Modelling and Optimization of a Two-Stage Compressor Train

Adriaen Verheyleweghen, *(verheyle@stud.ntnu.no)*

Supervisors: Sigurd Skogestad & Johannes Jäschke

December 10, 2014

Abstract

A model for a two-stage refrigeration cycle was implemented in MATLAB. Dynamic simulations were done to verify the model. The model was optimized at steady state to minimize the energy consumption and the nominal operating point was found. It was found that the model had no unconstrained degrees of freedom left. An alternative case were it was attempted to maximize the heat transfer, was also studied.

Contents

List of Figures

List of Tables

List of Symbols

Symbol	Explanation
γ_{avg}	Adiabatic ratios. Average ratio between specific heats, [-] Corresponding variable in script: POL
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, [-]
λ_i	Coefficients for calculating ν as a function of temperature.
ν	Liquid specific volume, [m^3/kg]
ϕ_i	Coefficients for calculating z as a function of temperature
A	Constant in the Antoine equation
B	Constant in the Antoine equation
C	Constant in the Antoine equation
C_i	Constants for calculating CP as a function of temperature
C_{ii}	Constants for compressor curves
CP	Heat capacity, [J/kg K]
CV	Valve constant, [kg/s$\sqrt{\text{bar}}$]
e_i	Constants for compressor curves
FCP	Product of flowrate and heat capacity of external cooling stream, [W/K]
FG	Mass flow of vapour stream, [kg/s]
FGCD	Discharge flow rate of vapour out of the compressor, [kg/s]
FGCS	Suction flow rate of vapour in to the compressor, [kg/s]
FGV	Volumetric flow rate of vapour into compressor, [m^3/s] (Corresponding variable in script: FGS)
FL	Mass flow of liquid stream , [kg/s]
g	Gravitational constant, [m/s^2]
h	Compressor head, [m] (Corresponding variable in script: HH)
hs	Scaled compressor head, [m] (Corresponding variable in script: HS)
H	Enthalpy in the tank, [J]
HFL	Specific enthalpy of the liquid stream, [J/kg]
HFG	Specific enthalpy of the vapour stream, [J/kg]
k	Polytropic constant, [-] (Corresponding variable in script: G)
M_w	Molecular weight, [kg/mol]
N	Fractional compressor speed based on normal operation, [-]
p	Constant in compressor curve. Value depends on compressor.
P	Pressure in tank, [bar]

PS	Suction pressure, [bar]
q	Constant used in compressor curve. Value depends on compressor.
Q	Heat transferred in evaporator or condenser, [W]
R	Universal gas constant, [J/K mol]
T	Temperature in tank, [°C]
TD	Discharge temperature from compressor, [°C]
TPI	Inlet temperature of process stream to evaporator or condenser, [°C]
TPO	Outlet temperature of process stream from evaporator or condenser, [°C]
TS	Interstage mixing temperature / suction temperature into the compressor, [°C]
UA	Product of heat transfer coefficient and heat transfer area for a heat exchanger, [W/K]
V	Volume of the tank [m^3]
VG	Specific volume of vapour refrigerant, [m^3/kg]
W	Mass accumulated in tank, [kg]
WL	Liquid inventory in tank, [kg]
WLV	Volumetric liquid inventory in tank, [m^3]
WV	Vapour inventory in tank, [kg]
XV	Fractional valve opening of valve, also used to refer to the valve itself, [-]
z	Compressability factor, [-]

List of Abbreviations

Abbreviation	**Explanation**
CV	Controlled variable
DAE	Differential algebraic equation
DOF	Degrees of freedom
HP	High pressure
IP	Intermediate pressure
LP	Low pressure
MV	Manipulated variable
SS	Steady-state
TPM	Throughput manipulator

1 Introduction

This report is the final product of a project on 'Modelling and Optimization of a Two-Stage Compressor Train'. The projected was conducted by Adriaen Verheyleweghen under supervision of Johannes Jäschke and Sigurd Skogestad.

The aim of this project is to model and simulate a two-stage refrigeration cycle. The finished model is to be optimized and a control structure will be proposed to keep the plant at optimal operating conditions despite disturbances. It will be attempted to come up with a self-optimizing control structure. The model will largely be based on previous work done by Basel Asmar [4]. The model will be implemented in MATLAB.

The report is divided into three parts. The first part contains some background theory on optimizing control. The main part of the report summarizes the results from the modelling work, including the dynamic behaviour of the model, the optimal operating conditions given the defined constraints and the self-optimizing control structure. The final section contains a discussion of the results. Additional material such as MATLAB code is placed in the appendix.

1.1 Process description

The process flow diagram of the studied process can be seen in Figure 1. The process is based on a similar plant operated by Exxon Mobile. The working fluid is propylene.

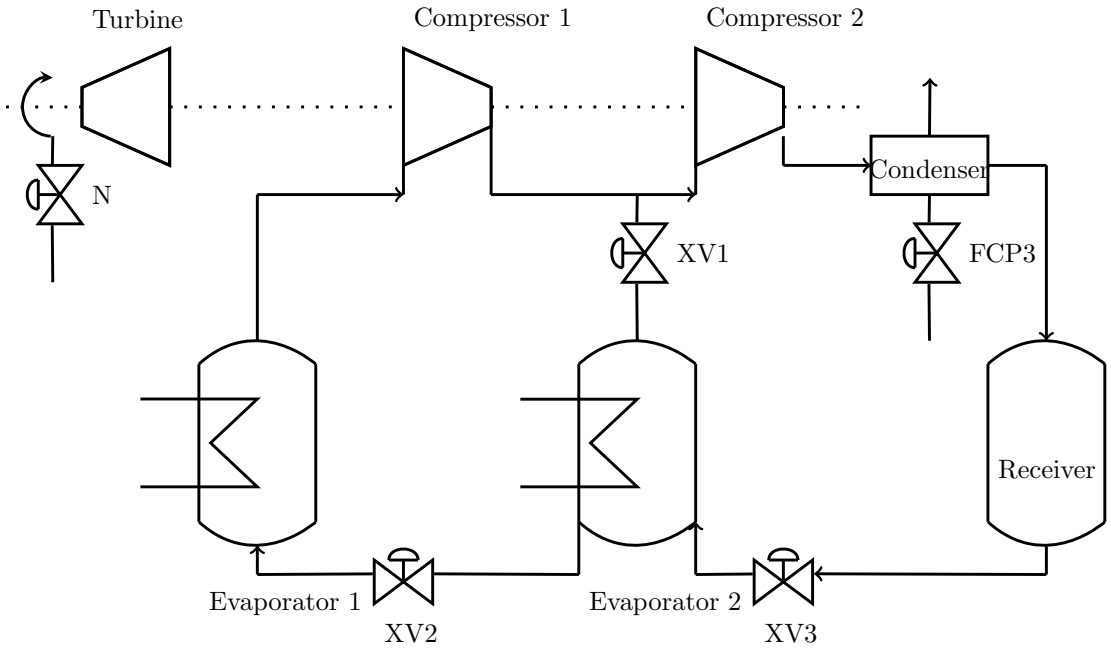


Figure 1: Process flow diagram of the studied process.

The process consists of two compressors in a series configuration which are driven by a single steam turbine. The compressed working fluid is condensed in the condenser using air cooling. A receiver is used as a buffer tank.

From the receiver the process stream is sent to an intermediate pressure (IP) flash evaporator. Only a small process load is removed at this stage. The fraction of gas and liquid can be adjusted by opening or closing the valves XV1 and XV2.

The saturated liquid is expanded over XV2 and evaporated in the low pressure (LP) kettle reboiler. The

main process load is removed in the kettle reboiler. The heating of the working fluid is achieved with heating coils which are submerged in the two evaporators. On the inside of the coils flows a hot process stream which is sought to be cooled down.

A detailed description of the model equations and the necessary assumptions are given in Section 3

Part I
Theory

This part contains some basic theory. It is meant to give a brief summary of the applied methods such as the steps in the top-down procedure for control structure design. It is assumed that the reader already has insight in the described theory. If the reader lacks the necessary background knowledge, it is recommended to read Skogestads paper on the subject [10] for a quick introduction. The idea behind this part is to give the reader a quick reminder of the applied theory, but it is not strictly necessary to read it in order to understand the rest of the report.

2 Top down procedure for control structure design

The top down procedure for control structure design is a useful tool for determining what needs to be controlled [10]. Skogestad defines four steps for the top down procedure:

2.1 Defining the operational objective

The first step is to define the objective. The objective is defined as a cost function which is to be minimized, subject to a set of equality constraints and a set of inequality constraints.

$$\min_u J(u,d) \quad s.t \quad g_1(u,d) = 0 \quad , \quad g_2(u,d) \leq 0 \tag{1}$$

Common objectives for control include maximization of throughput and minimization of energy consumption. Linear combinations of multiple objectives can also be used to find the optimal trade-off. The objectives very often relate to some kind of economic cost.

The constraints, i.e. the functions g_1 and g_2, must also be defined. The limits imposed on the variables depend on the physical constraints of the equipment, product specifications etc. The model equations are included as part of the equality constraints.

2.2 Degree of freedom analysis

The degrees of freedom of a system are used to control it. The number of dynamic degrees of freedom can be found by counting the number of MV's. Typically only those degrees of freedom which have a steady-state effect will have an effect on the cost. The number of steady-state degrees of freedom can be expressed as: [10]

$$N_{ss} = N_{MV} - N_0 \tag{2}$$

Where N_0 are the number of dynamic degrees of freedom which have no steady-state effect. Jensen and Skogestad describe a method for finding the steady-state degrees of freedom of refrigeration cycles [7]. For a single component, the steady-state degrees of freedom equal to the number of MVs minus the number of liquid receivers exceeding the first one.

After the degrees of freedom have been found, the process can be optimized using the remaining degrees of freedom. The problem is a mathematical programming problem and is solved using numerical tools. Constraints that are on their respective limits are called active constraints. Active constraints will need to be be controlled since they have a large effect on the cost function, as can be seen from Equation (3). Equation (3) shows the truncated Taylor expansion of the cost function around the nominal point, written in terms of the independent variables.

$$J(c,d) = J^*(c,d) + \lambda \left(c_{act} - c^*_{act}(d)\right) + (c - c^*(d))^T J_{cc}(c - c^*(d)) \tag{3}$$

In the above equation, c are the independent variables and c_{act} are active constraints. λ is the Lagrange multiplier. The star (*) subscript indicates values at the nominal point. As can be seen, the cost function is

linearly correlated to the active constraints. For small disturbances, this means that the active constraints contribute more to the loss than the unconstrained variables. This means that the active constraints must be controlled.

2.3 Implementation of optimal solution

The optimal solution that is found is implemented. As discussed previously, some of the degrees of freedom must be used to control the active constraints. Any remaining degrees of freedom can be used to control the process in such a way that the operation is optimal even when disturbances occur. This is called self-optimizing control, since the loss stays acceptably low even though the process is disturbed. It would be preferable to control the gradient of the cost directly, such that $J_u = 0$ at all times. Unfortunately this is often not possible since the gradient is difficult to measure. Instead, one chooses to control a combination of inputs such that it stays at a constant setpoint. If H is chosen correctly, this will ensure that $J_u \approx 0$.

$$c = Hy \tag{4}$$

Where H is called the selection matrix. There are different methods to find H:

- Brute force evaluation. Evaluate $Loss = J(c + \epsilon_n, d) - J^*(u)$, where ϵ_n is the implementation error, and select the variable c with the smallest loss.
- Nullspace method. Choose H such that $HF = 0$, where F is the sensitivity matrix defined as $F = \frac{\partial y^*}{\partial d}$. The nullspace method gives zero loss from optimal operation for any disturbance given that there is no implementation error, but it can only be used if there are enough measurements. The number of measurements must satisfy the inequality $n_u + n_d \leq n_y$. For a detailed description of the nullspace method, see Alstad and Skogestad [1].
- Exact local method. Since the nullspace method is only accurate if there is no implementation error, the exact local method was developed by Alstad et. al. [2] with roots in the exact method proposed by Halvorsen et. al. [6]. The exact local method involves minimizing the problem

  $$\min_H ||J_{uu}^{0.5}(HG^y)^{-1}H[FW_dW_{n^y}]||_F \tag{5}$$

  For information about the notation it is referred to the original paper by Alstad. Otherwise the method and the corresponding notation will be explained when used later in the report.

