NTNU – Trondheim
Norwegian University of
Science and Technology

Modelling and Simulation of a Two-Stage Refrigeration Cycle

Adriaen Verheyleweghen

Chemical Engineering and Biotechnology
Submission date: June 2015
Supervisor: Johannes Jäschke, IKP
Co-supervisor: Skogestad Sigurd, IKP

Norwegian University of Science and Technology
Department of Chemical Engineering

Preface

This thesis was written as the final work towards my degree of M.Sc. in Chemical Engineering at the Norwegian University of Science and Technology.

I would like to thank Johannes Jäschke for his invaluable advice and guidance throughout this project. Our many discussions have been very educational. He has always been able to provide useful feedback when my results were not as expected, and I am very grateful for that.

I would also like to thank my friends and fellow students for making these past five years fly by. Studying at NTNU would have been very boring without you!

Finally I would like to thank my family for their continued support. This is a big step in my life, and even though the path to get here has been hard, you have always encouraged me.

Declaration of Compliance

I hereby declare that this thesis is an independent work in agreement with the exam rules and regulations of the Norwegian University of Science and Technology

Trondheim, June 17, 2015

[illegible]

ABSTRACT

A two-stage refrigeration cycle was modelled and optimized in MATLAB. The optimum was found to be very flat, resulting in small losses from disturbances and implementation errors. The two unconstrained degrees of freedom were used to implement self-optimizing controllers. A subset of five measurements was used for the self-optimizing controller since this gave reasonably small losses. The controllers assured optimal steady-state operation of the refrigeration cycle even when disturbed. Studies of the dynamic responses of the closed-loop system showed relatively large initial deviations from the optimum caused by large time constants for the measurements. An alternative process model with constant temperature differences between the evaporator and the process stream was also investigated. The model was used to show the feasibility of including cost data in the measurements of the self-optimizing controller. It was found that the resulting controllers were able to keep the operation of the refrigeration cycle optimal despite fluctuations in the prices. In both the original and the alternative case it was found that the open-loop responses with constant inputs were almost as good as the closed-loop responses of the self-optimizing controllers. Control is thus not strictly necessary, and a constant input policy may give acceptable losses.

SAMMENDRAG

En to-trinns kjølesyklus har blitt modellert og optimalisert i MATLAB. Grunnet et svært flatt optimum er tapene fra forstyrrelser og implementeringsfeil små. Etter optimalisering gjenstår to frihetsgrader. Disse ble brukt til å implementere selvregulerende kontrollere med fem prosessmålinger som sørger for at syklusen opererer optimalt til tross for forstyrrelser. Den dynamiske responsen til systemet viser relativt store umiddelbare tap. Disse tapene skyldes de store tidskonstantene til målingene. En alternativ prosessmodel har også blitt studert. I den alternative modellen antas det at temperaturen mellom evaporatoren og prosesstrømmen holdes konstant. Det ble studert hvorvidt en selvoptimaliserende regulator kan brukes til å holde systemet optimalt gitt prisendringer. Det ble funnet at en slik regulator fungerer godt. Både den opprinnelige modellen og den alternative modellen har såpass flate optimum at det oppnåes akseptable tap ved å holde pådragene konstante.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

Symbol	Unit	Description
γ_{avg}	-	Adiabatic ratios. Average ratio between specific heats
δ_i		Coefficients for calculating *HFL* as a function of temperature
ζ_i		Coefficients for calculating *HFG* as a function of temperature
η	-	Polytropic efficiency of compressor
θ	s	Time delay
λ	-	Lagrange multiplier
λ_i		Coefficients for calculating ν as a function of temperature
$\mathbf{\Lambda}$		Relative gain array
ν	m^3/kg	Liquid specific volume
τ	s	Time constant
ϕ_i		Coefficients for calculating z as a function of temperature
A		Coefficient in the Antoine equation
B		Coefficient in the Antoine equation
C		Coefficient in the Antoine equation
$\mathbf{c}$		Controlled variables

C_i		Coefficients for calculating CP as a function of temperature
C_{ii}		Coefficients for compressor curves
CP	J/kgK	Heat capacity
CV	kg/s $\sqrt{\text{bar}}$	Valve constant
$\mathbf{d}$		Disturbances
e_i		Coefficients for compressor curves
$\mathbf{F}$		Sensitivity matrix
FCP	W/K	Product of flowrate and heat capacity of external cooling stream
FG	kg/s	Mass flow of vapour stream
FGCD	kg/s	Discharge flow rate of vapour out of the compressor
FGCS	kg/s	Suction flow rate of vapour in to the compressor
FGV	m^3/s	Volumetric flow rate of vapour into compressor
FL	kg/s	Mass flow of liquid stream
g	m/s^2	Gravitational constant
$\mathbf{G}$		Linearized process model from **u** to **y**
$\mathbf{G_d}$		Linearized process model from **d** to **y**
$\mathbf{G^y}$		Linearized process model from **u** to **c**
$\mathbf{G^y_d}$		Linearized process model from **d** to **c**
h	m	Scaled compressor head
hs	m	Compressor head
$\mathbf{H}$	-	Selection matrix
HFL	J/kg	Specific enthalpy of the liquid stream
HFG	J/kg	Specific enthalpy of the vapour stream
J	\$	Cost function
k	-	Polytropic constant
L	\$	Loss from optimal J
M_w	kg/mol	Molecular weight
N	-	Fractional compressor speed based on normal operation
p	-	Constant in compressor curve
p	bar	Pressure in vessel
PS	bar	Suction pressure

q	-	Constant in compressor curve
Q	W	Heat transferred in evaporator or condenser
R	J/K mol	Universal gas constant
T	K	Temperature in vessel
TD	K	Discharge temperature from compressor
TPI	K	Inlet temperature of process stream to evaporator or condenser
TPO	K	Outlet temperature of process stream from evaporator or condenser
TS	K	Interstage mixing temperature / suction temperature into the compressor
$\mathbf{u}$		Inputs
UA	W/K	Product of heat transfer coefficient and heat transfer area for a heat exchanger
V	m^3	Volume of the tank
VG	m^3/kg	Specific volume of vapour refrigerant
W	kg	Mass accumulated in tank
$\mathbf{W_d}$		Magnitudes of disturbances
$\mathbf{W_{n^y}}$		Magnitudes of measurement errors
W_s	W	Shaft work
WL	kg	Liquid inventory in tank
WLV	m^3	Volumetric liquid inventory in tank
WV	kg	Vapour inventory in tank
XV	-	Fractional valve opening of valve, also used to refer to the valve itself
$\mathbf{y}$		State variables / Outputs
z	-	Compressibility factor

List of Abbreviations

Abbreviation	Explanation
COP	Coefficient of performance
CV	Controlled variable
DAE	Differential algebraic equation
DOF	Degree of freedom
HP	High pressure
IP	Intermediate pressure
LP	Low pressure
MIMO	Multiple inputs, multiple outputs
MV	Manipulated variable
PID	Partial integral differential
RGA	Relative gain array
SIMC	Skogestad internal model control
SS	Steady-state

CHAPTER 1

INTRODUCTION

This report is the final product of the master thesis on 'Modelling and Optimization of a Two-Stage Refrigeration Cycle'. The thesis was written by Adriaen Verheyleweghen under supervision of Johannes Jäschke and co-supervision of Sigurd Skogestad.

1.1 Scope

The aim of this project is to model and optimize a two-stage refrigeration cycle. The principles of self-optimizing control will be used to derive a controller which keeps the plant operating optimally at all times. The methods described by Halvorsen et al. (2003), Alstad & Skogestad (2007), Alstad et al. (2009) will be used to derive the controllers in this thesis. Optimal selection of the controlled variables and degree of freedom analysis will be discussed.

Since self-optimizing control theory is valid for steady-state only, this will be the main focus of the thesis. The dynamic properties of the controllers will not be investigated in detail. The responses of the controllers will be looked at in order to ensure that the controllers are feasible. The optimal pairings for the MIMO system are found by applying the RGA to the system. The stability of the derived PI controllers will be ensured by applying a somewhat conservative SIMC tuning method to find the controller settings. This is the extent to which the dynamic properties of the system are investigated.

1.2 Previous work

This thesis is a continuation of a project work on the same subject (Verheyleweghen 2014). The model was originally written by Basel Asmar Asmar (1991) in ACSL, and was implemented by the author in MATLAB as part of the project (Verheyleweghen 2014). First attempts at optimizing the plant was also conducted as part of the project. However, the criterion for optimization were badly defined in the previous work, so the results from the optimization are of no use for the current thesis. It was previously found that no degrees of freedom were available for optimization, but this was because the objective of the optimization was set to be the minimization of the energy consumption. In the current thesis, a more realistic economic trade-off will be used as the objective. It will also be explored how different parameters for the optimization criterion effect the optimal solution.

It was also discovered that the original model by Asmar, and consequently the project by Verheyleweghen, contained some modelling errors. The model used in the current thesis is corrected. In which ways the current model differs from the previous model is covered in Section 3.1.2.

1.3 Structure of the thesis

The structure of the thesis is defined somewhat loosely with readability in mind. The first chapter contains the background theory on refrigeration cycles and process control and optimization. The findings from the modelling work are summarized in Chapter 3. The chapter is divided into three parts; the first part discussing the steady-state optimal solution, the second part discussing the performance of the derived controllers and the final part discussing an alternative process layout. Finally, a conclusion to the work is given in Section 4. Additional material such as MATLAB code is placed in the Appendix.

CHAPTER 2

THEORY

This chapter contains all the necessary background theory for this thesis. At first a general introduction to refrigeration cycles will be given. Control- and optimization theory follows.

2.1 Refrigeration cycles

This section aims to give a short introduction to refrigeration cycles. Fundamentals will be covered, as well as basic terminology. For an extensive coverage of refrigeration cycles it is referred to the literature. Granryd (2009) provides a broad coverage of the subject and is recommended if more theory is desired.

A refrigeration cycle is a process which, as the name suggests, is used to refrigerate a process stream. Refrigeration cycles and heat pumps are related processes with opposite directionalities of the energy flows, collectively referred to as vapour compression cycles. The main idea behind this type of process is to take heat from one reservoir and transfer it to another reservoir. The refrigerant is chosen such that it undergoes a phase change when heat is transferred to or from the system. Consequently the pressure can be manipulated to adjust the temperature of the working fluid, thus controlling the sign of the energy flow. Figure 2.1 shows an illustration of a basic refrigeration cycle.

The points in Figure 2.1 correspond to the four main states in a basic one-stage refrigeration cycle. A common tool for visualising vapour compression cycles is the pressure-enthalpy (p-h) diagram. It shows the phase diagram of the working fluid as a function of enthalpy and pressure. A p-h diagram cor-

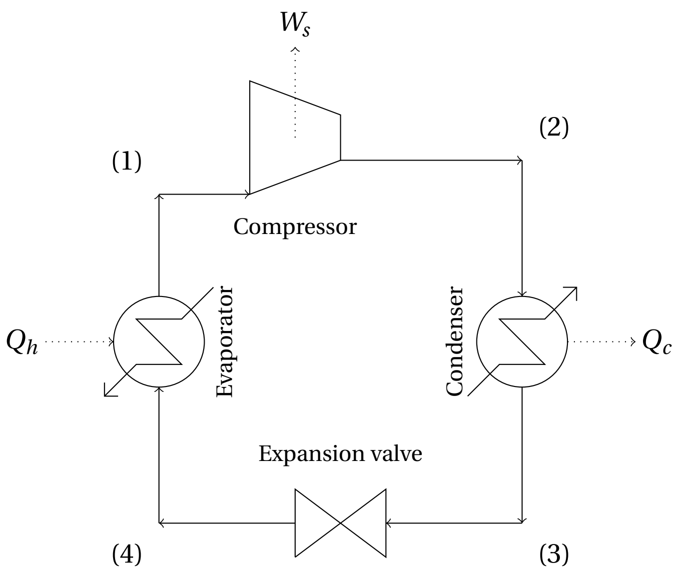


Figure 2.1: Illustration of a basic refrigeration cycle

responding to the refrigeration cycle shown in Figure 2.1 can be seen in Figure 2.2.

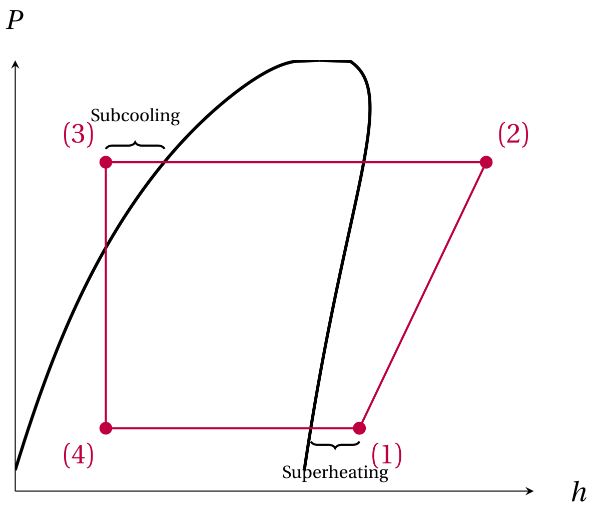


Figure 2.2: Pressure-Enthalpy diagram for a basic refrigeration cycle

2.1.1 Description

The four steps of a basic refrigeration cycle are explained here.

Compression

In the compression stage ($1 \rightarrow 2$), the pressure of the gaseous refrigerant is increased. The compression can either be isentropic or polytropic. Isentropic compression means that the entropy of the working fluid stays constant during the compression, i.e. that there is no heat loss to the surroundings since $dS = \frac{dQ^{\text{rev}}}{T} = 0$ (Skogestad 2011). This is only true if the process is reversible. Since all real processes have heat loss and thus are irreversible, isentropic compression is only used to calculate the ideal process. From thermodynamics it follows that

$$\frac{T_f}{T_i} = \left(\frac{p_f}{p_i}\right)^{\frac{\gamma-1}{\gamma}} \tag{2.1}$$

where γ is defined as $\gamma = C_p/C_v$ and the subscripts i and f refer to before and after compression, respectively. The reversible work can be written as: (Skogestad 2011)

$$W^{\text{rev}} = \frac{\gamma p_f V_f}{1-\gamma}\left(\left(\frac{p_f}{V_f}\right)^{\frac{\gamma-1}{\gamma}} - 1\right) \tag{2.2}$$

Polytropic compression, on the other hand, assumes that $p_i V_i^n = p_f V_f^n$, i.e. that γ in Equation 2.1 can be replaced by a coefficient n that satisfies $1 \leq n \leq \gamma$. This leads to a decrease in entropy, which is caused by heat loss to the surroundings. It follows from Equation 2.2 that the polytropic compression work always is smaller than the isentropic compression work. The extreme case where $n = 1$ is called isothermal compression.

Assuming that there is no cooling, the real compression work is always larger than the reversible compression work due to energy losses from friction, windage etcetera. The ratio between the real work and the reversible work is given by the efficiency η. η is typically between 0.6 and 0.8.Skogestad (2011)

Figure 2.3 shows the p-h diagram from Figure 2.2, focusing on the compression stage. The compression from state 1 to state 2B is isentropic. The working fluid undergoes polytropic compression when taking the path from state 1 to state 2A. The final temperature is lower than for isentropic compression, since heat has been lost to the environment. The compression from state 1 to state 2C shows a compression more akin to what is observed in a real compressor without external cooling.

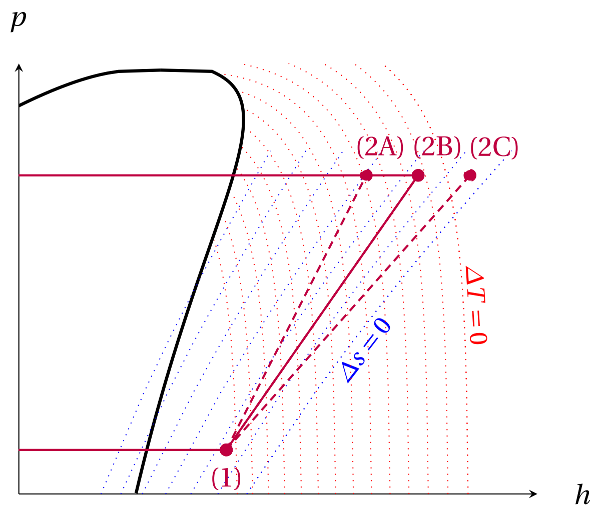


Figure 2.3: Excerpt of the pressure-enthalpy diagram for a basic refrigeration cycle, showcasing the differences between isentropic, polytropic and real compression work.

Condensation

After the compression, the gas is condensed isobarically to liquid in a condenser ($2 \longrightarrow 3$). The pressure stays constant during the condensation. Air or water are usually used as a coolants in the condenser, but special refrigeration cycles might require a coolant at a lower temperature. In a real process there might be some pressure drop over the condenser, but this is usually neglected as a first assumption.

Expansion

The refrigerant undergoes isenthalpic expansion over a Joule-Thomson expansion valve ($3 \longrightarrow 4$). The pressure decreases, causing the liquid to move into the two-phase-region of the phase diagram. The working fluid is at its coldest temperature after expansion. In theory one can use a turbine to extract work at this stage. In practice this design is not common due to the difficulties associated with the expansion of the liquid outlet from the condenser Granryd (2009).

Evaporation

The final step of the refrigeration cycle is the evaporation of the working fluid to return back to its gaseous state. The heat of evaporation is taken from the hot reservoir which is to be refrigerated.

2.1.2 Subcooling and superheating

As can be seen from Figure 2.2, point 1 can be placed to the right of the phase boundary. This is known as superheating. Superheating is a practical necessity to prevent liquid from entering the compressor. This phenomenon is known as slugging, and can cause serious damage to the compressor (Prasad 2002). The disadvantage of superheating the working fluid is that the entire cycle is moved to the right in the phase diagram. This pushes the compressor towards a region with flatter isentropes, leading to a larger energy consumption. The increased energy consumption in the compressor must be balanced by an increase in the heat transfer area in the evaporators. Superheating should thus be kept to a minimum.

The opposite is true for subcooling. Subcooling is used to prevent cavitation in the expansion valves. Cavitation can be equally destructive to valves as slugging is to compressors, and is therefore sought to be eliminated. Superheating and subcooling needs can be covered through internal heat exchanging. Heat flows from the subcooled liquid to the superheated vapour, resulting in zero net energy consumption if the amount of subcooling and superheating is balanced. Jensen also showed that subcooling may can be used to optimize the operation of refrigeration cycles (Jensen & Skogestad 2007). Subcooling reduces the thermodynamic loss from the isenthalpic expansion due to the formation of less vapour.

2.1.3 Performance of refrigeration cycles

The performance of refrigeration cycles can be quantified by the coefficient of performance (COP). The coefficient of performance is defined as

$$\mathrm{COP}_{\mathrm{refrigeration}} = \frac{Q_c}{W_s} = \frac{h_1 - h_4}{h_2 - h_1} \tag{2.3}$$

The COP is thus the ratio between the amount of heat transferred to the process and the work supplied to the process. The COP varies depending on the process conditions, but is typically larger than one.

2.1.4 Multi-stage compression cycles

If the temperature difference between the condenser and the evaporator is very large, it becomes attractive to consider a refrigeration cycle with two or more compression stages. This is to avoid large pressure ratios in the single-stage compression system, which cause undesired operating conditions. Typically the pressure ratio of a piston-driven compressor should not exceed 8-10 (Granryd 2009). Especially for real compressors with efficiencies lower than one, having a too large pressure ratio leads to a large energy loss when compared to the reversible process. A solution to this problem is to compress the vapour in multiple stages with interstage cooling. Interstage cooling is used to reduce the superheating as much as possible. This is advantageous because flat isentropes are avoided, thus increasing the efficiency and lowering the overall energy consumption of the compressors.

Figure 2.4 shows an illustration of a two-stage compressor train with interstage cooling. Figure 2.5 shows the corresponding p-h diagram. The dashed line shows a single-stage compression for comparison. The pressure ratios of the two compressors are equal, with the intermediate pressure being $p_{\text{intermediate}} = \sqrt{p_2 \cdot p_1}$. The inlet temperature to both compressors is equal to the saturation temperature of the refrigerant, i.e. the superheating is zero for both compressors. It can be seen that due to the somewhat steeper isentropes, the energy consumption of the second compressor is marginally lower than the first compressor, even though the pressure ratios are the same in both compressors. It has been assumed that the two compressors otherwise are identical. Due to the smaller pressure ratios, the total energy consumption of the two-stage compressor train is smaller than for the single-stage compressor.

Figure 2.4 shows the possibility of internal heat exchange as indicated by the dotted line between the evaporator and the interstage cooler. Internal heat exchange might be advantageous in some cases (Jensen & Skogestad 2007).

The two-stage design can be improved even further by realizing that vapour is produced in the expansion valve. This vapour can not contribute to cooling, so the expansion and subsequent compression of this vapour fraction is a waste of energy. By letting the expansion occur in a flash drum so that the vapour can be collected and fed directly between the two compressors, this can be avoided. The saturated vapour also acts as cooling for the superheated vapour exiting the first compressor, so the superheating is reduced before entering the second compressor. This method is known as two-stage throttling (Granryd 2009). An illustration of the described refrigeration cycle can be seen in Figure 2.6. The p-h diagram is shown in Figure 2.7.

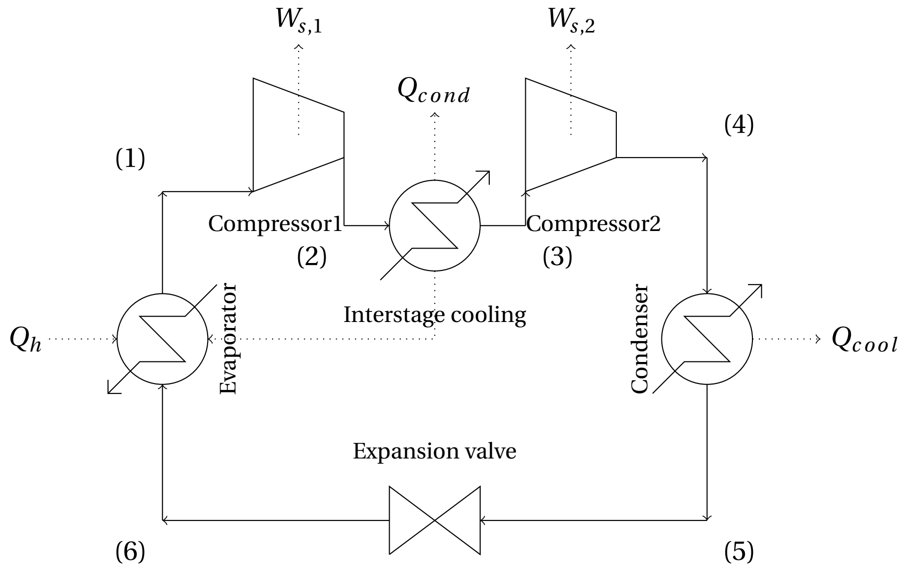


Figure 2.4: Illustration of a refrigeration cycle with a two-stage compressor train with interstage cooling

This design is a variation of the internal heat exchange discussed earlier. The refrigerant discharged from the first compressor does not have to be fed to the flash tank, however. Alternatively the vapour from the flash tank is mixed with the discharge from the first compressor and fed directly to the second compressor. The advantage of using a bubble design is that the superheat is kept to a minimum, at the expense of cooling capacity in the evaporators.

If the cooling load is distributed on different temperature levels, heat can be taken out on the intermediate pressure level as well. This can be achieved by installing a heat exchanger in the flash drum, for example. The p-h diagram in Figure 2.7 will remain unchanged, but the fraction of vapour will increase. This means that less liquid is available for cooling in the low temperature evaporator. The distribution of refrigerant between the two evaporators must be determined based on the loads on each temperature level.

2.1.5 Degrees of freedom in refrigeration cycles

For a simple refrigeration like the one shown in Figure 2.1, Jensen & Skogestad (2007, 2009) discuss the design specifications and operational degrees of freedom (DOF). The number of design specifications correspond to the number of variables which can be chosen freely by the operator. When all design specifi-

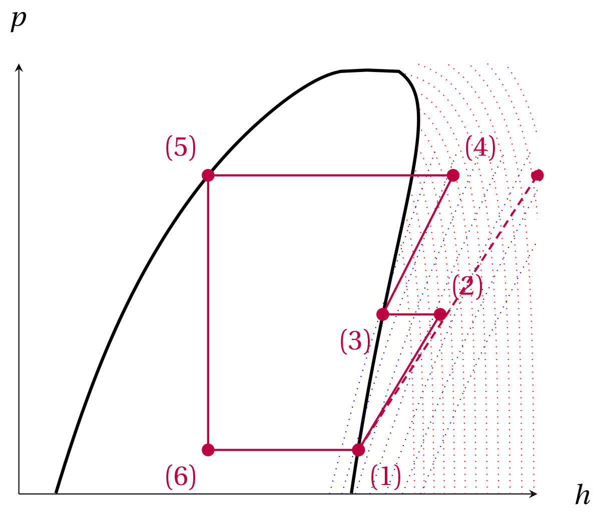


Figure 2.5: Pressure-Enthalpy diagram for the process shown in Figure 2.4. The dashed line shows a single-stage compression for comparison.

cations are set, they map to a unique state solution. To ensure that all design specifications are met, it is necessary to have an equal number of degrees of freedom that can be adjusted to keep the design specifications satisfied.

Jensen & Skogestad (2007, 2009) found that the design specifications for a refrigeration cycle include the heat load, the two pressures, the degree of subcooling and the degree of superheating. These five specification can be met by adjusting the compression work, the heat transfer in the evaporator and the condenser, the valve opening of the expansion valve and the active charge. Active charge is defined as the total mass of refrigerant that is present in the system, excluding storage (receiver) tanks. The active charge can be manipulated by introducing a receiver and an additional control valve to the cycle (Jensen & Skogestad 2007). Introducing additional tanks to the cycle requires all but one of the tanks to have level control in order to avoid fully filling or draining the tank (Aske & Skogestad 2009).

Jensen devised a simple method for determining the potential number of steady-state degrees of freedom. Each type of process equipment adds or removes potential degrees of freedom. The method is summarized in Table 2.1, which is taken from Jensen & Skogestad (2009)

The potential DOF attributed to the heat exchangers comes from the possibility to adjust the heat transfer. This can be done by introducing a bypass or

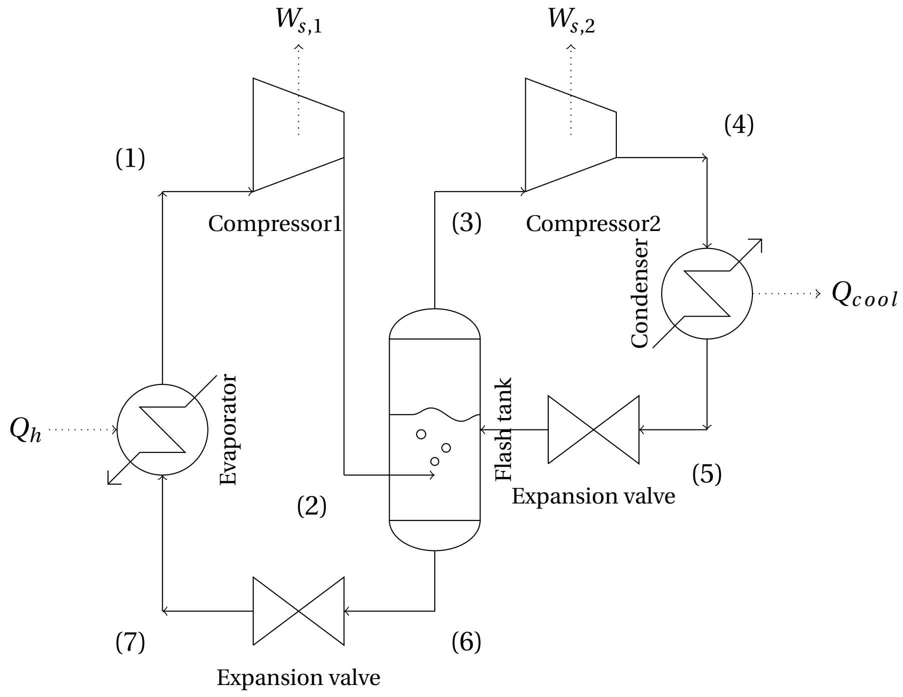


Figure 2.6: Illustration of a refrigeration cycle with a two-stage compressor train with two-stage throttling

by adjusting the flow rate of one of the streams into the heat exchanger. It is common to maximize the heat transfer in the evaporator. This removes a potential DOF. From an economic point of view, it makes sense to utilize the full heat transfer potential at all times, otherwise money could have been saved by investing in a smaller heat exchanger.

