NTNU – Trondheim
Norwegian University of
Science and Technology

TKP4140 Process Control

Project D: 2nd order reaction in a CSTR

Part 1, 2 & 3

Anne Kristin Smeby
Erlend Sørlie
Sunniva Gravdahl

November 18, 2022

Contents

List of Figures

List of Tables

1 Part 1: Modeling and Simulation

This part is the first of three parts of a project within the course *TKP4140 - Process Control*. In this part of the report, the modelling and simulation of a continuously stirred tank reactor (CSTR) will be carried out.

1.1 Project description

The process depicts a CSTR with one inlet stream, q_1, and one outlet stream, q_2. All the assumptions for the reactor is described in section 1.2.1. The feed, q_1, consists of pure component A. In the reactor a second order reactor takes place, see equation (1), consequently the product stream, q_2, consist of A and B. The reaction rate of this reaction is given in equation (2).

$$A \rightarrow B \tag{1}$$

$$r = k \cdot c_A^2, \quad [\text{kmol/m}^3 \cdot \text{min}] \tag{2}$$

where k is the rate constant, and c_A is the concentration of component A in the tank. The inlet stream, q_1, and the outlet stream, q_2, are manipulated in order to control the liquid height in the tank, h, and c_A. The inlet concentration, c_{Af}, and the rate constant k cannot be controlled, and are disturbances in the process. The temperature in the tank, T, is assumed to be constant by the use of a perfect temperature control controlling the heat supplied to the system, Q. A sketch of the system is shown in Figure 1.

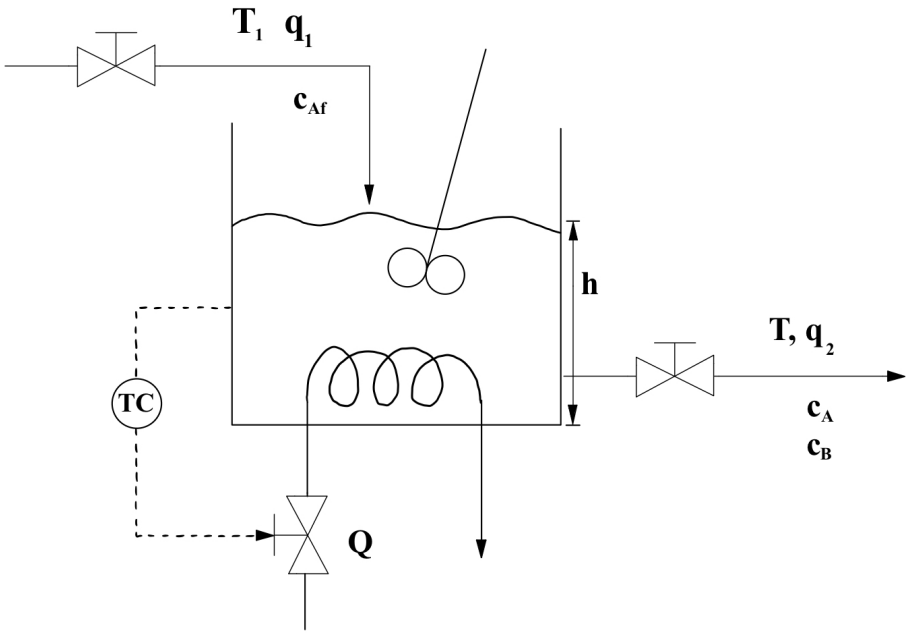

Figure 1: Sketch of the CSTR modelled in the report

1.1.1 Classification of variables

The variables in the process can be divided into states (x), inputs (u) ,outputs (y) and disturbances(d). The list below presents the classification for the system.

- states (x): h, c_A
- inputs (u): q_1, q_2
- outputs (measurements) (y): h, c_A
- disturbances (d): c_{Af}, k

The inputs are the MVs that will be manipulated to control the CVs, which are the outputs.

1.2 Nonlinear model

The nonlinear model is derived to describe the behaviours of the system. All assumptions, the model and steady state data are utilized in the derivation of the nonlinear model.

1.2.1 Assumptions

- T is constant (perfect control using Q is assumed).
- Density (ρ) is assumed constant and equal for all streams.
- Cascade is assumed on the controls, such that q_1 and q_2 are controlled directly.
- Perfect mixing in the CSTR is assumed. $c_{A,out} = c_A$
- The cross sectional area of the tank, A, is constant.

1.2.2 Model

To describe the process, a mass balance and a component balance will be necessary. The mass balance for the system can be written as:

$$\frac{\mathrm{d}m}{\mathrm{d}t} = \dot{m}_{\mathrm{in}} - \dot{m}_{\mathrm{out}}$$

where $\frac{\mathrm{d}m}{\mathrm{d}t}$ is the rate of mass accumulation in the tank and $\dot{m}_{\mathrm{in}}$ and $\dot{m}_{\mathrm{out}}$ are the mass flows in and out of the system. Using the fact that $m = \rho V = \rho A h$, this can be rewritten as follows:

$$\frac{\mathrm{d}\left(\rho A h\right)}{\mathrm{d}t} = \rho\left(q_1 - q_2\right)$$

Since ρ and A are assumed constant, they can be moved outside of the derivative.

$$\rho A \frac{\mathrm{d}h}{\mathrm{d}t} = \rho(q_1 - q_2)$$

Resulting in the mass balance for the system, given in the following equation:

$$\frac{\mathrm{d}h}{\mathrm{d}t} = \frac{q_1 - q_2}{A} \tag{3}$$

The component balance can be written as

$$\frac{\mathrm{d}n_A}{\mathrm{d}t} = F_{Af} - F_A + G_A$$

where $\frac{\mathrm{d}n_A}{\mathrm{d}t}$ is the rate of accumulation of A in the system, F_{Af} and F_A are the molar flows in and out of the tank, and G_A is the amount of A generated or consumed by the reaction. G_A can be expressed as:

$$G_A = r_A \cdot V$$

As A is consumed in the reaction, $r_A = -r = -k \cdot c_A^2$, then G_A is given by:

$$G_A = -k \cdot c_A^2 \cdot V$$

Inserting into the component balance, the resulting equation is

$$\frac{\mathrm{d}n_A}{\mathrm{d}t} = F_{Af} - F_A - k \cdot c_A^2 \cdot V$$

Utilizing that $n_A = c_A \cdot V$, $F_{Ai} = c_{Ai} \cdot q_i$ and $V = Ah$:

$$\frac{\mathrm{d}\left(c_A \cdot Ah\right)}{\mathrm{d}t} = q_1 c_{Af} - q_2 c_A - k \cdot c_A^2 \cdot Ah$$

Expanding the derivative, and using that A is constant, the equation can be rewritten

$$Ah\frac{\mathrm{d}c_A}{\mathrm{d}t} + c_A \cdot A\frac{\mathrm{d}h}{\mathrm{d}t} = q_1 c_{Af} - q_2 c_A - k \cdot c_A^2 \cdot Ah$$

Inserting the derived model equation for the liquid height, eq. (3):

$$Ah\frac{\mathrm{d}c_A}{\mathrm{d}t} + c_A\left(q_1 - q_2\right) = q_1 c_{Af} - q_2 c_A - k \cdot c_A^2 \cdot Ah$$

This is simplified to the model equation for c_A

$$\frac{\mathrm{d}c_A}{\mathrm{d}t} = \left(c_{Af} - c_A\right)\frac{q_1}{Ah} - k \cdot c_A^2 \tag{4}$$

1.2.3 Steady state calculations

The steady state values q_2^* and c_A^* are not known, and need to be calculated. At steady state, there are no changes. Applying steady state to eq. (3), gives

$$\frac{q_1^* - q_2^*}{A} = 0$$

Resulting in

$$q_1^* = q_2^* = 1\,\mathrm{kg\,s^{-1}} \tag{5}$$

Similarly, applying steady state to the model equation for c_A, eq. (4) can then be reformulated into a quadratic equation:

$$k^* {c_A^*}^2 + \frac{q_1^*}{V^*}c_A^* - \frac{q_1^*}{V^*}c_{Af} = 0$$

Using the quadratic formula to solve the expression, c_A^* can be calculated from the resulting equation

$$c_A^* = \frac{-\frac{q_1^*}{V^*} \pm \sqrt{\left(\frac{q_1^*}{V^*}\right)^2 - 4k^*\left(-\frac{q_1^*}{V^*}c_{Af}\right)}}{2k^*} \tag{6}$$

Inserting the provided steady state nominal values shown in Table 1, and ignoring the obviously impossible negaive c_A, it is found that

$$c_A^* = 0.05\,\mathrm{kmol\,m^{-3}}$$

The final missing steady sate nominal value is the liquid height h^*, which can be found using the volume and the area of the tank:

$$h^* = \frac{V^*}{A} = \frac{4\,\mathrm{m^3}}{4\,\mathrm{m^2}} = 1\,\mathrm{m} \tag{7}$$

1.2.4 Steady state data

The steady state values used for the calculations of this process are presented in Table 1. These values were given in the process description, or calculated in section 1.2.3. The * in the table indicates that the value is the steady state nominal value.

Table 1: Steady state nominal values, and constants given in the assignment.[1] q_2^*, c_A^* and h^* was calculated in section 1.2.3.

Variable	Description	Steady state value	Units
V^*	Volume	4	$\mathrm{m^3}$
q_1^*	Volumetric inlet flow	1	$\mathrm{m^3\,min^{-1}}$
c_{Af}^*	Inlet concentration of A	1	$\mathrm{kmol\,m^{-3}}$
k^*	Reaction constant	95	$\mathrm{m^3\,kmol^{-1}\,min^{-1}}$
q_2^*	Volumetric outlet flow	1	$\mathrm{m^3\,min^{-1}}$
c_A^*	Concentration of A in the tank	0.05	$\mathrm{kmol\,m^{-3}}$
h^*	Liquid height	1	m
A	Area	4	$\mathrm{m^2}$
ρ	Density	1000	$\mathrm{kg\,m^{-3}}$

1.3 Simulation results

This system consists of two input variables, q_1 and q_2, and two disturbances, c_{Af} and k. In order to study the system, a simulation was performed in MATLAB and Simulink, where a 10% step increase was performed in the input variable or disturbac, while all other inputs or disturbances was kept constant. The result of the step increases is shown in Figure 2. Where each column is a step increase in a different input variable or disturbance. The uppermost row shows what variable was increased, and the resulting graph, the middle row shows the response of the liquid height, h, and the bottom row shows the response of the concentration of A in the tank, c_A.