Alstad and Skogestad give the following criteria for the choice of self-optimizing variables

1. c should be insensitive to disturbances at the optimum, i.e. the optimum should not be expected to move when the process is disturbed.
2. The optimum should be flat to minimize the effect of implementation error. That is to say that J_{cc} should be small, i.e. that the concavity of J^* is small.
3. c should be easy to implement and control. This means for example that temperature and pressure measurements are preferred to composition measurements, as they are cheaper and more reliable.

2.4 Production rate and inventory control

Lastly, it is necessary to choose where to set the production rate. The throughput manipulator lets the operators choose how to operate the plant. If no degrees of freedom are left, the process does not have a possibility to set the production rate. Instead, the throughput is determined by upstream process units. The location of the TPM should be chosen such that it is close to any bottlenecks in the process [3].

Gas and liquid inventories must be controlled to keep them within acceptable bounds as defined by the constraints. Aske and Skogestad defined consistent inventory control 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"*[3]. In other words, the mass balance must be satisfied for the overall process

and for the individual units. Local consistency adds the additional criterion that each unit must fulfil the mass balance, even when viewed separately from the process as a whole.

For a closed system, it is advised to set the TPM to one of the streams inside the loop to avoid snowballing. The snowballing effect describes the steady-state phenomenon when a small change in the feed to the system results in a large change in the recycle flow rate [9]. This effect can be avoided by moving the TPM inside the recycle loop. It is also necessary to leave one of the inventories inside the loop uncontrolled. Not doing so will overspecify the system and violate the consistency rules [3].

Part II
Results

3 Modelling

The necessary equations and correlations for modelling the process were found in Chapter 3 of Asmar's thesis [4]. A complete description of the model can be found there. The core assumptions done in the modelling can be summarized as follows:

- Compressors are modelled based on compressor curves for existing equipment. This means that the model must be rewritten if applied to a different system.
- The coils in the evaporators are fully submerged at all times. The heat transfer coefficient for the heat transfer between the working fluid and the hot process streams is assumed to be independent of temperature. A constant value for UA is thus used. It is also assumed that there is no heat loss through the walls of the equipment.
- The process streams are modelled as single-phase streams. Their thermodynamic properties such as heat capacity are assumed to be constant on the given temperature interval.
- The thermodynamic properties of propylene are approximated as linear functions of the temperature over the studied temperature interval. The relationship between the saturation pressure and temperature is given by the Antoine equation.
- The process equipment has no capacity and does not undergo thermal expansion.
- The levels are assumed to be linearly correlated to the volumetric liquid hold-ups in the tanks. This assumption fails if the cross-sectional area of the tank is not constant.
- Propylene expands adiabatically over expansion valves. This results in a pressure and temperature drop, as heat is removed from the working fluid to form flash-vapour.

The system contains a number of implicit variables such as the temperatures. The system can be written as a combination of differential and algebraic equations. DAE-systems of this kind are solved in MATLAB with the built-in ode15s procedure.

Inspection shows that the system has 5 degrees of freedom. These 5 degrees of freedom are represented as control valves in Figure 1. XV1, XV2 and XV3 are valve openings, whereas N is to the rotational speed of the steam turbine and FCP3 corresponds to the heat removed from the condenser.

3.1 Model equations

The following sections describe the system of equations that has to be solved, in detail. The model equations will be presented in a logical order, though the final model does not depict this.

All model equations are taken from Asmar [4]. They are repeated here for reader convenience. Asmar describes a general system consisting of n compression stages. The notation has been kept even though this report considers a process consisting of only two evaporators and two compressors.

All parameters that are used in the equations are summarized in Appendix A.

3.1.1 Evaporators

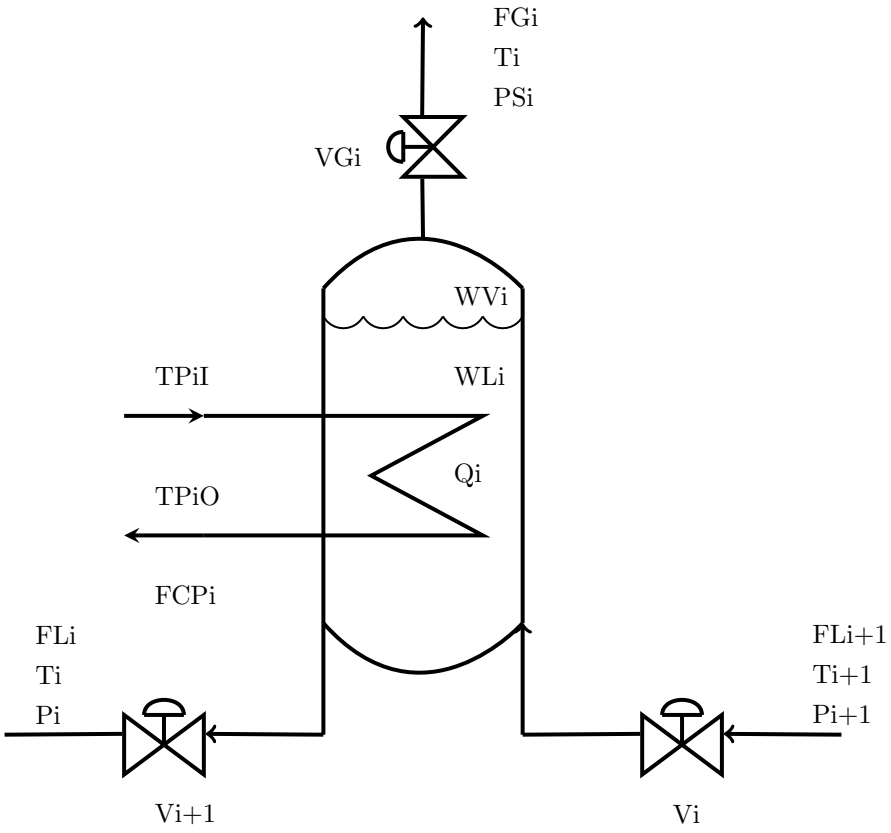


Figure 2: Illustration of an evaporator.

The mass balance for the evaporator shown in Figure 2 can be written as

$$\frac{dW_i}{dt} = FL_{i+1} - FL_i - FG_i \tag{6}$$

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. From the hold-up W, the level can be calculated indirectly. Assuming that the cross-sectional area of the evaporator is constant, i.e. that the evaporator is cylindrically-shaped, then the level of the refrigerant is proportional to the volumetric liquid hold-up, WLV.

$$L_i \propto WLV_i \tag{7}$$

The volumetric liquid hold-up can be calculated as

$$WLV_i = WL_i \cdot v_{f,i} \tag{8}$$

Where v_f is the liquid specific volume of propylene. The specific volume is assumed to be a linear function of temperature, and can thus be calculated as

$$v_{f,i} = \lambda_1 + \lambda_2 \cdot T_i \tag{9}$$

where λ_i are constant coefficients which are determined experimentally. The values for the coefficients are given in Table 8 in the Appendix.

The liquid hold-up WL can not be calculated directly, but must instead be calculated iteratively from the mass balance over the vessel inventory

$$W_i = WL_i + WV_i \tag{10}$$

Flow rates are determined by the fractional valve openings of the control valves. The flows are assumed to be proportional to the square root of the pressure drop over the valve.

$$FL_{i+1} = XVL_i \cdot CVL_i \sqrt{\Delta P} \tag{11}$$

In the above equation, XVL is the fractional valve opening and CVL is a constant coefficient. The same equation is used for calculating the vapour flow rates.

$$FG_i = XVG_i \cdot CVG_i \sqrt{\Delta P} \tag{12}$$

For the first evaporator, there is no valve on the vapour outlet, such that the flow rate solely depends on the suction pressure of the compressor.

The pressure in the evaporator is the saturation pressure P_i, which is a function of the temperature. Antoine's equation is used to calculate the saturation pressure as a function of temperature.

$$P_i = \exp\left(A - \frac{B}{T_i - C}\right) \tag{13}$$

In the above equation, A, B and C are constants. The values of the constants are given in Table 3 in the Appendix. The saturation temperature T_i must be calculated from the energy balance. Since the energy balance cannot be solved explicitly for the temperature, it must be calculated iteratively.

$$H_i = WL_i \cdot HFL_i + WV_i \cdot HFG_i \tag{14}$$

The dynamic energy balance for the evaporator can be written as

$$\frac{dH_i}{dt} = (HFL \cdot FL)_{i+1} - (HFL \cdot FL)_i - (HFG \cdot FG)_i + Q_i \tag{15}$$

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 inside the heating coils to the refrigerant in the vessel. The specific enthalpies are assumed to be linear functions of the temperature

$$HFG_i = \zeta_1 + \zeta_2 \cdot T_i \tag{16}$$

$$HFL_i = \delta_1 + \delta_2 \cdot T_i \tag{17}$$

The heat load Q_i is specified through the temperatures of the process stream in and out of the evaporator, in addition to the process stream mass flow rate. These parameters are either specified or used as disturbances in the calculations.

$$Q_i = FCP_i \left(TP_i i - TP_i o\right) \tag{18}$$

$$TP_i o = \left(1 - \alpha_i\right) T_{i+1} + \alpha_i TP_i i \tag{19}$$

where

$$\alpha_i = \exp\left(\frac{-U_i \cdot A_i}{FCP_i}\right) \tag{20}$$

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

3.1.2 Condenser and receiver

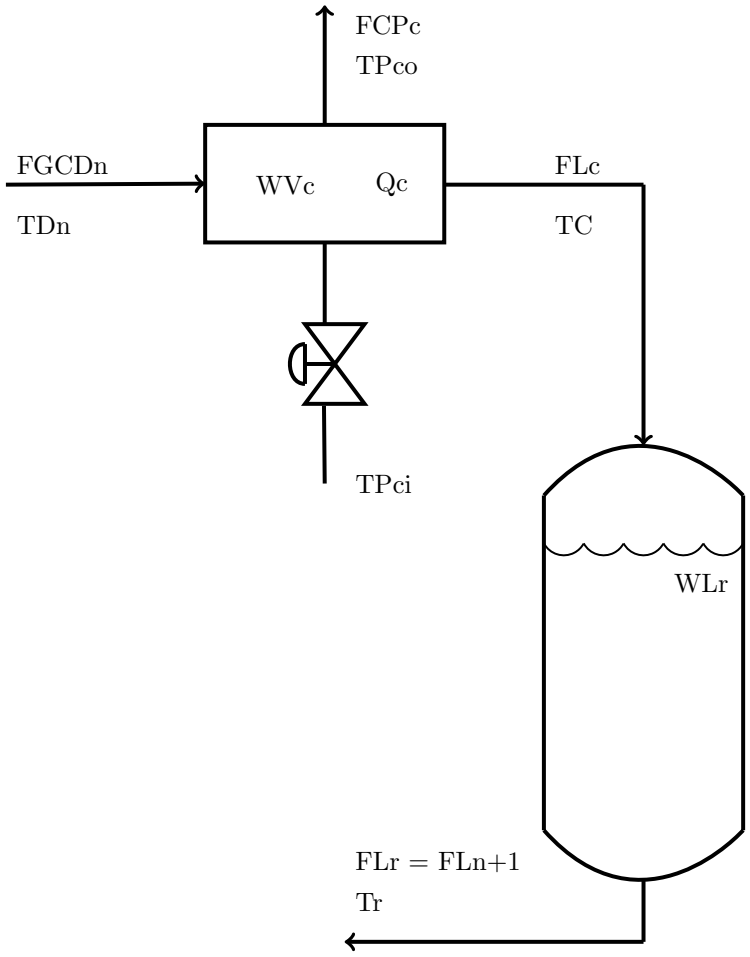


Figure 3: Illustration of the condenser and receiver

After the vapour refrigerant exits the second compressor, it enters the condenser with pressure equal to the discharge pressure of the compressor. The refrigerant exchanges heat with cold air and condenses. The temperature is calculated using the Antoine equation. Condensed refrigerant is stored inside the receiver, as it is assumed that no liquid refrigerant remains in the condenser. The mass balance over the condenser can thus be written as

$$\frac{dWV_c}{dt} = FGCD_n - FL_c \tag{21}$$

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

$$VG_c = V_c + V_r - (WL_r \cdot v_{f,r}) \tag{22}$$

The receiver liquid hold up WL_r is calculated from a total balance over all inventories in the system.

$$WL_r = W - \sum_{i}^{n} W_i - WV_c \tag{23}$$

For this system $n = 2$, meaning that the total mass in the system equals to the masses contained in the two evaporators plus the mass contained in the condenser and the receiver.