Potential DOF are the maximum possible degrees of freedom for a combination of process units. The actual DOF are not necessarily equal to the potential degrees of freedom. For example, the active charge of a cycle might not be adjustable since there is no receiver tank in the system. The amount of refrigerant is consequently fixed. The actual DOF can be found using the so-called valve-counting method. It can be summarized as follows: (Skogestad 2000)

1. Count MVs

2. Subtract MVs with no steady-state effect (such as additional liquid receivers in closed systems)

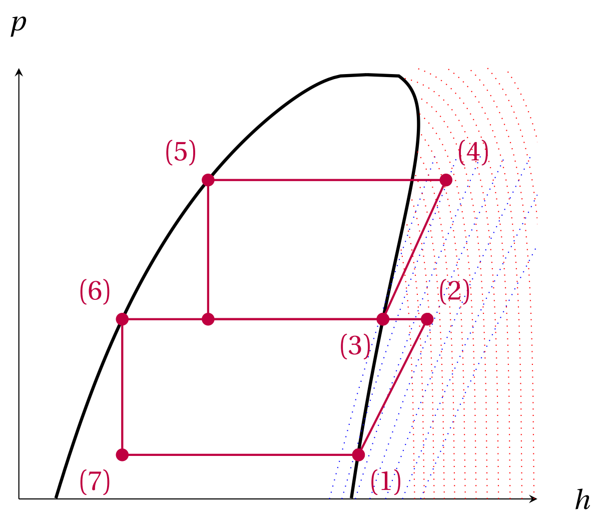


Figure 2.7: Pressure-Enthalpy diagram for the two-stage throttling process shown in Figure 2.6

For multicomponent refrigerants, the first $n_c - 1$ receivers have a steady-state effect. n_c signifies the number of unique components in the working fluid.

2.2 Steady-state optimization

This section contains the theoretical background that is necessary to study the behaviour of the process. It will mainly be focused on optimal operation of the plant at steady-state conditions. Dynamics will be covered in a later section.

2.2.1 Top-down procedure for control structure design

Designing a control structure for an entire plant rather than for a single unit can be a tedious task. There are often (seemingly) conflicting interests such as optimizing plant economics, having good controllability and achieving robustness. Another issue is finding out *what* to control, as the choice of MVs and CVs is seldom obvious. Even another issue is the linking of different control layers and time scales. The control system is often split into different layers which are operating on different time scales. Skogestad (2004) proposes the following control layers, with a rough approximation of the corresponding time scales given in parentheses

- Scheduling (weeks)

Table 2.1: Potential steady-state DOF for a refrigeration cycle. Table taken from Jensen & Skogestad (2009)

Process unit	Potential DOF
Feed	1
Splitter	$n_{\text{streams}} - 1$
Mixer	0
Compressor / Turbine / Pump	1
Adiabatic flash tank	0
Liquid phase reactor	1
Gas phase reactor	0
Heat exchanger	1
Column	0
Valve	0
Choke vale	1
For each closed cycle:	
Active charge	1
Composition of fluid	$n_c - 1$

- Plantwide optimization (day)
- Local optimization (hours)
- Supervisory control, including predictive and advanced control (minutes)
- Regulatory control (seconds)

It is possible to connect the control layers and solve the mentioned issues, but a systematic approach is needed. In a paper by Skogestad (2004), a method is proposed. The method consists of two parts, the so-called "Top-down analysis" and the "Bottom-up analysis". The top-down analysis deals with the definition of the control objective, identification of constraints, degree of freedom analysis and selection of controlled variables. The top-down analysis is only considering steady-state. Dynamics are covered by the bottom-up analysis, which deals with implementation of controllers, stabilization of the plant and real-time-optimization. The bottom-up analysis is not covered in its entirety because it is out of the scope of this thesis. Controller design, which is part of the bottom-up procedure, is covered in Section 2.3.2. The four steps in the top-down analysis are described below.

Defining the control objective

The first step in the top-down procedure is to define the control objective. The control objective is typically formulated as a cost function which has to be minimized. For example, it is often the objective to maximize the profit of a plant, in which case the cost function becomes the cost of the products minus the cost of the materials, the cost of the utilities and the cost of operation. The optimal solution is obtained when the marginal revenue equals the marginal cost. Other formulations of the cost function are possible as well, such as the minimization of environmental impact or the maximization of the throughput, with no regards to the economical cost. The cost function takes the form

$$\min_u J(\mathbf{y},\mathbf{u},\mathbf{d}) \quad s.t \quad \mathbf{g_1}(\mathbf{y},\mathbf{u},\mathbf{d}) = 0 \quad , \quad \mathbf{g_2}(\mathbf{y},\mathbf{u},\mathbf{d}) \leq 0 \tag{2.4}$$

where $\mathbf{y}$ are the outputs, $\mathbf{u}$ are the inputs and $\mathbf{d}$ are the disturbances. The minimization is subject to a set of equality constraints, $\mathbf{g_1}$, and a set of inequality constraints, $\mathbf{g_2}$.

The constraints are limits which are imposed on the states and the inputs. These constraints can be product specifications, physical limitations of the equipment or similar. If a constraint can not be violated, either because it is physically impossible (e.g. mass fractions larger than one) or because the violation of the constraint has very undesired effects (e.g. explosion of a reactor if the pressure becomes too large), the constraint is said to be hard. Soft constraints may be violated if necessary, but violation is undesired. An example for this would be a soft temperature constraint that is put in place to avoid the deterioration of process equipment at high temperatures. Soft constraints can be introduced by penalizing the violation of the constraint in the cost function. The penalty can be linear or non-linear, depending on the "softness" of the constraint.

Degree of freedom analysis

The second step in the top-down procedure is to identify the steady-state and dynamic degrees of freedom. Section 2.1.5 describes a method for identifying the degrees of freedom for a refrigeration cycle. For a general process, the following relationship can be used (Skogestad 2004)

$$n_{ss} = n_{MV} - n_0 \tag{2.5}$$

where n_0 are the number of dynamic degrees of freedom which have no steady-state effect, such as tanks where no chemical reactions occur.

Typically only steady-state degrees of freedom will effect the cost function.

Implementation of the optimal solution

Once the minimization problem has been defined, it can be solved to find the nominal operating point of the plant. The problem is a mathematical programming problem and can be quite difficult to solve, especially if the problem is non-linear and multivariate. The solution to such a problem must satisfy a set of conditions known as the Karusch-Kuhn-Tucker (KKT) conditions (Wright & Nocedal 1999). Algorithms such as the interior point method attempt to solve the KKT equations to find the global minimum of the problem. The interior point algorithm is used by MATLABs `fmincon`-function to solve the optimization problem in this work.

Based on the results from the optimization, a control structure can be found. Variables which have their nominal value on a constraint boundary, are called active. Active constraints must be controlled to avoid a large penalty on the cost function. As can be seen from Equation 2.6, active constraints are locally proportional to the loss. Variables which have a non-active nominal value only have a quadratic effect on the loss, as can be seen from Equation 2.8. For small perturbations from the nominal point, the active constraints therefore have a larger impact. Active constraints must therefore be controlled.

Wright & Nocedal (1999) show that the back-off of from the constraints which are optimally active is locally penalized linearly according to Equation 2.6. λ is the Lagrange multiplier and c are the constraints.

$$L_{\text{back-off}} = \left|J(\mathbf{c},\mathbf{d}) - J\left(\mathbf{c}^{\text{act}},\mathbf{d}\right)\right| = \lambda \cdot \left|\mathbf{c} - \mathbf{c}^{\text{act}}\right| \tag{2.6}$$

For the remaining unconstrained problem, the loss can be rewritten as the Taylor expansion of the cost function around the optimal point

$$L = \left|J(\mathbf{u},\mathbf{d}) - J\left(\mathbf{u}^{\text{opt}},\mathbf{d}\right)\right| = \mathbf{J_u} \cdot \left|\mathbf{u} - \mathbf{u}^{\text{opt}}\right| + \frac{1}{2}\left(\mathbf{u} - \mathbf{u}^{\text{opt}}\right)^{\mathsf{T}} \cdot \mathbf{J_{uu}} \cdot \left(\mathbf{u} - \mathbf{u}^{\text{opt}}\right) + \zeta^3 \tag{2.7}$$

Since $\mathbf{J_u} = 0$ at the optimum, the loss can be written as

$$L \approx \frac{1}{2}\left(\mathbf{u} - \mathbf{u}^{\text{opt}}\right)^{\mathsf{T}} \cdot \mathbf{J_{uu}} \cdot \left(\mathbf{u} - \mathbf{u}^{\text{opt}}\right) \tag{2.8}$$

If any steady-state degrees of freedom remain after controlling the active constraints, these can be used for optimization.

Inventory control

As mentioned earlier, gas and liquid inventories must be controlled to prevent them from emptying of overflowing. The controller pairings must be chosen such that the inventory control is locally consistent. Local consistency is defined by Aske & Skogestad (2009) as

"An inventory control system is consistent if it can achieve acceptable inventory regulation for any part of the process, including the individual units and the overall plant"

This means that both the global mass balance and the local mass balance must be satisfied for each unit. When the mass balance over a unit is satisfied even when it is viewed independently of the rest of the system, it is said to be locally consistent. A simple way to find out whether a system is consistent or not is to use the radiation rule described by Price & Georgakis (1993). Starting from the throughput manipulator and radiating outwards, each unit is checked to make sure it is consistent.

2.2.2 Self-optimizing control

Steady-state degrees of freedom which are not used to control active constraints can be used to keep the deviation from the nominal solution as small as possible, even when the process is disturbed. Such a control strategy would thus always ensure close to optimal operation, hence the name "self-optimizing control". Obviously the best controlled variable would be the gradient of the cost function. By keeping the gradient at zero, $J_u = 0$, the process would always be optimal. Unfortunately it is usually impossible to measure the gradient directly, so the optimality of the state must be estimated indirectly. Rather than controlling J_u, it is chosen to control a vector **c** which is a linear combination of the state that satisfies

$$\mathbf{c} = \mathbf{H} \cdot \mathbf{y} \tag{2.9}$$

H is known as the selection matrix. **c** is chosen such that the loss L is minimized

$$L = J(\mathbf{u}, \mathbf{c}) - J^{\text{opt}}(\mathbf{d}) \tag{2.10}$$

A few criteria can be used to choose the best possible **c**. First of all, it is desirable that the optimum value of **c** does not change when the process is disturbed. In other words, **c** should be insensitive to disturbances **d**. This is illustrated for a single c in Figure 2.8. If the optimal value of c is significantly different for the disturbed system and the nominal system, the loss ΔJ_d from keeping $c_{set} = c_{nom}^{\text{opt}}$ can be large. It is therefore desirable that $\Delta c^{\text{opt}} \approx 0$.

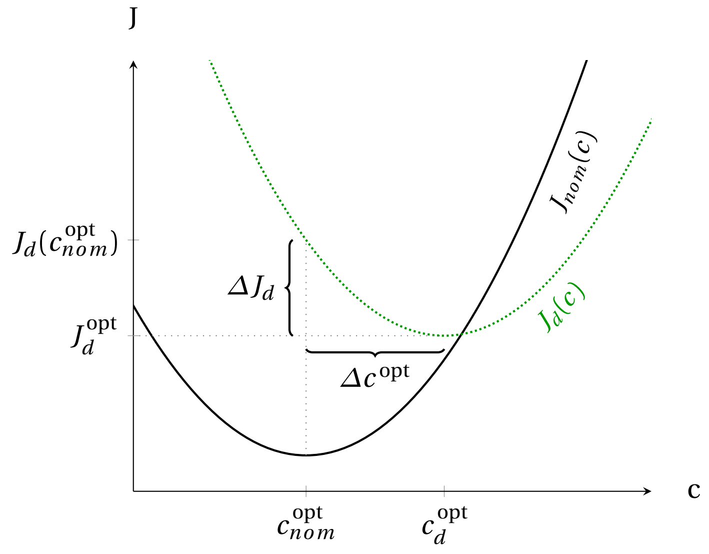


Figure 2.8: Sensitivity of c^{opt} to a disturbance d.

The optimum of the cost function should also be as flat as possible. This is to minimize the effect of implementation error. If the setpoint for c is offset by a value n from the true optimal value c_{opt}, this should not effect the cost function much. This means that that J_{cc} should be small or equivalently that the concavity of J should be small. The effect of implementation error for different concavities of the cost function is illustrated in Figure 2.9. It can be seen that the loss $\Delta\tilde{J}$ due to an implementation error δ is larger than the loss ΔJ since the concavity of $\tilde{J}$ is larger than the concavity of J.

Lastly, c should be easy to implement and measure. This means for example that temperature and pressure measurements are preferred to composition measurements, as they are easier to implement and more reliable.

As expected, the selection of **c** is usually not trivial. For single measurements, one can evaluate the loss directly and chose the measurement which gives the smallest loss when kept constant (Skogestad 2000). This method is known as the brute force method, because the system is simply solved again for each candidate measurement. The disadvantage is that one single measurement may not contain enough information about the system to keep the loss acceptably low. If linear combinations of measurements are to be used, the computational cost for evaluating the loss quickly becomes overwhelmingly large. Furthermore, this method is limited by the evaluation of just one disturbance or a specific combination of disturbances. This minimizes the loss for a given

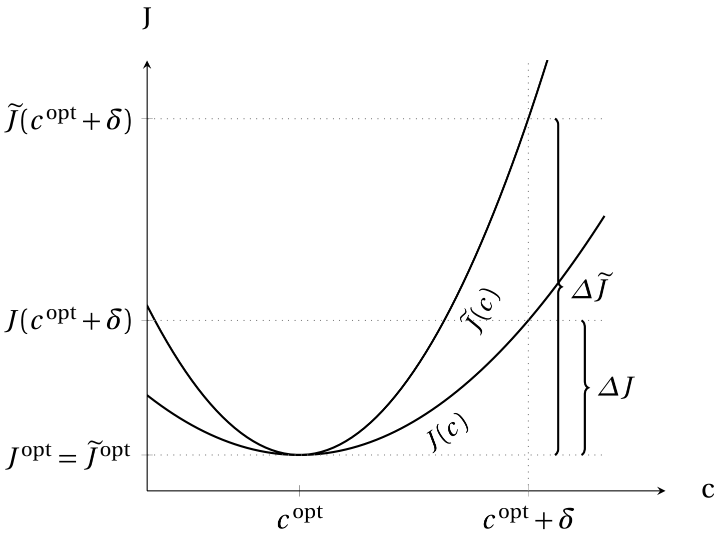


Figure 2.9: Sensitivity of c^{opt} to implementation error

combination of disturbances, but a different set of disturbances might give a very large loss.

The following sections present two methods to find the selection matrix **H** that works for any combination of disturbances. This is done by minimizing the average loss for all disturbances and implementation errors.

Nullspace method

The nullspace method is a simple method developed by Alstad & Skogestad (2007) to find **c** if it is reasonable to assume that there is no implementation error. It requires that the number of measurements is larger than the number of inputs and disturbances, $n_u + n_d \leq n_y$. The derivation is straightforward

Let the sensitivity matrix **F** be defined as

$$\mathbf{F} = \frac{\partial \mathbf{y}^{\text{opt}}}{\partial \mathbf{d}} \tag{2.11}$$

Locally, the truncated Taylor expansion can be used

$$\Delta \mathbf{y}^{\text{opt}} = \mathbf{F}\Delta\mathbf{d} \tag{2.12}$$

Combining Equation 2.12 and Equation 2.9,

$$\Delta\mathbf{c} = \mathbf{H}\,\mathbf{F}\Delta\mathbf{d} \tag{2.13}$$

As stated previously, **c** should be insensitive to disturbances. It follows that

$$\Delta\mathbf{c} = 0 \longrightarrow \mathbf{H}\,\mathbf{F}\Delta\mathbf{d} = 0 \tag{2.14}$$

Since $\Delta\mathbf{d}$ is non-zero, this must mean that $\mathbf{H}\,\mathbf{F} = \mathbf{0}$. Given the sensitivity matrix $\mathbf{F}$, the selection matrix $\mathbf{H}$ can thus be found in the left nullspace of $\mathbf{F}$

$$\mathbf{H} \in \mathcal{N}(\mathbf{F}^\mathsf{T}) \tag{2.15}$$

Any subspace of the left nullspace of $\mathbf{F}$ can be used as long as the dimensions are $n_u \times n_y$.

Exact local method

The exact local method is based on a linearized version of the model and a second order Taylor expansion of the cost function (Kariwala et al. 2008), as previously shown in Equation 2.7.

The loss from Equation 2.8 can be written as

$$L = \frac{1}{2}\mathbf{z}^\mathsf{T}\mathbf{z} \tag{2.16}$$

where **z** is defined as

$$\mathbf{z} = \mathbf{J}_{\mathbf{uu}}^{\mathbf{1/2}} \cdot (\mathbf{u} - \mathbf{u}^{\text{opt}}) \tag{2.17}$$

The models are linearized

$$\Delta\mathbf{y} = \mathbf{G}^\mathbf{y}\Delta\mathbf{u} + \mathbf{G}^\mathbf{y}_\mathbf{d}\Delta\mathbf{d} \tag{2.18}$$

$$\Delta\mathbf{c} = \mathbf{G}\Delta\mathbf{u} + \mathbf{G}_\mathbf{d}\Delta\mathbf{d} \tag{2.19}$$

Introducing the magnitudes of the disturbances and the measurement errors as

$$\Delta\mathbf{d} = \mathbf{W}_\mathbf{d}\mathbf{d}' \tag{2.20}$$

$$n^y = \mathbf{W}_{\mathbf{n}^\mathbf{y}}\mathbf{n}^{\mathbf{y}\prime} \tag{2.21}$$

where $\mathbf{W}_\mathbf{d}$ and $\mathbf{W}_{\mathbf{n}^\mathbf{y}}$ are diagonal scaling matrices with the magnitudes of the expected disturbances and measurement errors, respectively. $\mathbf{d}'$ and $\mathbf{n}^{\mathbf{y}\prime}$ are normalized vectors which are normally distributed with zero mean and unity variance (Kariwala et al. 2008).

$$\begin{bmatrix}\mathbf{d}' & \mathbf{n}^{\mathbf{y}\prime}\end{bmatrix}^\mathsf{T} \sim N\left(0, \mathbf{I}_{n_d+n_y}\right) \tag{2.22}$$

The linearized version of the input is

$$\Delta\mathbf{u}^{\text{opt}} = -\mathbf{J}_{\mathbf{uu}}{}^{-1}\,\mathbf{J}_{\mathbf{ud}}\Delta\mathbf{d} \tag{2.23}$$

The sensitivity matrix from Equation 2.12 gives the relationship between the disturbances and the optimal measurements. $\mathbf{F}$ can be written as

$$\mathbf{F} = \left(-\mathbf{G^y}\,\mathbf{J_{uu}}^{-1}\,\mathbf{J_{ud}} + \mathbf{G^y_d}\right) \tag{2.24}$$

Using these linearizations, it can be shown that the loss $\mathbf{z}$ from Equation 2.17 can be written as

$$\mathbf{z} = \mathbf{M_d d'} + \mathbf{M_{n^y} n^{y\prime}} \tag{2.25}$$

where

$$\mathbf{M_d} = -\mathbf{J_{uu}}^{1/2}\,(\mathbf{H\,G^y})^{-1}\,\mathbf{H\,F\,W_d} = \mathbf{J_{uu}}^{1/2}\left(\mathbf{J_{uu}}^{-1}\,\mathbf{J_{ud}} - \mathbf{G}^{-1}\,\mathbf{G_d}\right)\mathbf{W_d} \tag{2.26}$$

$$\mathbf{M_{n^y}} = -\mathbf{J_{uu}}^{1/2}\,(\mathbf{H\,G^y})^{-1}\,\mathbf{H\,W_{n^y}} = \mathbf{J_{uu}}^{1/2}\,\mathbf{G}^{-1}\,\mathbf{W_{n^y}} \tag{2.27}$$

Locally, the average loss which satisfies Equation 2.22 is shown by Kariwala et al. (2008) to be

$$L_{avg} = \frac{1}{2}\,\|[\mathbf{M_d} \quad \mathbf{M_{n^y}}]\|_F^2 \tag{2.28}$$

The subscript F indicates the Frobenius norm. Average loss means in this case the average loss between all possible combinations of disturbances and implementation errors. The magnitudes of the disturbances and the implementation errors are given by $\mathbf{W_d}$ and $\mathbf{W_{n^y}}$, respectively. It is assumed that all the disturbances and implementation errors are normally distributed. The term which is to be normed can be written as

$$\begin{aligned}
[\mathbf{M_d} \quad \mathbf{M_{n^y}}] &= \mathbf{J_{uu}}^{1/2}\mathbf{G}^{-1}\left(\left(\mathbf{G\,J_{uu}}^{-1}\,\mathbf{J_{ud}}^{-1} - \mathbf{G_d}\right)\mathbf{H\,W_{n^y}}\right) && (2.29)\\
&= \mathbf{J_{uu}}^{1/2}(\mathbf{H\,G^y})^{-1}\,\mathbf{H\,Y} && (2.30)
\end{aligned}$$

where

$$\mathbf{Y} = [\mathbf{F\,W_d} \quad \mathbf{W_{n^y}}] = \left[\left(\mathbf{G^y\,J_{uu}}^{-1}\,\mathbf{J_{ud}} - \mathbf{G^y_d}\right)\mathbf{W_d} \quad \mathbf{W_{n^y}}\right] \tag{2.31}$$

The goal is to find $\mathbf{H}$ such that the average loss in Equation 2.28 is minimized. The minimization problem can thus be written as

$$\min_{\mathbf{H}} = \left\|\mathbf{J_{uu}}^{1/2}\,(\mathbf{H\,G^y})^{-1}\,\mathbf{H\,Y}\right\|_F \tag{2.32}$$

An analytical expression for $\mathbf{H}$ is presented by Alstad et al. (2009). The explicit solution for $\mathbf{H}$ is

$$\mathbf{H}^\mathsf{T} = (\mathbf{Y\,Y}^\mathsf{T})^{-1}\,\mathbf{G^y}\left(\mathbf{G}^{\mathbf{y}\mathsf{T}}\,(\mathbf{Y\,Y}^\mathsf{T})^{-1}\,\mathbf{G^y}\right)^{-1}\,\mathbf{J_{uu}}^{1/2} \tag{2.33}$$

It is necessary that the matrix $(\mathbf{Y\,Y}^\mathsf{T})$ has full rank.

Yelchuru & Skogestad (2010) found that Equation 2.33 can be simplified by realizing that $\mathbf{H}$ is not a unique solution. Since $\widetilde{\mathbf{H}}^\mathsf{T} = \mathbf{H}^\mathsf{T}\mathbf{D}^\mathsf{T}$ also satisfies Equation 2.32 for any matrix $\mathbf{D}$, the solution from Equation 2.33 can be scaled by choosing $\mathbf{D}$ such that

$$\mathbf{D}^\mathsf{T} = \left(\left(\mathbf{G}^{\mathbf{y}\mathsf{T}}\left(\mathbf{Y}\,\mathbf{Y}^\mathsf{T}\right)^{-1}\mathbf{G}^\mathbf{y}\right)^{-1}\mathbf{J_{uu}}^{1/2}\right)^{-1} \tag{2.34}$$

The solution from Equation 2.33 then simplifies to

$$\widetilde{\mathbf{H}}^\mathsf{T} = \left(\mathbf{Y}\,\mathbf{Y}^\mathsf{T}\right)^{-1}\mathbf{G}^\mathbf{y} \tag{2.35}$$

Selection of controlled variables

Since there are usually a large number of measurements available in a plant, it is infeasible to use all of them in the calculation of the controlled variable (CV) for the self-optimizing controller. Thus a subset of measurements must be chosen. This introduces the problem of choosing the right measurements to get the optimal CV. If the plant has a large amount of measurements, the sheer amount of combinatorial possibilities means that evaluating every single possible subset of measurements to determine the lowest loss is computationally impossible. For example, consider a plant with 100 possible measurements of which a subset of 5 is chosen to calculate the CV. There are $\binom{100}{5} = 75,287,520$ possible subsets. Evaluating the loss for each of those is not feasible. Some of the measurements can be excluded from the analysis based on heuristic rules, but in general this is not enough to reduce the problem to a manageable size. The heuristic rules may also fail to result in a truly optimal CV.

Kariwala & Cao (2009) developed a bidirectional branch and bound (BAB) method which efficiently finds the subset of measurements which will yield the optimal CV. The general idea behind BAB methods is to divide the selection problem into smaller subproblems which are then solved recursively. The method does not waste time evaluating suboptimal branches as it discards branches that do not meet a certain selection criterion. The algorithm uses the exact local method to estimate the loss of a branch and uses this as the selection criterion. In this way it is ensured that the optimal CV is found. The exact algorithm will not be explained here, as it is outside of the scope of the thesis. It is referred to Kariwala & Cao (2009) for more information. However, the method will be used to find the optimal subset of measurements for the self-optimizing controller. A MATLAB script which implements the bidirectional BAB method can be downloaded from Mathworks [1].

[1] `http://www.mathworks.com/matlabcentral/fileexchange/25870-bidirectional-branch-and-bound-for-average-loss-minimization`

It follows from the branching method that the average loss decreases with an increasing number of measurements since more information about the system becomes available. It must also be true that each additional measurement contains less information than the previous measurement, so that the average loss approaches a minimum value (not necessarily zero, due to implementation error) when the number of measurements approaches infinity.

Caos algorithm calculates the average loss according to Equation 11 in Kariwala et al. (2008), which says that

$$L_{avg} = \frac{1}{6\left(n_y + n_d\right)} \|[\mathbf{M_d} \quad \mathbf{M_{n^y}}]\|_F^2 \tag{2.36}$$

The underlying assumption in Equation 2.36 is that all disturbances and implementation errors are uniformly distributed over the allowable region and that they have the same probability of occurring. This assumption is somewhat questionable, since the concept of the nominal point loses its meaning when all operating points have equal probabilities. However, although the value of L_{avg} may be wrong, the $\mathbf{H}$ matrix will still be the one that minimizes the loss (Kariwala et al. 2008, Alstad et al. 2009), therefore we will use the branch and bound algorithm to compute $\mathbf{H}$, but evaluate the loss using Equation 2.28.

2.3 Dynamic simulation and controller design

This section contains some theory about controller design and the dynamic simulation of the plant. The focus of this thesis is mainly steady-state optimization, and the dynamic simulation will only be used to confirm the validity of the developed control structure. This section only contains the most essential theory about controller design used in this work.