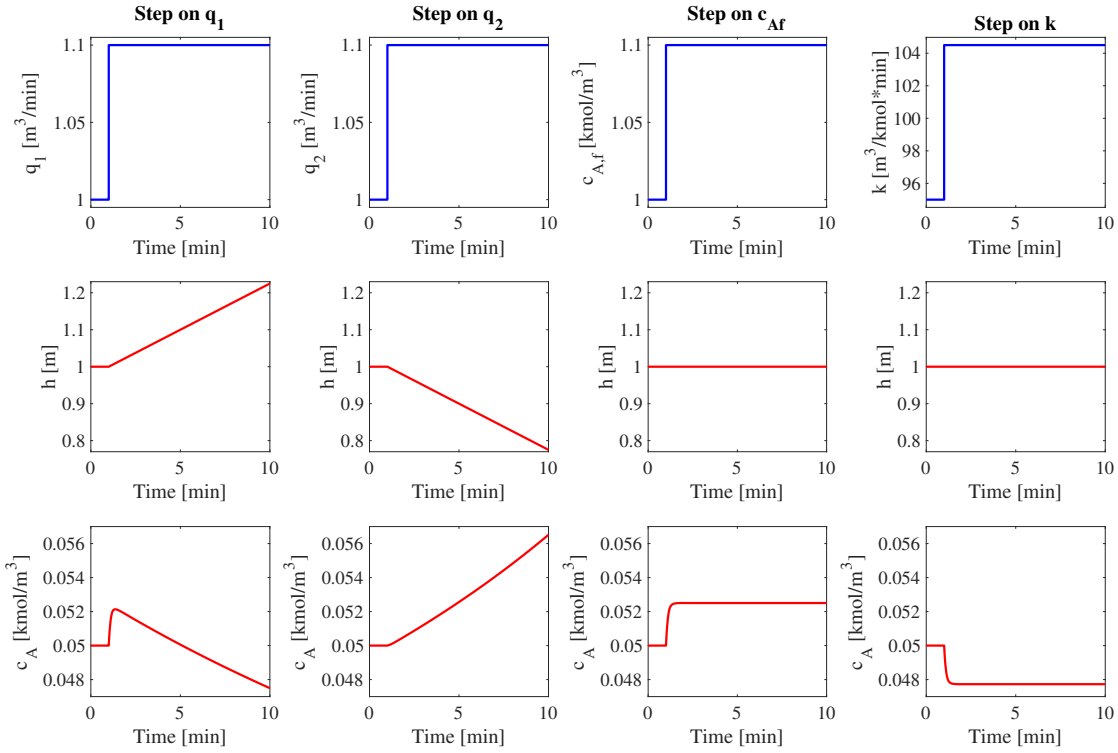

Figure 2: Simulation result for a 10% step increase in a single input variable or disturbances, with the resulting response for the output variables h and c_A.

1.4 Steady state gains

The gains for the steady state or intgrating processes were calculated for each step response of a corresponding input or disturbance. For systems that reached a steady state, the steady state gain was calculated using the following equation,

$$k = \frac{\Delta y}{\Delta u} \tag{8}$$

while for the integrating processes, the gain, k', was found using:

$$k' = \frac{\text{slope}}{\Delta u} \tag{9}$$

where the slope was found using

$$\text{slope} = \frac{\Delta y}{\Delta t}$$

on the last 5 datapoints of the simulation. The results from the calculations are presented in Table 2.

Table 2: The steady state gain or slope of the responses of the output variables after a 10% increase in an input variable or a disturbance.

Step change in	Output	Type	Value	Unit
q_1	h	Integrating	0.2500	m^{-2}
q_1	c_A	Integrating	-0.0048	$\mathrm{kmol/m}^6$
q_2	h	Integrating	-0.2500	m^{-2}
q_2	c_A	Integrating	0.0087	$\mathrm{kmol/m}^6$
c_{Af}	h	Steady state	0	$\mathrm{m}^4/\mathrm{kmol}$
c_{Af}	c_A	Steady state	0.0250	-
k	h	Steady state	0	$\mathrm{kmol} \cdot \mathrm{min/m}^2$
k	c_A	Steady state	$-2.39 \cdot 10^{-4}$	$\mathrm{kmol}^2 \cdot \mathrm{min/m}^6$

1.5 Discussion

As can be seen from Figure 2, a 10% step increase was done. The graphs on the leftmost part of Figure 2 shows a step increase on q_1. As can be seen, these step increases promoted changes in both the height, h, and the outlet concentration, c_A. It is observed that h increases with q_1. This is because the volume in the tank will increase when the mass flow is increased, but the outflow is kept constant, as can also be seen from eq. (3). The outlet concentration increased a bit, but then decreased with an increase in q_1. The first peak in concentration has relation to more A being introduced ($c_{Af} > c_A$). However, the outlet concentration then decreased as h increased. This might be due to the increasing volume, creating more time for A to be reacted in the tank, and thereby decreasing c_A.

From the second row of graphs, a step change on q_2 can be seen. Here, an opposite effect on h, is observed, which makes sense considering the volume will decrease when the outflow is increased. An increase in c_A can also be seen. This is probably due to reactant A having less time to react in the tank when the outflow is increased.

The third row of figures show a step on the concentration of A in the feed. These results show that increasing c_{Af} has no effect on the liquid level in the tank (h), as the in- and outflows are unchanged. However, the concentration of A in the tank will increase instantaneously, as more of the reactant has been added. It can, however, be seen that the concentration of A in the tank quickly stabilizes at a higher level.

Lastly, a step on the reaction constant, k, was performed, shown by the rightmost graphs in Figure 2. As can be observed, a change in k has no effect on the liquid level, h. This is understandable, as there is no change in q_1 or q_2. However, it can be seen from the figure that the step change leads to a drop in c_A. This is because the increased reaction constant for the reaction will result in more of reactant A consumed, leading to a lower concentration of A in the tank. This means that c_A will stabilize on a lower level than before the step change was performed. This is also consistent with what could be expected from eq. (4).

In conclusion, the responses shown in Figure 2 are explainable and agrees with the assumptions which were made for this system. Therefore, the simulation of the process seems reasonable.

2 Part 2: System Analysis

In this part of the project, a linear model approximation for the system responses will be analyzed. Furthermore, a simulation of the linearized model will be performed and compared with the nonlinear responses.

2.1 Linearization

In order to linearize the nonlinear model, we adapt it to fit the standard state-space form as shown in the following equations:

$$\dot{x} = Ax + Bu + Ed \tag{10a}$$
$$y = Cx + Du + Wd \tag{10b}$$

Where A, B, C, D, E and W are matrices. The vectors for the states, x, inputs, u, disturbances d and outputs, y are defined as:

$$x = [\Delta h \quad \Delta c_A]^T$$
$$u = [\Delta q_1 \quad \Delta q_2]^T$$
$$d = [\Delta c_{Af} \quad \Delta k]^T$$
$$y = [\Delta h \quad \Delta c_A]^T$$

For an arbitrary variable ψ, the deviation variable, $\Delta\psi$, is the difference between the current value and the nominal value:

$$\Delta\psi = \psi(t) - \psi^* \tag{11}$$

From part 1, using eq. (3) and eq. (4), we define

$$f_1 = \frac{\mathrm{d}h}{\mathrm{d}t} = \frac{q_1 - q_2}{A} \tag{12a}$$
$$f_2 = \frac{\mathrm{d}c_A}{\mathrm{d}t} = (c_{Af} - c_A)\,\frac{q_1}{Ah} - k \cdot c_A^2 \tag{12b}$$

Using the 1st-order Taylor expansion, and introducing the deviation variables from eq. (11), the equations can be rewritten:

$$\frac{d\Delta h}{dt} = \left.\frac{\partial f_1}{\partial q_1}\right|_* \Delta q_1 + \left.\frac{\partial f_1}{\partial q_2}\right|_* \Delta q_2 \tag{13a}$$
$$\frac{d\Delta c_A}{dt} = \left.\frac{\partial f_2}{\partial q_1}\right|_* \Delta q_1 + \left.\frac{\partial f_2}{\partial c_{Af}}\right|_* \Delta c_{Af} + \left.\frac{\partial f_2}{\partial k}\right|_* \Delta k + \left.\frac{\partial f_2}{\partial h}\right|_* \Delta h + \left.\frac{\partial f_2}{\partial c_A}\right|_* \Delta c_A \tag{13b}$$

all terms equal to zero to to a lack of dependency where neglected in the derivation. Calculating the remaining partial derivatives gives:

$$\left.\frac{\partial f_1}{\partial q_1}\right|_* = \frac{1}{A} = \frac{1}{4} = 0.25\,\mathrm{m}^{-2} \tag{14a}$$

$$\left.\frac{\partial f_1}{\partial q_2}\right|_* = -\frac{1}{A} = -\frac{1}{4} = -0.25\,\mathrm{m}^{-2} \tag{14b}$$

$$\left.\frac{\partial f_2}{\partial q_1}\right|_* = \frac{\left(c_{Af}^* - c_A^*\right)}{Ah^*} = \frac{(1-0.05)}{4\cdot 1} = 0.2375\,\mathrm{kmol} \tag{14c}$$

$$\left.\frac{\partial f_2}{\partial c_{Af}}\right|_* = \frac{q_1^*}{Ah^*} = \frac{1}{4\cdot 1} = 0.25\,\mathrm{min}^{-1} \tag{14d}$$

$$\left.\frac{\partial f_2}{\partial k}\right|_* = -{c_A^*}^2 = -\left(0.05\right)^2 = -0.0025\,\mathrm{kmol}^2/\mathrm{m}^6 \tag{14e}$$

$$\left.\frac{\partial f_2}{\partial h}\right|_* = -\frac{\left(c_{Af}^* - c_A^*\right)\cdot q_1^*}{A{h^*}^2} = -\frac{(1-0.05)\cdot 1}{4\left(1\right)^2} = -0.2375\,\mathrm{kmol\,m}^{-3}\,\mathrm{min}^{-1} \tag{14f}$$

$$\left.\frac{\partial f_2}{\partial c_A}\right|_* = -\frac{q_1^*}{Ah^*} - 2k^*c_A^* = -\frac{1}{4\cdot 1} - 2\cdot 95\cdot 0.05 = -9.75\,\mathrm{min}^{-1} \tag{14g}$$

By setting eq. (10a) equal to eq. (13), it becomes clear that in order to have the same system of equations:

$$A = \left.\begin{bmatrix} \frac{\partial f_1}{\partial h} & \frac{\partial f_1}{\partial c_A} \\ \frac{\partial f_2}{\partial h} & \frac{\partial f_2}{\partial c_A} \end{bmatrix}\right|_* = \begin{bmatrix} 0 & 0 \\ -0.2375 & -9.75 \end{bmatrix}$$

$$B = \left.\begin{bmatrix} \frac{\partial f_1}{\partial q_1} & \frac{\partial f_1}{\partial q_2} \\ \frac{\partial f_2}{\partial q_1} & \frac{\partial f_2}{\partial q_2} \end{bmatrix}\right|_* = \begin{bmatrix} 0.25 & -0.25 \\ 0.2375 & 0 \end{bmatrix}$$

$$E = \left.\begin{bmatrix} \frac{\partial f_1}{\partial c_{Af}} & \frac{\partial f_1}{\partial k} \\ \frac{\partial f_2}{\partial c_{Af}} & \frac{\partial f_2}{\partial k} \end{bmatrix}\right|_* = \begin{bmatrix} 0 & 0 \\ 0.25 & -0.0025 \end{bmatrix}$$

In our system, the states, x, and the outputs, y, are the same, meaning that $y = x$. In order for this to be the in the standard state-space form, in eq (10), C must be the identity matrix, and D = W must be zero-matrices:

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$W = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

2.2 Transfer functions

In order to find the transfer functions, firstly, the laplace transform of the standard state-space was taken. The transfer function matrices could then be computed using eq. (15) and eq. (16). Where I is the 2x2 identity matrix.

$$G(s) = C \cdot (s \cdot I - A)^{-1} \cdot B + D \tag{15}$$

$$G_d(s) = C \cdot (s \cdot I - A)^{-1} \cdot E + W \tag{16}$$

Inserting the matrices from section 2.1 gives the transfer function matrices. The detailed calculations can be found in appendix C

$$\begin{aligned} G &= \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} = \begin{bmatrix} \frac{0.25}{s} & \frac{-0.25}{s} \\ \frac{-6.090 \cdot 10^{-3}(-4s+1)}{s(0.1026s+1)} & \frac{6.090 \cdot 10^{-3}}{s(0.1026s+1)} \end{bmatrix} \\ G_d &= \begin{bmatrix} G_{d11} & G_{d12} \\ G_{d21} & G_{d22} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ \frac{2.564 \cdot 10^{-2}}{0.1026s+1} & \frac{-2.564 \cdot 10^{-4}}{0.1026s+1} \end{bmatrix} \end{aligned} \tag{17}$$

2.2.1 Zeros, poles and gains

For each of the transfer functions in G and G_d, the gains, zeros and poles were calculated. The gains were calculated using eq. (8) and eq. (9), in the same way as in Part 1, looking at the steady state, or the integrating gain at the end of the simulation. The zeros are all values of s where the numerator is zero, and the poles are the values of s where the denominator is zero.