It can be shown that the energy balance simplifies to

$$\frac{dT_r}{dt} = \frac{FL_c}{WL_r}(T_c - T_r) \tag{24}$$

In the above expression, the liquid refrigerant flow rate from the condenser, FL_c, can be calculated as

$$FL_c = \frac{Q_c}{HFG_c - HFL_c} \tag{25}$$

The specific enthalpies of the gas phase and the liquid phase are calculated using Equations (16) and (17) with the corresponding condenser temperature. Q_c is calculated the same way as for the two evaporators, using Equations (18), (19) and (20).

3.1.3 Compressor

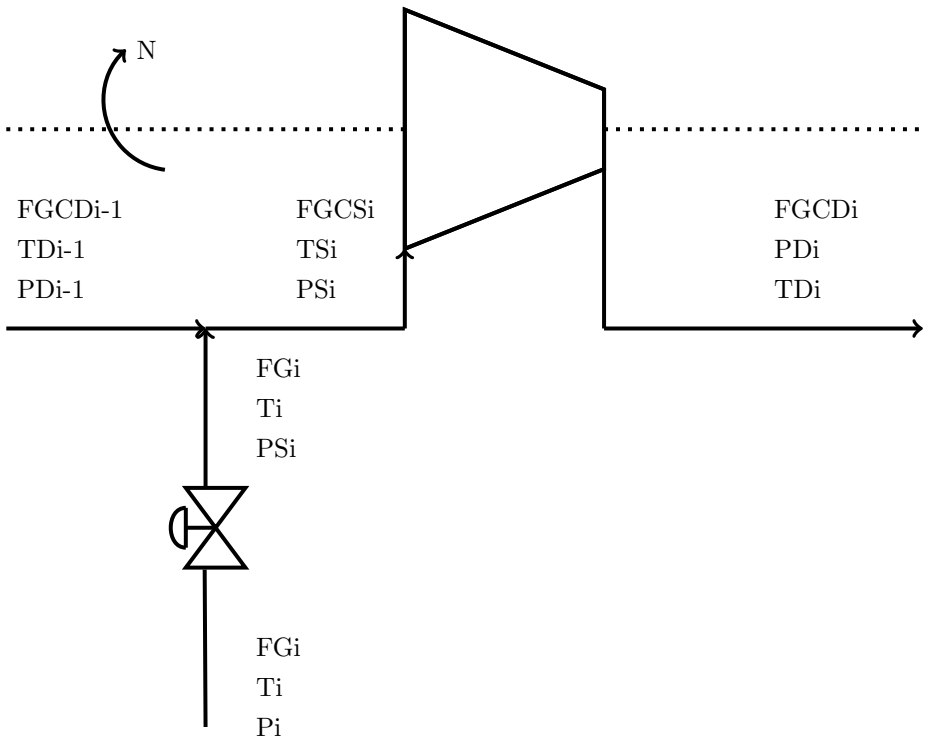


Figure 4: Illustration of a generic compressor stage. Notice that the figure must be modified for the first compression stage, as it does not have interstage mixing.

The suction mass flowrate into the compressor is given by

$$FGCS_i = FGV_i \frac{M_w \cdot PS_i}{TS_i \cdot R \cdot z_i} \tag{26}$$

This is the equation of state where TS and PS are the suction temperature and pressure, 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 compressability factor z is approximated as a linear function of the temperature

$$z = \phi_1 + \phi_2 \cdot T \tag{27}$$

As can be seen from Figure 4, the suction mass flowrate entering the compressor generally must satisfy the following mass balance

$$FGCS_i = FGCD_{i+1} + FG_i \tag{28}$$

where $FGCD$ is the discharge mass flow rate from the previous compressor and FG is the vapour flow rate from the corresponding evaporator. Equation (28) does not apply to the first compressor stage, as there is no interstage mixing.

The interstage mixing temperature TS is calculated from the energy balance.

$$TS_i = \frac{TD_{i-1} \cdot FGCD_{i-1} + T_i \cdot FG_i}{FGCD_{i-1} + FG_i} \tag{29}$$

The compressor equations are based on compressor curves. These are curves relating to the performance of the compressor which are found from experimental data.

$$\frac{FGV_i}{N^q} = f_i(hs_i) \tag{30}$$

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

$$hs_i = \frac{h_i}{N^p} \tag{31}$$

p is a constant value depending on the compressor.

For polytropic compression, the compression head can be expressed as

$$h_i = k_i \frac{R}{g \cdot M_w} \left(TD_i - TS_i\right) \tag{32}$$

g is the gravitational constant. The discharge temperature can be calculated as

$$TD_i = \frac{TS_i}{\left(\frac{PD_i}{PS_i}\right)^{\frac{1}{k_i}}} \tag{33}$$

k_i is a constant value which is defined as

$$k_i = \frac{n}{n-1} = \frac{\eta\gamma_{avg,i}}{\gamma_{avg,i} - 1} \tag{34}$$

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

$$\eta_i = f2_i(hs_i) \tag{35}$$

Similarly to the compressor curve f, $f2$ 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.

$$\gamma_{avg} = \frac{1}{2}\left(\frac{CPI_i}{CPI_i - R} + \frac{CPO_i}{CPO_i - R}\right) \tag{36}$$

CPI is the heat capacity at inlet (suction) conditions, whereas CPO is the heat capacity at outlet (discharge) conditions. The heat capacities are approximated as fifth order polynomials of the temperature.

$$CP = C_1 + C_2 \cdot T\left(C_3 + C_4 \cdot T\left(C_5 + C_6 \cdot T\right)\right) \tag{37}$$

The values of the coefficients in Equation (37) are given in Table 4 in the Appendix.

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

$$f_1(hs_1) = \frac{FGV_1}{N^q} = \frac{C_{11} \cdot hs_1 - C_{12}}{C_{13}} \tag{38}$$

For the second compressor

$$f_2(hs_2) = \frac{FGV_2}{N^q} = C_{24} + C_{25}\log\left(\left(\sqrt{TX^2+1}\right) - TX\right) \tag{39}$$

TX is defined as

$$TX = \frac{C_{21}}{\tan\left(\frac{C_{22}-hs_2}{C_{23}}\right)} \tag{40}$$

The parameters for Equation (38) can be seen in Table 9 in the Appendix. The parameters for Equation (39) can be seen in Table 10 in the Appendix.

In a similar fashion, $f2$ is found by fitting actual performance data. For both compressors, the following relationship can be used

$$\eta_i = f2_i(hs_i) = e_1 \cdot hs_i + e_2 - 10^{(e_3 \cdot hs_i - e_4)} \tag{41}$$

The parameters for Equation (41) can be seen in Table 11 in the Appendix.

3.2 Implementation in MATLAB

The original model by Asmar is written in ACSL and had to be translated to MATLAB. The differences between the original script by Asmar and the script that was implemented in MATLAB were mainly a result of the differences between the programming languages. Though the actual model equations stayed the same, the rest of the script had to be rewritten to work in MATLAB. ACSL is structured very unlike MATLAB. First, all variables are initialised and initial estimates are assigned to them. The equations are then stated as algebraic or differential equations and integrated directly through a call to the INTEG-function. The order of the equations does not seem to be of importance, so that variables can be used before values have assigned to them. In MATLAB it is necessary to define the set of equations in chronological order and then call an integrator such as ode15s from a different script to integrate the equation system.

The resulting MATLAB-code can be seen in Appendix **REF**

4 Dynamic simulation

In this section, the dynamic behaviour of the developed model is described. Though the focus of this report is not primarily to study the dynamic behaviour of the model, it is necessary to do so in order to find out which variables need to be controlled in the stabilizing layer of the control structure, i.e. finding variables which are not self-regulatory [8]

4.1 Controlling the levels

It was attempted to find a steady state solution of the equation system, i.e. solving the following equation

$$f(\mathbf{x}) = \begin{bmatrix} f_{\text{diff}} \\ f_{\text{alg}} \end{bmatrix} = \mathbf{0} \tag{42}$$

$$f_{\text{diff}} = \frac{\partial g(\mathbf{x})}{\partial t} \tag{43}$$

However, simulations revealed that the levels were not self-regulatory, and that they would integrate to positive or negative infinity. A fairly constant level is desired to make sure that the coils in the evaporators are submerged at all times. It was therefore decided to implement controllers before attempting to optimize the system. P-controllers were chosen because of their ease of implementation. The resulting setpoint offset can be accepted because there is no need to control the levels tightly.

In correspondence with the rules on consistent inventory control proposed by Aske and Skogestad [3] one of the inventories was left uncontrolled. This was done to avoid over-specification of the mass streams in the closed system. It was chosen to control the levels of the two evaporators, L_1 and L_2, and leave the level of the receiver, L_3, uncontrolled. The uncontrolled inventory will be indirectly controlled when all the other inventories are controlled (the total mass in the system is conserved), but the control will be slow due to the large time delay. This large time delay will cause large variations in the level. Since the receiver is the largest inventory in the loop, it is best suited to deal with level variations. The two smaller inventories have smaller tolerances for level variations, and thus require tighter level control. An illustration of the proposed control structure can be seen in Figure 5

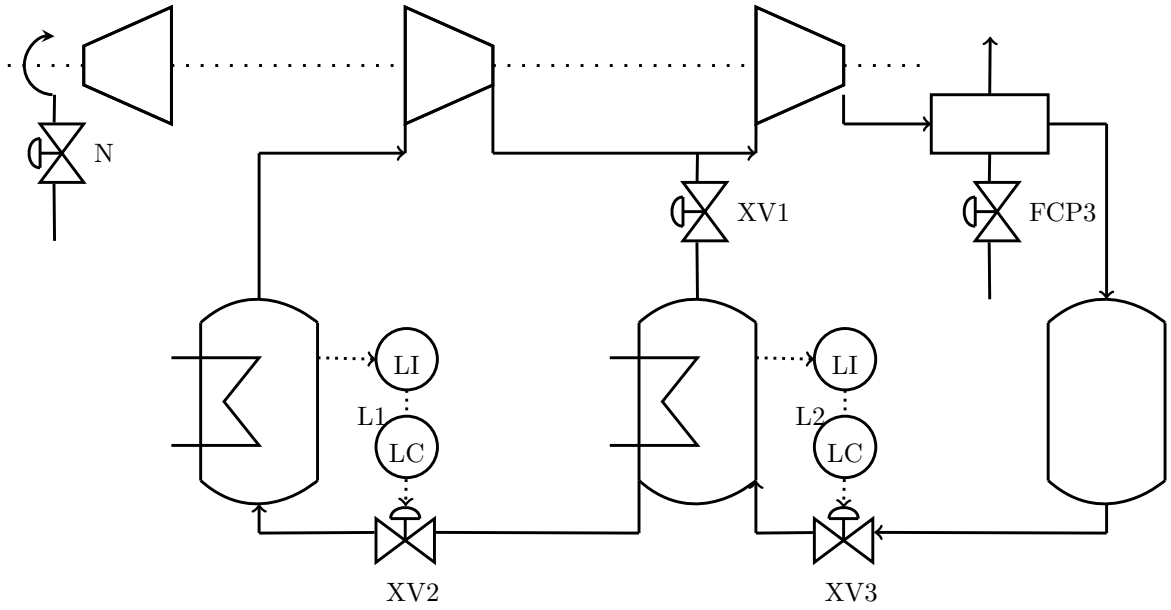


Figure 5: Process flow diagram of the studied process with the two added level control structures.