2.3.1 Relative Gain Array

The relative gain array (RGA) is a useful tool for determining the best pairings of MVs and CVs in a multiple-input-multiple-output (MIMO) system. In a MIMO system, there are interactions between the control loops. The RGA provides a measurement of these interactions. Skogestad & Postlethwaite (2007) derive the RGA as follows. Consider a plant $\mathbf{G}(s)$. For a given pair of input u_j and output y_i, there are two extreme cases which must be considered:

- All other loops open:

$$u_k = 0 \quad \forall \quad k \neq j \quad , \quad g_{ij} = \left(\frac{\partial y_i}{\partial u_j}\right)_{u_{k\neq j}=0} \tag{2.37}$$

- All other loops perfectly controlled:

$$y_k = 0 \quad \forall \quad k \neq i \quad , \quad \hat{g}_{ij} = \left(\frac{\partial y_i}{\partial u_j}\right)_{y_{k \neq i} = 0} \tag{2.38}$$

Here, g_{ij} and $\hat{g}_{ij}$ are the process gains of the pair u_j-y_i for the two considered extremes. In terms of the plant $\mathbf{G}(s)$, the gains can be written as

$$g_{ij} = [\mathbf{G}]_{ij} \tag{2.39}$$

$$\hat{g}_{ij} = 1/\left[\mathbf{G}^{-1}\right]_{ji} \tag{2.40}$$

The ratios between these two gives the elements in the RGA.

$$\lambda_{ij} = \frac{g_{ij}}{\hat{g}_{ij}} \tag{2.41}$$

And the full RGA becomes

$$\mathbf{\Lambda} = \mathbf{G} \circ \left(\mathbf{G}^{-1}\right)^{\mathsf{T}} \tag{2.42}$$

where $\circ$ denotes the Hadamard product.

The RGA thus gives the ratio between the gains of the open and the closed loops. The different values of λ_{ij} indicate the interaction between the pair and the other loops. $\lambda_{ij} = 0$ means that the input will have no effect whatsoever on the output, and pairing should thus be avoided. $\lambda_{ij} = 1$ means that the input effects the output without any other interaction from other control loops. This is the desired case. Negative λ_{ij} indicate that the current loop becomes unstable (i.e. the gain changes sign) when any of the other loops are opened. This is not desirable. $\lambda_{ij} \leqslant 0.5$ means that other control loops influence the pair, with the influence of the other loops being larger than or equal to the control pair. This should generally be avoided, since it makes control very difficult. Values of $\lambda_{ij} > 1$ indicate that the control pair is dominant, but that other loops drive the output in the opposite direction.

Since the RGA is calculated at steady-state, it is often assumed that the RGA can only be used to determine the optimal pairing at steady-state. Skogestad & Postlethwaite (2007) argue that the expression for the RGA is general, and can thus be used at the crossover frequency as well.

2.3.2 Model reduction and controller tuning

Proportional-integral-differential (PID) controllers are used to control the process in dynamic mode. In the time domain, the controller equation can be written as

$$u(t) = u_0 + K_c\left(e(t) + \frac{1}{\tau_i}\int_0^t e(\tau)d\tau + \tau_d\frac{d}{dt}e(t)\right) \tag{2.43}$$

where the error e is defined as the deviation from the setpoint of the controlled variable.

$$e(t) = y_s(t) - y(t) \tag{2.44}$$

The controller gain K_c, the integral time τ_i and the derivative time τ_d are the controller parameters, and the response of the controller depends on the choice of these three parameters. Multiple methods exist to find good controller settings, such as the Ziegler-Nichols method and the direct synthesis method. However, in this thesis the Skogestad Internal Model Control (SIMC) method is used to find the proper controller settings. The SIMC method is easy to use and results in a robust controller (Skogestad 2003).

The first step in the SIMC method is to reduce the model to a first- or second-order plus delay model on the form

$$y(s) = \frac{\theta}{(\tau_1 s + 1)(\tau_2 s + 1)}u(s) \tag{2.45}$$

Skogestad proposes a simple empirical rule called the "half-rule" to find the reduced process model. The half-rule states that largest neglected process lag is to be distributed evenly between the delay and the smallest time constant. The half-rule is fully explained in Skogestad (2003)

Using the derived second-order plus delay model, the recommended controller settings from the SIMC method are

$$\begin{aligned} K_c &= \frac{1}{k}\frac{\tau_1}{(\theta + \tau_c)} && (2.46)\\ \tau_i &= \min[\tau_1, 4(\theta + \tau_c)] && (2.47)\\ \tau_d &= \tau_2 && (2.48) \end{aligned}$$

τ_c is the desired closed-loop time constant. Skogestad proposes $\tau_c = \theta$ to be used, as this value gives a good trade-off between speed and robustness of the controller. Reducing τ_c leads to a more aggressive controller, whereas increasing τ_c gives better robustness.

The derivative part of the controller is often set to zero in practice (Skogestad 2003). The derivative action works against the proportional and the integral part in that it wants to counteract change in the system. The derivative action thus destabilizes the controller, especially if the signal is noisy. In order to reduce wear on the valves due to rapid input changes, the derivative action is omitted.

CHAPTER 3

RESULTS AND DISCUSSION

This chapter presents the results obtained during the work with this thesis. Firstly the process is described in detail, followed by the results of the steady-state simulation and optimisation. The derived controllers are implemented and evaluated. Lastly, a section about an alternative process model is included. As will be shown later, the losses due to disturbances and the losses due to implementation errors are expected to be similar. The performance of the controllers will be evaluated by introducing disturbances to the system, since these are easier to implement in the current model, but the results will be applicable to implementation errors as well. The results will be discussed continuously throughout the chapter to ease readability.

3.1 Process description

This section contains a detailed description of the studied process. A short overview of the process will be given before the model equations and assumptions are presented. Lastly, it will be described in which ways the model has been altered in comparison to the original model by Basel Asmar on which the model presented in this thesis is based.

The presented model was developed as part of the project on 'Modelling and Optimization of a Two-Stage Compressor Train', which was conducted by Adriaen Verheyleweghen during the autumn of 2014 (Verheyleweghen 2014). The content of the following chapter is based on said project, but is repeated here for the convenience of the reader. Some modifications of the model have also been done after the publication of the project, so it is necessary to address these changes here.

3.1.1 Overview

The process studied in this thesis is inspired by a refrigeration cycle which is part of a large petrochemical plant operated by Exxon Mobile. Cooling is needed on two different temperature levels in two evaporators. Propylene is used as the refrigerant. Figure 3.1 shows the process flow diagram of the studied refrigeration cycle.

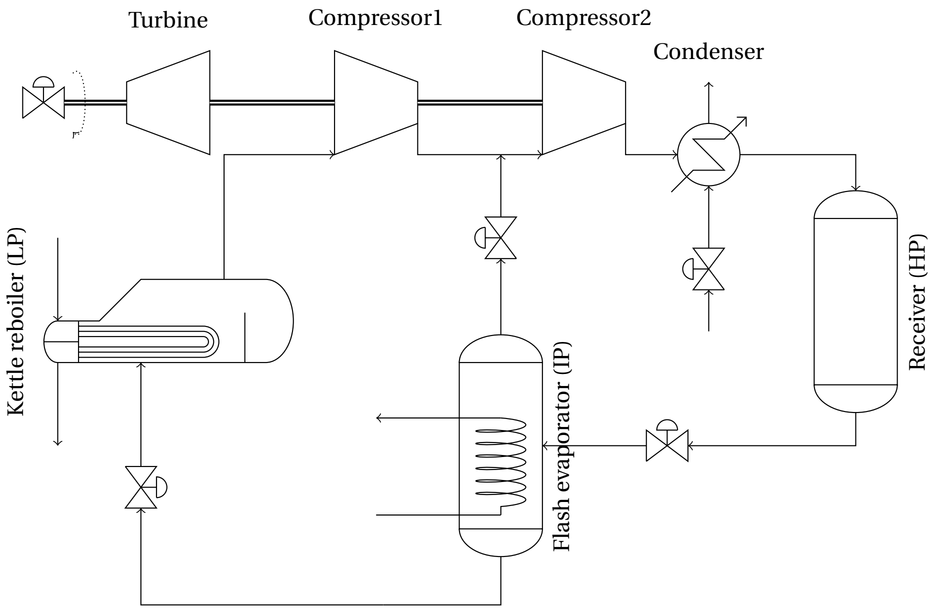


Figure 3.1: Process flow diagram of the studied process.

The refrigeration cycle is driven by a single steam turbine which runs two compressors in a series configuration. The steam turbine is a variable speed turbine, which means that the energy input to the system through the compressors can be adjusted. Cold gas is injected in the interstage node between the compressors to cool down the refrigerant before the inlet of the second compressor to avoid excessive overheat. After the final compression, the refrigerant is condensed with air cooling. The condensing liquid is collected in the receiver, which also acts as a buffer tank.

The heat loads are removed in the two evaporators. The first evaporator operates at intermediate pressure (IP) and removes a small heat load at high temperature. A flash evaporator is used for this purpose. By manipulating XV_1 and XV_2 the fraction of gas and liquid can be adjusted.

The main heat load is removed in a low pressure (LP) kettle reboiler. The size of the equipment differs by approximately a factor of ten between the LP and the IP stage. Due to the lower saturation pressure, the temperature level in this stage is much lower than in the IP evaporator.

The process bears similarities with the two-stage cycle with throttling that was discussed in Section 2.1. Additionally, it contains a liquid receiver to allow adjustment of the active charge in the system. The receiver is placed after the condenser, which is optimal according to Jensen & Skogestad (2007). The placement of the valve on the vapour outlet of the second evaporator is identical to Figure 2.10 in Jensen & Skogestad (2007), but contrary to what was written there, the pressure-enthalpy diagram of the cycle will not be identical to Figure 2.7. The pressure drop from the introduced valve is not taken into account there, so the diagram will look slightly for this cycle. The p-h diagrams for this cycle are shown in Appendix C.

3.1.2 Process model

This section contains all the model equations which were used to simulate the process. The model is based on work done by Basel Asmar (1991). A full description of the model can be found in Asmar's thesis. This section was previously published as part of the report 'Modelling and Optimization of a Two-Stage Compressor Train', by Verheyleweghen (2014). The model equations are largely unchanged, but some of the descriptive text has been updated since then.

The process described by Asmar is a generalized refrigeration cycle consisting of n compression stages. To keep the model as descriptive as possible, Asmar's notation will be kept when describing the studied process, even though it consists of only two stages.

The resulting model was implemented in the MATLAB file `'model.m'`. The parameters for the equations are contained in the file `'init_params.m'`. Both files are attached in Appendix D.

Evaporators

Figure 3.2 shows an illustration of a general evaporator i.

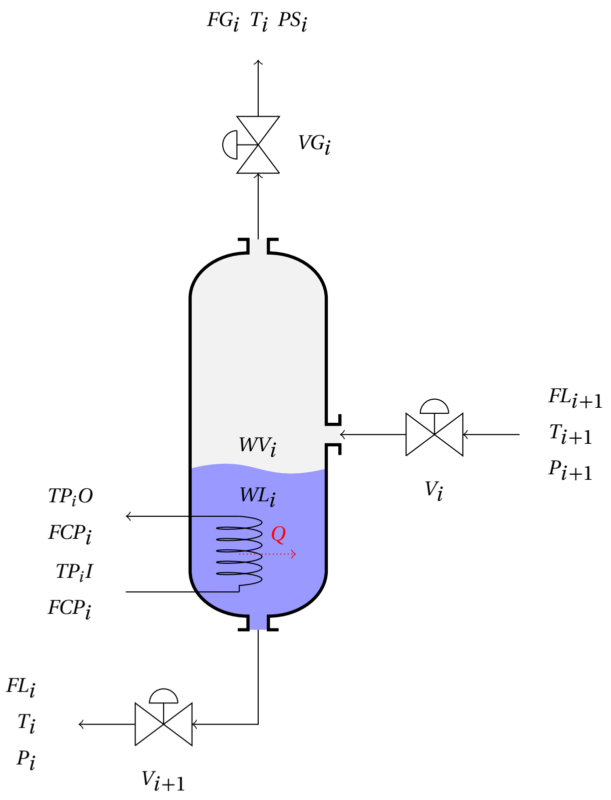


Figure 3.2: Illustration of an evaporator.

The mass balance over the evaporator from Figure 3.2 can be formulated as

$$\frac{dW_i}{dt} = FL_{i+1} - FL_i - FG_i \tag{3.1}$$

where *FL* are the liquid flows in and out of the evaporator and *FG* is the gas flow rate of the evaporator. This gives the change in refrigerant hold-up *W* in the evaporator. The liquid level in the evaporator can be calculated implicitly from the hold-up *W*.

Assuming that the cross-sectional area of the evaporator is constant, then the level of the refrigerant is proportional to the volumetric liquid hold-up, *WLV*.

$$L_i \propto WLV_i \tag{3.2}$$

The volumetric liquid hold-up is defined as the product of the liquid hold-up WL_i and the specific volume of the refrigerant v_f

$$WLV_i = WL_i \cdot v_{f,i} \tag{3.3}$$

The liquid hold-up *WL* is implicitly given by the mass balance over the vessel inventory

$$W_i = WL_i + WV_i \tag{3.4}$$

Flow rates are adjusted by the control valves. The driving force for the mass flow is the pressure difference over the valve.

$$FL_{i+1} = XVL_i \cdot CVL_i \sqrt{\Delta P} \tag{3.5}$$

In the above equation, *XVL* is the fractional valve opening and *CVL* is the valve constant.

The same equation is used for calculating the vapour flow rates.

$$FG_i = XVG_i \cdot CVG_i \sqrt{\Delta P} \tag{3.6}$$

The flow rate of the saturated vapour stream out of the LP evaporator can not be controlled without over-specifying the system, such that the flow rate is determined by the suction pressure of the compressor.

The saturation pressure P_i of the working fluid is related to the saturation temperature through Antoine's equation

$$P_i = \exp\left(A - \frac{B}{T_i - C}\right) \tag{3.7}$$

where A, B and C are constants. The values of the coefficients are given in Table A.1 in the Appendix. The saturation temperature T_i must be calculated implicitly from the energy balance.

$$H_i = WL_i \cdot HFL_i + WV_i \cdot HFG_i \tag{3.8}$$

The dynamic energy balance for the evaporator can be written as

$$\frac{dH_i}{dt} = (HFL \cdot FL)_{i+1} - (HFL \cdot FL)_i - (HFG \cdot FG)_i + Q_i \tag{3.9}$$

where *HFL* is the specific enthalpy of the liquid stream and *HFG* is the specific enthalpy of the gas stream. Q_i is the heat transferred from the process stream to the refrigerant in the evaporator.

The heat load Q_i can be calculated from the energy balance of the process stream as shown in Equation 3.10.

$$Q_i = FCP_i\,(TP_iI - TP_iO) \tag{3.10}$$

Given the saturation temperature in the evaporator, the exit temperature of the process stream can be calculated from the heat exchanger equation. Using the logarithmic mean temperature difference as a driving force for the heat transfer, the expression for TP_iO can be written as

$$TP_iO = (1-\alpha_i)\,T_{i+1} + \alpha_i\,TP_iI \tag{3.11}$$

where

$$\alpha_i = \exp\left(\frac{-U_i \cdot A_i}{FCP_i}\right) \tag{3.12}$$

here, U is the heat transfer coefficient and A is the available heat transfer area. FCP is the product of the heat capacity and the flowrate of the process stream.

Condenser and receiver

Figure 3.3 shows the condenser and the receiver units.

In the condenser, the compressed propylene is condensed at the discharge pressure using cold air as the coolant. It is assumed that the condenser contains only vapour. The saturated liquid is collected in the receiver, which simultaneously acts as a buffer tank to even out any fluctuations in mass flow or temperature due to its large size. The mass balance over the condenser can be written as

$$\frac{d\,WV_c}{d\,t} = FGCD_n - FL_c \tag{3.13}$$

The total vapour hold-up can be calculated by summation of the vapour hold-ups in the receiver and the condenser. Since the condenser does not contain any liquid, the vapour hold-up in the condenser equals to the total condenser volume.

$$VG_c = V_c + V_r - (WL_r \cdot v_{f,r}) \tag{3.14}$$

The liquid hold-up WL_r in the receiver is calculated implicitly from a total mass balance over all inventories in the system. This process has two evaporators in addition to the condenser, so $n = 2$ in the summation term in the following mass balance.

$$WL_r = W - \sum_i^n W_i - WV_c \tag{3.15}$$

The energy balance over the receiver simplifies to Equation 3.16, as shown by Asmar (1991).

$$\frac{d\,T_r}{d\,t} = \frac{FL_c}{WL_r}(T_c - T_r) \tag{3.16}$$

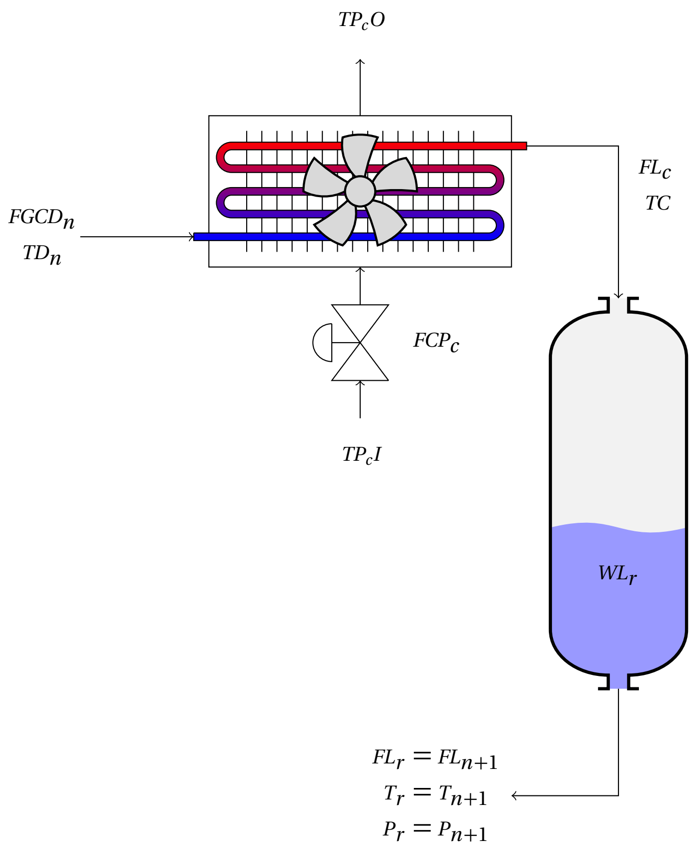


Figure 3.3: Illustration of the condenser and receiver

where the liquid refrigerant flow rate from the condenser, FL_c, can be expressed as

$$FL_c = \frac{Q_c}{HFG_c - HFL_c} \tag{3.17}$$

HFG_c and HFL_c are the specific enthalpies of the gas phase and the liquid phase, respectively. Q_i is the heat removed from the propylene in the condenser, and is calculated similarly to Q_i for the evaporators.

Compressor

Figure 3.4 shows a generic compressor stage, including an interstage mixing node for injection of saturated refrigerant vapour. The suction mass flowrate

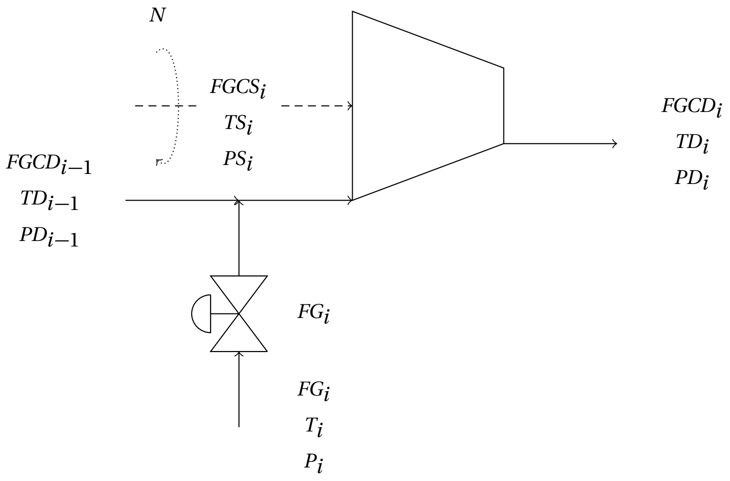


Figure 3.4: Illustration of a generic compressor stage. Notice that the figure does not apply for the first compression stage, as the first compression stage does not have interstage injection of saturated refrigerant vapour.

into the compressor is given by the equation of state.

$$FGCS_i = FGV_i \frac{M_w \cdot PS_i}{TS_i \cdot R \cdot z_i} \tag{3.18}$$

where *TS* and *PS* are the suction temperature and pressure respectively, z is the compressability factor and *FGV* is the inlet volumetric vapour flowrate of refrigerant. M_w is the molecular weight of propylene and R is the universal gas constant.

The suction mass flowrate must satisfy the mass balance over the mixing node

$$FGCS_i = FGCD_{i+1} + FG_i \tag{3.19}$$

where *FGCD* is the discharge mass flow rate from the previous compressor and *FG* is the vapour flow rate from the corresponding evaporator. For the first compressor stage *FG* will be zero, as there is no interstage mixing.

The suction temperature to the compressor, *TS*, is calculated from the energy balance over the mixing node. It is assumed that no heat loss occurs over the expansion valve and that there is no heat of mixing, so that the suction temperature is simply the weighted average of the two combined temperatures.

$$TS_i = \frac{TD_{i-1} \cdot FGCD_{i-1} + T_i \cdot FG_i}{FGCD_{i-1} + FG_i} \tag{3.20}$$

The compressor equations are based on empirical correlations which are found by fitting curves to real experimental data.

$$\frac{FGV_i}{N^q} = f_i(hs_i) \tag{3.21}$$

In the above expression, N is the fractional compressor speed, q is a constant and f is the compressor curve. Each compressor has a unique compressor curve and q-value. *hs* is the scaled compressor head, and is defined as

$$hs_i = \frac{h_i}{N^p} \tag{3.22}$$

p is a constant value depending on the compressor.

For polytropic compression, the compression head can be expressed as

$$h_i = k_i \frac{R}{g \cdot M_w}(TD_i - TS_i) \tag{3.23}$$

where g is the gravitational constant. Using the polytropic relationship, the discharge temperature can be expressed as a function of the suction temperature and suction- and discharge pressures.

$$TD_i = \frac{TS_i}{\left(\frac{PD_i}{PS_i}\right)^{\frac{1}{k_i}}} \tag{3.24}$$

k_i is a constant value which is defined as

$$k_i = \frac{n}{n-1} = \frac{\eta\gamma_{avg,i}}{\gamma_{avg,i} - 1} \tag{3.25}$$

In the above expression, n is the polytropic exponent and ν is the polytropic efficiency of the compressor, which is given by

$$\eta_i = g_i(hs_i) \tag{3.26}$$

Similarly to the compressor curve f, g is found by fitting experimental data from a specific compressor unit. Finally, γ_{avg} is the averaged ratios between the specific heats (adiabatic ratios) between suction and discharge.

$$\gamma_{avg} = \frac{1}{2}\left(\frac{CPI_i}{CPI_i - R} + \frac{CPO_i}{CPO_i - R}\right) \tag{3.27}$$

CPI is the heat capacity at suction conditions, whereas *CPO* is the heat capacity at discharge conditions.

By fitting supplier data to the performance of the compressors, the compressor curve for the first compressor was found to be

$$f_1(hs_1) = \frac{FGV_1}{N^q} = \frac{C_{11} \cdot hs_1 - C_{12}}{C_{13}} \tag{3.28}$$

For the second compressor

$$f_2(hs_2) = \frac{FGV_2}{N^q} = C_{24} + C_{25} \log\left(\left(\sqrt{TX^2+1}\right) - TX\right) \tag{3.29}$$

TX is defined as

$$TX = \frac{C_{21}}{\tan\left(\frac{C_{22}-hs_2}{C_{23}}\right)} \tag{3.30}$$

In a similar fashion, g is found by fitting actual performance data. For both compressors, the following relationship is used

$$\eta_i = g_i(hs_i) = e_1 \cdot hs_i + e_2 - 10^{(e_3 \cdot hs_i - e_4)} \tag{3.31}$$

Improved thermodynamic model equations

The above section contains the equations which make up the fundamental framework of the model. In order to flesh out the model, some more equations are necessary to relate the energy balances to the thermodynamics of the system. Some assumptions have already been made in the described model, mainly regarding the compressors. It was chosen to relate the compressor performance to empirical compressor curves rather than including entropy calculations in the model, for example. This subsection describes the equations which are used to calculate enthalpies and other thermodynamic properties. It will also be discussed why the equations used in this work differ from the ones proposed by Asmar (1991).

Asmar proposed first order linear approximations of the thermodynamic properties to be used. After the completion of the project (Verheyleweghen 2014), it was discovered that the overall energy balance of the model was not satisfied. Due to the large differences in magnitude for the compressor duties and the heat transferred in the evaporators, it was suspected that the original work by Asmar contained a misprint in the units of one or several of the variables. However, the issue persisted even when replacing the unit joules with kilojoules for the variables in question, so the error is most likely somewhere else. It was calculated that the COP of the cycle described in Verheyleweghen (2014) is 0.003, whereas the COP of the cycle in this thesis is a much more reasonable 1.3. This strengthens the hypothesis that the model by Asmar contains a misprint, or that the we did a mistake when implementing the model in MATLAB in Verheyleweghen (2014)

It was also observed that some of the linear relationships used by Asmar deviated somewhat from literature data for propylene (Angus et al. 2013, Chao & Zwolinski 1975). One example being that a constant compressibility factor was used in the LP evaporator. Figure 3.5 shows the compressibility factor as a function of the saturation temperature in the LP evaporator as calculated by Asmar's original model equation and as calculated by the new model equation. As can be seen, the deviation from the literature data is in the order of magnitude of approximately 5%, so one could argue that the deviation is insignificant. However, the errors accumulate through the model and are possibly causing the inconsistency in the global energy balance. Other deviations from the literature (Angus et al. 2013, Chao & Zwolinski 1975) were observed for the enthalpy and heat capacities (although less grave in the latter case).

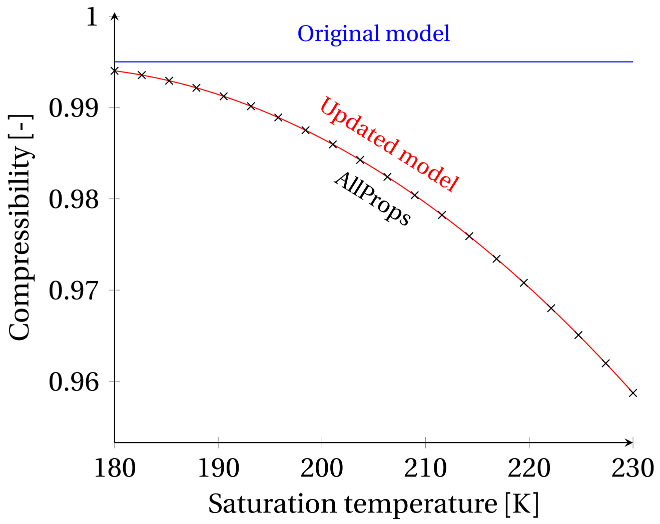


Figure 3.5: Values of the compressibility factor in the LP evaporator as a function of the saturation temperature according to Asmar (blue line), AllProps (black marks) and the updated model (red line).

Since Asmar does not cite a source for where the data on which the linearisations are based, it was not possible to find the source of the error easily. It was therefore decided that all equations relating to the energy balance would be replaced by new ones to make sure that all equations are consistent. The new thermodynamic equations were also chosen such that they are corresponding well with literature data over the entire range of operating conditions. The equations are based on the AllProps-model developed by the Center for Applied Thermodynamic Studies at the University of Idaho. AllProps can be downloaded from the web[1], but the code must be recompiled if a 64-bit operating system is used. gfortran 4.9 was used for this purpose.

It was first attempted to use the model directly by building a MEX file, but this was deemed difficult as there currently does not exist a functioning open-source Fortran compiler. It was considered to write a C wrapper from which the Fortran code was called, but this solution was abandoned because it was considered unnecessarily difficult. In the end it was decided that the compiled AllProps-model would be used to generate data to which polynomial models were fitted. The thermodynamic properties in question were calculated over the entire range of operating conditions dictated by the constraints for the pressures in the vessels. The constraints are shown in Table 3.1. A polynomial model was fitted to the data and used for interpolation. All properties were fitted as second order polynomials of temperature (and pressure in the case of the compressibility factor). It was found that second order polynomials gave a satisfactory R^2-value of approximately 0.999 for the fitted functions within the defined temperature ranges (see Table 3.1).

