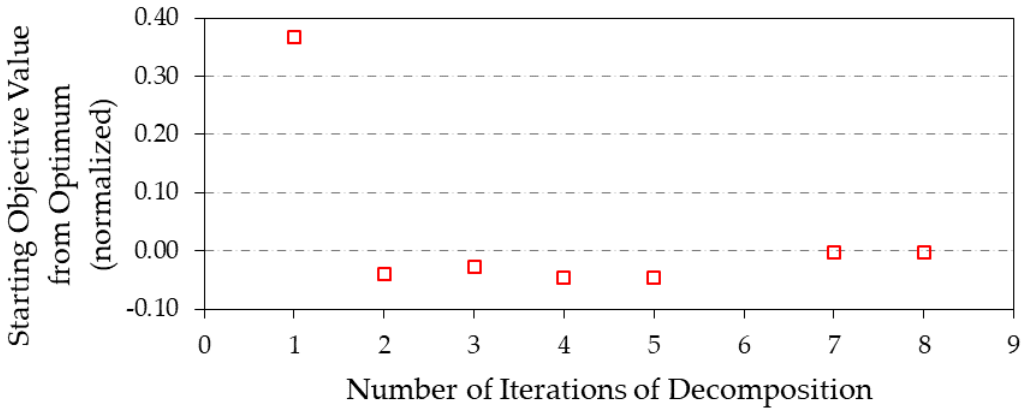

Figure 5.18: Starting objective values (from iteration 0) versus the number of decomposition iterations/runs for the 5 period model. The objective values are from iteration 0 using the end of the decomposition as the initial guess. The objective value (profit) is normalized based on the final profit value at optimality (of nominal operation). Using 6 runs of decomposition as the initial guess does not allow IPOPT to converge.

In summary, the Lagrangean decomposition scheme was implemented for the petroleum refinery multiperiod model. The inclusion of tank properties in the decomposition scheme increased the complexity of the algorithm, as the algorithm has previously been demonstrated for models with only the inventory as the interconnection constraint. As well, the nonlinearity and nonconvexity of the model proves difficulty in finding feasible solutions for the primal problem and part of the dual problem. Nonetheless, the end point of the decomposition may be used as an initial guess to solve the original model to optimality (to nominal operation). Varying the number of decomposition iterations can provide better initial guesses than running the decomposition until no improvement is made.

There are various avenues that may be explored in order to achieve feasible results or decrease computational times. A better initialization strategy is required in order to potentially allow feasible solutions and is an area of exploration that would be beneficial to study. The possibility of utilizing

parallelization may also be suggested for future work. As each time period is solved independently, it may be proposed to parallelize the problem. As it stands, each time period is being solved consecutively and although each period may only take one to two seconds to solve, these times start to add up in larger problems, such as with 30 periods. Since each period is independent of the others, then it may be possible to solve all time periods simultaneously using multiple cores/threads in processors and save more time during the decomposition phase. Potential future work could also include an improved initialization scheme for both the dual and primal subproblems. Currently, the initial point passed onto the dual problem is not completely feasible, and once the solution of the dual is passed onto the primal problem, it is infeasible.

References

[1] A. Wachter and L. T. Biegler. "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming". In: *Mathematical programming* 106.1 (2006), pp. 25–57.

[2] V. Gopal and L. T. Biegler. "Smoothing methods for complementarity problems in process engineering". In: *AIChE journal* 45.7 (1999), pp. 1535–1547.

[3] B. Baumrucker, J. Renfro, and L. T. Biegler. "MPEC problem formulations and solution strategies with chemical engineering applications". In: *Computers & Chemical Engineering* 32.12 (2008), pp. 2903–2913.

[4] A. Drud. *GAMS Documentation - CONOPT*. 25.1.1. ARKI Consulting and Development AS. Bagsvaerd, Denmark, May 2018. URL: `https://www.gams.com/latest/docs/S_CONOPT.html`.

[5] J. R. Jackson and I. E. Grossmann. "Temporal decomposition scheme for nonlinear multisite production planning and distribution models". In: *Industrial & engineering chemistry research* 42.13 (2003), pp. 3045–3055.

[6] M. Escobar, J. O. Trierweiler, and I. E. Grossmann. "A heuristic Lagrangean approach for the synthesis of multiperiod heat exchanger networks". In: *Applied Thermal Engineering* 63.1 (2014), pp. 177–191.

[7] S. M. Neiro and J. M. Pinto. "Langrangean decomposition applied to multiperiod planning of petroleum refineries under uncertainty". In: *Latin American applied research* 36.4 (2006), pp. 213–220.

[8] M. L. Fisher. "The Lagrangian relaxation method for solving integer programming problems". In: *Management science* 50.12_supplement (2004), pp. 1861–1871.