The selection of MV's for controlling the two levels was based on intuition and general pairing rules. XV2 was paired with L_1 since it has the most direct effect. XV3 could also have been chosen, but this would have lead to local inconsistency [3] and is a violation of the "pair close" tuning rule. After closing the first level control loop, XV3 and XV1 are the only remaining CV's having a direct effect on the L_2. Both fullfill the "pair close"-rule, but since XV3 has a more direct effect on the level, L_2 was paired with XV3.

The controllers were tuned by trail and error. It was found that a K_c-value of 0.5 gave satisfactory disturbance rejection without causing the controllers to saturate.

4.2 Dynamic responses

After the liquid inventories were controlled, the dynamic performance of the model was tested. The responses to a setpoint change in L1 are seen in Figure 6. A disturbance in TP1I was also simulated. The results of the simulation can be seen in Figure 7.

The responses seem okay at first glance. A more in-debt analysis of the results can be found in the discussion

in Section 7.1. The main focus of this project is the steady-state behaviour of the model, and the rest of the report will deal with the steady-state behaviour of the model unless something else is stated.

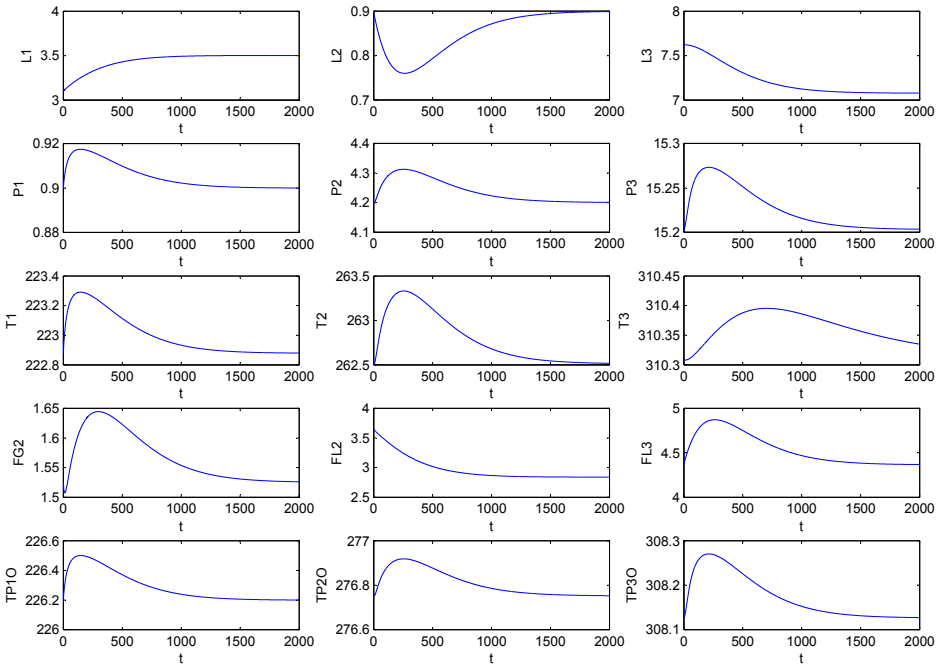


Figure 6: Dynamic responses for some selected variables for a 10% increase in the setpoint of L1

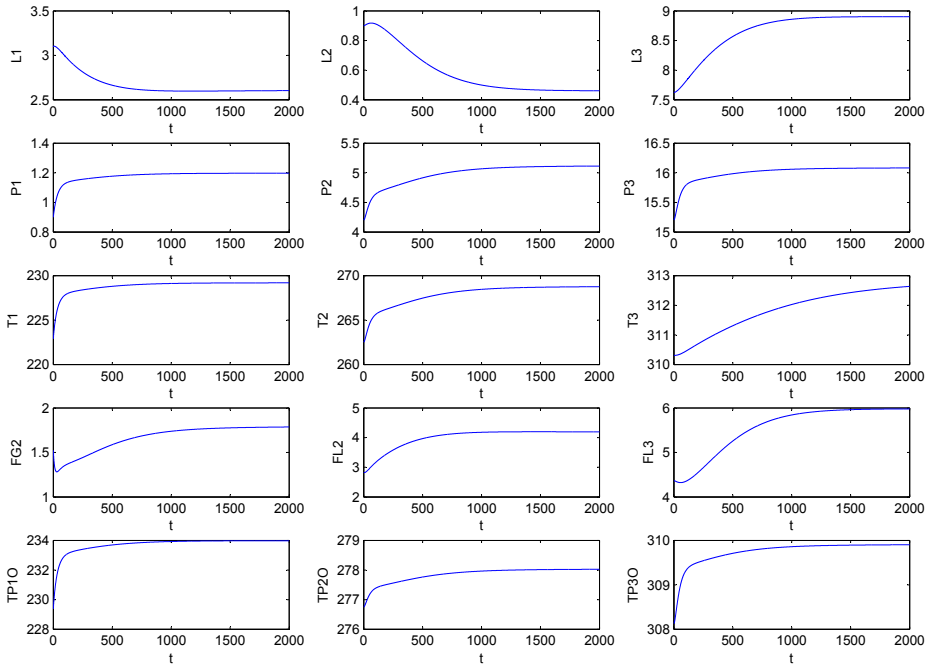


Figure 7: Dynamic responses for some selected variables for a 5% increase in TP1I.

5 Steady-state optimization

This section describes how the optimal operating conditions were found.

5.1 Determining the steady-state effect of the levels

At first the author ran the optimization with the setpoints for the level controllers being two optimizable degrees of freedom. The optimization showed that the setpoints influenced the total energy consumption. However, closer inspection showed that this behaviour was caused by numerical instability and that the cost function was not influenced by the levels after all. Figure 8 shows the optimal value of the cost function J as a function of the setpoints for L2 and L1. The cost function is more or less constant for all combinations of L2 and L1, except when both setpoints go towards their minimum values. Most likely this is a result of the optimizer not being able to find a solution that satisfies the constraints, and thus stalling prematurely. The figure also shows a small 'trench' for L1 approximately equal to 3.7. It is not known what causes this, but it is likely to be attributed to numerics and not represent an actual steady-state effect of the level setpoints.

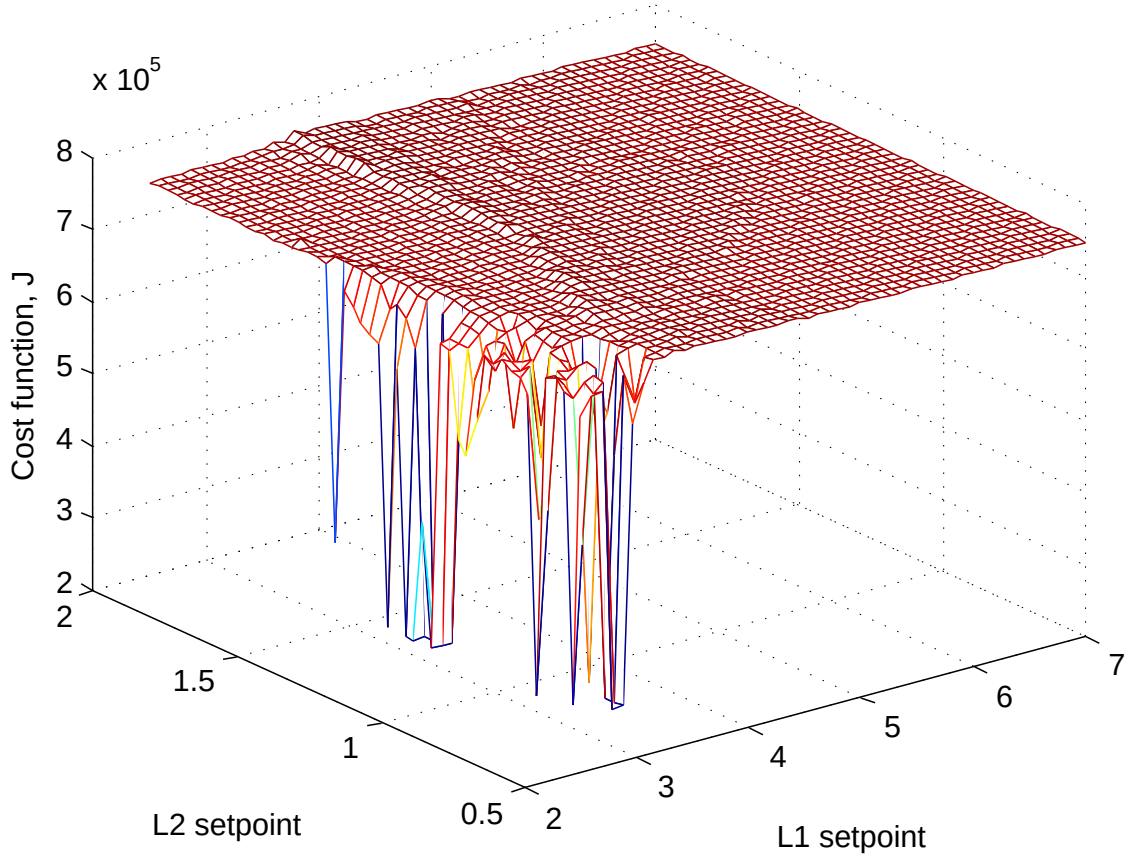


Figure 8: Optimal value of cost function J with fixed values for the setpoints of L1 and L2

Though levels can have a steady-state effect, as is the case for non-equilibrium liquid phase reactors [8], they do not effect the steady state in this case. The opposite would have been true if the heat-transfer in the cooling coils would have depended on the liquid level in the tank, but this effect is not included in the model.

The two level-controllers remove two steady-state degrees of freedom, so that the remaining steady-state

degrees of freedom is reduced to three. This follows from Equation (2). The same result can be found by applying the method to find steady-state degrees of freedom of refrigeration cycles described by Jensen and Skogestad.

5.2 Nominal operating conditions

The nominal operating conditions can be found by minimizing a nonlinear mathematical programming problem. The cost function that is to be minimized is the total energy consumption of the two compressors. Constraints were introduced to keep the variables within a reasonable operating ranges. In chapter 5 of Asmar's thesis[4], the ranges for some of the process variables are given. The constraints for the measured variables and the inputs are given in Table 1 and Table 2, respectively.