Table 3: Zeroes, poles, and steady state gain of all the G and G_d transfer functions.

Transfer function in	Gain	Gain unit	Zeros	Poles	Zeroes and poles unit
G_{11}	0.2500	m^{-2}	-	0	$[s^{-1}]$
G_{12}	-0.2500	m^{-2}	-	0	$[s^{-1}]$
G_{21}	-0.0061	$\mathrm{kmol/m}^6$	0.25	-9.75, 0	$[s^{-1}]$
G_{22}	0.0061	$\mathrm{kmol/m}^6$	-	-9.75, 0	$[s^{-1}]$
G_{d11}	0	$\mathrm{m}^4/\mathrm{kmol}$	-	-	$[s^{-1}]$
G_{d12}	0	$\mathrm{kmol \cdot min/m}^2$	-	-	$[s^{-1}]$
G_{d21}	0.0256	-	-	-9.75	$[s^{-1}]$
G_{d22}	$-2.564 \cdot 10^{-4}$	$\mathrm{kmol}^2\mathrm{\cdot min/m}^6$	-	-9.75	$[s^{-1}]$

2.3 Comparison of nonlinear and linearized responses

The equations used to describe the system, eq. (3) and eq. (4) are both non-linear. To see how the linearized model performed, step increases in the disturbances as well as the manipulated variables were performed. Both the responses for the non-linear and the linearized models were plotted together. In Figure 3 the responses to step increases in the manipulated variables are presented. In Figure 4 the responses to the disturbances are plotted.

From Figure 3 and Figure 4, it is apparent that the linearized model is quite a good approximation, being exact for the relations to h. However, for the reponses in c_A, the linearized model deviates from the non-linear model.

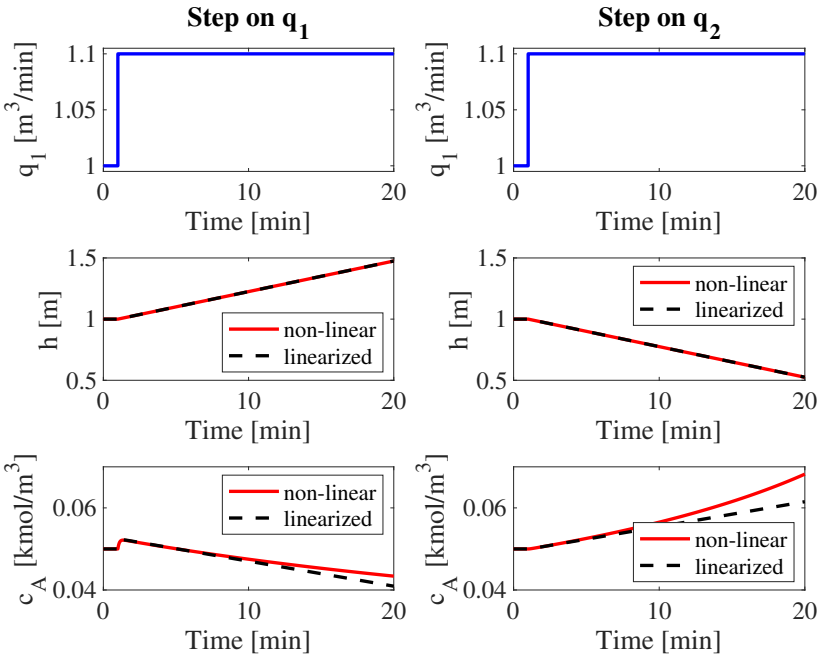

Figure 3: Simulation result for a 10% step increase in a single manipulated variable, with the resulting response for the output variables h and c_A both in the non-linear and linear models

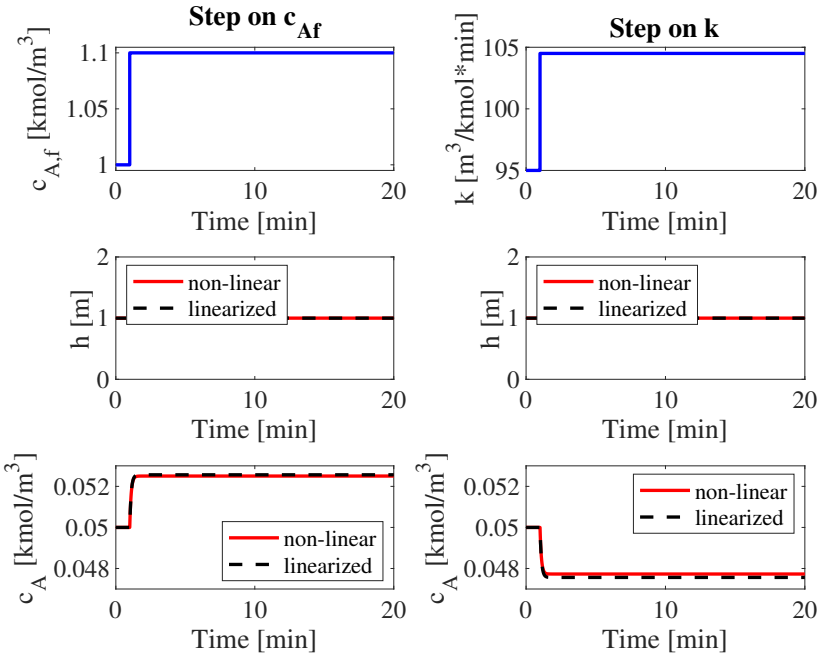

Figure 4: Simulation result for a 10% step increase in a single disturbance, with the resulting response for the output variables h and c_A both in the non-linear and linear models

2.4 Half-rule approximation of G

Looking at the different transfer functions in G, three of them are already first order. However, G_{21} and G_{22} are second order transfer functions:

$$G_{21}(s) = \frac{-6.090 \cdot 10^{-3}\,(-4s+1)}{s\,(0.1026s+1)}$$

$$G_{22}(s) = \frac{6.090 \cdot 10^{-3}}{s\,(0.1026s+1)}$$

In order to approximate them to a first order transfer functions, the half-rule is used[2]. The largest neglected time constant from the denominator is distributed evenly across the time delay and the smallest retained time constant. A first order transfer function is wanted, and the smallest of the factors in the denominator must be neglected. For G_{21}, Assuming that we can approximate:

$$\frac{1}{s} \approx \lim_{t\to\infty} \frac{\tau_1}{\tau_1 s + 1}$$

then the factor $\frac{1}{s}$ should be kept, with a time constant of ∞. Then:

$$\theta = \theta_0 + T_1 + \frac{\tau_2}{2} = 0 + 4 + \frac{2}{34} = \frac{69}{17}$$

$$\tau_I = \theta_1 + \frac{\tau_2}{2} = \infty + \frac{2}{34} = \infty$$

Where the exact value for $\tau_2 = \frac{4}{39}$ was reintroduced. The half rule approximation becomes:

$$G_{21}(s) \approx -6.090 \cdot 10^{-3} \cdot \frac{e^{-\frac{69}{17}s}}{s}$$

To test the approximation, the half-rule approximation was tested against the linearized transfer function, using the step-function in MATLAB. The result is shown in Figure 5. It is clear from the graphs that the half-rule is a valid approximation, as the graphs are perfectly aligned, except for the delay at the beginning.

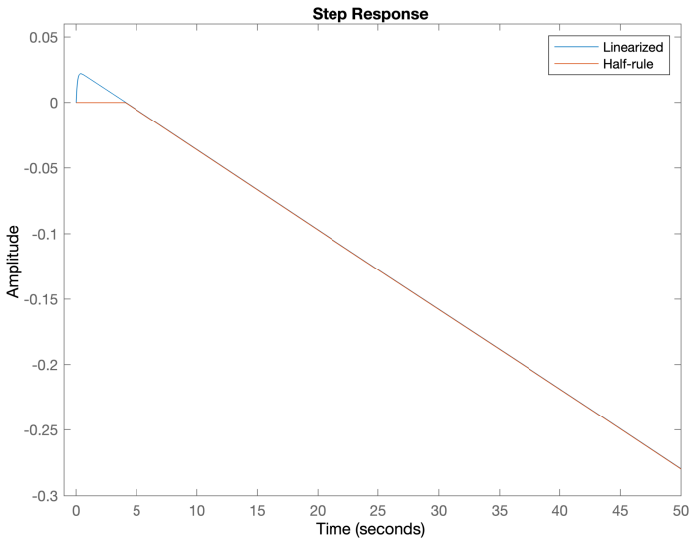

Figure 5: Simulation result of a step response in the linearized transfer function G_{21} and it's half rule approximation

For G_{22}, the process is almost the same, except for that there is no time constant in the numerator. The delay is approximated as:

$$\theta = \theta_0 + \frac{\tau_2}{2} = 0 + \frac{2}{34} = \frac{2}{34}$$

Resulting in:

$$G_{22}(s) \approx 6.090 \cdot 10^{-3} \cdot \frac{e^{-\frac{2}{34}s}}{s} \tag{18}$$

In Figure 6, the result of a step response comparing the responses of the half-rule and the linearized model is shown. The half-rule is very exact, and it is necessary to zoom in a lot in order to tell them apart.

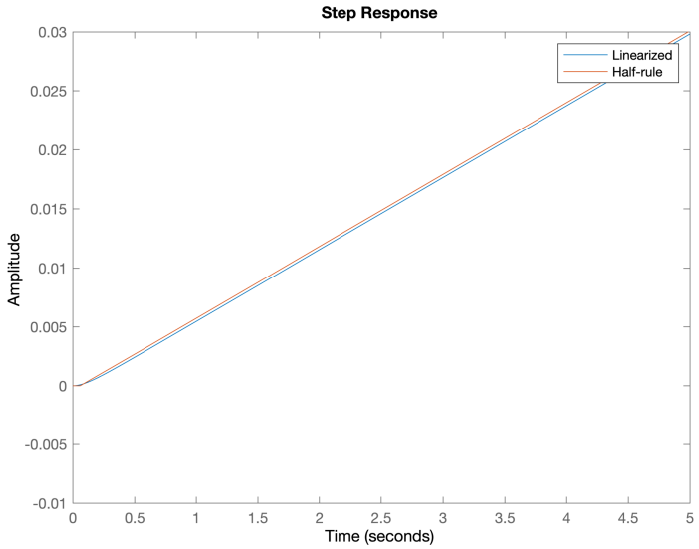

Figure 6: Simulation result of a step response in the linearized transfer function G_{22} and it's half rule approximation

2.5 Discussion

2.5.1 Properties of the transfer functions

The properties determining the behaviour of the responses of the transfer function are gains, poles and zeros.

The gains for the linearized transfer functions that completely matched the non-linear model, had the same gain as the results from the non-linear simulations had. This was all of the transfer functions related to the liquid height. Which makes sense as the partial derivative of model equation for the liquid height with regards to the MVs and DVs is constant, meaning that the non-linear model had a linear coherence, so it was expected that the linear model of these responses would be exact.