Chapter 6

Conclusions & Recommendations

6.1 Key Findings and Contributions

The development and applications of a multiperiod petroleum refinery optimization model were described and implemented in this work. Various aspects of the model were explored using termination criteria, computation time, as well as other metrics as the basis for comparison. Several studies illustrated model behaviour and included comparison of the method utilized by TOTAL to solve the model versus IPOPT/CONOPT for various case studies that deviate from normal operation of the refinery; testing the performance of IPOPT/CONOPT for various multiperiod model sizes; and comparing solution strategies for solving the model. The implementation of Lagrangean decomposition was a major component of the work as well; exploring the effect of the decomposition on the initial guess for the solution of the full space problem.

The key contributions and findings of this work are as follows:

- When solving the case studies (deviation from nominal operation), using IPOPT and CONOPT both have significant advantages in convergence time compared to the method utilized by TOTAL in solving their model. Although the models are not exactly the same (due to the TOTAL model having four additional tanks), both models behave in a similar fashion and all objective function values and variables solve within the same range and magnitude. Thus, the results provide context and estimation of the solution times.

- The single period model solves to the same points as the TOTAL model within a maximum of 0.2% of objective function difference for 13 different test cases and verifies that there is little mismatch between the two

models and can be extended to the multiperiod case for context.

- The computation times increase nonlinearly and significantly when the model is expanded from 5 periods up to 10 and 30 periods for solving both to the nominal operation point *and* to the case studies. Both IPOPT and CONOPT solve to the same objective value, which shows consistency between the solvers.

- The convergence tolerance set for IPOPT may be relaxed, as the larger tolerance does not affect the solution when solving from the initial point to the nominal operation point, but does provide a significantly faster convergence time for models larger than 5 periods.

- The Lagrangean decomposition was not able to find feasible solutions, as the inclusion of tank properties in the decomposition scheme increased the complexity of the algorithm and the nonlinearity and nonconvexity of the model proved difficult in finding feasible solutions.

- The end point of the decomposition proved to be an acceptable initial guess for solving the full space model to optimality at nominal operation, depending on the how long the decomposition was run for. It was found that a faster solution was found using a low number of decomposition iterations. This is indicative that the current decomposition model may drive the optimization further from the optimum, mainly due to the constraints. For longer decomposition runs, the objective function moves closer to the optimum but due to the infeasibilities in the primal and dual subproblems, the decomposition may be further violating the constraints.

6.2 Recommendations for Future Work

Several areas that may be considered for further exploration are identified. They are:

- Possibility of utilizing parallelization may be suggested for future work. As it stands, each time period is being solved consecutively and although each period may only take one to two seconds to solve, these times start to add up in larger problems, such as the 30 period model. By being able to parallelize the problem, all time periods could be solve simultaneously and thus, saving time spent in the decomposition phase.

- Improved initialization scheme for both the dual and primal subproblems in the decomposition. Currently, the initial point passed onto the dual problem is not completely feasible, and once the solution of the dual is passed onto the primal problem, it is also infeasible. An initialization strategy may provide benefit to finding feasible solutions for both the primal and dual subproblems and should be applied to the general case to allow flexibility in modifying the model (e.g. number of periods).

- Incorporation of additional intermediate product tanks. In the current model, only four intermediate tanks are included in the model, while the TOTAL model includes eight tanks - this was due to consideration of time, as the study was to study the multiperiod problem and not all tanks were required to model the problem.

- Formulation of uncertainty in the multiperiod model. There are various elements in the refinery model that may be explored in regards to uncertainty. These include supply uncertainty, fluctuations in feedstock and

product prices, and fluctuations in pricing of utilities (natural gas).

- Different configurations and further case studies. The same model was used throughout this work for both the single period and multiperiod studies. Various configurations of the refinery may be studied, such as inclusion of units that did not appear in this work. Additionally, other case studies such as catalyst scheduling and crude scheduling may be of interest. In catalyst scheduling, it may be pertinent to study the activity of the catalyst and provide conditions on when to shutdown the unit and replace the catalyst. In crude scheduling, it would be of interest to determine the optimal crude slate based on properties of the crude (e.g. sulfur content).

- Inclusion of integers and binary variables in the model. At the present moment, the model is a continuous NLP. With the implementation of binary and integer variables, other studies such as unit start-up and shutdown may be studied, as well as changing the operating mode of a unit (e.g. deciding on catalyst type or switching operating conditions).

- Further exploration of solver options, in both IPOPT and CONOPT - for example, a different strategy may be used to calculate the barrier parameter. There are two methods to calculate the barrier parameter: the monotone (Fiacco-McCormick) strategy (default), or an adaptive update strategy. These are further explained in the IPOPT documentation.