Table 1: Ranges for the measured variables, taken from [4]. 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	L_1 [m^3]	L_2 [m^3]	L_3 [m^3]	P_1 [bar]	P_2 [bar]	P_3 [bar]	$T_{P,1,o}$ [K]	$F_{L,2}$ [kg/s]	$F_{L,3}$ [kg/s]	$F_{G,2}$ [kg/s]
Lower boundary	2.9	0.6	6.9	0	3	12	200	0	0	0
Upper boundary	6.4	1.6	13.7	2	6	18	300	5.47	7.18	6.31

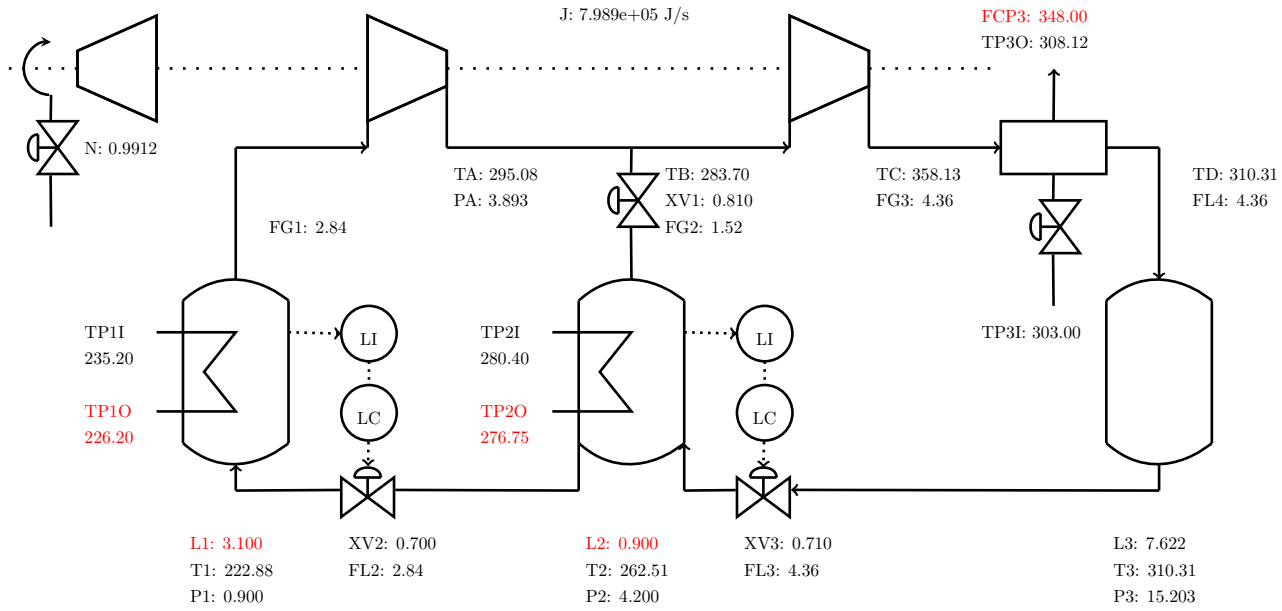
Table 2: Ranges for the input variables, taken from [4]

Variable	XV1 [-]	XV2 [-]	XV3 [-]	N [-]	FCP3 [$\frac{J}{s\,K}$]
Lower boundary	0	0	0	0.9	116
Upper boundary	1	1	1	1.1	348

Other, unconstrained state variables were specified to lie within the range zero to infinity. In addition to the constraints on the inputs and the state variables, the optimizer needs a set of equality and inequality constraints. The DAE-system was provided as an equality constraint, meaning that the steady-state optimal conditions are found. The setpoints for the levels were also set as equality constraints. The maximum allowable outflow temperatures of the process streams from the evaporators were specified as inequality constraints.

In addition to the specified constraints for the variables, one must also specify the load of the refrigeration cycle, otherwise the optimizer will attempt to shut down the process by letting N and FCP3 go to their lower limits. XV1 was approaching its lower limit, but the upper pressure constraint for P2 was reached before this happened. Relaxing the pressure constraint led to XV1 fully closing. The load was specified as inequality constraints for the outflow temperatures of the process streams from the evaporators. TP2O is required to lower than or equal to 276.75°C. The upper limit of TP1O is set to 226.20°C. These values were found in Appendix A of Asmar's thesis [4]

Figure 5.2 shows the calculated optimum given the set of constraints previously mentioned. It can be noted that FCP3 is at its upper limit, making it an active constraint. Both constrained process outflow temperatures, TP2O and TP1O, are at their upper allowable limits. This is to be expected, since lowering their temperatures would lead to higher energy consumption. These constraints are thus also active. All active constraints are coloured red in Figure 5.2. The cost function evaluates to an energy consumption of $7.989 \cdot 10^5$ W.



5.3 Self-optimizing control

It was originally planned to find a self-optimizing control structure for the process. However, this requires unconstrained degrees of freedom. The results from the optimization show that three constraints are active at the nominal point. This leaves zero unconstrained degrees of freedom left which can be used for self-optimizing control. In other words, the process is constrained in such a way that only one solution is possible at the optimal operating point.

6 Alternative process layout

In an attempt to free one degree of freedom in the system, it was suggested that one could connect the process streams such that the outflow from the second evaporator becomes the inflow of the first evaporator. The idea behind this is to decrease the number of independent heat load specifications from two to one. The cost function was also changed such that the objective was to maximise heat transfer rather than minimizing the energy consumption. It was thought that this would give an additional degree of freedom being the distribution of transferred heat between the two evaporators. The proposed process layout can be seen in Figure 9

Since the heat transfer is very different in the two evaporators, it was not possible to just connect them without adjusting the parameters. This proved to be a non-trivial task, as the optimizer struggled to find a solution of the updated process. The flow rates of the two process streams were combined to give the new flow rate of the process stream. The product of the heat transfer areas and the heat transfer coefficients, UA, were difficult to adjust. Hereafter, UA will be referred to simply as the heat transfer area. This makes logical sense because the heat transfer coefficient is expected to remain relatively constant, while the area can be moved from one evaporator to the other. It should be mentioned that the distribution of area is a question of design optimization and not control optimization. As such, it does not really concern this project. However, since the alternative case needs to have properly defined parameters, it will be explored how changing the design could lead to a less or more interesting case.

It was attempted to move heat transfer area from one evaporator to the other while keeping the overall area constant. This led to problems as the optimizer would not find a solution. Initial conditions were either badly defined or the system was set up in such a way that no solution existed. It was observed that the optimizer was able to find a solution if both heat transfer areas were reduced to about 30 W/K. It was also noticed that small changes in the distribution of the areas led to a change in the active constraints. In order to find the active constraint regions, the system was optimized for a range of areas around 30 W/K. The

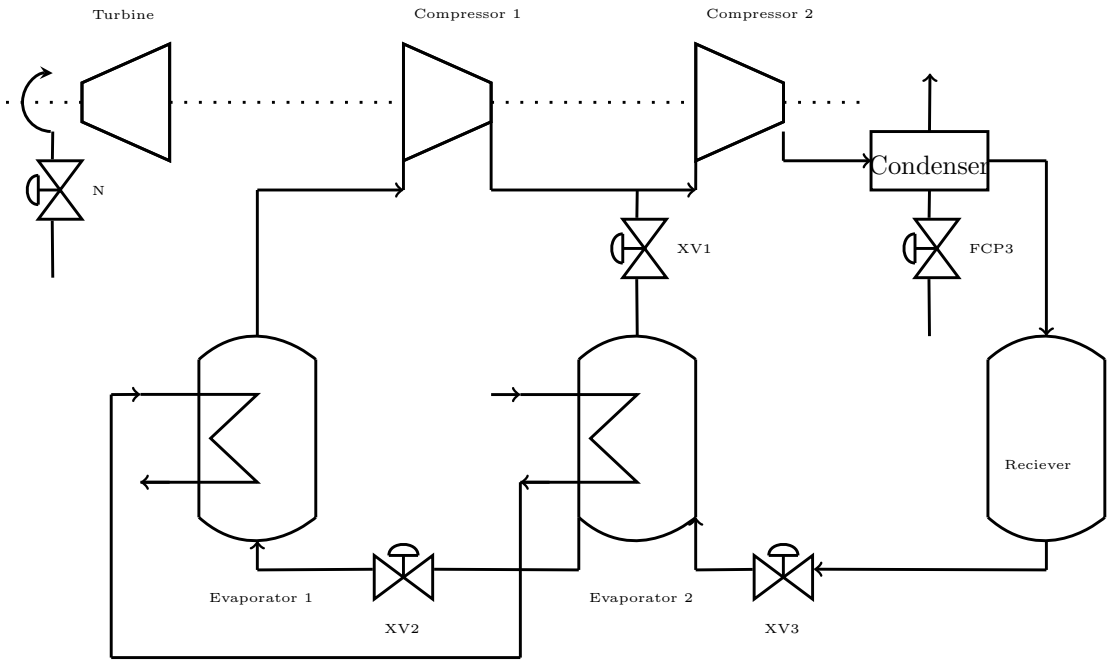


Figure 9: Process flow diagram of the considered alternative process.

resulting active constraint regions can be seen in Figure 10. The corresponding cost function can be seen in Figure 11

Eight main constraint regions are observed. The active constraints in each region are, beginning on the top, going clockwise:

- Region 1 (grey): XV1, FCP3 and XV2.
- Region 2 (dark blue): XV1, XV2 and XV3.
- Region 3 (red): XV1, XV3 and P3.
- Region 4 (purple): N, XV1 and XV3.
- Region 5 (orange): N, XV1 and FCP3.
- Region 6 (blue): N, FCP3 and XV2.
- Region 7 (pink): N, XV2 and XV3.
- Region 8 (green): XV2.

All the constraints are active and at their upper limits. There are some other constraint regions as well, but these regions are very small. A total of 21 regions were found.

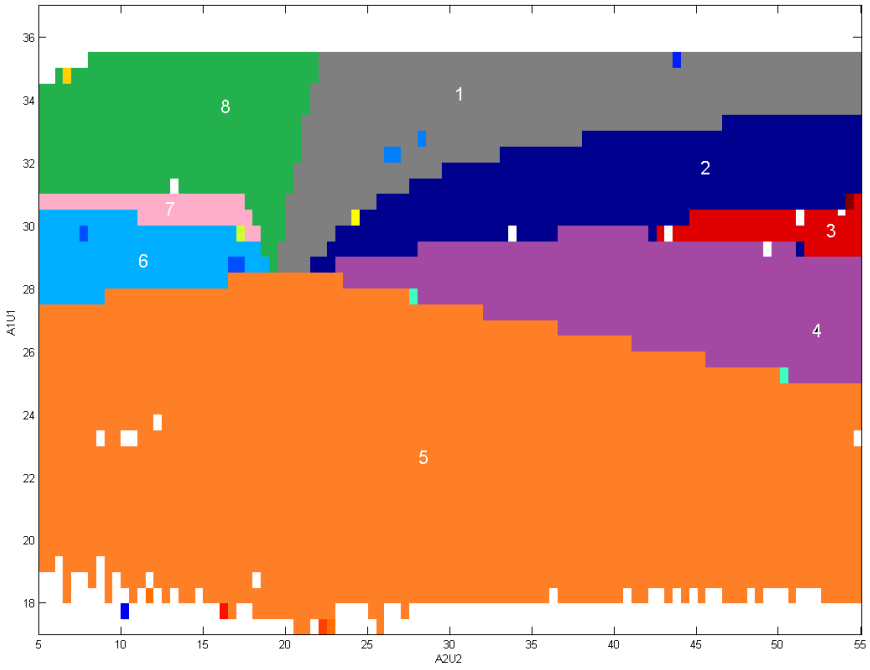


Figure 10: Active constraint regions for different combinations of the heat transfer areas in the two evaporators.

Figure 11: Cost as a function of the heat transfer areas U1A1 and U2A2.

Since there are three active constraints in all regions except for region eight, it is not possible to implement a self-optimizing control structure for most combinations of U1A1 and U2A2. It would be theoretically possible to implement self-optimizing control in region eight since there is only one constrained degree of freedom

in this region. That leaves two unconstrained degrees of freedom for optimization. It is also observed that the border between region two and region three gives the lowest possible value of the cost function for the studied intervals of U1A1 and U2A2. Which region it is best to be in will be discussed in greater detail in the discussion in Section 7

Part III
Discussion, conclusion and suggestions for further work

7 Discussion of the results

This section tries to explain the results from Part II.

7.1 Dynamic responses

The dynamic behaviour of the developed model is described in Section 4. Since the levels were not self-regulatory, they had to be controlled. As can be seen from Figure 6 and Figure 7, the levels are now controlled. The responses are relatively slow, as L1 seems to have a closed-loop time constant of about 250 seconds. This response can be attributed to the poor tuning of the controller. The P-controller was not tuned systematically. Instead, a Kc value was chosen such that the controller would not saturate given a reasonable (10%) disturbance. Faster control can be achieved by tuning the controller more aggressively, though this might lead to saturation. It is also observed that L2 reacts quite slowly to a setpoint change in L1 due to delay. This can be avoided by implementing a cascade or feed-forward controller. However, this was not deemed necessary. Finally, it is observed that L3 is as slow as the slowest of the two other levels. This is because it is not controlled directly.

From Figure 6 it can be seen that there is a steady-state offset for L1. As the setpoint for the level is increased by 10% from 3.1 to 3.41, the actual steady-state value of L1 is closer to 3.5. The offset is due to the lack of integral action, since the levels are controlled by pure P-controllers. It can also be seen that the response of T3 is very slow. This is probably due to the large capacity of the system and the transport delay.