The transfer functions that did not completely match, which was the transfer functions related to c_A, got different gains than the non-linear simulations, however, the gains from the linearized model were close. The reason for this is that in the model equation for c_A, the derivatives of the inputs and disturbances are not constant, and varies with either (or both) the liquid height and ca, as well as other inputs, either through the liquid height h or directly from the c_A terms. Creating a non-linear response, meaning that the linearization introduces errors.

The only zero, for G_{21} were positive, which corresponds to an inverse response. This can be seen in Figure 3.

The poles were either 0 or negative, this corresponds to stable, non-oscillating responses. Which from the plots, appears to be the case.

2.5.2 Linearization

A plot of step responses for the non-linar and the linearized model can be seen in Figure 3 and Figure 4. From the figures, it is easy to see that the linearized model is perfectly able to

simulate the respones of the liquid height h, as the lines completely overlap. This was to be expected, as the non-linear equation for $\frac{\mathrm{d}h}{\mathrm{d}}$ is of the first or zeroth order for all MVs and DVs.

However, when modelling c_A, the linearized model was not quite as good. For the responses in the inlet and outlet flows, q_1 and q_2 respectively, the linearized model is quite good at a short time interval after the step increase, but with an increasing deviaition as time progresses. The biggest deviation occurs for a step increase in q_2, as q_2 is not directly in the model equation for c_A, but indirectly through h. Which is why the linearized model for c_A performs poorly on the step increase in q_2.

For the step increases of the disturbances, modelling the response in c_A, the model performs quite well. There is a tiny difference in the steady state value, most likely caused by the tiny difference in the gain, however, this difference is small, and it can be concluded that the linearization performs adequately.

The linearized model performs well except for the response in c_A for increases in q_1 and q_2 where the deviation increases as time progresses. The response from part 1 is non-linear, therefore, the linear model introduces an error. As the response is a combination between an integrating and first order process, this error will increase with time.

2.5.3 Half-rule

There were only two transfer functions in the transfer function matrix G that had a higher order than 1. Therefore only G_{21} and G_{22} had to be approximated using the half rule.

From Figure 5 and Figure 6 it is clear from the graphs that the half-rule are a valid approximations, as the graphs are close to being perfectly aligned, except for the delay at the beginning of the G_{21} transfer function, but as this is an inverse response, the half-rule approximation is actually better with regards to tuning.

3 Part 3: Controller Design

This part of the project will use the linearized models found in part 2 to find and discuss design of the controllers. It will also include an application to the nonlinear process.

3.1 Control structure

Propose a control structure:

The system discussed in this report is a multiple input, multiple output (MIMO) system. This is because the system has two inputs (q_1, q_2) and two outputs (h, c_A). As can be seen from figure 3, the control pairings can be based on the scaled gain in the outputs when the inputs are increased. When pairing, it is ideal if the effect which the inputs have on the outputs will increase with increasing scaled gain for the outputs.

Based on this, and the trends shown in figure 3, q_1 was paired with h and q_2 was paired with c_A. A process matrix displaying these pairings are shown in figure 7.

		Inputs	
		q_1	q_2
Outputs	h	+	-
	c_A	-	+

Figure 7: Process matrix for the MIMO system.

The selected pairings are:

$$q_1 \leftrightarrow h$$

$$q_2 \leftrightarrow c_A$$

Where the selection was done mainly based based on the scaled gain in the output variables, when increasing the inputs, as seen in Figure 3. As both MVs have the same (but opposite) influence on h, the pairing was based on which MV that affected c_A most, which was q_2. As the pairing when MVs have a big effect on the CVs gives a more ideal control structure.

A block diagram was also drawn for the proposed control structure, which is displayed in Figure 8.

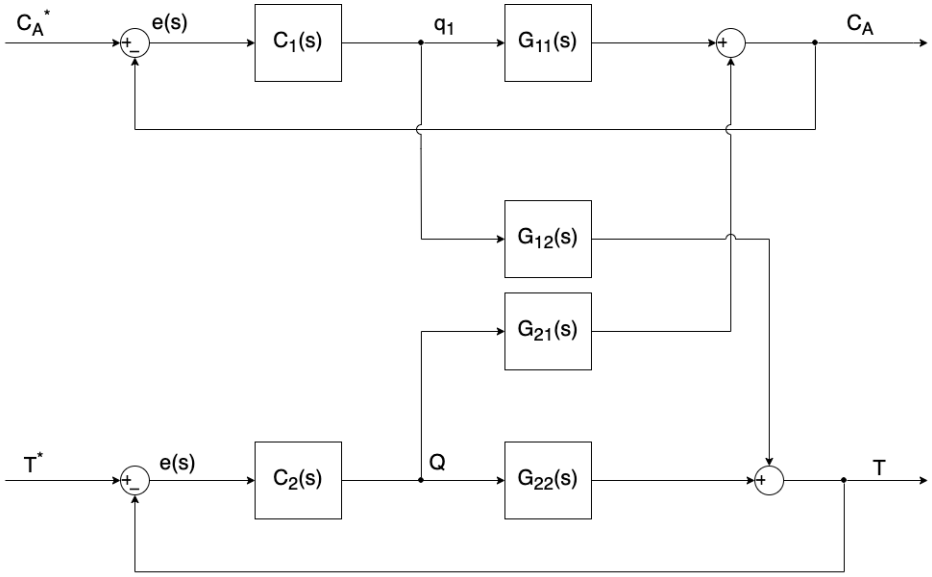

Figure 8: Block diagram representing the behaviour of the model and connecting the inputs and outputs.

A block diagram describes how the model responds to different inputs and changes. As can be seen from Figure 8, when a step change is made in the concentration C_A, it has close to no effect on the liquid height. This means that the transfer function $G_{12} \approx 0$. However, it can be seen from the block diagram that the block diagrams for the liquid height and the concentration output are linked by G_{12} and G_{21}. This means that a step change on the liquid height will have some effect on the concentration output, which is accurate for this model.

3.2 Controller tuning

From the selection of the pairings, the controls were tuned for the transfer functions G_{11} and G_{22}, shown in eq. (17). However, as none of these transfer functions are of the first order, they will require some special treatment.

G_{22} is an integrating system with an extra pole. A first order approximation of G_{22} was found using the half-rule, as shown in eq. (18). This approximation will be used instead of the integrating transfer function with the extra pole, in order to find a time delay. This allows us to use the SIMC-rules for PI-controllers. However, as both of the transfer functions are integrating responces, the "standard" SIMC-rules cannot be used.

The SIMC tuning rules for integrating responses are:

$$K_c = \frac{1}{k'} \frac{1}{\tau_c + \theta} \tag{19}$$

$$\tau_I = 4(\tau_c + \theta) \tag{20}$$

Where k' is the calculated gain for integrating response as shown in section 1.4.

The τ_c is the closed loop time constant, and it is selected based on how fast we want the loop to respond. Usually, for tight control, $\tau_c = \theta$, is selected. As G_{11} does not have a delay, τ_c is free to be chosen. τ_c should be chosen such that it is smaller or larger than τ_c for G_{11}. This is due to the fact that one τ_c needs to be faster than the other, in order for them not to "fight" each other, creating an unstable system. Furthermore, the height control has an immediate response, whereas the concentration control has a delay, meaning the height response should intuitively be faster. Therefore, τ_c for G_{11} is chosen in this report to be five times smaller than τ_c for G_{22}:

$$\tau_{c,G_{11}} = \frac{1}{5} \cdot \tau_{c,G_{12}}$$

However, selecting tight control did not provide a well tuned controller, instead, $\tau_{c,G_{12}} = 25 \cdot \theta$ was selected, as this provided better performance. The resulting tuning parameters are shown in Table 4.

Table 4: Tuning paramteters for the PI-controllers used in the simulation.

Transfer function	τ_c	K_c	τ_I
G_{11}	1.471/5	13.60	1.177
G_{22}	1.471	107.4	6.118

The controller transfer function for PI-controllers which was implemented in Simulink, is shown below

$$c(s) = K_c \frac{\tau_I s + 1}{\tau_I s} \tag{21}$$

3.3 Simulations

In order to test the controls, simulations were carried out with a 10 % step increase in the disturbances and the setpoints of the states. The results of the simulations with controllers are presented in the figures below.

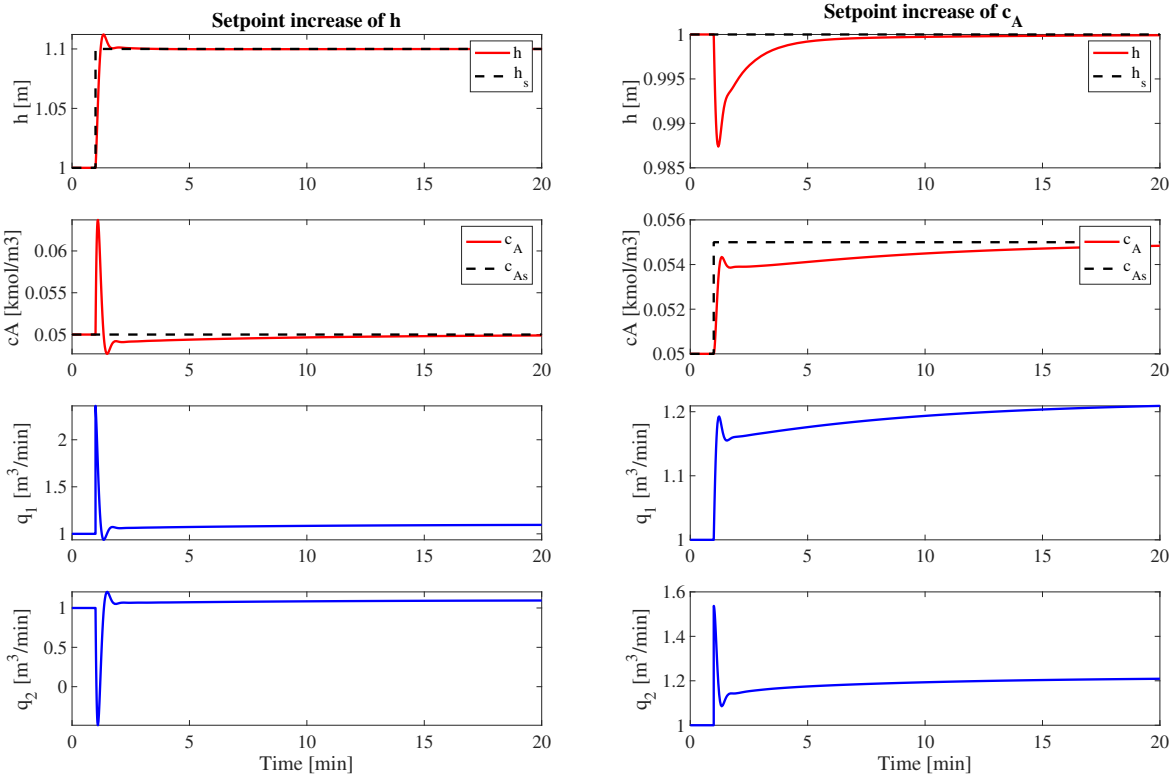

Figure 9: Simulation result of a 10 % step increase in the setpoints of the states, with implemented PI-controller tuned using the SIMC-rules.