Resulting equations

This section contains all the remaining equations which are used to complete the model described in Section 3.1.2. The associated coefficients are summarized in tables in the Appendix A with the rest of the parameters.

The specific volume is assumed to be a a second-order polynomial function of temperature, and can be written as

$$v_{f,i} = \lambda_1 \cdot T_i^2 + \lambda_2 \cdot T_i + \lambda_3 \tag{3.32}$$

where λ_i are constant coefficients which are found by curve fitting. The values for the coefficients are given in Table A.6 in the Appendix.

The specific enthalpies are assumed to be linear functions of the temperature

$$HFG_i = \zeta_1 \cdot T_i + \zeta_2 \tag{3.33}$$

$$HFL_i = \delta_1 \cdot T_i + \delta_2 \tag{3.34}$$

[1]`http://www.nt.ntnu.no/users/skoge/book-cep/diagrams/additional_diagrams/allprops/`

The compressibility factor z is approximated as a second-order polynomial function of temperature and pressure.

$$z_i = \phi_{00} + \phi_{10} \cdot P_i + \phi_{01} \cdot T_i + \phi_{20} \cdot P_i^2 + \phi_{11} \cdot P_i \cdot T_i + \phi_{02} \cdot T_i^2 \tag{3.35}$$

The compressibility factors in the LP evaporator and the condenser are only a function of the temperature since the saturation temperature dictates the pressure. This means that ϕ_{10}, ϕ_{20} and ϕ_{11} are zero in these cases.

The heat capacities are approximated as fifth order polynomials of the temperature.

$$CP = C_1 + C_2 \cdot T\,(C_3 + C_4 \cdot T\,(C_5 + C_6 \cdot T)) \tag{3.36}$$

Other differences from the original model

In addition to the aforementioned issues with the thermodynamic model equations, it was found that the original model by Asmar had unrealistically fast inputs. Asmar assumed that the valves adjusted immediately to changes in their setpoints. This is not very realistic, which is why first order filters were added to all inputs to simulate the dynamics of the valve. This also removes discontinuities in the outputs when the inputs are included as measurements for the controlled variables. Discontinuities can potentially lead to an unstable controller if derivative action is used.

The first order filters were modelled as

$$\frac{du_i}{dt} = \frac{u_i - u_{i,s}}{\tau} \tag{3.37}$$

where u_i is the actual input, $u_{i,s}$ is the setpoint of the input and τ is the time constant. The time constant for the inputs are shown in Table A.10 in Appendix A.

3.2 Steady-state simulation

This chapter contains the results from the steady-state simulations, including the nominal operating point as defined by the optimization problem and the self-optimizing control strategy resulting from the sensitivity analysis of the nominal solution.

3.2.1 Formulation of the optimization problem

In accordance with the top-down procedure described in Section 2.2.1, the constraints for the variables were defined. The constraints for this system were taken from Asmar (1991). The constraints for the measured variables are given in Table 3.1 and the constraints for the inputs are given in Table 3.2. It was assumed that all constraints were hard constraints.

Table 3.1: Ranges for the measured variables, taken from Asmar (1991). Notice that the unit for the levels is m^3. This is because the volumetric liquid hold-up is controlled rather than the level.

Variable	Unit	Description	Lower boundary	Upper boundary
L_1	[m^3]	Level LP evap.	2.9	6.4
L_2	[m^3]	Level IP evap.	0.6	1.6
L_3	[m^3]	Level HP evap.	6.9	13.7
P_1	[bar]	Pressure LP evap.	0	2
P_2	[bar]	Pressure IP evap.	3	6
P_3	[bar]	Pressure HP evap.	12	18
TP_1O	[K]	Temperature outlet LP evap.	200	300
FL_2	[kg/s]	Liquid flow rate to LP evap.	0	5.47
FL_3	[kg/s]	Liquid flow rate to IP evap.	0	7.18
FG_2	[kg/s]	Gas flow rate from IP evap.	0	6.31

Table 3.2: Ranges for the input variables, taken from Asmar (1991)

Variable	Unit	Description boundary	Lower boundary	Upper
XV_2	[-]	Valve 2 opening	0	1
XV_3	[-]	Valve 3 opening	0	1
N	[-]	Scaled shaft rotation speed	0.9	1.1
XV_1	[-]	Valve 1 opening	0	1
FCP_3	[J/sK]	Flow rate and heat cap. of air	116	348

The cost function for the refrigeration cycle gives the economic trade-off between the energy consumption of the compressors and the recovery of valuable molecules on the process side, as indicated by the exit temperatures of the process stream from the evaporators, TP_1O and TP_2O. One might be tempted to use the transferred energy instead of the temperature in the cost function to have the same units (J/s) for each term, but this is not correct. The recovery of valuable molecules is temperature dependent, with lower temperatures giving better recovery. Since temperature is a state function, the final temperature of the fluid is independent on the thermodynamic path taken to reach it.

The cost function becomes

$$J = p_W \cdot W_{tot} + p_{TP_1O} \cdot TP_1O + p_{TP_2O} \cdot TP_2O \tag{3.38}$$

where p_i is the marginal profit associated with each term. If the equation is linearized, the marginal profits can be expressed as

$$\Delta J = p_W \cdot \Delta W_{tot} + p_{TP_1O} \cdot \Delta TP_1O + p_{TP_2O} \cdot \Delta TP_2O \tag{3.39}$$

such that

$$p_y = \left(\frac{\Delta J}{\Delta y}\right) \tag{3.40}$$

Alternatively, one can write the cost function as

$$J = W_{tot} + \alpha \cdot TP_1O + \beta \cdot TP_2O \tag{3.41}$$

where

$$\alpha = \left(\frac{p_{TP_1O}}{p_W}\right) \tag{3.42}$$

$$\beta = \left(\frac{p_{TP_2O}}{p_W}\right) \tag{3.43}$$

$$\tag{3.44}$$

The exact values of α and β will fluctuate from a day-to-day basis due to changes in the prices of the products, the raw materials and utilities. Real-time optimization (RTO) must be used to keep the values of α and β updated during dynamic operation. Alternatively, one can implement a self-optimizing controller which has the prices as measurements. Such a controller is discussed in Section 3.4.

Assuming that the marginal profit of the compressors only depends on the energy consumption of the compressors (that is to say that operating costs are neglected), p_W can easily be found from the current energy prices. Unfortunately we have no information about the marginal profits associated with the outlet temperatures, so we can not find α and β. That would require detailed information about the effect that the outlet temperatures have on the economics of the real plant.

It is reasonably safe to assume that α is bigger than β by approximately one order of magnitude. This can be assumed since the process flow rate through the second evaporator is much smaller than through the first evaporator. Additionally, the lower outlet temperature leads to a larger marginal profit in the first evaporator since the recovery is higher.

Even though marginal profit data for the plant is not available, some guesses can be made to limit the ranges of α and β. Table 3.3 lists what effect α has on the nominal active constraint set. The same arguments can be used for β, since it effects the cost function in the same way as α does, albeit in a much smaller scale, as discussed above.

It is to be expected that $113.5 < \alpha < 214.5$ since there is observed a trade-off in the real plant. As mentioned, β will have a similar effect on the cost function, though the values will be different. For the remainder of the thesis, it will be used that

$$\alpha = 125$$

and

$$\beta = 1$$

unless stated differently. These values result in reasonable operating conditions. As will be seen later, these values result in an optimal value of $N = 1$ and $XV_1 = 0.5$. Since the range for N is normalized around the operating point, a value of 1 corresponds with the actual operating point used in the real plant.

While it is reasonable to assume that α is bigger than β, it may be unlikely to be 125 times larger. The values were chosen because they give an interesting case with two unconstrained degrees of freedom for optimization. A more likely case with $\frac{\alpha}{\beta} \approx 10$ would result in only one degree of freedom, N, as XV_1 would be fully open. Such a case is considered in the alternative process described in Section 3.4.

3.2.2 DOF analysis

Applying the method described in Section 2.1.5 to the model described in Chapter 3.1, the steady-state degrees of freedom can be found.

Potential degrees of freedom

There are a total of 8 process specifications: 2 unique heat loads, one for each evaporator; 2 pressures; 2 levels which must be controlled; 1 subcooling and 1 superheating. Note that the intermediate pressure is not a process specification since it is given indirectly through the heat loads.

Table 3.3: Optimal active constraints for different values of α. The arrows indicate whether the active constraint is on its upper limit (↑) or lower limit (↓).

Case	Constraints	Comment
$\alpha < 0$	-	Not possible. The recovery of valuable molecules becomes better with decreasing temperature, so the marginal profit term p_{TP_1O} must be negative. Since p_W is negative (energy costs money), α must be positive.
$\alpha < 63.5$	$N \downarrow$ $FCP_3 \downarrow$ $P_2 \uparrow$	Both N and FCP_3 at their lower limits. This means that the energy cost of the compressor is so large that the best strategy is to shut off the entire refrigeration cycle in order to save energy. This does not make practical sense, since the refrigeration cycle was considered economically feasible enough to be built.
$\alpha < 85$	$N \downarrow$ $FCP_3 \uparrow$ $P_2 \uparrow$	Same as above. The decrease in outlet temperature due to having FCP_3 fully open now outweighs the energy cost of the slightly increased compression cost. The overall compression cost is still high, which is why the compressor speed is at its lower boundary. The transition between this region and the previous one seems to be discontinuous, or at least very steep.
$\alpha < 113.5$	$FCP_3 \uparrow$ $P_2 \uparrow$	A trade-off between the energy consumption of the compressors and the recovery of valuable molecules has been achieved.
$\alpha < 140.5$	$FCP_3 \uparrow$	Same as above. It becomes increasingly important to cool TP_1O as much as possible, which is why XV_1 opens gradually to lower the overall temperature in the system. This leads to somewhat increased compression cost due to larger pressure ratios.
$\alpha < 214.5$	$FCP_3 \uparrow$ $XV_1 \uparrow$	Same as above
$\alpha > 214.5$	$N \uparrow$ $FCP_3 \uparrow$ $XV_1 \uparrow$	The marginal compression cost p_W is negligible compared to the marginal profit term p_{TP_1O}, so N is at its upper limit to achieve maximum cooling.

These eight process specifications must be controlled using the eight potential degrees of freedom in the system, which can be found using the method summarized in Table 2.1 in Section 2.2.2. See Figure 3.6 for the potential degrees of freedom for the studied process.

Figure 3.6: Potential steady-state DOF for the studied refrigeration cycle

+ 2 choke valves
+ 2 compressors
+ 3 heat exchangers
+ 1 active charge
= 8 potential DOF

Only two choke valves are included, namely XV_2 and XV_3. XV_1 is a vapour valve and not a choke valve, and does therefore not contribute to the degrees of freedom.

Actual degrees of freedom

There are fewer actual degrees of freedom than potential degrees of freedom. Firstly it is noticed that the two compressors are connected to the turbine and driven by the same driveshaft. Since the speed of the two compressors is adjusted simultaneously using the steam turbine, one potential degree of freedom is removed. In the case of the studied process, both evaporators are utilizing the maximum heat transfer potential, so these two process units do not provide actual degrees of freedom. The condenser duty is commonly also maximized (Jensen & Skogestad 2009), but it can be seen that the degree of freedom is kept in the studied case. By manipulating the air flow rate, FCP_3, the heat transfer can be adjusted. The actual degrees of freedom is five, which can be confirmed by counting the number of physical valves in the system. The five MVs in the process include the three valves, the cooling air flow rate and the compressor speed.

Steady-state degrees of freedom

The process has three vessels, two of which must have their liquid levels controlled in accordance with the rules about consistent inventory control proposed by Aske & Skogestad (2009). Since the receiver is the largest vessel, it was chosen to leave the level of the receiver, L_3, uncontrolled. The two smaller inventories have smaller tolerances for level variations, since it must be made sure that the heating coils are always submerged, and thus require tighter level control.

It was chosen to pair XV_2 with L_1 and XV_3 with L_2 according to the "pair close" rule. These pairings were chosen because the valves have the most direct effect on the levels . Other combinations are possible as well, such as pairing XV_3 with L_1 and XV_1 with L_2, but these are not as well suited as the proposed pairings. In the case of pairing XV_3 and L_1, this would have led to local inconsistency Aske & Skogestad (2009). The gain from XV_1 to L_2 is small, so this control scheme would not work very well. Using FCP_3 or N to control the levels is not recommended, as both MVs have little effect on the levels. An illustration of the proposed control structure can be seen in Figure 3.7

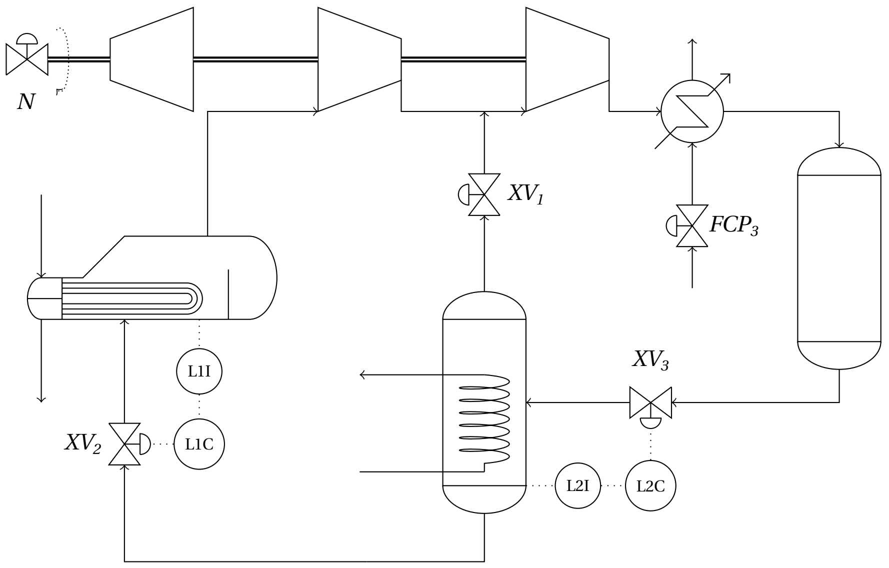


Figure 3.7: Process flow diagram of the studied process with the two added level control structures.

3.2.3 Nominal operating point

The three remaining steady-state degrees of freedom, namely XV_1, FCP_3 and N, can be used to find the nominal point. Using the interior-point algorithm in the `fmincon`-function in MATLAB, the nonlinear optimization problem described in Section 3.2.1 is solved.

The nominal steady-state conditions are shown in Figure B.1 in Appendix B.

The corresponding pressure-enthalpy diagram of the nominal solution can be seen in Figure C.1 in Appendix C. The nominal solution has one active constraint, FCP_3, which is at its upper limit. This active constraint must be controlled, as previously discussed in Section 2.2.1. N and XV_1 are not active and can thus be used for optimization. It is discussed in Table 3.3 why these inputs are not active for the chosen values of α and β.

The COP of the cycle is

$$\text{COP} = \frac{Q_1 + Q_2}{W_1 + W_2} = \frac{1044.2 + 74.5}{288.5 + 545.2} = 1.342 \tag{3.45}$$

The maximum obtainable COP for this cycle is achieved when the compressor duty is minimized and the condenser duty is maximized. The active constraints are $N \downarrow, XV_1 \uparrow, FCP_3 \uparrow$.

$$\text{COP}_{\text{max}} = \frac{Q_1 + Q_2}{W_1 + W_2} = \frac{524.8 + 89.8}{134.0 + 243.3} = 1.629 \tag{3.46}$$

The COP of the nominal solution is worse than the maximum COP. This is caused by the large values of α and β, which weigh TP_1O and TP_2O more heavily. The COP for this cycle is lower than other propylene cycles from the literature (Prapainop & Suen 2006), but this is mainly due to the very low evaporator temperature. For comparison, a single stage cycle with an evaporator temperature equal to T_1 has a COP of 1.06 [2]. The two-stage cycle is thus more efficient than a single-stage cycle.

3.2.4 Self-optimizing control

Using $\alpha = 125$ and $\beta = 1$, the optimal solution contains two unconstrained degrees of freedom, namely N and XV_1. The selection matrix **H** will therefore have dimensions $2 \times n_y$, where n_y is the number of measurements. The expression for the controlled variable **c** is

$$\mathbf{c} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \mathbf{H} \cdot \mathbf{y} = \begin{bmatrix} H_1 \\ H_2 \end{bmatrix} \cdot \mathbf{y} \tag{3.47}$$

H is calculated using the exact local method described in Section 2.2.2. As it is relatively difficult to calculate the exact derivatives needed for $\mathbf{F}$, $\mathbf{G_y}$ and $\mathbf{J_{uu}}$, numerical finite difference approximations were used instead. $\mathbf{W_{n_y}}$ was

[2]Calculated with CoolPack 1.5
`http://en.ipu.dk/Indhold/refrigeration-and-energy-technology/coolpack.aspx`

constructed by assuming that the standard deviation of the implementation error for each variable is 1% of the nominal value.

Five major disturbances have been identified, namely the inlet flow rates and temperatures of the process streams in the two evaporators, as well as the inlet temperature of the cooling air in the condenser. The standard deviations of the disturbances are assumed to be

$$\mathbf{W_d} = \mathrm{diag}\left(\begin{bmatrix}\sigma_{TP_1I} & \sigma_{TP_2I} & \sigma_{TP_3I} & \sigma_{FCP_1} & \sigma_{FCP_2}\end{bmatrix}\right)$$

$$\mathbf{W_d} = \mathrm{diag}\left(\begin{bmatrix}2 & 2 & 3 & 4 & 1\end{bmatrix}\right)$$

Selection of controlled variables

Using the bidirectional branch and bound algorithm, the best subset of variables can be found. The corresponding average loss is calculated using Equation 2.28. Figure 3.8 shows the average loss as a function of the number of measurements. Only measurable variables such as the inputs, temperatures, pressures and flow rates have been included in the subset of measurements. Other state variables such as enthalpy have been omitted as they are not physically measurable in the real plant. It can be seen from Figure 3.8 that the aver-

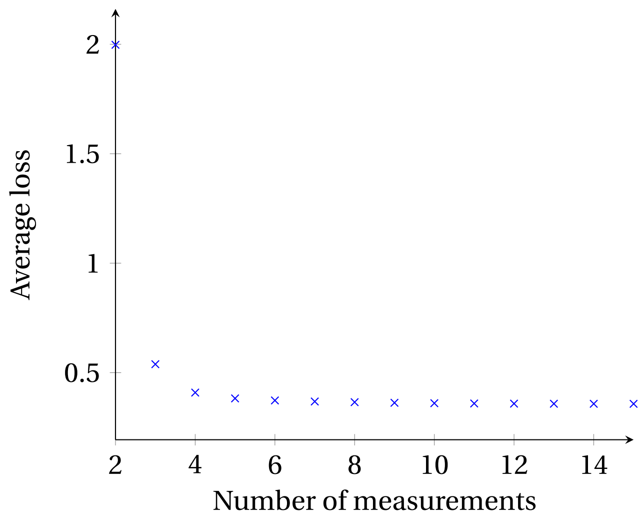


Figure 3.8: Average loss as a function of the number of measurements

age loss decreases exponentially. In order to successfully reject all major disturbances, it is necessary to use at least as many measurements as there are major disturbances. This is why the loss in Figure 3.8 becomes relatively large when less than five measurements are used. The improvement of using more than five measurements is small, especially considering that the losses in Figure 3.8 are only a fraction of the nominal value of the cost function, which is $2.934 \cdot 10^4$ cost units.

Each additional sensor increases the complexity and the investment cost of the control structure. The probability of failure of the controller also increases with the number of sensors. For these reasons, as few measurements as possible should be used. It was decided that 5 measurements would give sufficiently low loss in this case.

Using the partial bidirectional branch and bound algorithm, it is found that the best subset of five measurements is

$$XV_1,\ P_1,\ P_3,\ FG_1 \text{ and } FG_3$$

It is observed that the best subset of measurements usually includes flow- and pressure measurements. Temperature measurements are generally not included because their implementation errors are larger than the implementation errors of the corresponding pressure measurements. Consequently the same information about the system can be obtained with higher accuracy by using pressure measurements rather than temperature measurements. The same seems to be at least partially true for flow measurements, which are generally given higher priority than the corresponding pressure measurements. Important measurements such as FG_3 are sometimes duplicated by including measurements of FL_3 or FL_4 in addition. This reduces the average implementation error for the measurement. Since the average loss decreases so rapidly in Figure 3.8, it seems that only a selected few measurements are required to provide information about the entire process. Additional measurements are used to reduce the implementation error, which is why these "duplicate" measurements are common when using more than a handful of measurements.

Calculating the selection matrix H

This subset of measurements gives the following selection matrix **H** when using the explicit expression from Equation 2.35.

$$\mathbf{H} = \begin{bmatrix} -78.44 & -172.69 & -7.40 & 86.19 & 36.10 \\ 4.49 & 7.67 & 0.48 & -4.56 & -1.76 \end{bmatrix}$$

The corresponding setpoints for the controllers are found from the nominal point $\mathbf{c}_{\text{set}} = \mathbf{H} \cdot \mathbf{y}^{\text{nom}}$

$$\mathbf{c}_{\text{set}} = \begin{bmatrix} 111.38 \\ -5.14 \end{bmatrix}$$

Optimality of controlled variables

The actual losses for some selected disturbances are given in Table 3.5. The actual losses are found by subtracting the optimal cost from the cost that is obtained by holding c constant, see Equation 2.10. These particular disturbances were chosen because their magnitudes are well within one standard deviation from the mean, and they thus represent typical disturbances in the real plant.

Table 3.5 also shows the losses that are obtained by holding all inputs constant at their nominal values. This corresponds to choosing a set of two measurements, N and XV_1, with $\mathbf{H} = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]$. Since there is not added any implementation error to the simulation, the losses will be smaller than if they were included. In some cases the derived controller actually performs worse than a constant input policy, but overall the average loss is reduced notably when using self-optimizing control.

Table 3.5: Losses for self-optimizing control versus constant setpoint policy for some disturbances. Units for the losses are the same as for the cost.

Variable	Disturbance	Self-optimizing	Constant inputs
TP_1I	+1K	$1.80 \cdot 10^{-3}$	$164.51 \cdot 10^{-3}$
TP_2I	+1K	$0.30 \cdot 10^{-3}$	$0.30 \cdot 10^{-3}$
TP_3I	+1K	$0.35 \cdot 10^{-3}$	$46.91 \cdot 10^{-3}$
FCP_1	$+2\text{W}\,\text{K}^{-1}$	$98.30 \cdot 10^{-3}$	$121.4 \cdot 10^{-3}$
FCP_2	$+0.5\text{W}\,\text{K}^{-1}$	$0.70 \cdot 10^{-3}$	$0.70 \cdot 10^{-3}$

As discussed in Section 2.2.2, it is desired that the controlled variables should be insensitive to disturbances and implementation errors, and that the sensors should be easy to install. The latter criterion has been ensured by only using temperature-, pressure- and flow measurements in the construction of $\mathbf{H}$, since these measurements generally are cheap and easy to implement. The sensitivity to disturbances and implementation error is ensured by minimizing the average loss by using the exact local method.

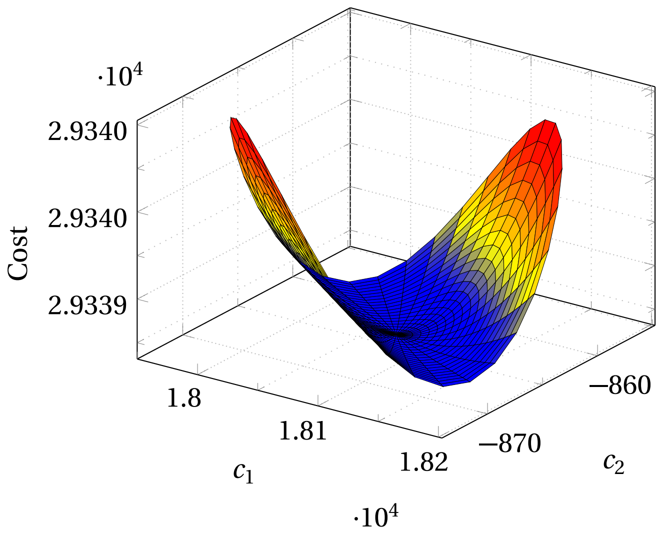
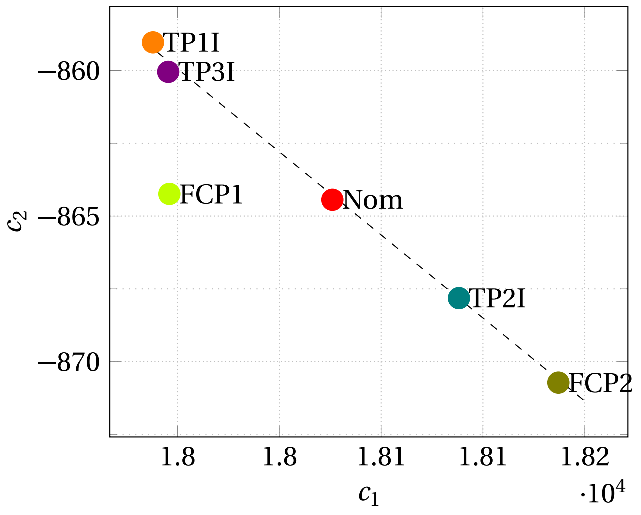


Figure 3.9: Cost as a function of the controlled variables

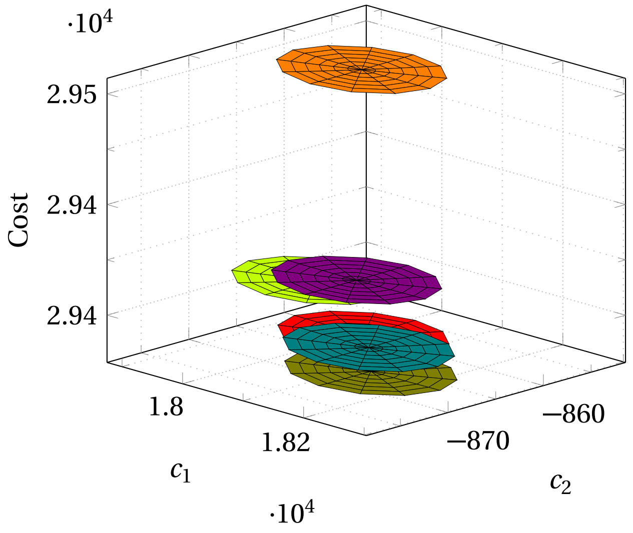
Figure 3.9 shows the cost as a function of the controlled variables. The optimal value of the CV is the minimum of a elliptic hyperboloid. The hyperboloid curves much more in one direction than the other. A seemingly flat valley runs diagonally between the two major axis. Upon closer inspection it is seen that the change in the cost function around the minimum is very small, only about 0.1% of the nominal minimum in the plotted region. This means that for all practical purposes, the cost function surface is more or less flat around the optimum, as can be seen from Figure 3.10(b). Losses due to implementation errors are thus expected to be very small. Assuming that the implementation errors are no bigger than 1% of the nominal value, the resulting deviation in the calculated CV is at most 1%. As seen from Figure 3.10, the consequent losses should be even smaller than the losses caused by disturbances.

Figure 3.10 shows the same plot for a set of disturbances. The same disturbances as those presented in Table 3.5 were used. The cost function for each disturbance has been plotted as a function of the controlled variables, each being centred around the optimal value for the disturbance.

It can be seen from Figure 3.10(a) that the optimal setpoints of the controlled variables are grouped relatively tightly together. This is another criterion for a good controlled variable, as discussed in Section 3.2.4. The loss associated with keeping the controlled variable at a constant setpoint is accordingly small. For the five studied disturbances it is observed that a 1°C increase in TP_1leads



((a)) Optimal values of **c** for the studied disturbances.



((b)) Cost function surfaces for the studied disturbances.