Figure 7 shows the dynamic responses to a disturbance in TP1I. The 5% increase in TP1I leads to a larger amount of heat transferred to the refrigerant, which causes more refrigerant to evaporate. This leads to the observed drop of the level in the first evaporator. The level controller reacts by opening XV2, which causes the level in the second evaporator to drop as well. L3 follows after a little delay. T1 rises due to the increased heat transfer. T2 follows quite quickly due to the pressure equilibrium that is established in the saturated refrigerant. T3 increases much more slowly due to the large capacity of the tank. It takes much more time to heat up the large hold-up of refrigerant in the receiver. The hold-up of liquid in the receiver is more than eight times larger than that of the second evaporator, so it makes sense that the temperature increases much more slowly. The increased pressure spreads almost immediately from P1 to P3 to P2, as is to be expected.

Overall, the dynamic responses seem to correspond well with what is expected from intuition and theory. It is therefore safe to assume that the implemented model gives a reasonable approximation of the actual process.

7.2 Steady-state optimization

As already discussed, it is expected that the setpoints for the levels do not have a steady-state effect based on literature. A brute-force evaluation of the cost function for different setpoints shows the same result. With the exception of when both setpoints were close to their lower boundaries, the cost function is approximately constant. The observed deviations for small values of L1 and L2 were most likely due to violations of the upper constraint for L3. The border between the stable region and the unstable region seems to follow a linear relationship, which strengthens this theory. The border line represents the case where the upper constraint for L3 is active. The linearity of the border follows directly from the mass balance.

At the nominal operating point it was found that both the upper temperature specifications for the process

outflows were active. This makes sense, as the objective was to minimize the energy consumption. Cooling the process streams more than necessary would be a give-away of energy and thus increase the cost. Since both heat loads can be specified independently, it is possible to have both active. In addition, it was observed that the condenser duty is maximised by letting the flow of air being at its upper constraint. This is to be expected since more cooling leads to a lower pressure and a lower temperature in the evaporators, allowing for more cooling. Since there is no cost associated with the condenser, this constraint will always be active when trying to minimize the energy consumption.

When no heat load was specified, it was observed that N went to its lower boundary. This is to be expected, since the optimizer tries to minimize the energy consumption by shutting of the compressor train. FCP3 also went to its lower boundary to avoid pressure drop which would cost unnecessary energy loss in the compressor train. P2 was at its upper boundary since the valve XV2 was trying to close completely to avoid bypass of gas. By closing XV2 it can be made sure that all the gas goes through both compressors, thus utilizing the energy as efficiently as possible.

Based on these two cases, it seems that the optimizer was set up properly and that the model gives reasonable results.

7.3 Alternative process layout

The alternative case was defined in order to free some degrees of freedom by combining the two heat load specifications to one. It was hoped that this would open up a degree of freedom being the distribution of liquid in the two tanks, i.e. how much cooling is done in each of the evaporators.

Problems arose when trying to find fitting parameters for the new model. The distribution of heat transfer area in the two evaporators caused trouble as it was not possible to distribute the old heat transfer areas equally between the two evaporators. A solution was found when U1A1 was set equal to approximately 30 W/K. However, the active constraints are very sensitive to the exact distribution of area, as can be seen in Figure 10. A total of 21 different constraint combinations were observed, though most of them only existed in a single point or in very small regions. It is therefore relatively safe to assume that the eight main regions are the correct ones, whereas the smaller ones were caused by numerical error or strange model behaviour on the border between two constraint regions.

All except for one of the main constraint regions have three active constraint. Region eight is the exception with only one active constraint. It is theoretically possible to implement self-optimizing control in this region, but this might not be the best overall solution. As can be seen from Figure 11, the total heat transfer is larger in region two and three. Whether or not it is better to stay in region eight depends on the sensitivity of the cost function there. It would be preferable to have a flat cost function curve at this point [8][5] In addition, it is desirable that changes in the inputs do not cause any constraints to become active, as that would make control very difficult as some sort of event detection would be needed to switch between different control structures. The desire for a flat optimum obviously goes for all constraint regions. The sensitivity has not yet been found, so it is difficult to say which of these eight regions would be the best choice for implementation.

It is worthwhile to notice that the attempt to gain additional degrees of freedom by connecting the two evaporators, failed. There are still three active constraints and zero unconstrained degrees of freedom in the system, with the exception of the previously mentioned region eight. It is hard to spot a reason for why the active regions are placed where they are, but it could seem as if the optimizer tries to maximise the heat transfer in the most effective evaporator first and then maximise the other one if possible. This means that the distribution of heat in the two evaporators does not give a degree of freedom as previously assumed. However, this is just a very early hypothesis that needs more investigation to be verified.

It should also be mentioned at this point that while the above discussion assumes that the distribution of heat transfer area can be chosen freely to optimize the operation, this is generally not possible. The

plant has predetermined equipment that needs to be operated as optimal as possible, even though the plant might run more efficiently if the process layout was changed. It is not the objective of the control engineer to change the layout of the plant, but rather to operate the existing plant to the best of its abilities. But the determination of the constraint regions is not completely useless from a control point of view. Once the design has been finalized and the plant has been built, the process can still go from one constraint region to another. Fouling inside the evaporator can lead to a decrease in the heat transfer coefficient, thus lowering the UA-value. If the plant is designed in such a way that it operates in region three, for example, it is very likely that it switches from region three to region two or four over time due to fouling. This knowledge will be valuable to the operators of the plant.

8 Conclusion

A model for a two-stage refrigeration cycle was implemented in MATLAB. Dynamic simulations were done to verify the model. The model was optimized at steady state and the nominal operating point was found. Degree of freedom analysis showed that not enough unconstrained degrees of freedom were left to implement self-optimizing control. An alternative case was created to get more degrees of freedom. Finding the optimal parameters for the new models is non-trivial, since a large amount of active constraint regions are present. One of the constraint regions only contains one active constraint, so it would be possible to implement a self-optimizing control structure in this region. This was not done due to time restrictions.

9 Suggestions for further work

- Look at the alternative case in depth. Determine what region is most realistic or interesting to study.
- Exxon suggested that one should let the differences TP1I-T1 and TP2I-T2 be constant instead of TP1I and TP2I. The cost function should be changed to include a trade-off between the energy consumption and the recovery of high-value molecules as indicated by the heat transfer in the evaporators.
- Dynamic optimization / MPC

References

[1] Vidar Alstad and Sigurd Skogestad. Null space method for selecting optimal measurement combinations as controlled variables. *Industrial & engineering chemistry research*, 46(3):846–853, 2007.

[2] Vidar Alstad, Sigurd Skogestad, and Eduardo S Hori. Optimal measurement combinations as controlled variables. *Journal of Process Control*, 19(1):138–148, 2009.

[3] Elvira Marie B Aske and Sigurd Skogestad. Consistent inventory control. *Industrial & engineering chemistry research*, 48(24):10892–10902, 2009.

[4] Basel Nashat Asmar. Control of a two-stage refrigeration cycle, 1991.

[5] James J. Downs and Sigurd Skogestad. An industrial and academic perspective on plantwide control. *Annual Reviews in Control*, 35(1):99 – 110, 2011.

[6] Ivar J. Halvorsen, Sigurd Skogestad, John C. Morud, and Vidar Alstad. Optimal selection of controlled variables†. *Industrial & Engineering Chemistry Research*, 42(14):3273–3284, 2003.

[7] Jørgen Bauck Jensen and Sigurd Skogestad. Steady-state operational degrees of freedom with application to refrigeration cycles. *Industrial & Engineering Chemistry Research*, 48(14):6652–6659, 2009.

[8] Truls Larsson and Sigurd Skogestad. Plantwide control-a review and a new design procedure. *Modeling, Identification and Control*, 21(4):209–240, 2000.

[9] William L. Luyben. Snowball effects in reactor/separator processes with recycle. *Industrial & Engineering Chemistry Research*, 33(2):299–305, 1994.

[10] Sigurd Skogestad. Control structure design for complete chemical plants. *Computers & Chemical Engineering*, 28(1–2):219 – 234, 2004. Escape 12.

Appendices

A Parameters used for the simulations

In the following tables, the coefficients for the model equations will be given. All values were taken from Asmar [4]

Table 3: Coefficients for the Antoine equation in Equation (13)

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

Table 4: Coefficients for calculating the heat capacity of propylene in Equation (37). Calculated heat capacity has units J/(kgK)

Variable	C_1	C_2	C_3	C_4	C_5	C_6
Value	3.707	0.01	23.439	0.001	-11.594	$2.2033 \cdot 10^{-3}$

Table 5: Coefficients for calculating the compressibility of propylene as a function of temperature in Equation (27)

Variable	ϕ_1 [-]	ϕ_2 [K^{-1}]
Low pressure	0.995	0
Intermediate pressure	0.566	$1.26 \cdot 10^{-3}$
High pressure	$2.97 \cdot 10^{-3}$	-0.194

Table 6: Coefficients for calculating the specific vapour enthalpy of propylene as a function of temperature in Equation (16)

Variable	ζ_1 [J/kg]	ζ_2 [J/kgK]
Low pressure	747.441	1.36908
Intermediate pressure	663.375	1.63599
High pressure	474.0188	2.09759

Table 7: Coefficients for calculating the specific liquid enthalpy of propylene as a function of temperature in Equation (17)

Variable	δ_1 [J/kg]	δ_2 [J/kgK]
Low pressure	7.16036	2.63768
Intermediate pressure	7.16036	2.63768
High pressure	95.5601	2.32367

Table 8: Coefficients for calculating the specific volume of propylene as a function of temperature in Equation (9)

Variable	λ_1 [J/kg]	λ_2 [J/kgK]
Low pressure	$8.05418 \cdot 10^{-4}$	$3.934 \cdot 10^{-6}$
Intermediate pressure	$5.91479 \cdot 10^{-4}$	$4.777 \cdot 10^{-6}$
High pressure	$7.493 \cdot 10^{-6}$	$-2.00589 \cdot 10^{-4}$

Table 9: Coefficients for the compressor curve for the first compressor in Equation (38)

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

Table 10: Coefficients for the compressor curve for the second compressor in Equation (39) and Equation (40)

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

Table 11: Coefficients for the compressor curves for both compressors in Equation (41)

Variable	e_1 [m^{-1}]	e_2 [-]	e_3 [m^{-1}]	e_4 [-]
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

B MATLAB-code

This section contains the MATLAB-code which was used to calculate all the results. Only the model for the first case is included, as the alternative case is almost identical with the exception of some minor changes to the parameter initialisation, a new cost function and a new model equation being the condition that the inlet to the first evaporator is the outlet from the second evaporator.

B.1 Model equations

DAE-system which is to be integrated.

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

% Description

%% Extracting states:
H1  = x( 1); % Diff.
H2  = x( 2); % Diff.
T3  = x( 3); % Diff.
W1  = x( 4); % Diff.
W2  = x( 5); % Diff.
W4  = x( 6); % Diff.

T1  = x( 7); % Alg.
T2  = x( 8); % Alg.
PA  = x( 9); % Alg.
TA  = x(10); % Alg.
TC  = x(11); % Alg.
TD  = x(12); % Alg.
HH1 = x(13); % Alg.
HH2 = x(14); % Alg.
WV1 = x(15); % Alg.
WV2 = x(16); % Alg.