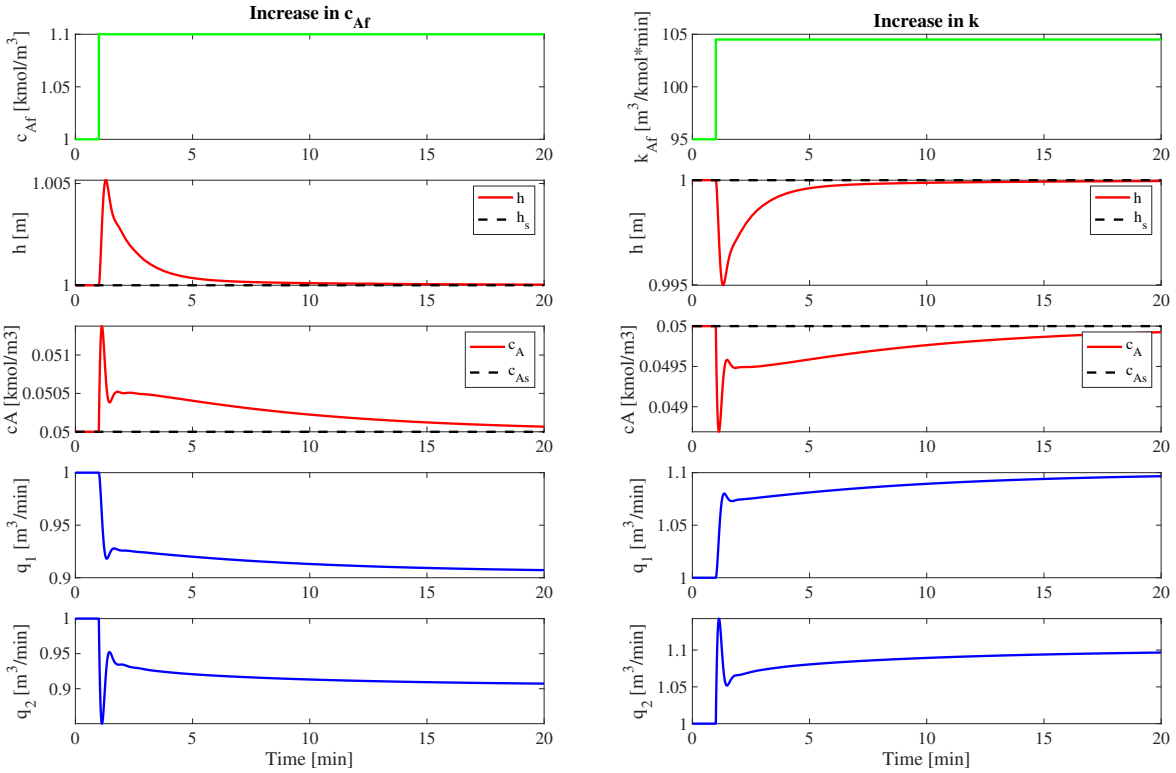

Figure 10: Simulation result of a 10 % step increase in the disturbances, with implemented PI-controller tuned using the SIMC-rules.

3.3.1 Integral Absolute Error

The integral of absolute error, IAE, is the integration of the error of the states compared to their setpoints, integrated over the simulation time. IAE is a performance index that can be used to evaluate controllers. The IAE for the step increases can be found in Table 5

Table 5: The IAE for the responses of the controllers for setpoint increases and step increases in the disturbances.

Change description	IAE for h-controller	IAE for c_A-controller
Setpoint increase in h	0.0162	0.0101
Setpoint increase in c_A	0.0180	0.0109
Step change in c_{Af}	0.0080	0.0049
Step change in k	0.0083	0.0050

Even though the IAE for C_A and h cannot be compared, it may be used as an indicator for the quality of each of the two controllers.

3.4 Bode plot

A bode plot was drawn for the height-inlet flow controller. The closed loop transfer function for this controller is defined as:

$$\begin{aligned} L(s) &= c_1 \cdot G_{11} \\ &= 13.60 \frac{1.177s + 1}{1.177s} \cdot \frac{0.25}{s} \\ &= 2.889 \frac{1.177s + 1}{s^2} \end{aligned} \tag{22}$$

The resulting bode plot from L is shown in Figure 11

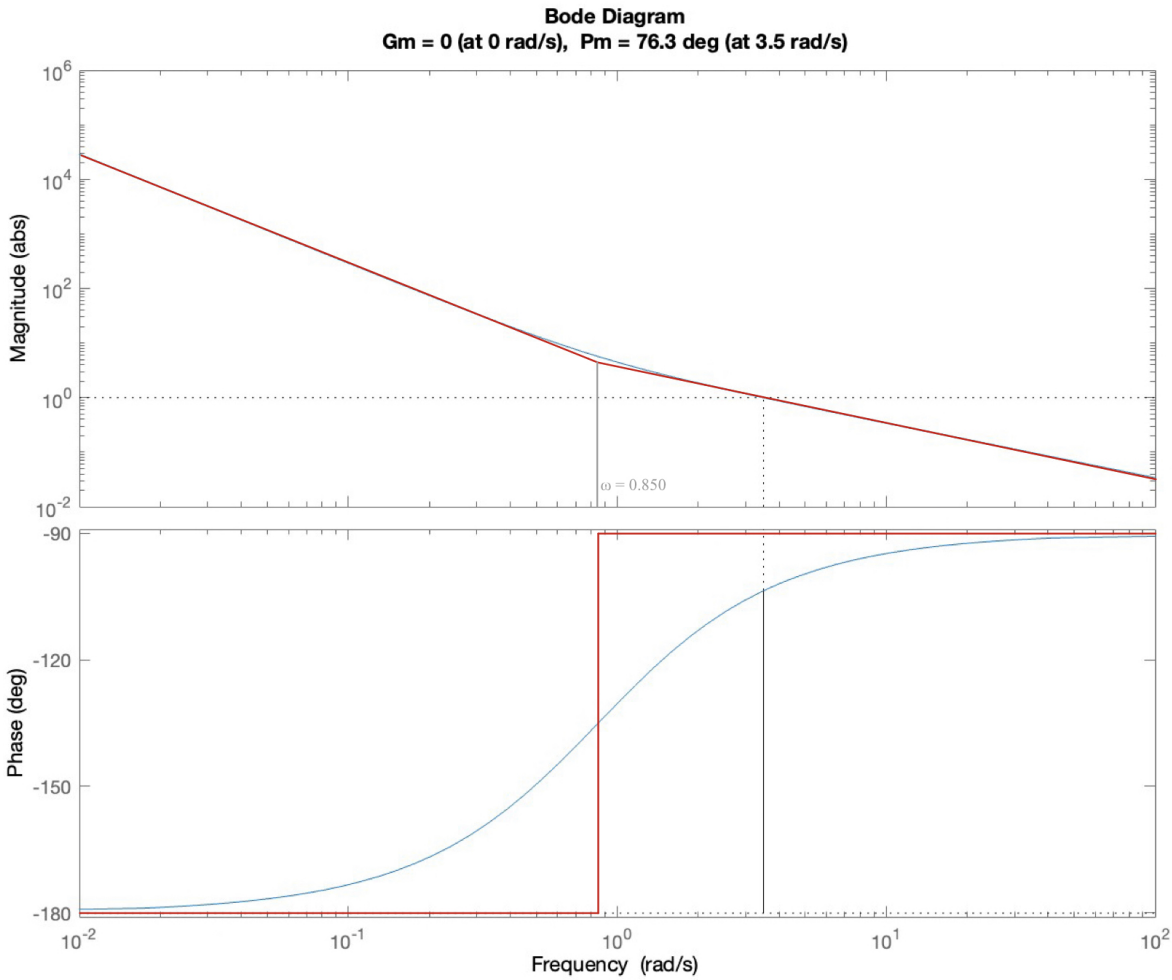

Figure 11: The bode plot of the closed loop transfer function for the pairing $q_1 \leftrightarrow h$ with asymptotes drawn in red.

From the bode plot, it can be observed that there is only one zero, which is at approximately $\omega = 0.85$. The poles are not visible on the plot, as they are 0. This is the reason for the steeper downward slope for the magnitude (-2 for the log-log plot), and the phase beginning at $-180°$

3.4.1 Margins

By applying polar form rules to the transfer function in eq. (22), the following expression were used to calculate the magnitude and the phase respectively:

$$|L(j\omega)| = 2.889\frac{\sqrt{1.177^2\omega^2 + 1}}{\omega^2}$$

$$\angle L(j\omega) = -180° + \arctan(1.177\omega)$$

From these equations, $\omega_c = 0$ and $\omega_{180} = 3.499$. From these values, the margins could be calculated:

- Gain margin: ∞ $(\theta = 0)$
- Phase margin: $76.32° = 1.33\,\text{rad}$
- Time delay margin: $0.38\,\text{s}$

And we can see that the calculated values coincide with the values from the bode plot shown in Figure 11.

3.5 Discussion

3.6 Observed closed loop responses

Figure 9 and Figure 10 shows the responses of the proposed control structure. The quality of the control structure may then be discussed based on these figures. As can be observed, the controller was able to counteract the step changes made in both inputs (q_1, q_2) and both outputs (h, c_A).

In Figure 9, the responses when performing a step point change on the output variables are shown. As can be seen, the three figures on the left shows the responses when a step change is made in h. As can be seen from the graphs, h, c_A, q_1 and q_2 all quickly adapt to the new setpoint. It is, however, important to notice that the response for q_2 is negative in the first oscillation. This seems impossible, as the flowrate cannot be negative. Nevertheless, in a real system, this would only mean that q_2 is saturated, and this would be controlled by anti-windup. Furthermore, it is not a large negative value, which means the results may still be feasible. Aside from this, the h-controller is, based on the responses, a good controller. All the variables reach the new setpoint quickly and does not oscillate.

The graphs on the right side in Figure 9 displays the responses when a setpoint increase in c_A is performed. It can be seen that there is slightly more delay for c_A than for h, which is logical as τ_c for this controller is 5 times larger than for h. Furthermore, it is necessary that the concentration tuning is a bit slower than the height tuning in order to avoid unwanted interaction between the two controllers. All variables also reach the new setpoint fairly quickly with the concentration tuning, which suggests that this tuning is good. As can be seen from the c_A response for the c_A setpoint increase in Figure 9, it doesnt's seem to quite reach the setpoint within the same time as the other variables. However, this difference to the setpoint is considered to be negligible, as the y-axis for this variable is much smaller than for the other variables.

Figure 10 shows the closed-loop behaviour when making a 10 % step increase in the disturbances, c_{Af} and k. From these responses, it can also be observed that all variables adapt to the new setpoint value fairly quickly and without much oscillation. This suggests that the controller works well. From the graphs on the left, it can be observed that when making a step change in c_{Af}, c_A increases. This is reasonable, as initial concentration will have this effect on c_A. To compensate for the increase in c_A, q_2 decreases. Consequently, the height also increases quickly. It can be seen that q_1 decreases in order to compensate for the increase in h. The system then stabilizes.

On the right part of Figure 10, a step change on the reaction constant, k, was made. Understandably, the concentration of A will decrease when the reaction constant is increased. This is due to the fact that more of reactant A will be consumed when k is increased. Then, q_2 will increase to compensate for the decrease in concentration. When q_2 is decreased, a spike in h is also observed. To compensate for this, q_1 will decrease before the system then quickly stabilizes at the new setpoint.

Overall, both the h-controller and the k-controller can be concluded to be well working controllers. All variables quickly adjust to the new setpoints, both when a step increase is performed on the output variables and on the disturbances.