Figure 3.10: Optimality of the controlled variables as illustrated by the cost function surfaces of the disturbances.

to the largest loss, followed by a 2W/K increase in FCP_1. It is to be expected that these two disturbances lead to the largest losses as they have a large, direct effect on the cost function. The two disturbances associated with the IP evaporator, namely TP_2I and FCP_2, lead to much smaller losses since this term of the cost function is weighted less than the corresponding term associated with the first evaporator. In other words, α is much larger than β in Equation 3.41

It is worth noting that the optimal operating points (i.e. the centres of the paraboloids) for all disturbances except for the 2W/K disturbance in *FCP_1* lie on a straight line. The straight line incidentally goes through the valleys of the cost function surfaces. It is therefore expected that these four disturbances have smaller losses associated with them than *FCP_1*. Since the optimal value of **c** for a disturbance in *FCP_1* lies on a line perpendicular to the cost function valleys, the associated loss must be higher. Indeed, this is observed since the loss is two orders of magnitude larger than for any of the other disturbances.

3.3 Dynamic simulation

This chapter contains the results from the dynamic analysis of the system, including the selection of the input-output pairings and the controller tunings. Finally, the dynamic performance of the controllers is studied.

3.3.1 Input-output pairings

The best pairings of the inputs and outputs are found from the RGA, $\mathbf{\Lambda}$. The steady-state gain matrix, $\mathbf{G}$, is

$$\mathbf{G} = \begin{bmatrix} 1.543 & -0.081 \\ -0.076 & 0.005 \end{bmatrix} \cdot 10^5$$

Which gives the following RGA

$$\mathbf{\Lambda} = \begin{bmatrix} 3.783 & -2.783 \\ -2.783 & 3.783 \end{bmatrix}$$

In accordance with the rules in Section 2.3.1, it was chosen to pair on the positive diagonal elements, to avoid the negative elements on the off-diagonal. The large elements ($\lambda_{ii} > 1$) on the diagonal mean that the gain is reduced when the loops are closed, but at least the sign is unchanged. N was paired with c_1 and *XV_1* was paired with c_2

3.3.2 Controller tuning

The dynamic open-loop responses of the controlled variable c_2 to a 1% step in *XV_1* is shown in Figure 3.11. First order approximations of the responses are needed to tune the corresponding PI controllers. Since the system has a non-linear response, fitting a regular n-th order approximation to the response does not yield a very good fit. The half-rule model reduction method can consequently not be used. It was decided to graphically fit the first order approximation directly to the non-linear response. The resulting first order approximation is plotted as the dashed red curve in Figure 3.11.

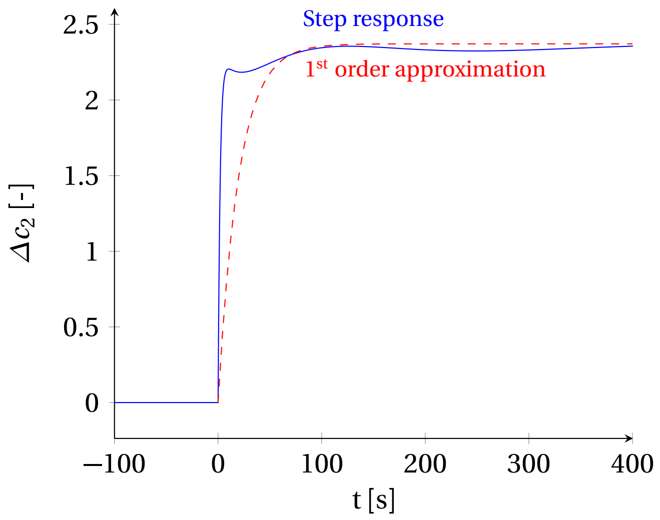


Figure 3.11: Open-loop response of Δc_2 to a 1% step in XV_1.

The approximated first-order transfer function from XV_1 to c_2 is

$$G_{XV_1} = \frac{542}{20s+1}$$

The resulting controller settings are found from the SIMC rules

$$\tau_c = 2$$
$$K_c = \frac{1}{k} \cdot \frac{\tau_1}{\tau_c + \theta} = 0.0185$$
$$\tau_i = \min(\tau_1, 4(\tau_c + \theta)) = 8$$

The loop is closed and the step response from N to c_2 is found. The response to a 1% increase in N, along with the first order approximation, can be seen in Figure 3.12.

The approximated first-order transfer function from N to c_1 is

$$G_N = \frac{2.33 \cdot 10^4}{10s+1}$$

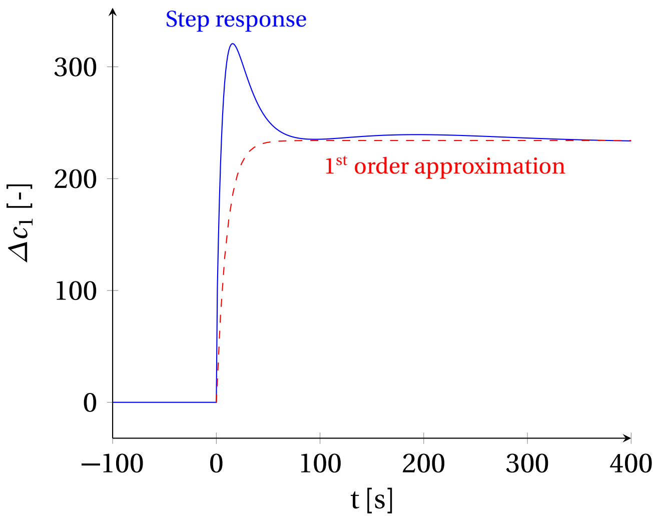


Figure 3.12: Partially open-loop response of Δc_1 to a 1% step in N. The loop between Δc_2 and XV_1 has been closed.

The resulting controller settings are found from the SIMC rules from Section 2.3.2

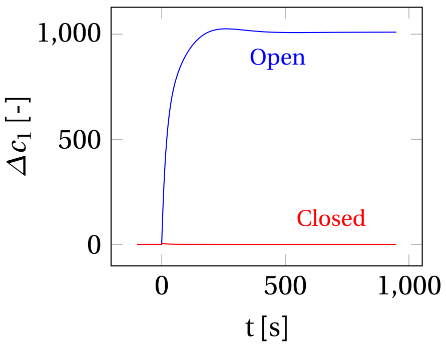
$$\tau_c = 2$$
$$K_c = \frac{1}{k} \cdot \frac{\tau_1}{\tau_c + \theta} = 2.14 \cdot 10^{-4}$$
$$\tau_i = \min(\tau_1, 4(\tau_c + \theta)) = 8$$

3.3.3 Dynamic behaviour of the controllers

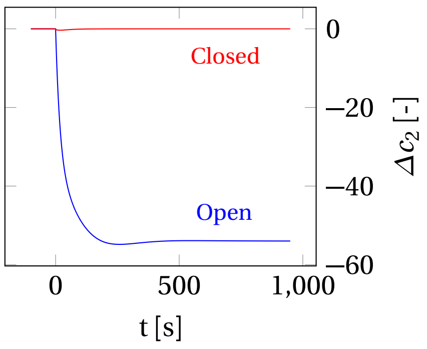
The open-loop and closed-loop responses to a 1°C increase in TP_1I can be seen in Figure 3.13. The responses to other disturbances are very similar.

It is observed that the losses are more or less identical in Figure 3.13(e). Only a very small decrease in steady-state loss is observed when closing the loop, as previously calculated and shown in Table 3.5.

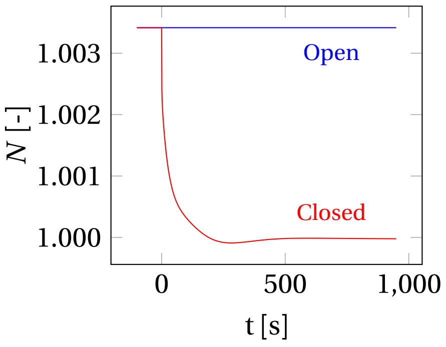
The relatively large loss that occurs at $t = 0$ only slowly goes to zero. The loss has such a large time constant because it takes time for the state variables to reach their new set points after the inputs were changed. Even if the controllers were tuned more aggressively such that they were able to reject the disturbance almost immediately and $\Delta c = 0$, the loss would still look similar due to the slow measurements. The absolute integrated losses are included in Fig-



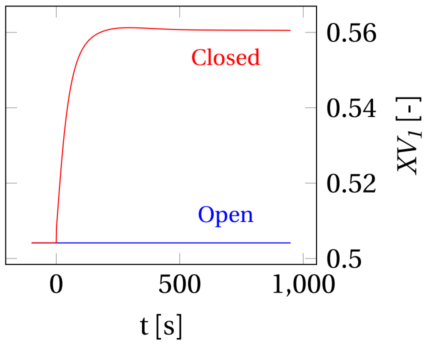
((a)) Response of Δc_1 to a step in TP_1I



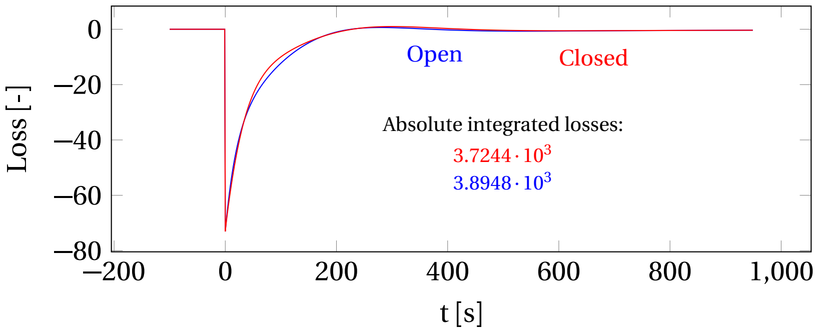
((b)) Response of Δc_2 to a step in TP_1I



((c)) Response of N to a step in TP_1I



((d)) Response of XV_1 to a step in TP_1I



((e)) Losses due to a step in TP_1I

Figure 3.13: Closed- and open-loop responses to a 1°C increase in TP_1I

ure 3.13(e), and it can be seen that they are very similar, with the closed loop loss being slightly smaller. The majority of the loss comes from the immediate peak, which can not be prevented with control.

Since the controllers are limited in their usefulness by the large time delay, and because the steady state loss is only marginally better when the loop is closed, it seems that a good control strategy is to not control the process at all. By keeping the controllers at their nominal set-points, the loss stays acceptably low. Alternatively, the controllers can be used to control the pressures in the vessels or the process outlet temperatures directly, thus giving better controllability over the product specifications on the process side.

3.4 Alternative process model

This chapter considers a alternative process layout. Whenever it is referred to the original case, the process described in the previous chapters is meant.

The model considered up to this point in the thesis does not take into consideration what happens on the process side of the plant. It was assumed that the inlet conditions to the two evaporators were independent of the refrigeration cycle. The inlet flow rates and temperatures were assumed to be normally distributed around the nominal operating conditions. Disturbances are thus random and independent. However, this is not the case in the real plant. In conversation with Exxon it was suggested that the inlet temperature to the evaporators should be dependent on the suction pressure to the compressor. This behaviour is observed in the real plant since there is a loop connecting the outlet to the inlet. We do not know exactly what the nature of these unspecified processes are, so they are represented by the boxes Π_1 and Π_2 in Figure 3.14.

As a first approach to simulate the real plant it was suggested to maintain a constant temperature difference between the process stream inlet and the refrigerant in the evaporator. The underlying assumption that the heat losses in the unspecified processes Π_1 and Π_2 are equal to the transferred heat in the evaporators is not necessarily good, but the assumption is reasonable over a small temperature range. As a first attempt the temperature differences were found from the nominal temperature differences from the steady-state solution of the original case. Given the nominal temperatures shown in Figure B.1, the temperature differences were calculated to be

$$\Delta T_1 = TP_1I - T_1 = 12.8°\text{C}$$

and

$$\Delta T_2 = TP_1I - T_2 = 15.0°\text{C}$$

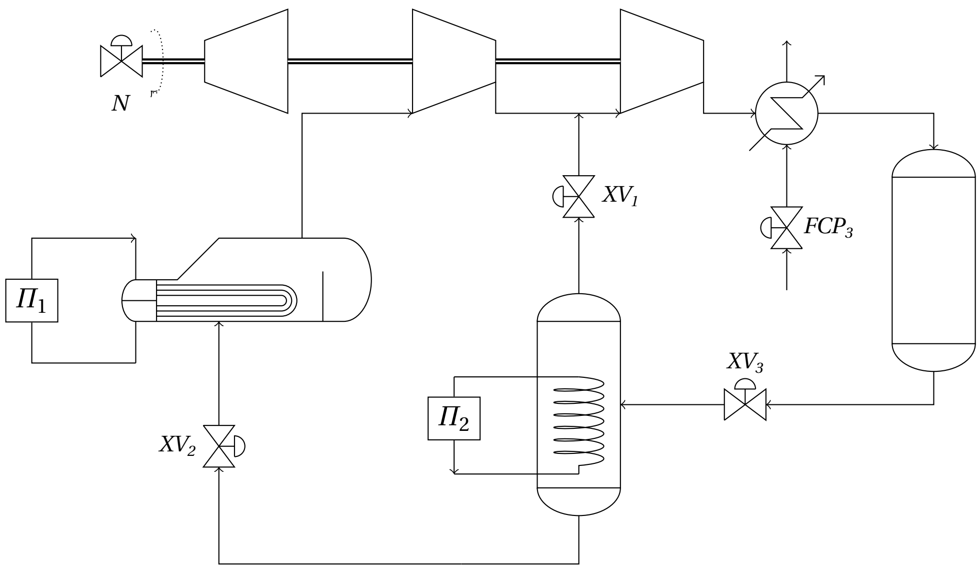


Figure 3.14: Process flow diagram of the alternative process layout with the two added level control structures.

Since the objective remains unchanged, the cost function for the optimization is as previously defined in Section 3.2.1. However, the alteration of the model necessitates new values of α and β to be found. Like in the original case, the lack of economic data from the plant makes it difficult to calculate the exact parameters. A quick survey of possible combinations of α and β revealed that only three constraint regions exist for the alternative model, assuming that other disturbances are relatively small and do not cause large variations in e.g. the pressure levels. The three observed regions are:

1. $FCP_3 \uparrow$, $XV_1 \uparrow$, $N \downarrow$ for

$$\begin{aligned} \beta &\geq 15 - \frac{3}{5}\alpha \\ \alpha &\geq 0 \quad , \quad \beta \geq 0 \end{aligned}$$

2. $FCP_3 \uparrow$, $XV_1 \uparrow$, $N \uparrow$ for

$$\begin{aligned} \beta &\leq 30 - \frac{6}{7}\alpha \\ \alpha &\geq 0 \quad , \quad \beta \geq 0 \end{aligned}$$

3. $FCP_3 \uparrow$, $XV_1 \uparrow$ for values of α and β in the transitional region.

The system is fully constrained in region 1 and 2, with N being at the lower and upper boundary, respectively. For combinations of α and β in region 3, the system has one degree of freedom for optimization. The optimal value of N as a function of α and β can be seen in Figure 3.15

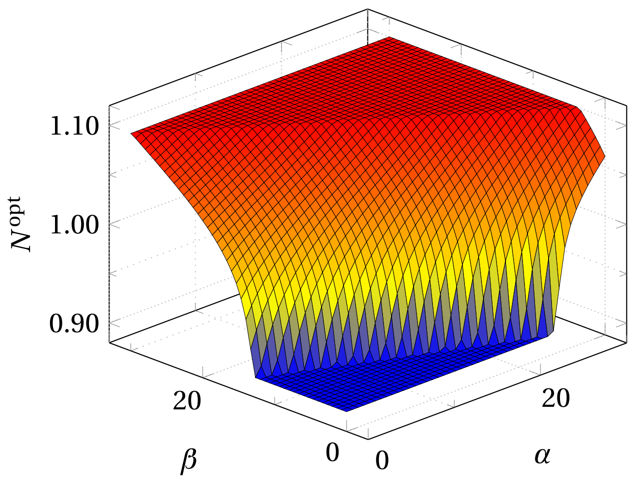


Figure 3.15: Optimal value of N as a function of α and β

It is observed that FCP_3 is at its upper constraint in all three regions. This was not observed in the original case. As discussed in Section 3.2.3, it is optimal to use as little cooling as possible if the only concern is the energy consumption of the compressors. A small cooling load results in better pressure ratios in the compressors, consequently lowering the energy consumption. In the alternative case there is a hidden constraint in the form of the specified temperature difference between the process stream and the evaporator. Whereas it was possible to let the temperature difference approach the pinch point in the original case (which was indeed observed for small α and β), the fixed temperature differences of the alternative case results in a larger heat load. In order to satisfy the overall energy balance of the cycle, the increased heat load must be balanced by an increase in the condenser duty. In order to reach the specified temperature differences of 12.8°C and 15.0°C, the condenser duty must always be at the upper limit.

It is also observed that XV_1 is at its upper boundary in all three constraint regions. Contrary to what was observed in the original case, it is suboptimal to keep the valve partially closed. In the original case it was possible to achieve a trade-off between the increased compressor duty and the lowered outlet temperature from the LP evaporator by partially closing XV_1. The increased flow to the LP evaporator results in a larger temperature difference between the process stream and the refrigerant, thus giving better heat transfer and ultimately a lower outlet temperature. In the alternative case it is not possible to increase the temperature difference since it is locked. Instead, the outlet temperature must be lowered by lowering the temperatures of the entire cycle. This is achieved by increasing the compressor speed N. XV_1 can no longer be used to create additional driving forces in the evaporator, and the trade-off thus disappears. The optimal strategy is consequently to keep XV_1 fully open at all times to avoid unnecessary compression of supersaturated gas.

N^{opt} increases as the effect of the outlet temperatures is weighted more heavily through the parameters α and β in the cost function. This is to be expected, since more energy must be put into the system to lower the temperature of the cycle. Since the condenser is already at its upper constraint, the energy must come from the compressors. It is noteworthy that α has a bigger impact on the cost function than β. This can be seen from Figure 3.15, which shows that a higher compressor speed is required to operate at optimum conditions for a given value of α than for the same value of β. This follows readily from the fact that the outlet temperature from the IP evaporator is much higher than the outlet temperature from the LP evaporator.

For the remainder of the chapter, the values $\alpha = 15$ and $\beta = 2.5$ will be used. This gives a $\frac{\alpha}{\beta}$-ratio of 6, which is more realistic than the previous ratio of |125.

3.4.1 Self-optimizing control

It was suggested that the cost parameters α and β should be included as measurements in the self-optimizing controller. The advantage of such a controller is that it would react to prize changes immediately. Re-optimization and parameter tuning can be done less frequently for such a controller. A similar controller is discussed by Jäschke & Skogestad (2011). Since the self-optimizing controller is based on a local optimization method, this method can only be used if it assumed that the fluctuations in α and β (or any other disturbance, for that matter) are normally distributed around an expected value. Once the prices start changing permanently, e.g. due to a change in the market, the controller will give a persistent offset from the optimum and the parameters must be updated. A self-optimizing controller containing economic data must thus be updated every few days or so, but it is regardless an improvement over a controller where the economic parameters are assumed to be constant.

In the original case, it was assumed that the disturbances were not measured. Instead, information about the disturbances was provided indirectly by the other state variables through the sensitivity matrix $\mathbf{F}$. However, since α and β do not effect the other state variables, these disturbances must be measured directly. The augmented sensitivity matrix for the alternate case thus becomes

$$\mathbf{F}^* = \begin{bmatrix} \ddots & & & \vdots & \vdots \\ & \mathbf{F} & & \mathbf{f}_\alpha & \mathbf{f}_\beta \\ & & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 1 & 0 \\ 0 & \cdots & 0 & 0 & 1 \end{bmatrix} \tag{3.48}$$

Here, $\mathbf{f}_\alpha = \left(\frac{\partial \mathbf{y}^{\text{opt}}}{\partial \alpha}\right)$ and $\mathbf{f}_\beta = \left(\frac{\partial \mathbf{y}^{\text{opt}}}{\partial \beta}\right)$ are the optimal sensitivities of the states to disturbances in α and β. $\mathbf{F}$ is the sensitivity matrix found in the original case. Similarly, the augmented linearized model $\mathbf{G_y}^*$ is

$$\mathbf{G_y}^* = \begin{bmatrix} \mathbf{G_y} \\ 0 \\ 0 \end{bmatrix} \tag{3.49}$$

It is assumed that α and β have zero implementation error, since such an error would be impossible to reduce by measuring other variables ($f_{i \neq \alpha,\beta} = 0$). Since α and β are not limited by the accuracy of any physical equipment that could cause measurement error, this assumption seems reasonable. $\mathbf{W_{n^y}}^*$ thus becomes

$$\mathbf{W_{n^y}}^* = \begin{bmatrix} w_{y_1} & & & & & \\ & w_{y_2} & & & & \\ & & \ddots & & & \\ & & & w_{y_n} & & \\ & & & & 0 & \\ & & & & & 0 \end{bmatrix} \tag{3.50}$$

The choice of the variances σ_α and σ_β is as non-trivial as the selection of α and β themselves. Statistic estimation of the variances can be done, but this requires economic data. In the following section, it will be assumed that σ_i is 10% of the nominal value of i.

Since the measurements of α and β are not physical, there is little cost and risk associated with them. It was previously found that five measurements gave a good trade-off between cost and accuracy. The two measurements of α and β are added on top of that, resulting in a total of seven measurements.

Using

$$\sigma_\alpha = 10\% \cdot \alpha \quad \alpha = 15$$
$$\sigma_\beta = 10\% \cdot \beta \quad \beta = 2.5$$

it was found that the best subset of seven measurements includes

$$P_A,\ P_1,\ FG_1,\ FL_2,\ FL_3,\ \alpha \text{ and } \beta$$

The selection matrix **H** and the corresponding set-point **c** is calculated from Equation 2.35. The step response of c to a 1% step in N, along with the approximated first order response can be seen in Figure 3.16.

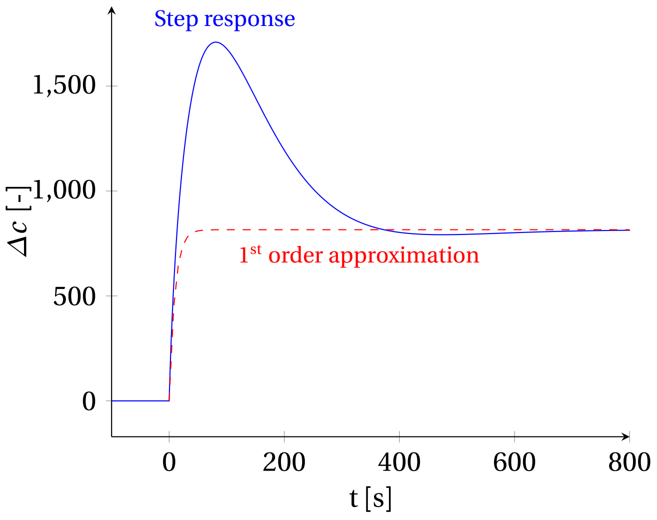


Figure 3.16: Open loop step response

The controller was tuned using the SIMC rules, resulting in the following controller parameters

$$\begin{aligned} \tau_c &= 10 \\ K_c &= 2.5 \cdot 10^{-5} \\ \tau_i &= 20 \end{aligned}$$

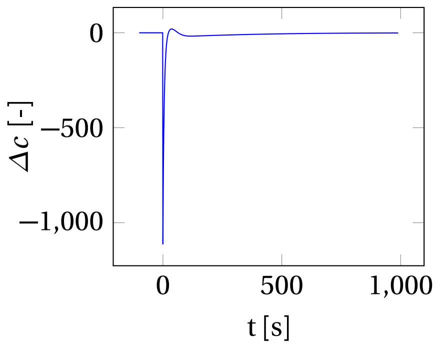
The closed loop response to a disturbance in α is seen in Figure 3.17. The magnitude of the disturbance in α is $+1$.

It can be seen from Figure 3.17(c) that the immediate loss is relatively large compared to the steady-state loss. Despite the controller being fairly fast (c is driven to c_{set} after less than 100 seconds), it takes more than 1000 seconds for the loss to approach zero. This is due to the slow dynamics of the system. The adjustment of the input N leads to a perturbation of the states, which in turn causes c to change.

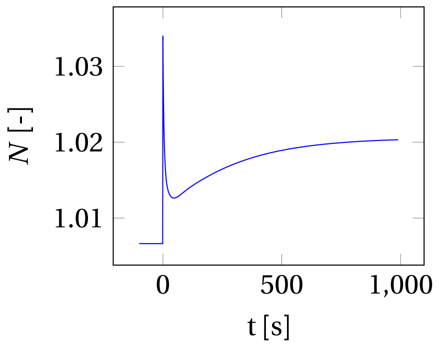
Due to the large time constant, the integrated loss becomes significant. It takes over 1000 seconds before the self-optimizing controller outperforms a constant input controller (not shown) in terms of the loss.

In order to overcome the issues associated with the large time constant of the system, an alternative controller was derived. This controller does not use any of the slow plant measurements, but instead relies entirely on measurements of N, α and β. Obviously such a controller is not able to reject disturbances, but this might be acceptable if the expected disturbances are relatively small compared to the price variations.

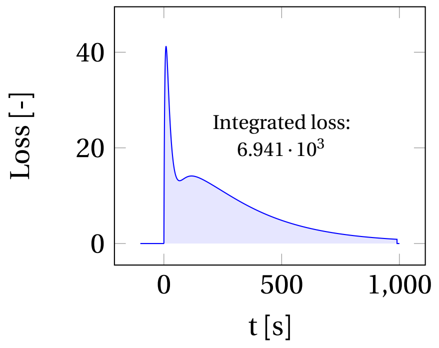
Figure 3.18 shows the closed loop responses of the derived controller. As can be seen, it rejects a disturbance in α faster than the controller from Figure 3.17 since N is set to the optimal value N^{opt} almost immediately (the time constant of the input N is two seconds). However, the shorter time required to reach zero loss is at the expense of a somewhat larger immediate loss. The five extra plant measurements in the first controller measure the loss caused by the set-point change in N, and partially reject it. This information is not available in the second controller, thus causing a larger initial peak. The absolute integrated loss over the first 1000 seconds is more or less the same, being marginally in favour of the second controller. Which of the two controllers is better thus depends on the expected frequency of the disturbances. If disturbances are very frequent, the first controller is better due to somewhat lower immediate loss. If disturbances are infrequent, that is to say the mean time between disturbances is more than approximately 1000 seconds, then the second controller is better. Since the second controller can not reject disturbances other than price changes, the first controller is likely to be the better choice in a real application.



((a)) Response of $\varDelta c$ to a step in α

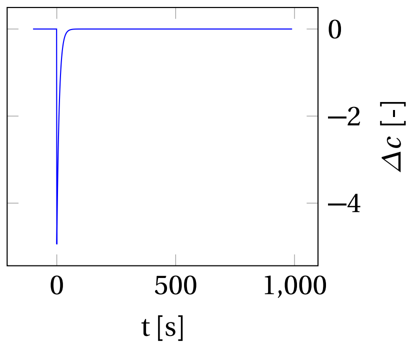


((b)) Response of N to a step in α

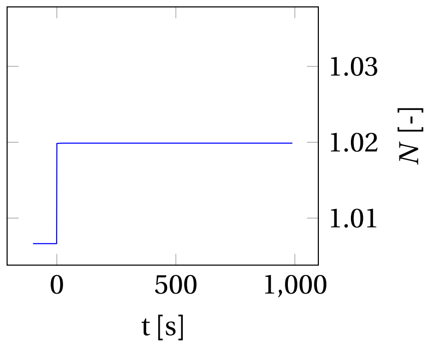


((c)) Loss due to a step in α

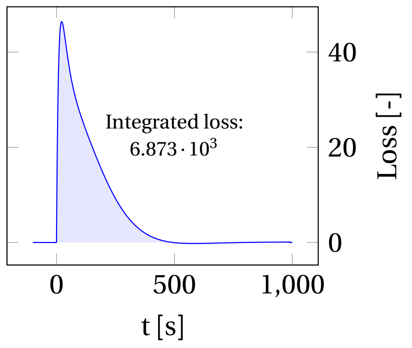
Figure 3.17: Closed-loop response to a +1 step in α. CV includes α, β and five plant measurements.