XV2 = x(17); % Controller
XV3 = x(18); % Controller

L1dummy   = x(19); % Dummy
L2dummy   = x(20); % Dummy
L3dummy   = x(21); % Dummy
P1dummy   = x(22); % Dummy
P2dummy   = x(23); % Dummy
P3dummy   = x(24); % Dummy
TP1Odummy = x(25); % Dummy
TP2Odummy = x(26); % Dummy
TP3Odummy = x(27); % Dummy
FG1dummy  = x(28); % Dummy
FG2dummy  = x(29); % Dummy
FG3dummy  = x(30); % Dummy
FL2dummy  = x(31); % Dummy
FL3dummy  = x(32); % Dummy
FL4dummy  = x(33); % Dummy
TBdummy   = x(34); % Dummy

%% Extracting inputs and disturbances
```

```
45
46  L1sp = u(1);
47  L2sp = u(2);
48  N    = u(3);
49  XV1  = u(4);
50  FCP3 = u(5);
51
52  %% Defining parameters
53
54  if ~exist('params','var')
55      params = init_params();
56  end
57
58  W = params.W;
59  V1 = params.V1;
60  V2 = params.V2;
61  V3 = params.V3;
62  VC = params.VC;
63  C11 = params.C11;
64  C12 = params.C12;
65  C13 = params.C13;
66  C21 = params.C21;
67  C22 = params.C22;
68  C23 = params.C23;
69  C24 = params.C24;
70  C25 = params.C25;
71  E11 = params.E11;
72  E12 = params.E12;
73  E13 = params.E13;
74  E14 = params.E14;
75  E21 = params.E21;
76  E22 = params.E22;
77  E23 = params.E23;
78  E24 = params.E24;
79  U1A1 = params.U1A1;
80  U2A2 = params.U2A2;
81  U3A3 = params.U3A3;
82  CV1 = params.CV1;
83  CV2 = params.CV2;
84  CV3 = params.CV3;
85  FCP1 = params.FCP1;
86  FCP2 = params.FCP2;
87  TP1I = params.TP1I;
88  TP2I = params.TP2I;
89  TP3I = params.TP3I;
90  A = params.A;
91  B = params.B;
92  C = params.C;
93  R = params.R;
94  MW = params.MW;
95  G = params.G;
96  C1 = params.C1;
97  C2 = params.C2;
98  C3 = params.C3;
```

```
C4 = params.C4;
C5 = params.C5;
C6 = params.C6;

%% Algebraic equations

P1  = exp(A - B/(T1 - C));
Z1  = 0.955 + 0.0*T1;

HS1 = HH1/(N^2.19);
FG1S= (N^1.56)*(C11*HS1 - C12)/(HS1 - C13);
FG1 = FG1S*MW*P1*100/(T1*R*Z1);

% ----------------------

P2  = exp(A - B/(T2 - C));
FG2 = XV1*CV1*sqrt(P2-PA);

TB  = (TA*FG1 + T2*FG2)/(FG1+FG2);
Z2  = 0.566 + 1.26E-3*TB;

% ----------------------

P3  = exp(A - B/(TD - C));
Z3 = -0.194 + 2.97E-3*TD;

HS2 = HH2/(N^2.11);
TX = C21/(tan((C22-HS2)/C23));
FG3S = (N^1.79)*(C24 + C25*log(sqrt(TX^2+1)-TX));
FG3 = FG3S*MW*PA*100/(TB*R*Z2);

% ----------------------
% Enthalpies
HFG1 = 747.441 + 1.36908*T1;
HFG2 = 663.375 + 1.63599*T2;
HFG3 = 474.018 + 2.09759*TC;

HFL2 = 7.16036 + 2.63768*T2;
HFL3 = 95.5601 + 2.32367*T3;
HFL4 = 95.5601 + 2.32367*TD;

HW1V = 747.441 + 1.36908*T1;
HW1L = 7.16036 + 2.63768*T1;

HW2V = 663.375 + 1.63599*T2;
HW2L = 7.16036 + 2.63768*T2;

% ----------------------

A1 = exp(-U1A1/FCP1);
TP1O = (1-A1)*T1 + A1*TP1I;
Q1 = FCP1*(TP1I - TP1O)

A2 = exp(-U2A2/FCP2);
```

```
153 TP2O = (1-A2)*T2 + A2*TP2I;
154 Q2 = FCP2*(TP2I - TP2O)
155
156 A3 = exp(-U3A3/FCP3);
157 TP3O = (1-A3)*TD + A3*TP3I;
158 QC = FCP3*(TP3I - TP3O);
159
160 % ----------------------
161
162 FL2 = XV2*CV2*sqrt(P2-P1);
163 FL3 = XV3*CV3*sqrt(P3-P2);
164 FL4 = -QC/(HFG3-HFL4);
165
166 ETA1 = E11*HS1 + E12 - 10^(E13*HS1 - E14);
167
168 CPI1 = C1 + C2*T1*(C3 + C4*T1*(C5 + C6*T1));
169 CPO1 = C1 + C2*TA*(C3 + C4*TA*(C5 + C6*TA));
170
171 G1 = 0.5*(CPI1/(CPI1-R) + CPO1/(CPO1-R));
172 POL1 = ETA1*G1/(G1-1);
173
174 ETA2 = E21*HS2 + E22 - 10^(E23*HS2 - E24);
175
176 CPI2 = C1 + C2*TB*(C3 + C4*TB*(C5 + C6*TB));
177 CPO2 = C1 + C2*TC*(C3 + C4*TC*(C5 + C6*TC));
178
179 G2 = 0.5*(CPI2/(CPI2-R) + CPO2/(CPO2-R));
180 POL2 = ETA2*G2/(G2-1);
181
182 WL1 = W1 - WV1;
183 WL2 = W2 - WV2;
184 W3  = W - W1 - W2 - W4;
185
186 HL1 = WL1*HW1L;
187 HV1 = WV1*HW1V;
188
189 HL2 = WL2*HW2L;
190 HV2 = WV2*HW2V;
191
192 VF1 = 8.05418E-4 + 3.934E-6*T1;
193 VF2 = 5.91479E-4 + 4.777E-6*T2;
194 VF3 = -2.00589E-4 + 7.493E-6*TD;
195
196 V1G = V1 - WL1*VF1;
197 V2G = V2 - WL2*VF2;
198 V4G = V3 + VC - W3*VF3;
199
200 VG1 = V1G/WV1;
201 VG2 = V2G/WV2;
202 VG4 = V4G/W4;
203
204 L1 = WL1*VF1;
205 L2 = WL2*VF2;
206 L3 = W3*VF3;
```

```
207
208 PTH1 = 0.5*FG1*(CPO1+CPI1)*(TA-T1)*1000/MW;
209 PTH2 = 0.5*FG3*(CPO2+CPI2)*(TC-TB)*1000/MW;
210 PTHTOT = PTH1 + PTH2;
211
212 %% Defining the derivatives
213
214 dH1 = HFL2*FL2 - HFG1*FG1 + Q1;
215 dH2 = HFL3*FL3 - HFL2*FL2 - HFG2*FG2 + Q2;
216 dT3 = FL4*(TD - T3)/W3;
217 dW1 = FL2 - FG1;
218 dW2 = FL3 - FL2 - FG2;
219 dW4 = FG3 - FL4;
220
221 dx = [dH1; dH2; dT3; dW1; dW2; dW4];
222
223 %% Defining the residuals
224
225 % Algebraic
226 T1resid = H1 - HL1 - HV1;
227 T2resid = H2 - HL2 - HV2;
228 PAresid = FG3 - FG1 - FG2;
229 TAresid = TA - T1*((PA/P1)^(1/POL1));
230 TCresid = TC - TB*((P3/PA)^(1/POL2));
231 TDresid = P3 - Z3*R*TD/(MW*VG4*100);
232 HH1resid = HH1 - POL1*(R*1000/(G*MW))*(TA - T1);
233 HH2resid = HH2 - POL2*(R*1000/(G*MW))*(TC - TB);
234 WV1resid = P1 - Z1*R*T1/(MW*VG1*100);
235 WV2resid = P2 - Z2*R*T2/(MW*VG2*100);
236
237 % Controllers
238 KcL1C = 0.5;
239 L1Cresid = (0.2441 - XV2) + (L1sp - L1)*KcL1C;
240
241 KcL2C = 0.6;
242 L2Cresid = (0.2961 - XV3) + (L2sp - L2)*KcL2C;
243
244 resids = [T1resid; T2resid; PAresid; TAresid; TCresid; TDresid;...
245          HH1resid; HH2resid; WV1resid; WV2resid; L1Cresid; L2Cresid;...
246          ];
247
248 % Dummies
249 L1resid = L1-L1dummy;
250 L2resid = L2-L2dummy;
251 L3resid = L3-L3dummy;
252
253 P1resid = P1-P1dummy;
254 P2resid = P2-P2dummy;
255 P3resid = P3-P3dummy;
256
257 TP1Oresid = TP1O-TP1Odummy;
258 TP2Oresid = TP2O-TP2Odummy;
259 TP3Oresid = TP3O-TP3Odummy;
260
```

```
FG1resid = FG1-FG1dummy;
FG2resid = FG2-FG2dummy;
FG3resid = FG3-FG3dummy;

FL2resid = FL2-FL2dummy;
FL3resid = FL3-FL3dummy;
FL4resid = FL4-FL4dummy;

TBresid  = TB-TBdummy;

resids = [resids; L1resid; L2resid; L3resid; P1resid; P2resid; P3resid; ...
          TP1Oresid; TP2Oresid; TP3Oresid; FG1resid; FG2resid; FG3resid;...
          FL2resid; FL3resid; FL4resid; TBresid];

%% Output

out = [dx; resids];

end
```

Initialisation of the parameters

Creates struct of parameters.

```
function params = init_params()

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

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

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

% Coefficients for curve fittings for second compressor performance
params.C21 = 0.45615;
params.C22 = 10296;
params.C23 = 2956.2;
params.C24 = 0.816051;
params.C25 = 0.27585;

% Coefficients for curve fittings for first compressor isentropic efficiency
params.E11 = 6.47E-5;
params.E12 = 0.353;
params.E13 = 7.101E-4;
params.E14 = 6.5963;

% Coefficients for curve fittings for second compressor isentropic efficiency
params.E21 = 4.098E-5;
```

```
params.E22 = 0.364;
params.E23 = 1.121E-3;
params.E24 = 12.445;

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

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

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

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

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

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

% Specific heat capacity equation coefficients for propylene
params.C1 = 3.707;
params.C2 = 0.01;
params.C3 = 23.439;
params.C4 = 0.001;
params.C5 = -11.594;
params.C6 = 2.2033E-3;

end
```

B.2 Integrator

Integrates the model using ode15s. First call to fsolve is to create good initial values for the integrator, but this step seems unnecessary.

```
function [t,x] = integrator()
% Description

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Calculating the steady state with the given inputs
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
7
8  % Initial estimates for the states
9
10 % Differential states
11 W1ic = 1.8790e+03;
12 W2ic = 509.5381;
13 T3ic = 310.3064;
14 H1ic = 1.1346e+06;
15 H2ic = 3.6506e+05;
16 W4ic = 524.1002;
17
18 x0_diff = [H1ic; H2ic; T3ic; W1ic; W2ic; W4ic]; % Diff.
19
20 % Algebraic states
21 T1ic  = 222.8800;
22 T2ic  = 262.5103;
23 PAic  = 3.8933;
24 TAic  = 295.0822;
25 TCic  = 358.1332;
26 TDic  = 310.3064;
27 HH1ic = 7.5895e+03;
28 HH2ic = 8.7647e+03;
29 WV1ic = 36.1681;
30 WV2ic = 21.8629;
31
32 XV2ic = 0.7000;
33 XV3ic = 0.7096;
34
35 x0_alg = [T1ic; T2ic; PAic; TAic; TCic; ...
36     TDic; HH1ic; HH2ic; WV1ic; WV2ic; XV2ic; XV3ic]; % Alg.
37
38 % Additional (non-state) variables
39 L1ic     = 3.1;
40 L2ic     = 0.9;
41 L3ic     = 7.6216;
42
43 P1ic     = 0.9;
44 P2ic     = 4.2;
45 P3ic     = 15.2025;
46
47 TP1Oic   = 226.2000;
48 TP2Oic   = 276.7510;
49 TP3Oic   = 308.1249;
50
51 FG1ic    = 2.8400;
52 FG2ic    = 1.5245;
53 FG3ic    = 4.3646;
54
55 FL2ic    = 2.8400;
56 FL3ic    = 4.3646;
57 FL4ic    = 4.3646;
58
59 TBic     = 283.7050;
60
```

```
x0_add = [L1ic; L2ic; L3ic; P1ic; P2ic; P3ic; TP1Oic; TP2Oic; TP3Oic; ...
    FG1ic; FG2ic; FG3ic; FL2ic; FL3ic; FL4ic; TBic];

x0 = [x0_diff; x0_alg; x0_add]; % Combining diff, alg and additional

% Initial conditions for the inputs
L1spic = 4.0118;
L2spic = 1.5892;
Nic    = 0.9912;
XV1ic  = 0.8097;
FCP3ic = 348.00;

u0 = [L1spic; L2spic; Nic; XV1ic; FCP3ic];