3.7 Controller performance

The IAE values for C_A and h from the simulation were presented in Table 5. As can be seen for the h-controller, the IAE values were small but positive for all changes. The IAE values

being so small indicates that the h-controller works well for these changes. The controller is fast, and as can be seen from the step change responses, all variables quickly return to the new setpoints. It can be seen that the setpoint increase in c_A creates a slightly larger IAE than for the step change in c_{Af}. This means that the h-controller performs better for step changes on the disturbance than the setpoint change on outputs. The IAEs being small can also be understood from Figure 9 and Figure 10, where it can be seen that a change in concentration does not have a large impact on the height. As can be seen, this is confirmed from the fact that the IAE for the setpoint increase in h is also larger than the IAE for the step change in k.

The IAEs for the c_A-controllers are also compared in Table 5. Here, it can also be seen that the controller performes better when counteracting the changes in disturbance than for the setpoint changes for the outputs. However, all the IAEs are small also for this controller, meaning that the controller is well suited for these changes.

In Figure 9, for the response in a setpoint increase in h, we can see that q_2 becomes negative, which is not realistic for this system. In a more realistic set-up. The controller would have saturated when the valve was fully closed, and anti-windup would have to be implemented. However, in this project, this was not part of the assignment.

To conclude, both the h-controller and the c_A-controller have sufficient tuning parameters. This can be seen from the fact that the IAEs are small. Furthermore, it can be seen from the simulated graphs for the responses. There are little observed oscillations and overshoots for the responses. As can be observed, all outputs return to the desired values as well. Nevertheless, it can be concluded that the controllers work slightly better at counteracting disturbances than setpoint changes on the outputs.

3.8 Bode plot

The bode plot, shown in Figure 11, is done for the feedback loop, as well as for the tunings. As can be seen from this plot, the asymptotes are good fits. This indicates that the frequency window is large enough for the bode plot to have time to respond to fit the asymptotes.

As was shown, the gain margin goes to zero. Furthermore, the phase margin is found to be 1.33 rad. This indicates how much the phase can be shifted before the system gets unstable.

4 Final discussion

In this project, the necessary steps to tune a CSTR-reactor with a perfect temperature control was performed. This was done in three main parts. In part 1, a non-linear model of the system was derived, and simulations was performed on the model to test how changes in inputs and disturbances would affect the system. The system responses were sensible, as for example, changes in the feed consentration or the reaction constant did not affect the level in the tank.

In part 2, a linear approximation of the model was found, and analyzed. It was found that for the most part, the linear model gave good approximations, except for responses in c_A for input changes, where the linearized model showed an increasing deviation from the non-linear model as time progressed. Due to the responses not being linear in the non-linear model.

In part 3 the pairings of MVs and CVs were found using the results from part 1 and part 2. The pairings were used to create controllers for the system. Tuning parameters for PI-controlles were obtained from the linearized model using the SIMC-rules. Then the model was simulated, and tested for disturbances, and set-point increases. The responses and IAE values verified the control parameters which were calculated.

In conclusion, it was found that the controllers work well for this system, with the exception of q_2 being negative for one of the responses. In the real world, this would be fixed with

anti-windup, and should not be much of an issue. The fact that the controllers (mostly) work well means that the system was modelled, and controlled properly through the analyses and simulations which were performed throughout the three parts. The results were verified by observing the responses.

4.1 Additional feedback

This project has helped create a better understanding for the entire course, by combining different parts of the curriculum and applying them to a problem. The teaching assistants provided valuable help, and insight for solving the project tasks.

For next year, it might be beneficial if the teaching assistants have "the solutions" for the project, for example, the equations and gains for the non-linear model, the transfer functions, gains, poles and zeroes for the linearized model, and tuning parameters for the tuning, as well as all graphs we were expected to create. Most of the assignments were unique, and it would benefit everyone if the teaching assistants would have some extra material, on each problem so that they could help us more efficiently, by verifying if the students were doing the tasks correctly.

Furthermore, the project was a great learning experience. However, when the report is not graded, as in previous years, it becomes more like an exercise. It would therefore maybe be beneficial if there was a pause in the regular exercises in the same weeks as the projects. If not, this subject becomes very large.

Bibliography

[1] NTNU: TKP4140 - Prosessregulering, "Projects description," 2022. Accessed 21.09.2022.

[2] S. Skogestad and C. Grimholt, "Chapter 5: The SIMC Method for Smooth PID Controller Tuning." `https://folk.ntnu.no/skoge/publications/2012/skogestad-improved-simc-pid/PIDbook-chapter5.pdf`, 2012. Accessed 22.10.22.

A Matlab scripts part 1 & 2

The following MATLAB scripts were used to simulate the step increases and calculate the gains for part 1 and 2 (with the exception of a few changes in the plotting to create the plots in part 2.)

A.1 Simulation

Listing 1: The code in the file *Simulation.m* used to simulate the step increases and calculate the gains in part 1 and 2

```
% TKP4140 Process Control Project
%% Example script showing how to run the Simulink model
% Example script that:
% - calls the computeTransferFunctions function
% - runs the Simulink model
% - plots the results

warning off
clc
clear
close all

%% Set default options for plotting. You can change this to your preferences
set(0,'DefaultTextFontName','Times',...
'DefaultTextFontSize',18,...
'DefaultAxesFontName','Times',...
'DefaultAxesFontSize',18,...
'DefaultLineLineWidth',2,...
'DefaultLineMarkerSize',5)

%% Initializing

%Compute the transfer functions
[G,Gd]=computeTransferFunctions; %Get G and Gd from computeTransferFunctions

%Model parameters
A=4; %area of the tank in m^2
p = [A]; % vertical vector with model parameters. This is used in Simulink

% Nominal  values for u and d.
% These will go to the Workspace and will be used in the "nominal u" and
% "nominal d" blocks.
q1_nom = 1;
q2_nom = 1;
caf_nom = 1;
k_nom = 95;

% The vectors that get sent to the simulink simulation
u_nom=[q1_nom;q2_nom];
d_nom=[caf_nom;k_nom];

% Define the number of variales and model parameters
% These are used in the Interpreted Matlab Fcn block in Simulink
Nx = 2; %--CHANGE HERE-- number of states--NUMBER OF DIFFERENTIAL EQUATIONS
Nu = 2; %--CHANGE HERE-- number of inputs--NUMBER OF MVS
Nd = 2; %--CHANGE HERE-- number of disturbances--NUMBER OF DVS
Np = 1; %--CHANGE HERE-- number of model parameters--CONSTANT VALUES
Ny = Nx;% Do not change--number of outputs---equal to number of states

%INITIAL CONDITIONS FOR THE STATES
%Set the initial conditions you found for every state.
%These intial conditions will be used to solve the differential equations.
x0 = [1;1/20];  % [h;cA]
```

```
%% Set the time for giving the steps
u_t_step = 1; % In the "q0 increase" block, the step time will be set to 1.
d_t_step = 1; % In the "q1 increase" block, the step time will be set to 1.

%% Set simulation time
tsim = 20; % same units as your model

%% Step in q1
% Simulating a 10% increase in q1:
q1_step=0.1*q1_nom;
q2_step=0;
k_step=0;
caf_step=0;
% Inserting updated values into step-vectors, which are sent to simulink
u_step = [q1_step;q2_step];
d_step = [caf_step;k_step];

%run Simulink model to test the step
sim('SimulinkPart2')


figure(1)
subplot(321)
plot(time,u(:,1),'blue')
title('Step on q_1')
ylabel('q_1 [m^3/min]')
xlabel('Time [min]')
ylim([0.995, 1.105]);

figure(1)
subplot(323)
plot(time,x(:,1),'red')
hold on
plot(time,ylinear(:,1),'black', LineStyle='--')
legend('non-linear','linearized','Location','southeast')
ylabel('h [m]')
xlabel('Time [min]')
ylim([0.5,1.5]);
hold off

figure(1)
subplot(325)
plot(time,x(:,2),'red')
hold on
plot(time,ylinear(:,2),'black', LineStyle='--')
legend('non-linear','linearized','Location','northeast')
ylabel('c_A [kmol/m^3]')
ylim([0.04, 0.07]);
xlabel('Time [min]')
hold off

% Finding the slope, integrating process eq(9)
%delta u
du_q1 = 0.1*q1_nom;
%delta h
dh_q1 = x(end,1) - x((end-4),1);
dh_q1_linear = ylinear(end,1) - ylinear((end-4),1);
%delta ca
dcaq1 = x(end,2) - x((end-4),2);
dcaq1_linear = ylinear(end,2) - ylinear((end-4),2);
%delta t
dtq1 = time(end) - time(end-4);

%calculating the gain of h
h_slopeq1 = dh_q1/dtq1;
```

```
h_gain_q1 = h_slopeq1/du_q1;
h_slopeq1_linear = dh_q1_linear/dtq1;
h_gain_q1_linear = h_slopeq1_linear/du_q1;

%calculating the gain of ca
ca_slopeq1 = dcaq1/dtq1;
ca_gain_q1 = ca_slopeq1/du_q1;
ca_slopeq1_linear = dcaq1_linear/dtq1;
ca_gain_q1_linear = ca_slopeq1_linear/du_q1;

%Printing the result
disp('For an increase in q1:')
fprintf('The non-linear gain in h is: %1.4f \n', h_gain_q1)
fprintf('The linearized gain (G_11) in h is: %1.4f \n', h_gain_q1_linear)
fprintf('The non-linear gain in cA is: %1.4f \n', ca_gain_q1)
fprintf('The linearized gain (G_21) in h is: %1.4f \n\n', ca_gain_q1_linear)


%% Step in q2
% Simulating a 10% increase in q2:
q1_step=0;
q2_step=0.1*q2_nom;
k_step=0;
caf_step=0;
% Inserting updated values into step-vectors, which are sent to simulink
u_step = [q1_step;q2_step];
d_step = [caf_step;k_step];

%run Simulink model to test the step
sim('SimulinkPart2')


figure(1)
subplot(322)
plot(time,u(:,2),'blue')
title('Step on q_2')
ylabel('q_1 [m^3/min]')
xlabel('Time [min]')
ylim([0.995, 1.105]);

figure(1)
subplot(324)
plot(time,x(:,1),'red')
hold on
plot(time,ylinear(:,1),'black', LineStyle='--')
legend('non-linear','linearized','Location','northeast')
ylabel('h [m]')
xlabel('Time [min]')
ylim([0.5,1.5]);
hold off

figure(1)
subplot(326)
plot(time,x(:,2),'red')
hold on
plot(time,ylinear(:,2),'black', LineStyle='--')
legend('non-linear','linearized','Location','southeast')
ylabel('c_A [kmol/m^3]')
ylim([0.04, 0.07]);
xlabel('Time [min]')
hold off

figure(1)
print('-depsc2','-r600','Part2_inputs.eps') % save figure as eps

% Finding the slope, integrating process eq(9)
```

```
%delta u
du_q2 = 0.1*q2_nom;
%delta h
dh_q2 = x(end,1) - x((end-4),1);
dh_q2_linear = ylinear(end,1) - ylinear((end-4),1);
%delta ca
dcaq2 = x(end,2) - x((end-4),2);
dcaq2_linear = ylinear(end,2) - ylinear((end-4),2);
%delta t
dtq2 = time(end) - time(end-4);

%calculating the gain of h
h_slopeq2 = dh_q2/dtq2;
h_gain_q2 = h_slopeq2/du_q2;
h_slopeq2_linear = dh_q2_linear/dtq2;
h_gain_q2_linear = h_slopeq2_linear/du_q2;

%calculating the gain of ca
ca_slopeq2 = dcaq2/dtq2;
ca_gain_q2 = ca_slopeq2/du_q2;
ca_slopeq2_linear = dcaq2_linear/dtq2;
ca_gain_q2_linear = ca_slopeq2_linear/du_q2;

%Printing the result
disp('For an increase in q2:')
fprintf('The non-linear gain in h is: %1.4f \n', h_gain_q2)
fprintf('The linearized gain (G_12) in h is: %1.4f \n', h_gain_q2_linear)
fprintf('The non-linear gain in cA is: %1.4f \n', ca_gain_q2)
fprintf('The linearized gain (G_22) in ca is: %1.4f \n\n', ca_gain_q2_linear)

%% Step in cAf
% Simulating a 10% increase in cAf:
q1_step=0;
q2_step=0;
k_step=0;
caf_step=0.1*caf_nom;
% Inserting updated values into step-vectors, which are sent to simulink
u_step = [q1_step;q2_step];
d_step = [caf_step;k_step];

%run Simulink model to test the step
sim('SimulinkPart2')

figure(2)
subplot(321)
plot(time,d(:,1),'blue')
title('Step on c_{Af}')
ylabel('c_{A,f} [kmol/m^3]')
xlabel('Time [min]')
ylim([0.995, 1.105]);

figure(2)
subplot(323)
plot(time,x(:,1),'red')
hold on
plot(time,ylinear(:,1),'black', LineStyle='--')
legend('non-linear','linearized','Location','northwest')
ylabel('h [m]')
xlabel('Time [min]')
hold off

figure(2)
subplot(325)
plot(time,x(:,2),'red')
hold on
plot(time,ylinear(:,2),'black', LineStyle='--')
```

```
legend('non-linear','linearized','Location','southeast')
ylabel('c_A [kmol/m^3]')
ylim([0.047, 0.053])
xlabel('Time [min]')
hold off

%Finding the gains, SS-gain, eq(8)
ducaf = 0.1*caf_nom;

%calculating the gains of h
dycaf_h = max(x(:,1)) - min(x(:,1));
dycaf_linear_h = max(ylinear(:,1)) - min(ylinear(:,1));
h_gain_caf = dycaf_h/ducaf;
h_gain_caf_linear = dycaf_linear_h/ducaf;

%calculating the gains of ca
dycaf_ca = max(x(:,2)) - min(x(:,2));
dycaf_linear_ca = max(ylinear(:,2)) - min(ylinear(:,2));
ca_gain_caf = dycaf_ca/ducaf;
ca_gain_caf_linear = dycaf_linear_ca/ducaf;

%Printing the result
disp('For an increase in caf:')
fprintf('The non-linear gain in h is: %1.4f \n', h_gain_caf)
fprintf('The linearized gain (Gd_11) in h is: %1.4f \n', h_gain_caf_linear)
fprintf('The non-linear gain in cA is: %1.4f \n', ca_gain_caf)
fprintf('The linearized gain (Gd_12) in ca is: %1.4f \n\n', ca_gain_caf_linear)

%% Step in k
% Simulating a 10% increase in k:
q1_step=0;
q2_step=0;
k_step=0.1*k_nom;
caf_step=0;
% Inserting updated values into step-vectors, which are sent to simulink
u_step = [q1_step;q2_step];
d_step = [caf_step;k_step];

%run Simulink model to test the step
sim('SimulinkPart2')


figure(2)
subplot(322)
plot(time,d(:,2),'blue')
title('Step on k')
ylabel('k [m^3/kmol*min]')
xlabel('Time [min]')

figure(2)
subplot(324)
plot(time,x(:,1),'red')
hold on
plot(time,ylinear(:,1),'black', LineStyle='--')
legend('non-linear','linearized','Location','northwest')
ylabel('h [m]')
xlabel('Time [min]')
hold off

figure(2)
subplot(326)
plot(time,x(:,2),'red')
hold on
plot(time,ylinear(:,2),'black', LineStyle='--')
legend('non-linear','linearized','Location','northeast')
ylabel('c_A [kmol/m^3]')
```

```
ylim([0.047, 0.053])
xlabel('Time [min]')
hold off

figure(2)
print('-depsc2','-r600','Part2_disturbances.eps') % save figure as eps

%Finding the gains, SS-gain, eq(8)
duk = 0.1*k_nom;

%calculating the gains of h
dyk_h = max(x(:,1)) - min(x(:,1));
dyk_linear_h = max(ylinear(:,1)) - min(ylinear(:,1));
h_gain_k = dyk_h/duk;
h_gain_k_linear = dyk_linear_h/duk;

%calculating the gains of ca
dyk_ca = min(x(:,2)) - max(x(:,2));
dyk_linear_ca = min(ylinear(:,2)) - max(ylinear(:,2));
ca_gain_k = dyk_ca/duk;
ca_gain_k_linear = dyk_linear_ca/duk;

%Printing the result
disp('For an increase in k:')
fprintf('The non-linear gain in h is: %1.4f \n', h_gain_k)
fprintf('The linearized gain (Gd_21) in h is: %1.4f \n', h_gain_k_linear)
fprintf('The non-linear gain in cA is: %1.4e \n', ca_gain_k)
fprintf('The linearized gain (Gd_22) in ca is: %1.4e \n\n', ca_gain_k_linear)
```