((a)) Response of $\varDelta c$ to a step in α



((b)) Response of N to a step in α



((c)) Loss due to a step in α

Figure 3.18: Closed-loop response to a +1 step in α. CV includes only α and β.

CHAPTER 4

CONCLUSION

A two-stage refrigeration cycle was modelled and optimized. Due to the lack of economic data from the real plant, it was not possible to derive exact parameters for the cost function. After an investigation of possible candidates, the chosen set of parameters resulted in an optimal steady-state solution containing two degrees of freedom. As expected, the condenser duty was at the upper constraint, leaving the compressor speed and the valve opening for control. Two self-optimizing controllers were derived and implemented. Analysis of the steady-state disturbance rejection showed that the derived controllers reduced the steady-state loss compared to a constant input policy. However, since the cost surface is flat around the optimum, both the self-optimizing controllers and the constant input policy gave very small losses. The self-optimizing controller outperformed the constant set-point policy by one or two orders of magnitude, but this hardly makes a difference since the open-loop loss is negligible. Studies of the dynamic performance of the controllers revealed that the large time delay inherent to the system lead to somewhat large initial losses to disturbances. Only after the process settles after around 1000 seconds, the controlled system consistently outperforms the uncontrolled system.

An alternative case with constant temperature differences between the process side and the evaporators was also investigated. The degree of freedom associated with the valve opening to the mixing node disappeared, as the trade-off between the energy consumption and the outlet temperature was lost due to the fixed temperature differences. Using the remaining degree of freedom, being the compressor speed, a self-optimizing controller was developed. This controller also included measurements of the economic parameters α and β in addition to regular plant measurements. The plant thus remains optimal despite changes in the prices, without having to re-calculate the controller set-points. Similarly to what was observed for the original case, it was found that the derived controller did improve the disturbance rejection somewhat. However, the decrease in loss was relatively small compared to the nominal value of the cost function. A second controller was derived, using only measurements of α and β. Said controller had quicker rejection of variations in the prices. This comes at the cost of no rejection of process disturbances. With this in mind, the first controller is most likely the best choice in practice.

The overall conclusion from this thesis is that self-optimizing control can be applied to two-stage refrigeration cycles with some success. Due to the very flat shape of the cost surface, it is not strictly necessary to control the system directly in order to achieve a satisfactory degree of optimality. By having constant set-points equal to the nominal solution, the steady-state loss is less than 0.1% of the optimal cost.

4.1 Further work

The model can be improved upon in some ways:

- Derive a model for the interaction between the inlet of the process streams to the evaporators and the refrigerant inside the evaporators. The current approach using constant temperature differences is a simplified version of the actual plant behaviour.
- Use model predictive control to predict the optimal inputs given the disturbances to eliminate the effect of the large time delays in the system.
- Obtain proper economic data from a real refrigeration cycle to estimate α, β and their variances.

BIBLIOGRAPHY

Alstad, V. & Skogestad, S. (2007), 'Null space method for selecting optimal measurement combinations as controlled variables', *Industrial & engineering chemistry research* **46**(3), 846–853.

Alstad, V., Skogestad, S. & Hori, E. S. (2009), 'Optimal measurement combinations as controlled variables', *Journal of Process Control* **19**(1), 138–148.

Angus, S., Armstrong, B. & De Reuck, K. (2013), *International Thermodynamic Tables of the Fluid State: Propylene (Propene)*, number v. 7 *in* 'IUPAC chemical data series', Elsevier Science.

Aske, E. M. B. & Skogestad, S. (2009), 'Consistent inventory control', *Industrial & engineering chemistry research* **48**(24), 10892–10902.

Asmar, B. N. (1991), 'Control of a two-stage refrigeration cycle'.

Chao, J. & Zwolinski, B. J. (1975), 'Ideal gas thermodynamic properties of ethylene and propylene'.

Granryd, E. (2009), *Refrigerating engineering*, Royal Institute of Technology, KTH, Department of Energy Technology, Division of Applied Thermodynamics and Refrigeration.

Halvorsen, I. J., Skogestad, S., Morud, J. C. & Alstad, V. (2003), 'Optimal selection of controlled variables†', *Industrial & Engineering Chemistry Research* **42**(14), 3273–3284.

Jäschke, J. & Skogestad, S. (2011), Optimal operation by controlling the gradient to zero, *in* 'World Congress', Vol. 18, pp. 6073–6078.

Jensen, J. B. & Skogestad, S. (2007), 'Optimal operation of simple refrigeration cycles: part i: degrees of freedom and optimality of sub-cooling', *Computers & chemical engineering* **3**1(5), 712–721.

Jensen, J. B. & Skogestad, S. (2009), 'Steady-state operational degrees of freedom with application to refrigeration cycles', *Industrial & Engineering Chemistry Research* **4**8(14), 6652–6659.

Kariwala, V. & Cao, Y. (2009), 'Bidirectional branch and bound for controlled variable selection. part ii: Exact local method for self-optimizing control', *Computers & Chemical Engineering* **3**3(8), 1402 – 1412.

Kariwala, V., Cao, Y. & Janardhanan, S. (2008), 'Local self-optimizing control with average loss minimization', *Industrial & Engineering Chemistry Research* **4**7(4), 1150–1158.

Prapainop, R. & Suen, K. (2006), 'Simulation of potential refrigerants for retrofit replacement'.

Prasad, B. S. (2002), 'Effect of liquid on a reciprocating compressor', *Journal of energy resources technology* **1**24(3), 187–190.

Price, R. M. & Georgakis, C. (1993), 'Plantwide regulatory control design procedure using a tiered framework', *Industrial & engineering chemistry research* **3**2(11), 2693–2705.

Skogestad, S. (2000), 'Plantwide control: the search for the self-optimizing control structure', *Journal of Process Control* **1**0(5), 487 – 507.

Skogestad, S. (2003), 'Simple analytic rules for model reduction and pid controller tuning', *Journal of Process Control* **1**3(4), 291 – 309.

Skogestad, S. (2004), 'Control structure design for complete chemical plants', *Computers & Chemical Engineering* **2**8(1–2), 219 – 234. Escape 12.

Skogestad, S. (2011), *Chemical and energy process engineering*, CRC press.

Skogestad, S. & Postlethwaite, I. (2007), *Multivariable feedback control: analysis and design*, Vol. 2, Wiley New York.

Verheyleweghen, A. (2014), 'Modelling and optimization of a two-stage compressor train'.

Wright, S. J. & Nocedal, J. (1999), *Numerical optimization*, Vol. 2, Springer New York.

Yelchuru, R. & Skogestad, S. (2010), ‘Miqp formulation for optimal controlled variable selection in self optimizing control’, *PSE Asia* pp. 25–28.

APPENDIX A

PARAMETERS USED FOR THE SIMULATIONS

In the following tables, the coefficients for the model equations will be given. The values in Table A.1 to Table A.6 are based on linearizations of the AllProps-model, as described in Section 3.1.2. The values in Table A.7, Table A.8 and Table A.8 were taken from Asmar (1991)

Table A.1: Coefficients for the Antoine equation in Equation 3.7

Variable	A	B	C
	[-]	[-]	[-]
Value	9.0825	1807.53	26.15

Table A.2: Coefficients for calculating the heat capacity of propylene in Equation 3.36. Calculated heat capacity has units J/(kgK)

Variable	C_1	C_2	C_3	C_4	C_5	C_6
Value	18.4794	0.00727	23.4390	0.00143	−11.095	0.01941

Table A.3: Coefficients for calculating the compressibility of propylene as a function of temperature in Equation 3.35

	ϕ_{00}	ϕ_{10}	ϕ_{01}	ϕ_{20}	ϕ_{11}	ϕ_{02}
	[-]	[bar^{-1}]	[K^{-1}]	[bar^{-2}]	[(Kbar)$^{-1}$]	[K^{-2}]
LP	0.659	0	$3.870 \cdot 10^{-3}$	0	0	$-1.116 \cdot 10^{-5}$
IP	0.415	−0.089	$4.317 \cdot 10^{-3}$	$-3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$-7.917 \cdot 10^{-6}$
HP	−0.345	0	$1.112 \cdot 10^{-2}$	0	0	$-2.437 \cdot 10^{-5}$

Table A.4: Coefficients for calculating the specific vapour enthalpy of propylene as a function of temperature in Equation 3.33

Variable	ζ_1	ζ_2
	[J/kgK]	[J/kg]
Low pressure	1.26898	734.198
Intermediate pressure	1.41275	698.814
High pressure	1.75257	592.804

Table A.5: Coefficients for calculating the specific liquid enthalpy of propylene as a function of temperature in Equation 3.34

Variable	δ_1	δ_2
	[J/kgK]	[J/kg]
Low pressure	2.18222	84.2375
Intermediate pressure	2.40251	29.8187
High pressure	2.81837	−91.3762

Table A.6: Coefficients for calculating the specific volume of propylene as a function of temperature in Equation 3.32

Variable	λ_1	λ_2	λ_3
	[J/kgK^2]	[J/kgK]	[J/kg]
Low pressure	$9.802 \cdot 10^{-9}$	$-1.053 \cdot 10^{-6}$	0.001381
Intermediate pressure	$2.223 \cdot 10^{-8}$	$-7.145 \cdot 10^{-6}$	0.002129
High pressure	$6.736 \cdot 10^{-8}$	$-3.349 \cdot 10^{-5}$	0.00598

Table A.7: Coefficients for the compressor curve for the first compressor in Equation 3.28

Variable	C_{11}	C_{12}	C_{13}
	[m^3/s]	[m^4/s]	[m]
Value	1.9928	16791	8756.4

Table A.8: Coefficients for the compressor curve for the second compressor in Equation 3.29 and Equation 3.30

Variable	C_{21}	C_{22}	C_{23}	C_{24}	C_{25}
	[-]	[m]	[m]	[m^3/s]	[m^3/s]
Value	0.45615	10296	2956.2	0.816051	0.27585

Table A.9: Coefficients for the compressor curves for both compressors in Equation 3.31

Variable	e_1	e_2	e_3	e_4
	[m^{-1}]	[-]	[m^{-1}]	[-]
Compressor 1	$6.47 \cdot 10^{-5}$	0.353	$7.101 \cdot 10^{-4}$	6.5963
Compressor 2	$4.098 \cdot 10^{-5}$	0.364	$11.218 \cdot 10^{-4}$	12.445

Table A.10: Time constants for the low pass filters for the inputs

Variable	τ_{XV2} [s]	τ_{XV3} [s]	τ_N [s]	τ_{XV1} [s]	τ_{FCP3} [s]
Value	2	2	10	5	2

Appendix B

Process flow diagrams

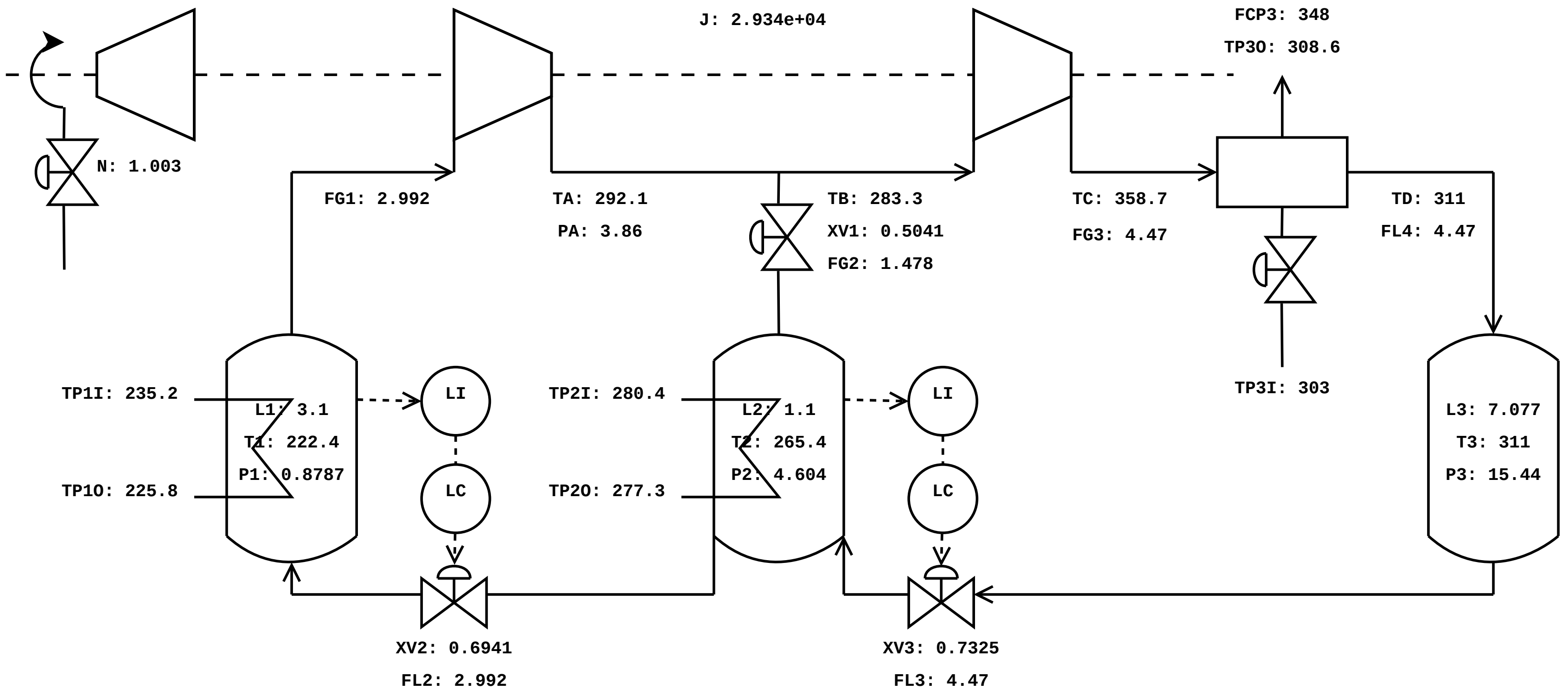


Figure B.1: PFD of the process with nominal values of key variables shown.

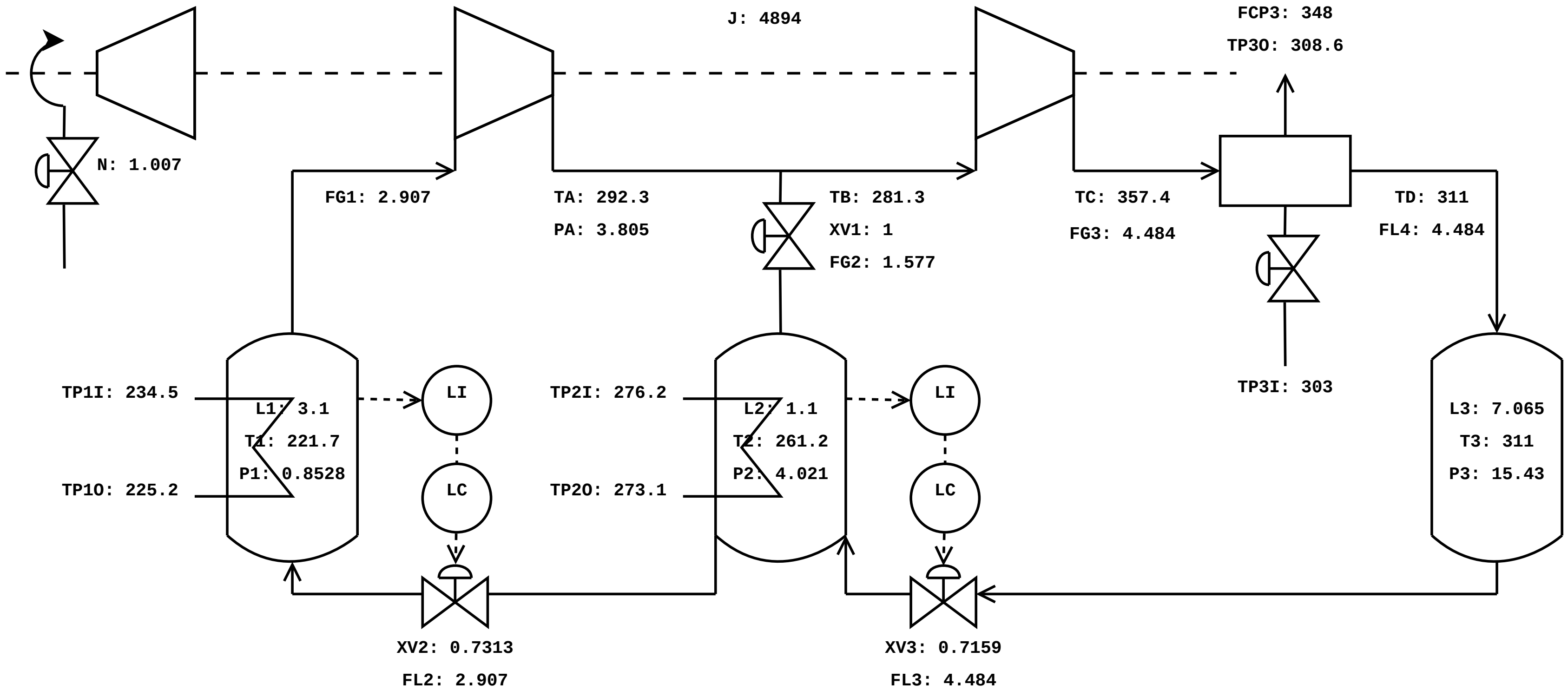


Figure B.2: PFD of the alternative process with nominal values of key variables shown.

APPENDIX C

PRESSURE-ENTHALPY DIAGRAMS

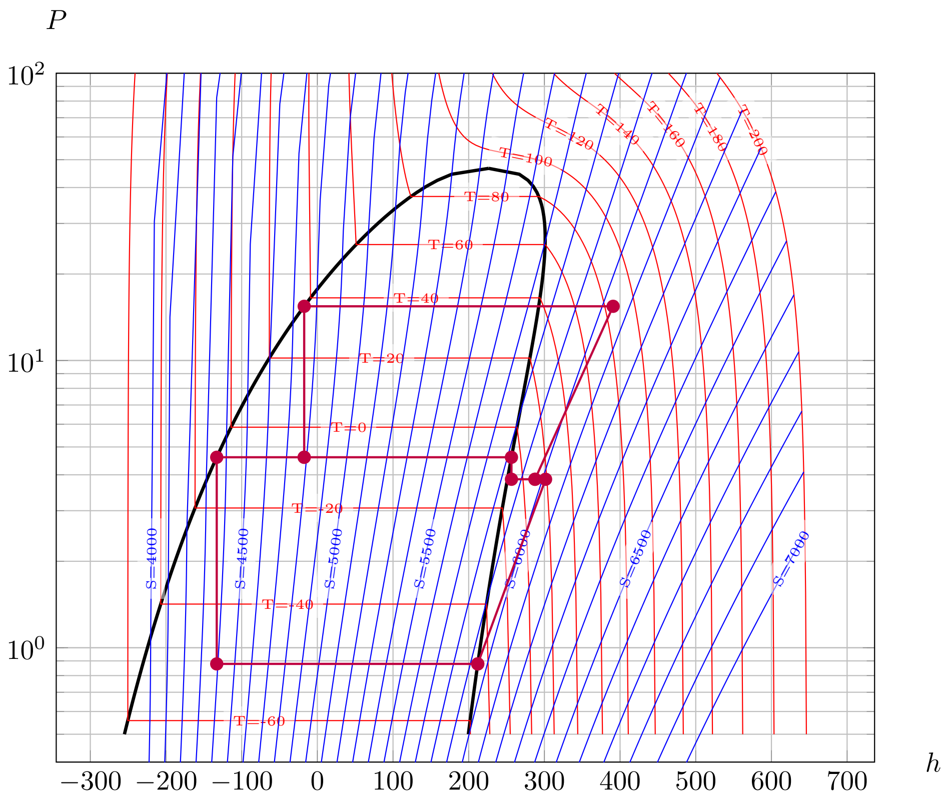


Figure C.1: Pressure-Enthalpy diagram of the nominal solution.

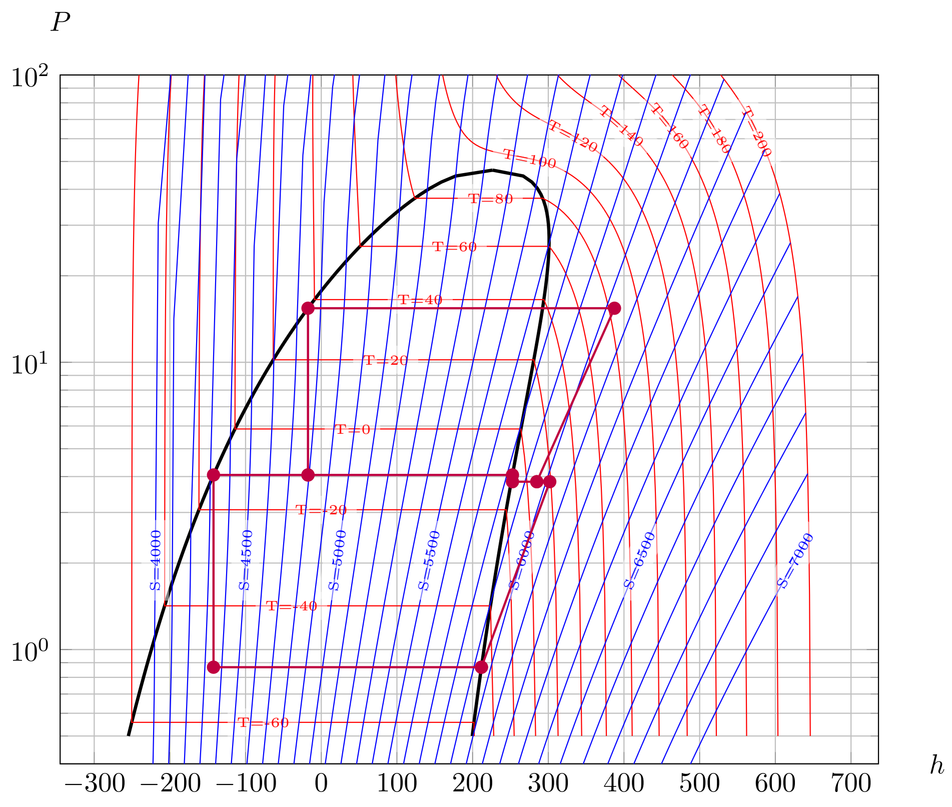


Figure C.2: Pressure-Enthalpy diagram of the nominal solution of the alternative process.