% Calculating the steady state
g = @(x) model([],x,u0);

x_SS = fsolve(g,x0);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Calculating responses to disturbances / setpoint changes
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Doing a step change in L1 setpoint
u = u0;
%u(1) = u(1)*1.1;

% Parametrizing the function to pass extra parameters to it (u and params)
params = init_params;
%params.TP1I = params.TP1I*1.05;
g = @(t,x) model(t,x,u,params);

% Defining the constant, singular mass matrix M
M = diag([ones(1,length(x0_diff)),zeros(1,length(x0_alg)+length(x0_add))]);

J = @(t,x) jac(t,x,u);

% Setting the options for the ODEsolver
%options = odeset('Jacobian',J,'Mass',M,'MStateDependence','none',...
%    'MassSingular','yes','RelTol',1e-4,'Vectorized','off');

 options = odeset('Mass',M,'MStateDependence','none',...
     'MassSingular','yes','RelTol',1e-4,'Vectorized','off');

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

% Calculating the corresponding outputs from the states
L1 = x(:,19);
L2 = x(:,20);
L3 = x(:,21);
P1 = x(:,22);
P2 = x(:,23);
P3 = x(:,24);
```

```
T1 = x(:,7);
T2 = x(:,8);
T3 = x(:,3);
TP1O = x(:,25);
TP2O = x(:,26);
TP3O = x(:,27);
FG2 = x(:,29);
FL2 = x(:,31);
FL3 = x(:,32);

% Plotting the results
figure()

hold on
subplot(5,3,1); plot(t,L1)
xlabel('t'); ylabel('L1')
subplot(5,3,2); plot(t,L2)
xlabel('t'); ylabel('L2')
subplot(5,3,3); plot(t,L3)
xlabel('t'); ylabel('L3')
subplot(5,3,4); plot(t,P1)
xlabel('t'); ylabel('P1')
subplot(5,3,5); plot(t,P2)
xlabel('t'); ylabel('P2')
subplot(5,3,6); plot(t,P3)
xlabel('t'); ylabel('P3')
subplot(5,3,7); plot(t,T1)
xlabel('t'); ylabel('T1')
subplot(5,3,8); plot(t,T2)
xlabel('t'); ylabel('T2')
subplot(5,3,9); plot(t,T3)
xlabel('t'); ylabel('T3')
subplot(5,3,10); plot(t,FG2)
xlabel('t'); ylabel('FG2')
subplot(5,3,11); plot(t,FL2)
xlabel('t'); ylabel('FL2')
subplot(5,3,12); plot(t,FL3)
xlabel('t'); ylabel('FL3')
subplot(5,3,13); plot(t,TP1O)
xlabel('t'); ylabel('TP1O')
subplot(5,3,14); plot(t,TP2O)
xlabel('t'); ylabel('TP2O')
subplot(5,3,15); plot(t,TP3O)
xlabel('t'); ylabel('TP3O')
hold off

end
```

B.3 Steady-state optimization

Optimizing the model using fmincon.

```
function [y,fval,exitflag] = opt(u0,x0,params)

if (~exist('params','var'))
```

```
4       params = init_params();
5   end
6
7   if (~exist('u0','var')) || isempty(u0)
8       % Inputs
9       L1spic = 4.0118;
10      L2spic = 1.5892;
11      Nic    = 0.9912;
12      XV1ic  = 0.8097;
13      FCP3ic = 348.00;
14      u0 = [L1spic; L2spic; Nic; XV1ic; FCP3ic];
15  end
16  if (~exist('x0','var')) || isempty(x0)
17      % Differential states
18      H1ic = 1.1346e+06;
19      H2ic = 3.6506e+05;
20      W1ic = 1.8790e+03;
21      W2ic = 509.5381;
22      T3ic = 310.3064;
23      W4ic = 524.1002;
24      x0_diff = [H1ic; H2ic; T3ic; W1ic; W2ic; W4ic];          % Differential
25
26      % Algebraic states
27      T1ic  = 222.8800;
28      T2ic  = 262.5103;
29      PAic  = 3.8933;
30      TAic  = 295.0822;
31      TCic  = 358.1332;
32      TDic  = 310.3064;
33      HH1ic = 7.5895e+03;
34      HH2ic = 8.7647e+03;
35      WV1ic = 36.1681;
36      WV2ic = 21.8629;
37      XV2ic = 0.7000;
38      XV3ic = 0.7096;
39      x0_alg = [T1ic; T2ic; PAic; TAic; TCic; ...
40                TDic; HH1ic; HH2ic; WV1ic; WV2ic; XV2ic; XV3ic]; % Algebraic
41
42      % Additional (non-state) variables
43      L1ic     = 3.1;
44      L2ic     = 0.9;
45      L3ic     = 7.6216;
46      P1ic     = 0.9;
47      P2ic     = 4.2;
48      P3ic     = 15.2025;
49      TP1Oic   = 226.2000;
50      TP2Oic   = 276.7510;
51      TP3Oic   = 308.1249;
52      FG1ic    = 2.8400;
53      FG2ic    = 1.5245;
54      FG3ic    = 4.3646;
55      FL2ic    = 2.8400;
56      FL3ic    = 4.3646;
57      FL4ic    = 4.3646;
```

```
    TBic     = 283.7050;
    x0_add = [L1ic; L2ic; L3ic; P1ic; P2ic; P3ic; TP1Oic; TP2Oic; ...
      TP3Oic; FG1ic; FG2ic; FG3ic; FL2ic; FL3ic; FL4ic; TBic]; % Additional

    x0 = [x0_diff; x0_alg; x0_add];    % Combining diff, alg and additional
end

g = @(x) model([],x,u0,params);

x0 = fsolve(g,x0); % Get initial guess

y0 = [u0; x0]; % Combining inputs and states

%% Constraints
lb=zeros(length(y0),1);
lb(3)  = 0.0; % N
lb(4)  = 0; % XV1
lb(5)  = 116; % FCP3
lb(22) = 0; % XV2
lb(23) = 0; % XV3
lb(24)  = 2.9; % L1
lb(25)  = 0.6; % L2
lb(26) = 6.9; % L3
lb(27) = 0; % P1
lb(28) = 3; % P2
lb(29) = 12; % P3
lb(30) = 200; % TP1O
lb(34) = 0; % FG2
lb(36) = 0; % FL2
lb(37) = 0; % FL3

ub=ones(length(y0),1)*1e12;
ub(3)  = 1.1; % N
ub(4)  = 1; % XV1
ub(5)  = 348; % FCP3
ub(22) = 1; % XV2
ub(23) = 1; % XV3
ub(24)  = 6.4; % L1
ub(25)  = 1.6; % L2
ub(26) = 13.7; % L3
ub(27) = 2; % P1
ub(28) = 6; % P2
ub(29) = 18; % P3
ub(30) = 300; % TP1O
ub(34) = 6.31; % FG2
ub(36) = 5.47; % FL2
ub(37) = 7.18; % FL3

%% Optimization

% fmincon options
alg = 'Interior-Point';
%alg = 'sqp';
options = optimset('TolFun',10e-7,'TolCon',10e-7,'MaxFunEvals',1e4,...
```

```
'MaxIter',1e4,'Display','none','Algorithm',alg,'Diagnostics','off'...
);

% Call fmincon to optimize the model
nlconpar = @(y) nlcon(y,params);
[y,fval,exitflag] = fmincon(@cost,y0,[],[],[],[],lb,ub,nlconpar,options);

name = {'L1sp','L2sp','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'...
        };

if exitflag ~= 1
    fprintf('Exitflag not equal to 1! \n\n')
end

upper_limit = abs(ub./y)-1;
lower_limit = 1-abs(lb./y);
err_tol = 1e-7;
for i = 1:length(y0)
    if upper_limit(i) < err_tol
        fprintf('Active constraint: %s at upper limit \n',name{i})
    elseif lower_limit(i) < err_tol
        fprintf('Active constraint: %s at lower limit \n',name{i})
    end
end

function [j] = cost(y)

%% Objective function

% Unpacking the parameters
T1  = y(12);
TA  = y(15);
TB  = y(39);
TC  = y(16);
FG1 = y(33);
FG3 = y(35);

%% Calculating the cost

% General constants
MW = 42.081;

% Specific heat capacity equation coefficients for propylene
C1 = 3.707;
C2 = 0.01;
C3 = 23.439;
C4 = 0.001;
C5 = -11.594;
C6 = 2.2033E-3;

CPI1 = C1 + C2*T1*(C3 + C4*T1*(C5 + C6*T1));
```

```
CPO1 = C1 + C2*TA*(C3 + C4*TA*(C5 + C6*TA));

CPI2 = C1 + C2*TB*(C3 + C4*TB*(C5 + C6*TB));
CPO2 = C1 + C2*TC*(C3 + C4*TC*(C5 + C6*TC));

% Energy consumption
PTH1 = 0.5*FG1*(CPO1+CPI1)*(TA-T1)*1000/MW;
PTH2 = 0.5*FG3*(CPO2+CPI2)*(TC-TB)*1000/MW;
PTHTOT = PTH1 + PTH2;

j = PTHTOT;

function [c ,ceq ] = nlcon(y,params)

% Inputs
u = y(1:5);
% State variables
x = y(6:end);

% Nonlinear inequality constraints C(x)<0
c=[x(25)-226.2,...
   x(26)-276.751];

% Nonlinear equality constraints C(x)=0
ceq=[model([],x,u,params);...
    y(24)-3.1 ; ...
    y(25)-0.9 ; ...
    ];

```

B.4 Finding active constraint regions

```
A1int = linspace(5,55,101);
A2int = linspace(5,55,101);

map1 = zeros(length(A1int),length(A2int));
map2 = zeros(length(A1int),length(A2int));
map3 = zeros(length(A1int),length(A2int));

for i = 1:length(A1int)
    for j = 1:length(A2int)
        try
            p = init_params;
            p.U1A1 = A1int(i);
            p.U2A2 = A2int(j);
            [y,fval,exit,cvu,cvd] = opt(u0,x0,p);
            if exit > 0
                map1(i,j) = cvu;
                map2(i,j) = cvd;
                map3(i,j) = fval;
            else
                map1(i,j) = exit;
                map2(i,j) = exit;
            end
```

```
        catch err
            if strcmp(err.identifier,'optim:barrier:UsrNonlConstrUndefAtX0')
                map1(i,j) = -10;
                map2(i,j) = -10;
            else
                rethrow(err)
            end
        end
    end
end
combimap = (map1+map2*1i);
for i = 1:101
    for j = 1:101
        Z(i,j) = getflag(combimap(i,j));
    end
end
X = A1int;
Y = A2int;
h = pcolor(Y,X,Z);
set(h, 'EdgeColor', 'none');
```

```
function flag = getflag(z)
        switch z
            case -10 - 10i;
                    flag = -1;
            case 0;
                flag = -1;
            case -2 - 2i
                flag = -1;
            case 0 + 32i
                flag = 1;
            case 56
                flag = 2;
            case 4194304
                flag = 3;
            case 4194312
                flag = 4;
            case 4194320
                flag = 5;
            case 4194344
                flag = 6;
            case 4194352
                flag = 7;
            case 8388632
                flag = 8;
            case 8388664
                flag = 9;
            case 12582920
                flag = 10;
            case 12582928
                flag = 11;
            case 12582952
                flag = 12;
            case 12582960
```

```
34                  flag = 13;
35              case 4194304 + 16777216i
36                  flag = 14;
37              case 0 + 67108864i
38                  flag = 15;
39              case 16 + 67108864i
40                  flag = 16;
41              case 536870912
42                  flag = 17;
43              case 536870912 + 67108864i
44                  flag = 18;
45              case 545259536
46                  flag = 19;
47              case 545259544
48                  flag = 20;
49              case 549453840
50                  flag = 21;
51              otherwise
52                  fprintf('unknown z: %14.14f, %14.14f\n',real(z),imag(z))
53                  flag = 0;
54          end
55      end
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
```