A.2 SysODE

Listing 2: The code in the file *SysODE.m* used to iterate over the differential equations in part 1

```
% TKP4140 - Autumn19.
% This function defines the system of ordinary differential equations (ODE)
function  dxdt = SysODE(x,u,d,p)
% The order of the argument corresponds to the order in which
% you connect the signals to the Interpreted Matlab Fcn in Simulink
% x-states; u-inputs; d-disturbances; p-model parameters

% ==========MODEL PARAMETERS============================

A = p(1); % [m^2]


%======Derivatives: WRITE YOUR DIFFERENTIAL EQUATIONS HERE =======

q1=u(1);
q2=u(2);

caf=d(1);
k=d(2);

h=x(1);
ca=x(2);


%Write the differential equations
dxdt(1) = (q1-q2)/A;
dxdt(2) = ((caf-ca)*q1/(A*h)) - k*ca^2;

dxdt = dxdt(:); % system of differential equations as vertical vector

end
```

A.3 ComputeTransferfunctions

Listing 3: The code in the file *computeTransferFunctions.m* was used to calculate the transfer function matrices in part 2

```
%% In this function you:
% - Define the differential equations for your model
% - Linearize and get your model in the the state-space form.
% - Obtain the transfer function matrices G(s) and Gd(s)
%
% To obtain the transfer functions G and Gd from the state-space form:
% Suppose you have the linearized system x_dot=Ax+Bu+Ed, y=C*x; Eq.(1)
% where x=delta states, u= delta MV, d= delta DV and y= delta CV
% From this we want to compute the transfer function matrices G(s) and Gd(s)
%
% We apply Laplace transform to Eq (1) to obtain:
% G(s)=C*(I*s-A)^-1*B and Gd(s)=C*(I*s-A)^-1*E
% You should know how to derive this by hand! possible exam question
%
% You can also consider that u inclues MVs and DVs. In this case the model
% would be x_dot=Ax+Bd*u, where Bd is a nx-by-(nMV+nDV) matrix.
% In this case you would get only one transfer function matrix G(s).
% Both forms are equivalent, as long you know what your are doing.
%
%%

function [G,Gd,A,B,C,D,E]=computeTransferFunctions


%%Definition of differential equations

% Define/create symbolic variables for your MV, DV, CV
syms h q1 q2 cA cAf k

% Tip: if your equations have additional variables that depend on
% your MVs, DVs or CVs and you want to define these additional variables,
% so that you can write your differential equations in a simpler way,
% you can declare these "aid" variables as symbolic variables too. For
% examples: valve_flow, enthalpy_in, etc.

%Define vectors of DV(d), MV(u), states(x) and CV(y)
d = [cAf;k];           %deviation variables; vertical vector (if more than 1 DV)
u = [q1;q2];                        %inputs; vertical vector (if more than 1 MV)
x = [h;cA];                  %states; vertical vector (if more than one states)
y = [h;cA];                     % outputs; vertical vector (if more than one CV)
% y can be different than the states x, but we recommend to select y=x as
% in part 3 you may decide to control all your states


%Parameters that will be used in the differential equations
Area = 4;  %m^2 - Note that we are not calling it A because below A will be
          % matrix A from x_dot=Ax+Bu+Ed, y=C*x. In other words, be careful
                                              % when naming your parameters.

%Here you can write additional algebraic equations for the variables that
%you will use in the differential equations.

%Differential equations
f(1,1) = 1/Area*(q1-q2);
f(1,2) = (cAf - cA)*(q1/(Area*h)) - k*cA^2;
%For more than 1 state: f(1,1)=...; f(1,2)=...;
%Alternatively, write f directly as an horizontal vector

%% Linearization
% Define the Jacobian - for the state space form x_dot=Ax+Bu+Ed
A=jacobian(f,x);
```

```
B=jacobian(f,u);
E=jacobian(f,d);

% Define the Jacobian from states to output: y=Cx+Du
C=jacobian(y,x);                        % Select the outputs from the states
D=jacobian(y,u);                % Matrix from input to output. Usually  D=0

%Define the nominal point for x,y, u, d
h = 1;
cA = 0.05;

q1 = 1;
q2 = 1;

cAf = 1;
k = 95;

%Replace the symbolic variables by their corresponding nominal value
A=double(subs(A));
B=double(subs(B));
E=double(subs(E));

C=double(subs(C));
D=double(subs(D));

%Laplace variable
s=tf('s');

%Identity matrix
n=length(A); I=eye(n);

%Obtaining transfer function matrix - y = G*u+Gd*d
% you must know how to do this by hand! possible exam question.
G=C*inv(s*I-A)*B+D;                  % tranfer function from input to output
Gd=C*inv(s*I-A)*E;            % tranfer function from disturbance to output

%Simplifying equation - minimum realization
%(cancels common roots in numerator and denominator)
G=minreal(G);
Gd=minreal(Gd);

%You may change the display format.
G=set(zpk(G), 'DisplayFormat','time constant');
Gd=set(zpk(Gd), 'DisplayFormat','time constant');
end
```

B Simulink part 1 & 2

In Figure 12, the setup for the simulink model used in part 1 and 2 is shown.