APPENDIX D

MATLAB CODE

D.1 Steady-state

D.1.1 Model

```matlab
function out = SS_model(x,u,p)
%{
    Defining the equations that make up the DAE system.
%}

y = [u;x];

%% Extracting inputs:
\Delta

% Inputs
N     = y( 1);                        % Compressor speed [-]
XV1   = y( 2);                           % Valve opening [-]
FCP3  = y( 3); % prod of F and cp of cooling air [kJ/(s.K)]

%% Extracting states:
% Differential
H1    = y( 4);                   % Enthalpy LP evaporator [kJ]
H2    = y( 5);                   % Enthalpy IP evaporator [kJ]
T3    = y( 6);                   % Temperature HP receiver [K]
W1    = y( 7);                % Ref. holdup LP evaporator [kg]
W2    = y( 8);                % Ref. holdup IP evaporator [kg]
W4    = y( 9);               % Vap. ref. holdup condenser [kg]

% Algebraic
```

```
T1   = y(10);               % Temperature LP evaporator [K]
T2   = y(11);               % Temperature IP evaporator [K]
PA   = y(12);        % Pressure inter-stage mixing node [bar]
TA   = y(13);    % Discharge temperature first compressor [K]
TC   = y(14);  % Discharge temperature second compressor [K]
TD   = y(15);                   % Temperature condenser [K]
HH1  = y(16);          % Compressor head first compressor [m]
HH2  = y(17);         % Compressor head second compressor [m]
WV1  = y(18);             % Vapour holdup LP evaporator [kg]
WV2  = y(19);             % Vapour holdup IP evaporator [kg]
XV2  = y(20);           % Valve opening, L1-constroller [-]
XV3  = y(21);           % Valve opening, L2-constroller [-]
L1   = y(22); % Volumetric liquid holdup LP evaporator [m3]
L2   = y(23); % Volumetric liquid holdup IP evaporator [m3]
L3   = y(24);      % Volumetric liquid holdup receiver [m3]
P1   = y(25);                % Pressure LP evaporator [bar]
P2   = y(26);                % Pressure IP evaporator [bar]
P3   = y(27);                     % Pressure receiver [bar]
TP1O = y(28);   % T process stream outlet LP evaporator [K]
TP2O = y(29);   % T process stream outlet IP evaporator [K]
TP3O = y(30);         % T cooling air outlet condenser [K]
FG1  = y(31);     % Vapour mass flowrate from LP evap [kg/s]
FG2  = y(32);     % Vapour mass flowrate from IP evap [kg/s]
FG3  = y(33);% Vapour mass flowrate from second comp [kg/s]
FL2  = y(34);     % Liquid mass flowrate from IP evap [kg/s]
FL3  = y(35);    % Liquid mass flowrate from receiver [kg/s]
FL4  = y(36);   % Liquid mass flowrate from condenser [kg/s]
TB   = y(37);     % Liquid mass flowrate from IP evap [kg/s]
PTH1 = y(38);   % Power consumption first compressor [kJ/s]
PTH2 = y(39);  % Power consumption second compressor [kJ/s]
Q1   = y(40);             % Heat load LP evaporator [kJ/s]
Q2   = y(41);             % Heat load IP evaporator [kJ/s]

%% Defining parameters

if ~exist('p','var')
    p = SS_init_params();
end

W    = p.W;                    % Refrigerant inventory [kg]
V1   = p.V1;                   % Size of LP evaporator [m3]
V2   = p.V2;                   % Size of IP evaporator [m3]
V3   = p.V3;                        % Size of receiver [m3]
VC   = p.VC;                       % Size of condenser [m3]
C11  = p.C11;     % Coefficient for fitting compressor curve
C12  = p.C12;                                        % --||--
C13  = p.C13;                                        % --||--
C21  = p.C21;                                        % --||--
C22  = p.C22;                                        % --||--
```

```
C23  = p.C23;                                      % --||--
C24  = p.C24;                                      % --||--
C25  = p.C25;                                      % --||--
E11  = p.E11;                                      % --||--
E12  = p.E12;                                      % --||--
E13  = p.E13;                                      % --||--
E14  = p.E14;                                      % --||--
E21  = p.E21;                                      % --||--
E22  = p.E22;                                      % --||--
E23  = p.E23;                                      % --||--
E24  = p.E24;                                      % --||--
LAM1 = p.LAM1; % Coeffs for liq. ref. spec. vol. LP [m3/kg]
LAM2 = p.LAM2; % Coeffs for liq. ref. spec. vol. IP [m3/kg]
LAM3 = p.LAM3; % Coeffs for liq. ref. spec. vol. HP [m3/kg]
PHI1 = p.PHI1;       % Coeffs compressability factor LP [-]
PHI2 = p.PHI2;         % Coeffs compr. factor mix. node [-]
PHI3 = p.PHI3;    % Coeffs compressability factor cond. [-]
ZET1 = p.ZET1;    % Coeffs for vap. ref. specific H [kJ/kg]
ZET2 = p.ZET2;                                     % --||--
ZET3 = p.ZET3;                                     % --||--
DEL1 = p.DEL1;    % Coeffs for liq. ref. specific H [kJ/kg]
DEL2 = p.DEL2;                                     % --||--
DEL3 = p.DEL3;                                     % --||--
DEL4 = p.DEL4;                                     % --||--
U1A1 = p.U1A1; % Heat trnsf coeff & area LP evap [kJ/(s.K)]
U2A2 = p.U2A2; % Heat trnsf coeff & area IP evap [kJ/(s.K)]
U3A3 = p.U3A3;    % Heat trnsf coeff & area cond [kJ/(s.K)]
CV1  = p.CV1;          % Valve constant XV1 [kg/(s.bar0.5)]
CV2  = p.CV2;          % Valve constant XV2 [kg/(s.bar0.5)]
CV3  = p.CV3;          % Valve constant XV3 [kg/(s.bar0.5)]
FCP1 = p.FCP1;       % F and cp of process stream [J/(s.K)]
FCP2 = p.FCP2;
TP1I = p.TP1I;
TP2I = p.TP2I;           % T of process stream at inlet [K]
TP3I = p.TP3I;              % T of cooling air at inlet [K]
A    = p.A;             % Coefficient for Antoine equation
B    = p.B;                                       % --||--
C    = p.C;                                       % --||--
R    = p.R;                       % Gas constant [J/(mol.K)]
MW   = p.MW;         % Propylene molecular weight [kg/kmol]
G    = p.G;                    % Gravity acceleration [m/s2]
C1   = p.C1;           % Coefficient for heat cap. equation
C2   = p.C2;                                       % --||--
C3   = p.C3;                                       % --||--
C4   = p.C4;                                       % --||--
C5   = p.C5;                                       % --||--
C6   = p.C6;                                       % --||--
L1sp = p.L1sp;
L2sp = p.L2sp;
```

```

%% Algebraic equations

                                      % Compressability factors [-]
Z1 = PHI1.a*T1^2 + PHI1.b*T1 + PHI1.c;                         % LP
Z2 = PHI2.a + PHI2.b*PA/10 + PHI2.c*TB +                        ...
     PHI2.d*(PA/10)^2 + PHI2.e*PA/10*TB + PHI2.f*TB^2; % IP
Z3 = PHI3.a*TD^2 + PHI3.b*TD + PHI3.c;                         % HP

HS1  = HH1 / (N^2.19);   % Scaled head first compressor [m]
                                   % Vapour vol. flow rate [m3/s]
FG1S = (N^1.56)*(C11*HS1 - C12)/(HS1 - C13);
HS2  = HH2 / (N^2.11);   % Scaled head first compressor [m]
TX   = C21 / tan((C22-HS2)/C23);                              % [-]
                                   % Vapour vol. flow rate [m3/s]
FG3S = (N^1.79) * (C24 + C25*log(sqrt(TX^2 + 1) - TX));

HFG1 = ZET1.a*T1 + ZET1.b;    % Vap. ref. specific H [kJ/kg]
HFG2 = ZET2.a*T2 + ZET2.b;                          % [kJ/kg]
HFG3 = ZET3.a*TC + ZET3.b;                          % [kJ/kg]
HFL2 = DEL2.a*T2 + DEL2.b;    % Liq. ref. specific H [kJ/kg]
HFL3 = DEL3.a*T3 + DEL3.b;                          % [kJ/kg]
HFL4 = DEL4.a*TD + DEL4.b;                          % [kJ/kg]
HW1V = ZET1.a*T1 + ZET1.b;       % Vap. ref. spec. H [kJ/kg]
HW1L = DEL1.a*T1 + DEL1.b;       % Liq. ref. spec. H [kJ/kg]
HW2V = ZET2.a*T2 + ZET2.b;       % Vap. ref. spec. H [kJ/kg]
HW2L = DEL2.a*T2 + DEL2.b;       % Liq. ref. spec. H [kJ/kg]

A1 = exp(-U1A1/FCP1);                                     % [-]
A2 = exp(-U2A2/FCP2);                                     % [-]
A3 = exp(-U3A3/FCP3);                                     % [-]
QC = FCP3*(TP3I - TP3O);   % Heat transfered in cond [kJ/s]

                                    % Polytropic efficiency [-]
ETA1 = E11*HS1 + E12 - 10^(E13*HS1 - E14);
                                  % Heat capacities [J/(mol.K)]
CPI1 = C1 + C2*T1*(C3 + C4*T1*(C5 + C6*T1));         % Inlet
CPO1 = C1 + C2*TA*(C3 + C4*TA*(C5 + C6*TA));        % Outlet
GAM1 = 0.5*(CPI1/(CPI1-R) + CPO1/(CPO1-R));% Avg. gamma [-]
POL1 = ETA1*GAM1/(GAM1-1);                                % [-]

ETA2 = E21*HS2 + E22 - 10^(E23*HS2 - E24);                % [-]
CPI2 = C1 + C2*TB*(C3 + C4*TB*(C5 + C6*TB));  % [J/(mol.K)]
CPO2 = C1 + C2*TC*(C3 + C4*TC*(C5 + C6*TC));  % [J/(mol.K)]
GAM2 = 0.5*(CPI2/(CPI2-R) + CPO2/(CPO2-R));               % [-]
POL2 = ETA2*GAM2/(GAM2-1);                                % [-]

WL1 = W1 - WV1;          % Liquid holdup LP evaporator [kg]
WL2 = W2 - WV2;          % Liquid holdup IP evaporator [kg]
```

```
W3  = W - W1 - W2 - W4;       % Liquid holdup receiver [kg]

HL1 = WL1 * HW1L;  % Enthalpy liquid holdup in LP evap [kJ]
HV1 = WV1 * HW1V;  % Enthalpy vapour holdup in LP evap [kJ]
HL2 = WL2 * HW2L;  % Enthalpy liquid holdup in IP evap [kJ]
HV2 = WV2 * HW2V;  % Enthalpy vapour holdup in IP evap [kJ]

                  % Liquid refrigerant specific volume [m3/kg]
VF1 = LAM1.a*T1^2 + LAM1.b*T1 + LAM1.c;                  % LP
VF2 = LAM2.a*T2^2 + LAM2.b*T2 + LAM2.c;                  % IP
VF3 = LAM3.a*TD^2 + LAM3.b*TD + LAM3.c;                  % HP

V1G = V1 - WL1*VF1;           % Vapour volume in vessel [m3]
V2G = V2 - WL2*VF2;                                  % --||--
V4G = V3 + VC - W3*VF3;                              % --||--
VG1 = V1G/WV1; % Vapour refrigerant specific volume [m3/kg]
VG2 = V2G/WV2;                                       % --||--
VG4 = V4G/W4;                                        % --||--

%% Defining the derivatives
dH1      =  HFL2 * FL2 - HFG1 * FG1 + Q1;              % [kW]
dH2      =  HFL3 * FL3 - HFL2 * FL2 - HFG2 * FG2 + Q2;% [kW]
dT3      =  FL4 * (TD - T3) / W3;                     % [K/s]
dW1      =  FL2 - FG1;                               % [kg/s]
dW2      =  FL3 - FL2 - FG2;                         % [kg/s]
dW4      =  FG3 - FL4;                               % [kg/s]

dx       = [dH1; dH2; dT3; dW1; dW2; dW4];

%% Defining the residuals
T1res    =  H1 - HL1 - HV1;                            % [kJ]
T2res    =  H2 - HL2 - HV2;                            % [kJ]
PAres    =  FG3 - FG1 - FG2;                         % [kg/s]
TAres    =  TA - T1 * (PA / P1)^(1 / POL1);             % [K]
TCres    =  TC - TB * (P3 / PA)^(1 / POL2);             % [K]
TDres    =  P3 - Z3 * R * TD / (MW * VG4 * 100);      % [bar]
HH1res   =  HH1 - POL1 * (R*1000/(G*MW))*(TA - T1);     % [m]
HH2res   =  HH2 - POL2 * (R*1000/(G*MW))*(TC - TB);     % [m]
WV1res   =  P1 - Z1 * R * T1 / (MW * VG1 * 100);      % [bar]
WV2res   =  P2 - Z2 * R * T2 / (MW * VG2 * 100);      % [bar]
L1res    =  L1 - WL1 * VF1;                            % [m3]
L2res    =  L2 - WL2 * VF2;                            % [m3]
L3res    =  L3 - W3 * VF3;                             % [m3]
P1res    =  P1 - exp(A - B/(T1 - C));                 % [bar]
P2res    =  P2 - exp(A - B/(T2 - C));                 % [bar]
P3res    =  P3 - exp(A - B/(TD - C));                 % [bar]
TP1Ores =  TP1O - ((1 - A1)*T1 + A1*TP1I);              % [K]
TP2Ores =  TP2O - ((1 - A2)*T2 + A2*TP2I);              % [K]
TP3Ores =  TP3O - ((1 - A3)*TD + A3*TP3I);              % [K]
```

```
FG1res   =  FG1 − FG1S * MW * P1 * 100 / (T1*R*Z1); % [kg/s]
FG2res   =  FG2 − XV1 * CV1 * sqrt(P2 − PA);        % [kg/s]
FG3res   =  FG3 − FG3S * MW * PA * 100 / (TB*R*Z2); % [kg/s]
FL2res   =  FL2 − XV2 * CV2 * sqrt(P2 − P1);        % [kg/s]
FL3res   =  FL3 − XV3 * CV3 * sqrt(P3 − P2);        % [kg/s]
FL4res   =  FL4 + QC / (HFG3 − HFL4);               % [kg/s]
TBres    =  TB  − (TA*FG1 + T2*FG2) / (FG1+FG2);       % [K]
PTH1res =  PTH1 − 0.5*FG1*(CPO1+CPI1)*(TA−T1)/MW;  % [kJ/s]
PTH2res =  PTH2 − 0.5*FG3*(CPO2+CPI2)*(TC−TB)/MW;  % [kJ/s]
Q1res    =  Q1 − FCP1 * (TP1I − TP1O);              % [kJ/s]
Q2res    =  Q2 − FCP2 * (TP2I − TP2O);               % [kJ/s]

%% Controllers
L1Cres   = L1sp − L1;
L2Cres   = L2sp − L2;

resids = [T1res; T2res; PAres; TAres; TCres; TDres;      ...
          HH1res; HH2res; WV1res; WV2res; L1Cres;        ...
          L2Cres; L1res; L2res; L3res; P1res; P2res;     ...
          P3res; TP1Ores; TP2Ores; TP3Ores; FG1res;      ...
          FG2res; FG3res; FL2res; FL3res; FL4res;        ...
          TBres; PTH1res; PTH2res; Q1res; Q2res];

out     = [dx; resids];

end
```

D.1.2 Optimizer

```
function [y,fval,exitflag,active_list] = ...
         SS_opt(u0,x0,p,const,socbool)
%{
              Solving the DAE system
%}

if (~exist('u0','var')) || isempty(u0)
    u0 = SS_init_u;
end
if (~exist('x0','var')) || isempty(x0)
    x0 = SS_init_x;
end
if (~exist('p','var')) || isempty(p)
    p = SS_init_params;
end
if (~exist('socbool','var')) || isempty(socbool)
   socbool = false;
end
```

```

g = @(x) SS_model(x,u0,p);
options = optimset('Display','none');
x0 = fsolve(g,x0,options); % Get initial guess (not needed)

y0 = [u0; x0];

%% Constraints
lb=zeros(length(y0),1);
lb(1)  = 0.9;  % N
lb(2)  = 0;    % XV1
lb(3)  = 116;  % FCP3
lb(20) = 0;    % XV2
lb(21) = 0;    % XV3
lb(25) = 0;    % P1
lb(26) = 3;    % P2
lb(27) = 12;   % P3
lb(28) = 200;  % TP1O
lb(32) = 0;    % FG2
lb(34) = 0;    % FL2
lb(35) = 0;    % FL3

ub=ones(length(y0),1)*1e8;
ub(1)  = 1.1;  % N
ub(2)  = 1;    % XV1
ub(3)  = 348;  % FCP3
ub(20) = 1;    % XV2
ub(21) = 1;    % XV3
ub(25) = 2;    % P1
ub(26) = 6;    % P2
ub(27) = 18;   % P3
ub(28) = 300;  % TP1O
ub(32) = 6.31; % FG2
ub(34) = 5.47; % FL2
ub(35) = 7.18; % FL3

%% Optimization

% fmincon options
options = optimset(                                          ...
    'ScaleProblem', 'obj-and-constr',                        ...
    'TolFun', 10e-9,                                         ...
    'TolCon', 10e-9,                                         ...
    'MaxFunEvals', 1e6,                                      ...
    'MaxIter', 1e6,                                          ...
    'Display','none',                                        ...
    'Algorithm','Interior-Point',                            ...
    'Diagnostics','off',                                     ...
    'FinDiffType','central',                                 ...
    'SubproblemAlgorithm','cg');

% Call fmincon to optimize the model
if ~exist('const','var') || isempty(const)
    const = zeros(0,2);
end
nlconpar = @(y) nlcon(y,p,const,socbool);
[y,fval,exitflag] = fmincon(@cost,y0,[],[],[],[],...
                            lb,ub,nlconpar,options);

% Print active constraints and exitflag
active_list = SS_printactive(y,exitflag,lb,ub);

% Create process flowsheet
if length(dbstack) == 1
    SS_create_pfd(y,fval,p);
end

function j = cost(y)
    %% Objective function
    alpha = 125;%125;
    beta  = 1;%1;
    j = (y(38)+y(39)) + alpha*y(28)+beta*y(29);

function [cineq,ceq] = nlcon(y,p,const,socbool)
     global H c subset
    %% Nonlinear inequality constraints C(y)<0
    cineq = [];

    %% Nonlinear equality constraints C(y)=0
    if ~exist('H','var') || isempty(H)
        load('H_matrices.mat')
    end
    if socbool                      % For testing the obtained H
        ceq=[SS_model(y(4:end),y(1:3),p);                  ...
            y(const(:,1))-const(:,2);                      ...
            H*y(subset)-c;                                 ...
        ];

    else
        ceq=[SS_model(y(4:end),y(1:3),p);                 ...
            y(const(:,1))-const(:,2);                     ...
        ];
    end
```

D.1.3 Initialization

```matlab
function p = SS_init_params()
%{
             Initialises all the parameters.
%}

%% Refrigerant inventory [kg]
p.W = 6500;

%% Vessels sizes [m3]
p.V1 = 20;                                        % LP evaporator
p.V2 = 3.393;                                     % IP evaporator
p.V3 = 18.85;                                         % Receiver
p.VC = 4.15;                                         % Condenser

%% Coefficients for curve fittings for performance
% First compressor
p.C11 = 1.9928;
p.C12 = 16791;
p.C13 = 8756.4;

% Second compressor
p.C21 = 0.45615;
p.C22 = 10296;
p.C23 = 2956.2;
p.C24 = 0.816051;
p.C25 = 0.27585;

%% Coeffs. for curve fittings for isentropic efficiency
% First compressor
p.E11 = 6.47E-5;
p.E12 = 0.353;
p.E13 = 7.101E-4;
p.E14 = 6.5963;

% Second compressor
p.E21 = 4.098E-5;
p.E22 = 0.364;
p.E23 = 1.121E-3;
p.E24 = 12.445;

%% Coeffs. for calculating specific volumes of propylene
% Low pressure
p.LAM1.a =   9.802E-9;
p.LAM1.b =  -1.053E-6;
p.LAM1.c =   0.001381;

% Intermediate pressure
p.LAM2.a =   2.223E-8;
p.LAM2.b =  -7.145E-6;
```

```
p.LAM2.c =   0.002129;

% High pressure
p.LAM3.a =   6.736E-8;
p.LAM3.b =  -3.349E-5;
p.LAM3.c =    0.00598;

%% Coefficients for calculating compressibility factors [-]
% Low pressure, first evaporator
p.PHI1.a = -1.115653785185266E-5;
p.PHI1.b =  0.003869991483971;
p.PHI1.c =  0.659220459377467;

% Intermediate pressure, mixing node
p.PHI2.a = 0.414607511696639;
p.PHI2.b =  -0.898851421515544;
p.PHI2.c = 0.004316778331929;
p.PHI2.d = -0.029680863703624;
p.PHI2.e = 0.002581535259452;
p.PHI2.f = -7.916965454791605E-6;

% High pressure, condenser
p.PHI3.a = -2.437079326095500E-5;
p.PHI3.b =  0.011121262139455;
p.PHI3.c = -0.344907274735881;

%% Enthalpies
p.ZET1.a = 1.26898;
p.ZET1.b = 734.198;
p.ZET2.a = 1.41275;
p.ZET2.b = 698.814;
p.ZET3.a = 1.75257;
p.ZET3.b = 592.804;
p.DEL1.a = 2.18222;
p.DEL1.b = 84.2375;
p.DEL2.a = 2.40251;
p.DEL2.b = 29.8187;
p.DEL3.a = 2.81837;
p.DEL3.b = -91.3762;
p.DEL4.a = 2.81837;
p.DEL4.b = -91.3762;

%% Combined  overall  heat  transfer  coefficients and heat
% transfer areas in the two evaporators  and  the condenser
% [J/(s.K)]
p.U1A1 = 146.066;                                 % LP evaporator
p.U2A2 = 5.5479;                                  % IP evaporator
p.U3A3 = 420.643;                                     % Condenser

```

```
%% Valve constants [kg/(s.bar0.5)]
p.CV1 = 3.39814;                                 % Vapour valve
p.CV2 = 2.233338;                          % First liquid valve
p.CV3 = 1.85435;                          % Second liquid valve

%% Combined process stream flowrate and specific heat
% capacity [J/(s.K)]
p.FCP1 = 111.394;                              % LP evaporator
p.FCP2 = 24.32;                                % IP evaporator

%% Process stream inlet temperatures [K]
p.TP1I = 235.2;          % Process stream in LP evaporator
p.TP2I = 280.4;          % Process stream in IP evaporator
p.TP3I = 303.0;             % Cooling air in the condenser

%% Antoine Equation coefficients for propylene
p.A = 9.0825;
p.B = 1807.53;
p.C = 26.15;

%% General constants
p.R = 8.314;                      % Gas constant [J/(mol.K)]
p.MW = 42.081;       % Propylene molecular weight [kg/kmol]
p.G = 9.81;                  % Gravity acceleration [m/s2]

%% Specific heat capacity coefficients for propylene
p.C1 = 18.479424911751948;
p.C2 = 0.007270343051175;
p.C3 = 23.438998780670026;
p.C4 = 0.001429363603578;
p.C5 = -11.095379346410985;
p.C6 = 0.019414066400671;

%% Controller settings
p.L1sp   = 3.1;                                  % L1 setpoint
p.L2sp   = 1.1;                                  % L2 setpoint

p.XV2opt = 0.7;          % Initial guess optimal value XV2
p.KcL1C  = 2.86;                             % Controller gain
p.KiL1C  = 1/37.61;                            % Integral gain

p.XV3opt = 0.7;          % Initial guess optimal value XV3
p.KcL2C  = 10;                               % Controller gain
p.KiL2C  = 1/37.61;                            % Integral gain

% SOC controllers
p.XV1opt = 0.5;          % Initial guess optimal value XV1
k     = 2.371/(0.01*0.504);                     % Process gain
tau1  = 20;                            % Process time constant
```

```
theta = 0;                                    % Process time delay
tauc  = 2;                     % Desired closed loop time constant
p.KcXV1C = (tau1/k)*(1/(tauc+theta));
p.KiXV1C = p.KcXV1C/min([tau1,4*(tauc+theta)]);

p.Nopt = 1;                        % Initial guess optimal value N
k      = 234.1/(0.01*1.003);                        % Process gain
tau1   = 10;                                % Process time constant
theta  = 0;                                    % Process time delay
tauc   = 2;                    % Desired closed loop time constant
p.KcNC = (tau1/k)*(1/(tauc+theta));
p.KiNC = p.KcNC/min([tau1,4*(tauc+theta)]);

% Low pass filter time constants [s]
p.tau_XV2f  = 0.002;
p.tau_XV3f  = 0.002;
p.tau_Nf    = 0.010;
p.tau_XV1f  = 0.002;
p.tau_FCP3f = 0.005;

end
```

```
function  u = SS_init_u()
% Initialize the u vector containing the inputs

    % Inputs
    Nic    = 1;
    XV1ic  = 0.355612;
    FCP3ic = 348;

    u = [Nic; XV1ic; FCP3ic];
end
```

```
function x = SS_init_x()
    % Differential states
    H1     = 1.117982E6;                                      % [J]
    H2     = 4.347168E5;                                      % [J]
    T3     = 310.842;                                         % [K]
    W1     = 1934.69;                                        % [kg]
    W2     = 630.168;                                        % [kg]
    W4     = 526.521;                                        % [kg]
    x_diff = [H1; H2; T3; W1; W2; W4]; % Diff.

    % Algebraic states
T1      = 222.486;                                            % [K]
    T2     = 268.474;                                         % [K]
    PA      = 3.84945;                                    % [bar]
    TA      = 291.760;                                    % [bar]
    TC      = 359.871;                                    % [bar]
    TD      = 310.842;                                    % [bar]
    HH1     = 7574.36;                                      % [m]
    HH2     = 8951.01;                                      % [m]
    WV1     = 35.0968;                                      % [m]
    WV2     = 23.4576;                                      % [m]
    XV2     = 0.66247;
    XV3     = 0.73245;
    x_alg   = [T1; T2; PA; TA; TC; TD; HH1; HH2; WV1;     ...
               WV2; XV2; XV3]; % Alg.

    % Additional (non—state) variables
    L1      = 3.1;                                         % [m3]
    L2      = 1.1;                                         % [m3]
    L3      = 7.085;                                       % [m3]
    P1      = 0.883654987841408;                          % [bar]
    P2      = 5.070104271470663;                          % [bar]
    P3      = 15.385558898601653;                         % [bar]
    TP1O    = 2.259140730949896e+02;                        % [K]
    TPIM    = 2.779674765318184e+02;                        % [K]
    TP3O    = 3.085006399364720e+02;                        % [K]
    FG1     = 3.027199259360120;                         % [kg/s]
    FG2     = 1.335098297645051;                         % [kg/s]
    FG3     = 4.362297557005171;                         % [kg/s]
    FL2     = 3.027199259360120;                         % [kg/s]
    FL3     = 4.362297557005171;                         % [kg/s]
    FL4     = 4.362297557005171;                         % [kg/s]
    TB      = 2.846334106435063e+02;                        % [K]
    PTH1    = 2.897784557586991e+02;                        % [J]
    PTH2    = 5.325813792750275e+02;                        % [J]
    Q1      = 1.034396541656729e+03;                        % [J]
    Q2      = 59.158970746175093;

    x_add   = [L1; L2; L3; P1; P2; P3; TP1O; TPIM; TP3O; ...
               FG1; FG2; FG3; FL2; FL3; FL4; TB; PTH1;   ...
               PTH2; Q1; Q2]; % Additional

    % Combining diff, alg and additional
    x = [x_diff; x_alg; x_add];
end
```

D.1.4 Calculating H

```
function [Juu,Gy,F,y,fval] = getSensMats(u0,x0,p,const)

```

```
if (~exist('u0','var')) || isempty(u0)
    u0 = SS_init_u;
end
if (~exist('x0','var')) || isempty(x0)
    x0 = SS_init_x;
end
if (~exist('p','var')) || isempty(p)
    p = SS_init_params;
end
if ~exist('const','var') || isempty(const)
    const = zeros(0,2);
end

%% Calculates Juu

% Get nominal solution
[y,fval] = SS_opt(u0,x0,p,const);
nom = [[1:3]',y(1:3)];

f = @(c) SS_opt(u0,x0,p,c);

% Create steps for the inputs
h = [0,0,0;nom(1,2)*0.01,0,0]';
k = [0,0,0;0,nom(2,2)*0.01,0]';

[~,ff] = f(nom+h+k);
[~,bb] = f(nom-h-k);

[yf1,f1] = f(nom+h);
[yb1,b1] = f(nom-h);
[yf2,f2] = f(nom+k);
[yb2,b2] = f(nom-k);
[~,ce] = f(nom);

% Calculate the derivative using finite difference methods
Juu_1 = (f1-2*ce+b1)/sum(h(:))^2;
Juu_2 = (f2-2*ce+b2)/sum(k(:))^2;
Juu_d = (ff-f1-f2+2*ce-b1-b2+bb)/(sum(h(:))*sum(k(:)));

Juu = [Juu_1 Juu_d; Juu_d Juu_2];

%% Calculates Gy = dy/du

% Calculate the derivative using finite difference methods
GyN   = 0.5*(yf1-yb1)/sum(h(:));
GyXV1 = 0.5*(yf2-yb2)/sum(k(:));
Gy = [GyN, GyXV1];

%% Calculates sensitivity matrix F
```

```

ds = {'TP1I','TP2I','TP3I','FCP1','FCP2'};
F  = [];

delta  = 0.001;
for d  = ds
    f  = getfield(p,d{1});
    pf = p; pf = setfield(pf,d{1},(f+f*delta));
    yf = SS_opt(u0,x0,pf,const);
    pb = p; pb = setfield(pb,d{1},(f-f*delta));
    yb = SS_opt(u0,x0,pb,const);
    F  = [F ((yf-yb)/(2*f*delta))];
end

end
```

```
function [H,c,subset,Loss,names] = ...
    optimal_measurements(n,s,l,Juu,Gy,F,y_opt)

% Function to calculate H using exact local method

if ~exist('n','var')
    n = 5;
end
if ~exist('s','var')
    s = true;
end
if ~exist('l','var')
    l = true;
end

if ~l && nargin < 7
  error('Juu, Gy, F and y_opt must be provided when l = 0')
elseif l
  load('sensitivity_matrices.mat')
end

% Choose temperature, pressure and flow measurements, and u
subset = [1 2 6 10:15 25:37];
F   = F(subset,:);
Gy = Gy(subset,:);
Wd = diag([2,2,3,4,1]);
Wn2 = diag([0.01,0.01,1,1,1,0.1,1,1,1,0.1,0.1,0.1,...
            1,1,1,0.1,0.1,0.1,0.1,0.1,0.1,1]);
Wn = diag(y_opt(subset)*0.01);

% Call  branch and bound  algorithm  to  find  subset  with
% minimal average loss.
```

```
addpath([pwd '/b3av'])
[~,sset] = pb3av(Gy,F,Wd,Wn,Juu,n,inf,1);

sset

names = {'N','XV1','FCP3','H1','H2','T3','W1','W2','W4',...
         'T1','T2','PA','TA','TC','TD','HH1','HH2',      ...
         'WV1','WV2','XV2','XV3','L1','L2','L3','P1',    ...
         'P2','P3','TP1O','TP2O','TP3O','FG1','FG2',     ...
         'FG3','FL2','FL3','FL4','TB','PTH1','PTH2',     ...
         'Q1','Q2'                                        ...
        };
subset = subset(sset);
names = names(subset);

y_opt = y_opt(subset);
Wn_   = Wn(sset,sset);
Gy_   = Gy(sset,:);
F_    = F(sset,:);
Y_    = [F_*Wd Wn_];

H = (Gy_'/(Y_*Y_'));
c = H*y_opt;
%c2 = H(1,:)*diag(y_opt)

diffr = ((diag(Wn)-diag(Wn2))./(diag(Wn2)))'
diffr(sset)

% Calculating the average loss (formulae in pb3av is wrong)
X = Juu^(1/2)/(H*Gy_)*(H*Y_);
Loss = 1/2*norm(X,'fro')^2;

% Average loss for constant inputs
Hcon = [1 0; 0 1];
sset = [1,2];
Wncon = Wn(sset,sset);
Gycon = Gy(sset,:);
Fcon  = F(sset,:);
Ycon  = [Fcon*Wd Wncon];
X2 = Juu^(1/2)/(Hcon*Gycon)*(Hcon*Ycon);
Loss2 = 1/2*norm(X2,'fro')^2;

% Save H,c and subset for easy access
if s
    save('H_matrices.mat','H','c','subset')
end

end
```

D.1.5 Additional functions

```
function active_list = SS_printactive(y,exitflag,lb,ub)
% Function that prints active constraints and non-zero
% exit flags.

names = {'N','XV1','FCP3','H1','H2','T3','W1','W2','W4',...
         'T1','T2','PA','TA','TC','TD','HH1','HH2',      ...
         'WV1','WV2','XV2','XV3','L1','L2','L3','P1',    ...
         'P2','P3','TP1O','TP2O','TP3O','FG1','FG2',     ...
         'FG3','FL2','FL3','FL4','TB','PTH1','PTH2',     ...
         'Q1','Q2'                                       ...
        };

if exitflag <= 0
    fprintf(['Exitflag negative or equal to zero!\n'     ...
    '\nExitflag: %i',exitflag)
end

upper_limit = abs(ub./y)-1;
lower_limit = 1-abs(lb./y);
err_tol = 1e-4;
active_list = [];
for i = 1:length(y)
  if upper_limit(i) < err_tol
    if length(dbstack)<=2;
       fprintf('Active constraint: %s at upper limit \n',...
       names{i})
    end
    if isempty(active_list)
      active_list = [names{i},'u '];
    else
      active_list = [active_list,[names{i},'u ']];
    end
  elseif lower_limit(i) < err_tol
    if length(dbstack)<=2;
      fprintf('Active constraint: %s at lower limit \n', ...
      names{i})
    end
    if isempty(active_list)
      active_list = [names{i},'l '];
    else
      active_list = [active_list,[names{i},'l ']];
    end
  end
end

end
```

D.2 Dynamic

D.2.1 Open loop

```
function out = OL_model(t,x,u,params)

% Description

%% Extracting (filtered) inputs
Nf    = x( 1);
XV1f  = x( 2);
FCP3f = x( 3);

%% Extracting states:

% Differential
H1    = x( 4);
H2    = x( 5);
T3    = x( 6);
W1    = x( 7);
W2    = x( 8);
W4    = x( 9);

% Algebraic
T1    = x(10);
T2    = x(11);
PA    = x(12);
TA    = x(13);
TC    = x(14);
TD    = x(15);
HH1   = x(16);
HH2   = x(17);
WV1   = x(18);
WV2   = x(19);
XV2   = x(20);
XV3   = x(21);
L1    = x(22);
L2    = x(23);
L3    = x(24);
P1    = x(25);
P2    = x(26);
P3    = x(27);
TP1O  = x(28);
TP2O  = x(29);
TP3O  = x(30);
FG1   = x(31);
FG2   = x(32);
FG3   = x(33);
```

```
FL2  = x(34);
FL3  = x(35);
FL4  = x(36);
TB   = x(37);
PTH1 = x(38);
PTH2 = x(39);
Q1   = x(40);
Q2   = x(41);

% First order filters for MVs
XV2f   = x(42);
XV3f   = x(43);

% Integrated errors for level controllers
IE_L1C = x(44);
IE_L2C = x(45);

%% Extracting inputs and disturbances

L1sp = u( 1);
L2sp = u( 2);
N    = u( 3);
XV1  = u( 4);
FCP3 = u( 5);

%% Defining parameters
W    = params.W;
V1   = params.V1;
V2   = params.V2;
V3   = params.V3;
VC   = params.VC;
C11  = params.C11;
C12  = params.C12;
C13  = params.C13;
C21  = params.C21;
C22  = params.C22;
C23  = params.C23;
C24  = params.C24;
C25  = params.C25;
E11  = params.E11;
E12  = params.E12;
E13  = params.E13;
E14  = params.E14;
E21  = params.E21;
E22  = params.E22;
E23  = params.E23;
E24  = params.E24;
LAM1 = params.LAM1;
LAM2 = params.LAM2;
```

```
LAM3 = params.LAM3;
PHI1 = params.PHI1;
PHI2 = params.PHI2;
PHI3 = params.PHI3;
ZET1 = params.ZET1;
ZET2 = params.ZET2;
ZET3 = params.ZET3;
DEL1 = params.DEL1;
DEL2 = params.DEL2;
DEL3 = params.DEL3;
DEL4 = params.DEL4;
U1A1 = params.U1A1;
U2A2 = params.U2A2;
U3A3 = params.U3A3;
CV1  = params.CV1;
CV2  = params.CV2;
CV3  = params.CV3;
FCP1 = params.FCP1;
FCP2 = params.FCP2;
TP1I = params.TP1I;
TP2I = params.TP2I;
TP3I = params.TP3I;
A    = params.A;
B    = params.B;
C    = params.C;
R    = params.R;
MW   = params.MW;
G    = params.G;
C1   = params.C1;
C2   = params.C2;
C3   = params.C3;
C4   = params.C4;
C5   = params.C5;
C6   = params.C6;

%% Algebraic equations

Z1 = PHI1.a*T1^2 + PHI1.b*T1 + PHI1.c;
Z2 = PHI2.a+PHI2.b*PA/10+PHI2.c*TB+PHI2.d*(PA/10)^2 +   ...
     PHI2.e*PA/10*TB + PHI2.f*TB^2;
Z3 = PHI3.a*TD^2 + PHI3.b*TD + PHI3.c;

HS1  = HH1 / (Nf^2.19);
FG1S = (Nf^1.56)*(C11*HS1 - C12)/(HS1 - C13);
HS2  = HH2 / (Nf^2.11);
TX   = C21 / tan((C22-HS2)/C23);
FG3S = (Nf^1.79) * (C24 + C25*log(sqrt(TX^2 + 1) - TX));

HFG1 = ZET1.a*T1 + ZET1.b;
```

```
HFG2 = ZET2.a*T2 + ZET2.b;
HFG3 = ZET3.a*TC + ZET3.b;
HFL2 = DEL2.a*T2 + DEL2.b;
HFL3 = DEL3.a*T3 + DEL3.b;
HFL4 = DEL4.a*TD + DEL4.b;
HW1V = ZET1.a*T1 + ZET1.b;
HW1L = DEL1.a*T1 + DEL1.b;
HW2V = ZET2.a*T2 + ZET2.b;
HW2L = DEL2.a*T2 + DEL2.b;

A1 = exp(-U1A1/FCP1);
A2 = exp(-U2A2/FCP2);
A3 = exp(-U3A3/FCP3f);
QC = FCP3f*(TP3I - TP3O);

ETA1 = E11*HS1 + E12 - 10^(E13*HS1 - E14);
CPI1 = C1 + C2*T1*(C3 + C4*T1*(C5 + C6*T1));
CPO1 = C1 + C2*TA*(C3 + C4*TA*(C5 + C6*TA));
GAM1 = 0.5*(CPI1/(CPI1-R) + CPO1/(CPO1-R));
POL1 = ETA1*GAM1/(GAM1-1);

ETA2 = E21*HS2 + E22 - 10^(E23*HS2 - E24);
CPI2 = C1 + C2*TB*(C3 + C4*TB*(C5 + C6*TB));
CPO2 = C1 + C2*TC*(C3 + C4*TC*(C5 + C6*TC));
GAM2 = 0.5*(CPI2/(CPI2-R) + CPO2/(CPO2-R));
POL2 = ETA2*GAM2/(GAM2-1);

WL1 = W1 - WV1;
WL2 = W2 - WV2;
W3  = W - W1 - W2 - W4;

HL1 = WL1 * HW1L;
HV1 = WV1 * HW1V;
HL2 = WL2 * HW2L;
HV2 = WV2 * HW2V;

VF1 = LAM1.a*T1^2 + LAM1.b*T1 + LAM1.c;
VF2 = LAM2.a*T2^2 + LAM2.b*T2 + LAM2.c;
VF3 = LAM3.a*TD^2 + LAM3.b*TD + LAM3.c;

V1G = V1 - WL1*VF1;
V2G = V2 - WL2*VF2;
V4G = V3 + VC - W3*VF3;
VG1 = V1G/WV1;
VG2 = V2G/WV2;
VG4 = V4G/W4;

E_L1C   = L1sp-L1;
E_L2C   = L2sp-L2;
```

```matlab

%% Defining the derivatives
dH1      =  HFL2 * FL2 - HFG1 * FG1 + Q1;
dH2      =  HFL3 * FL3 - HFL2 * FL2 - HFG2 * FG2 + Q2;
dT3      =  FL4 * (TD - T3) / W3;
dW1      =  FL2 - FG1;
dW2      =  FL3 - FL2 - FG2;
dW4      =  FG3 - FL4;

dx       = [dH1; dH2; dT3; dW1; dW2; dW4];

%% Defining the residuals
T1res    =  H1 - HL1 - HV1;
T2res    =  H2 - HL2 - HV2;
PAres    =  FG3 - FG1 - FG2;
TAres    =  TA - T1 * (PA / P1)^(1 / POL1);
TCres    =  TC - TB * (P3 / PA)^(1 / POL2);
TDres    =  P3 - Z3 * R * TD / (MW * VG4 * 100);
HH1res   =  HH1 - POL1 * (R * 1000/(G * MW)) * (TA - T1);
HH2res   =  HH2 - POL2 * (R * 1000/(G * MW)) * (TC - TB);
WV1res   =  P1 - Z1 * R * T1 / (MW * VG1 * 100);
WV2res   =  P2 - Z2 * R * T2 / (MW * VG2 * 100);
L1res    =  L1 - WL1 * VF1;
L2res    =  L2 - WL2 * VF2;
L3res    =  L3 - W3 * VF3;
P1res    =  P1 - exp(A - B/(T1 - C));
P2res    =  P2 - exp(A - B/(T2 - C));
P3res    =  P3 - exp(A - B/(TD - C));
TP1Ores  =  TP1O - ((1 - A1)*T1 + A1*TP1I);
TP2Ores  =  TP2O - ((1 - A2)*T2 + A2*TP2I);
TP3Ores  =  TP3O - ((1 - A3)*TD + A3*TP3I);
FG1res   =  FG1 - FG1S * MW * P1 * 100 / (T1 * R * Z1);
FG2res   =  FG2 - XV1f * CV1 * sqrt(P2 - PA);
FG3res   =  FG3 - FG3S * MW * PA * 100 / (TB * R * Z2);
FL2res   =  FL2 - XV2f * CV2 * sqrt(P2 - P1);
FL3res   =  FL3 - XV3f * CV3 * sqrt(P3 - P2);
FL4res   =  FL4 + QC / (HFG3 - HFL4);
TBres    =  TB  - (TA*FG1 + T2*FG2) / (FG1+FG2);
PTH1res  =  PTH1 - 0.5 * FG1*(CPO1 + CPI1) * (TA - T1) / MW;
PTH2res  =  PTH2 - 0.5 * FG3*(CPO2 + CPI2) * (TC - TB) / MW;
Q1res    =  Q1 - FCP1 * (TP1I - TP1O);
Q2res    =  Q2 - FCP2 * (TP2I - TP2O);

%% Controllers
% L1 controller
XV2opt  = params.XV2opt;
KcL1C   = params.KcL1C;
KiL1C   = params.KiL1C;
L1Cres   = ((XV2opt - XV2) + E_L1C*KcL1C + IE_L1C*KiL1C);
```

```

% L2 controller
XV3opt = params.XV3opt;
KcL2C  = params.KcL2C;
KiL2C  = params.KiL2C;
L2Cres   = ((XV3opt − XV3) + E_L2C*KcL2C + IE_L2C*KiL2C);

resids = [T1res; T2res; PAres; TAres; TCres; TDres;     ...
          HH1res; HH2res; WV1res; WV2res; L1Cres;        ...
          L2Cres; L1res; L2res; L3res; P1res; P2res;     ...
          P3res; TP10res; TP20res; TP30res; FG1res;      ...
          FG2res; FG3res; FL2res; FL3res; FL4res; TBres;...
          PTH1res; PTH2res; Q1res; Q2res];

%% First order low−pass filters (differential)
tau_XV2f  = params.tau_XV2f;
tau_XV3f  = params.tau_XV3f;
tau_Nf    = params.tau_Nf;
tau_XV1f  = params.tau_XV1f;
tau_FCP3f = params.tau_FCP3f;

dXV2f  = (XV2−XV2f)/tau_XV2f;
dXV3f  = (XV3−XV3f)/tau_XV3f;
dNf    = (N−Nf)/tau_Nf;
dXV1f  = (XV1−XV1f)/tau_XV1f;
dFCP3f = (FCP3−FCP3f)/tau_FCP3f;

%% Integrated controller errors

dE_L1C   =  E_L1C;
dE_L2C   =  E_L2C;

out = [dNf; dXV1f; dFCP3f; dx; resids; dXV2f; dXV3f;    ...
       dE_L1C; dE_L2C];

end
```

```
function [t,x,u] = OL_integrator(c,x0,u0)

% Add path to get access to SS folder
addpath(genpath('\MATLAB\Original'))

% Initial estimates for the states
if ~exist('c','var');
    c = zeros(0,2);
end

if ~exist('x0','var');
```

```
    x0 = SS_init_x;
end

if ~exist('u0','var');
    u0 = SS_init_u;
end

% Calculating the steady state
y_SS = SS_opt(u0,x0,c);

x_SS = [y_SS;y_SS(20:21);0;0];
u_SS = [3.1; 1.1; y_SS(1:3)];

%% Calculating responses to disturbances / setpoint changes

u = u_SS;
% Doing a step change in u setpoint
u(4) = u(4)*1.01;

% Additional parameters
p = SS_init_params;
%p.TP1I   = p.TP1I + 1;

p.XV2opt = y_SS(20);
p.XV3opt = y_SS(21);

% Creating a function handle to pass extra parameters to
% it (u and params)
g = @(t,x) OL_model(t,x,u,p);

% Defining the constant, singular mass matrix M
M = diag([ones(1,9),zeros(1,32),ones(1,4)]);

options = odeset('Mass',M,'MStateDependence','none',     ...
    'MassSingular','yes','RelTol',1e-6,'Vectorized','off');

tspan = 0:0.05:1000;
[t,x] = ode15s(g,tspan,x_SS,options);

close all
plot(t,x(:,29))

tbz = (-100:0.5:-0.5)';
t = [tbz;t];
x = [(x_SS*ones(size(tbz))')';x];
u = [(u_SS*ones(size(tbz))')';                           ...
     (u*ones(length(t)-length(tbz),1)')'];

% Plot results
```

```
if length(dbstack)==1
    OL_plotter(t,x,x_SS,u_SS);
end


end
```

D.2.2 Closed loop

```
function out = CL_model(t,x,u,params)

% Description

%% Extracting (filtered) inputs
Nf    = x( 1);
XV1f  = x( 2);
FCP3f = x( 3);

%% Extracting states:

% Differential
H1    = x( 4);
H2    = x( 5);
T3    = x( 6);
W1    = x( 7);
W2    = x( 8);
W4    = x( 9);

% Algebraic
T1    = x(10);
T2    = x(11);
PA    = x(12);
TA    = x(13);
TC    = x(14);
TD    = x(15);
HH1   = x(16);
HH2   = x(17);
WV1   = x(18);
WV2   = x(19);
XV2   = x(20);
XV3   = x(21);
L1    = x(22);
L2    = x(23);
L3    = x(24);
P1    = x(25);
P2    = x(26);
P3    = x(27);
```

```
TP1O = x(28);
TP2O = x(29);
TP3O = x(30);
FG1  = x(31);
FG2  = x(32);
FG3  = x(33);
FL2  = x(34);
FL3  = x(35);
FL4  = x(36);
TB   = x(37);
PTH1 = x(38);
PTH2 = x(39);
Q1   = x(40);
Q2   = x(41);

% First order filters for MVs
XV2f   = x(42);
XV3f   = x(43);

% Integrated errors for level controllers
IE_L1C = x(44);
IE_L2C = x(45);

% More states (for control)
N    = x(46);
c1   = x(47);
IE_c1= x(48);
XV1  = x(49);
c2   = x(50);
IE_c2= x(51);

%% Extracting inputs and disturbances
L1sp = u( 1);
L2sp = u( 2);
N    = u( 3);
XV1  = u( 4);
FCP3 = u( 5);

%% Defining parameters
W    = params.W;
V1   = params.V1;
V2   = params.V2;
V3   = params.V3;
VC   = params.VC;
C11  = params.C11;
C12  = params.C12;
C13  = params.C13;
C21  = params.C21;
C22  = params.C22;
```

```
C23  = params.C23;
C24  = params.C24;
C25  = params.C25;
E11  = params.E11;
E12  = params.E12;
E13  = params.E13;
E14  = params.E14;
E21  = params.E21;
E22  = params.E22;
E23  = params.E23;
E24  = params.E24;
LAM1 = params.LAM1;
LAM2 = params.LAM2;
LAM3 = params.LAM3;
PHI1 = params.PHI1;
PHI2 = params.PHI2;
PHI3 = params.PHI3;
ZET1 = params.ZET1;
ZET2 = params.ZET2;
ZET3 = params.ZET3;
DEL1 = params.DEL1;
DEL2 = params.DEL2;
DEL3 = params.DEL3;
DEL4 = params.DEL4;
U1A1 = params.U1A1;
U2A2 = params.U2A2;
U3A3 = params.U3A3;
CV1  = params.CV1;
CV2  = params.CV2;
CV3  = params.CV3;
FCP1 = params.FCP1;
FCP2 = params.FCP2;
TP1I = params.TP1I;
TP2I = params.TP2I;
TP3I = params.TP3I;
A    = params.A;
B    = params.B;
C    = params.C;
R    = params.R;
MW   = params.MW;
G    = params.G;
C1   = params.C1;
C2   = params.C2;
C3   = params.C3;
C4   = params.C4;
C5   = params.C5;
C6   = params.C6;

%% Algebraic equations
```

```

Z1 = PHI1.a*T1^2 + PHI1.b*T1 + PHI1.c;
Z2 = PHI2.a+PHI2.b*PA/10+PHI2.c*TB+PHI2.d*(PA/10)^2 +    ...
     PHI2.e*PA/10*TB + PHI2.f*TB^2;
Z3 = PHI3.a*TD^2 + PHI3.b*TD + PHI3.c;

HS1  = HH1 / (Nf^2.19);
FG1S = (Nf^1.56)*(C11*HS1 - C12)/(HS1 - C13);
HS2  = HH2 / (Nf^2.11);
TX   = C21 / tan((C22-HS2)/C23);
FG3S = (Nf^1.79) * (C24 + C25*log(sqrt(TX^2 + 1) - TX));

HFG1 = ZET1.a*T1 + ZET1.b;
HFG2 = ZET2.a*T2 + ZET2.b;
HFG3 = ZET3.a*TC + ZET3.b;
HFL2 = DEL2.a*T2 + DEL2.b;
HFL3 = DEL3.a*T3 + DEL3.b;
HFL4 = DEL4.a*TD + DEL4.b;
HW1V = ZET1.a*T1 + ZET1.b;
HW1L = DEL1.a*T1 + DEL1.b;
HW2V = ZET2.a*T2 + ZET2.b;
HW2L = DEL2.a*T2 + DEL2.b;

A1 = exp(-U1A1/FCP1);
A2 = exp(-U2A2/FCP2);
A3 = exp(-U3A3/FCP3f);
QC = FCP3f*(TP3I - TP3O);

ETA1 = E11*HS1 + E12 - 10^(E13*HS1 - E14);
CPI1 = C1 + C2*T1*(C3 + C4*T1*(C5 + C6*T1));
CPO1 = C1 + C2*TA*(C3 + C4*TA*(C5 + C6*TA));
GAM1 = 0.5*(CPI1/(CPI1-R) + CPO1/(CPO1-R));
POL1 = ETA1*GAM1/(GAM1-1);

ETA2 = E21*HS2 + E22 - 10^(E23*HS2 - E24);
CPI2 = C1 + C2*TB*(C3 + C4*TB*(C5 + C6*TB));
CPO2 = C1 + C2*TC*(C3 + C4*TC*(C5 + C6*TC));
GAM2 = 0.5*(CPI2/(CPI2-R) + CPO2/(CPO2-R));
POL2 = ETA2*GAM2/(GAM2-1);

WL1 = W1 - WV1;
WL2 = W2 - WV2;
W3  = W - W1 - W2 - W4;

HL1 = WL1 * HW1L;
HV1 = WV1 * HW1V;
HL2 = WL2 * HW2L;
HV2 = WV2 * HW2V;

```

```
VF1 = LAM1.a*T1^2 + LAM1.b*T1 + LAM1.c;
VF2 = LAM2.a*T2^2 + LAM2.b*T2 + LAM2.c;
VF3 = LAM3.a*TD^2 + LAM3.b*TD + LAM3.c;

V1G = V1 - WL1*VF1;
V2G = V2 - WL2*VF2;
V4G = V3 + VC - W3*VF3;
VG1 = V1G/WV1;
VG2 = V2G/WV2;
VG4 = V4G/W4;

E_L1C   = L1sp-L1;
E_L2C   = L2sp-L2;

E_L1C   = L1sp-L1;
E_L2C   = L2sp-L2;

E_c1  = c1sp-c1;
E_c2  = c2sp-c2;

%% Defining the derivatives
dH1      =  HFL2 * FL2 - HFG1 * FG1 + Q1;
dH2      =  HFL3 * FL3 - HFL2 * FL2 - HFG2 * FG2 + Q2;
dT3      =  FL4 * (TD - T3) / W3;
dW1      =  FL2 - FG1;
dW2      =  FL3 - FL2 - FG2;
dW4      =  FG3 - FL4;

dx       = [dH1; dH2; dT3; dW1; dW2; dW4];

%% Defining the residuals
T1res    =  H1 - HL1 - HV1;
T2res    =  H2 - HL2 - HV2;
PAres    =  FG3 - FG1 - FG2;
TAres    =  TA - T1 * (PA / P1)^(1 / POL1);
TCres    =  TC - TB * (P3 / PA)^(1 / POL2);
TDres    =  P3 - Z3 * R * TD / (MW * VG4 * 100);
HH1res   =  HH1 - POL1 * (R * 1000/(G * MW)) * (TA - T1);
HH2res   =  HH2 - POL2 * (R * 1000/(G * MW)) * (TC - TB);
WV1res   =  P1 - Z1 * R * T1 / (MW * VG1 * 100);
WV2res   =  P2 - Z2 * R * T2 / (MW * VG2 * 100);
L1res    =  L1 - WL1 * VF1;
L2res    =  L2 - WL2 * VF2;
L3res    =  L3 - W3 * VF3;
P1res    =  P1 - exp(A - B/(T1 - C));
P2res    =  P2 - exp(A - B/(T2 - C));
P3res    =  P3 - exp(A - B/(TD - C));
TP1Ores =  TP1O - ((1 - A1)*T1 + A1*TP1I);
TP2Ores =  TP2O - ((1 - A2)*T2 + A2*TP2I);
```

```matlab
TP3Ores =  TP3O - ((1 - A3)*TD + A3*TP3I);
FG1res  =  FG1 - FG1S * MW * P1 * 100 / (T1 * R * Z1);
FG2res  =  FG2 - XV1f * CV1 * sqrt(P2 - PA);
FG3res  =  FG3 - FG3S * MW * PA * 100 / (TB * R * Z2);
FL2res  =  FL2 - XV2f * CV2 * sqrt(P2 - P1);
FL3res  =  FL3 - XV3f * CV3 * sqrt(P3 - P2);
FL4res  =  FL4 + QC / (HFG3 - HFL4);
TBres   =  TB  - (TA*FG1 + T2*FG2) / (FG1+FG2);
PTH1res =  PTH1 - 0.5 * FG1*(CPO1 + CPI1) * (TA - T1) / MW;
PTH2res =  PTH2 - 0.5 * FG3*(CPO2 + CPI2) * (TC - TB) / MW;
Q1res   =  Q1 - FCP1 * (TP1I - TP1O);
Q2res   =  Q2 - FCP2 * (TP2I - TP2O);

%% Controllers
% L1 controller
XV2opt = params.XV2opt;
KcL1C  = params.KcL1C;
KiL1C  = params.KiL1C;
L1Cres  = ((XV2opt - XV2) + E_L1C*KcL1C + IE_L1C*KiL1C);

% L2 controller
XV3opt = params.XV3opt;
KcL2C  = params.KcL2C;
KiL2C  = params.KiL2C;
L2Cres  = ((XV3opt - XV3) + E_L2C*KcL2C + IE_L2C*KiL2C);

resids = [T1res; T2res; PAres; TAres; TCres; TDres;     ...
          HH1res; HH2res; WV1res; WV2res; L1Cres;       ...
          L2Cres; L1res; L2res; L3res; P1res; P2res;    ...
          P3res; TP1Ores; TP2Ores; TP3Ores; FG1res;     ...
          FG2res; FG3res; FL2res; FL3res; FL4res; TBres;...
          PTH1res; PTH2res; Q1res; Q2res];


%% First order low-pass filters (differential)
tau_XV2f  = params.tau_XV2f;
tau_XV3f  = params.tau_XV3f;
tau_Nf    = params.tau_Nf;
tau_XV1f  = params.tau_XV1f;
tau_FCP3f = params.tau_FCP3f;

dXV2f  = (XV2-XV2f)/tau_XV2f;
dXV3f  = (XV3-XV3f)/tau_XV3f;
dNf    = (N-Nf)/tau_Nf;
dXV1f  = (XV1-XV1f)/tau_XV1f;
dFCP3f = (FCP3-FCP3f)/tau_FCP3f;

%% Integrated controller errors
dE_L1C  =  E_L1C;
```

```
dE_L2C   =  E_L2C;

% SOC1 controller
XV1opt   = params.XV1opt;
KcXV1C   = params.KcXV1C;
KiXV1C   = params.KiXV1C;

Nopt   = params.Nopt;
KcNC   = params.KcNC;
KiNC   = params.KiNC;

load('H_matrices')
SOC1res  = ((Nopt - N) + E_c1*KcNC + IE_c1*KiNC);
cres1    =  c1 - H(1,:)*x(subset);
dE_c1 =  E_c1;

SOC2res  = ((XV1opt - XV1) + E_c2*KcXV1C + IE_c2*KiXV1C);
cres2    =  c2 - H(2,:)*x(subset);
dE_c2 =  E_c2;

out = [dNf; dXV1f; dFCP3f; dx; resids; dXV2f; dXV3f;    ...
       dE_L1C; dE_L2C;SOC1res; cres1; dE_c1; SOC2res;   ...
       cres2; dE_c2];

end
```

```
function [t,x,u] = CL_integrator(const,x0,u0)
% Description

% Add path to get access to SS folder
addpath(genpath('\MATLAB\Original'))

% Initial estimates for the states
if ~exist('c','var');
    const = zeros(0,2);
end

if ~exist('x0','var');
    x0 = SS_init_x;
end

if ~exist('u0','var');
    u0 = SS_init_u;
end

% Calculating the steady state
y_SS = SS_opt(u0,x0,const);

```

```
load('H_matrices.mat')
csp   = c;

x_SS = [y_SS;y_SS(20:21);0;0;y_SS(1);csp(1);0;          ...
        y_SS(2);csp(2);0];
u_SS = [3.1; 1.1; csp(1); csp(2); y_SS(3)];

%% Calculating responses to disturbances / setpoint changes

% Doing a step change in L1 setpoint
u = u_SS;
%u(3)  = u(3)*1.01;

% Parametrizing the function to pass extra parameters to
% it (u and params)
params = SS_init_params;
params.XV2opt = y_SS(20);
params.XV3opt = y_SS(21);
params.Nopt    = y_SS(1);
params.XV1opt = y_SS(2);

% Disturbances
%params.TP1I = params.TP1I + 5;
%params.TP2I = params.TP2I + 2;
%params.TP3I = params.TP3I + 2;
%params.FCP1 = params.FCP1 + 5;

g = @(t,x) CL_model(t,x,u,params);

% Defining the constant, singular mass matrix M
M = diag([ones(1,9),zeros(1,32),ones(1,4),0,0,1,0,0,1]);

 options = odeset('Mass',M,'MStateDependence','none',...
    'MassSingular','yes','RelTol',1e-6,'Vectorized','off');

tspan = 0:0.5:2000;
[t,x] = ode15s(g,tspan,x_SS,options);


% Plot results
if length(dbstack)==1
    CL_plotter(t,x,x_SS,u_SS);
end


end
```