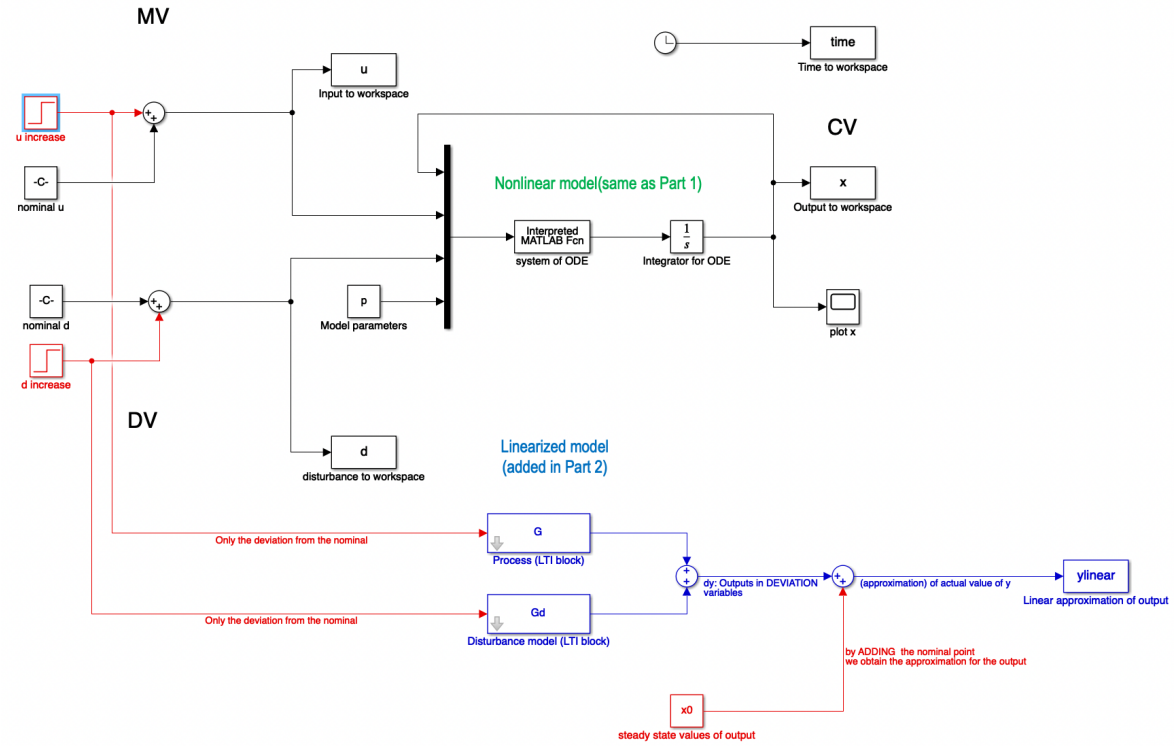

Figure 12: The simulink model used in the simulation. The handout model was adapted to handle a system of two differential equations. *u increase*, *nominal u*, *nominal d*, *d increase* and *model parameters* were defined in the MATLAB script *Simulation.m* shown in appendix A.1. The differential equations were solved numerically using the *system of ODE* block, which calls on the function defined in *SysODE.m* script shown in appendix A.2.

C Calculations of the transfer functions

The transfer functions were calculated using eq. (15) and eq. (16). Looking at the equations, the contain a common term, which will be calculated first.

$$\begin{aligned} C \cdot (s \cdot I - A)^{-1} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \left(s \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 0 \\ -0.2375 & -9.75 \end{bmatrix} \right)^{-1} \\ &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s & 0 \\ 0.2375 & s + 9.75 \end{bmatrix}^{-1} \end{aligned}$$

For a general matrix, M:

$$M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Its inverse can be calculated using:

$$M^{-1} = \frac{1}{\det(M)} \cdot \text{adj}(M) = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

Applying this, the result is:

$$\begin{aligned} C \cdot (s \cdot I - A)^{-1} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{s} & 0 \\ \frac{-0.2375}{s(s+9.75)} & \frac{1}{s+9.75} \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{s} & 0 \\ \frac{-0.2375}{s(s+9.75)} & \frac{1}{s+9.75} \end{bmatrix} \end{aligned}$$

C.1 Calculating G

Inserting the common term into eq. (15), the result is:

$$G(s) = \begin{bmatrix} \frac{1}{s} & 0 \\ \frac{-0.2375}{s(s+9.75)} & \frac{1}{s+9.75} \end{bmatrix} \cdot \begin{bmatrix} 0.25 & -0.25 \\ 0.2375 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} \frac{0.25}{s} & \frac{-0.25}{s} \\ \frac{0.2375s - 0.059375}{s(s+9.75)} & \frac{0.059375}{s(s+9.75)} \end{bmatrix}$$

Finally, rewriting to time constant form:

$$G(s) = \begin{bmatrix} \frac{0.25}{s} & \frac{-0.25}{s} \\ \frac{-6.090 \cdot 10^{-3}(-4s+1)}{s(0.1026s+1)} & \frac{6.090 \cdot 10^{-3}}{s(0.1026s+1)} \end{bmatrix}$$

C.2 Calculating G_d

Inserting the common term into eq. (15), the result is:

$$G_d(s) = \begin{bmatrix} \frac{1}{s} & 0 \\ \frac{-0.2375}{s(s+9.75)} & \frac{1}{s+9.75} \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 \\ 0.25 & -0.0025 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ \frac{0.25}{s+9.75} & \frac{-0.0025}{s+9.75} \end{bmatrix}$$

Rewriting to time constant form:

$$G_d(s) = \begin{bmatrix} 0 & 0 \\ \frac{2.564 \cdot 10^{-2}}{0.1026s+1} & \frac{-2.564 \cdot 10^{-4}}{0.1026s+1} \end{bmatrix}$$

D Matlab scripts part 3

D.1 Simulation

Listing 4: The code in the file *Part3.m* used to perform the simulations in part 3

```
% TKP4140 Process Control Project part 3
warning off
clc
clear
close all

%% Set default options for plotting
set(0,'DefaultTextFontName','Times',...
'DefaultTextFontSize',18,...
'DefaultAxesFontName','Times',...
'DefaultAxesFontSize',18,...
'DefaultLineLineWidth',2,...
'DefaultLineMarkerSize',7.75)

%% Tuning parameters for the controllers
% Collecting the gains from the output of part 2
h_gain_q1 = 0.25;
cA_gain_q2 = 0.006089743589744;
% The delay in the transfer functions as shown in the report
theta11 = 0;
theta22 = 2/34;
% Determining the closed loop time constant, as explained in the report
tauc2 = theta22*25;
tauc1 = tauc2/5;
% Calculating Kc and tauI
Kc1 = (1/h_gain_q1)*(1/(tauc1+theta11));
Kc2 = (1/cA_gain_q2)*(1/(tauc2+theta22));
tauI1 = 4*(tauc1+theta11);
tauI2 = 4*(tauc2+theta22);

%% Initializing the model

%Model parameters
A=4; %area of the tank in m^2
p = [A]; % vertical vector with model parameters. This is used in Simulink

% Nominal  values for u and d.
% The nominal values are used for
q1_nom = 1;
q2_nom = 1;
caf_nom = 1;
k_nom = 95;

% The vectors that get sent to the simulink simulation
u_nom=[q1_nom;q2_nom];
d_nom=[caf_nom;k_nom];

% Define the number of variales and model parameters
% These are used in the Interpreted Matlab Fcn block in Simulink
Nx = 2; %--CHANGE HERE-- number of states--NUMBER OF DIFFERENTIAL EQUATIONS
Nu = 2; %--CHANGE HERE-- number of inputs--NUMBER OF MVS
Nd = 2; %--CHANGE HERE-- number of disturbances--NUMBER OF DVS
Np = 1; %--CHANGE HERE-- number of model parameters--CONSTANT VALUES
Ny = Nx;% Do not change--number of outputs---equal to number of states

%INITIAL CONDITIONS FOR THE STATES
%Set the initial conditions you found for every state.
%These intial conditions will be used to solve the differential equations.
x0 = [1;1/20];  % [h;cA]
```

```
%% Set the time for giving the steps
setpoint_t_step = 1; % Sets the time for changes in the setpoint
d_t_step = 1; % Sets the time for disturbance

%% Setting the setpoints to be nominal values of states
s_h = 1;
s_cA = 1/20;

%% Simulation time
tsim = 20;

%% Testing the model

%% 10% increase in setpoint of h
s_h_step = 0.1*s_h;
s_cA_step = 0;
k_step=0;
caf_step=0;
% Inserting updated disturbance values into step-vectors, which are sent to simulink
d_step = [caf_step;k_step];

%% Running the model
sim('part3sim.mdl')

%% Plotting the result
figure(1)
subplot(421)
plot(time,x(:,1),'red', 'DisplayName','h')
hold on;
plot(time, h_set, '--', 'color', 'black', 'DisplayName', 'h_s')
hold off;
title('Setpoint increase of h')
ylabel('h [m]')
legend()

figure(1)
subplot(423)
plot(time,x(:,2),'red', 'Displayname', 'c_A')
hold on;
plot(time, cA_set, '--', 'color', 'black', 'DisplayName', 'c_{As}')
hold off;
ylabel('cA [kmol/m3]')
legend()

figure(1)
subplot(425)
plot(time,u(:,1),'blue')
hold on;
ylabel('q_1 [m^3/min]')

figure(1)
subplot(427)
plot(time,u(:,2),'blue')
hold on;
ylabel('q_2 [m^3/min]')
xlabel('Time [min]')

IAE_h_h = IAE_h(end);
IAE_cA_h = IAE_cA(end);

%% 10% increase in setpoint of cA
s_h_step = 0;
s_cA_step = 0.1*s_cA;
k_step=0;
caf_step=0;
```

```
% Inserting updated disturbance values into step-vectors, which are sent to simulink
d_step = [caf_step;k_step];

%% Running the model
sim('part3sim.mdl')

%% Plotting the result
figure(1)
subplot(422)
plot(time,x(:,1),'red', 'DisplayName','h')
hold on;
plot(time, h_set, '--', 'color', 'black', 'DisplayName', 'h_s')
hold off;
title('Setpoint increase of c_A')
ylabel('h [m]')
legend()

figure(1)
subplot(424)
plot(time,x(:,2),'red', 'Displayname', 'c_A')
hold on;
plot(time, cA_set, '--', 'color', 'black', 'DisplayName', 'c_{As}')
hold off;
ylabel('cA [kmol/m3]')
legend()

figure(1)
subplot(426)
plot(time,u(:,1),'blue')
hold on;
ylabel('q_1 [m^3/min]')

figure(1)
subplot(428)
plot(time,u(:,2),'blue')
hold on;
ylabel('q_2 [m^3/min]')
xlabel('Time [min]')

IAE_h_cA = IAE_h(end);
IAE_cA_cA = IAE_cA(end);

%% Export figure in .eps format. This gives optimal results in Latex

set(gcf,'Position',[100 100 1200 800])                % set the figure size
                        % first 2 numbers are the coordinates on your screen
                                              % 3rd number is the figure width
                                              % 4th number is the figure hight

print('-depsc2','-r600','SetPointIncrease.eps') % save eps.

%% 10% increase in setpoint of cAf
s_h_step = 0;
s_cA_step = 0;
k_step=0;
caf_step=0.1*caf_nom;
% Inserting updated disturbance values into step-vectors, which are sent to simulink
d_step = [caf_step;k_step];

%% Running the model
sim('part3sim.mdl')

%% Plotting the result
figure(2)
subplot(521)
plot(time,d(:,1),'green')
```

```
title('Increase in c_{Af}')
ylabel('c_{Af} [kmol/m^3]')

figure(2)
subplot(523)
plot(time,x(:,1),'red', 'DisplayName','h')
hold on;
plot(time, h_set, '--', 'color', 'black', 'DisplayName', 'h_s')
hold off;
ylabel('h [m]')
legend()

figure(2)
subplot(525)
plot(time,x(:,2),'red', 'Displayname', 'c_A')
hold on;
plot(time, cA_set, '--', 'color', 'black', 'DisplayName', 'c_{As}')
hold off;
ylabel('cA [kmol/m3]')
legend()

figure(2)
subplot(527)
plot(time,u(:,1),'blue')
hold on;
ylabel('q_1 [m^3/min]')

figure(2)
subplot(529)
plot(time,u(:,2),'blue')
hold on;
ylabel('q_2 [m^3/min]')
xlabel('Time [min]')

IAE_h_caf = IAE_h(end);
IAE_cA_caf = IAE_cA(end);

%% 10% increase in setpoint of k
s_h_step = 0;
s_cA_step = 0;
k_step=0.1*k_nom;
caf_step=0;
% Inserting updated disturbance values into step-vectors, which are sent to simulink
d_step = [caf_step;k_step];

%% Running the model
sim('part3sim.mdl')

%% Plotting the result
figure(2)
subplot(522)
plot(time,d(:,2),'green')
ylabel('k_{Af} [m^3/kmol*min]')
title('Increase in k')

figure(2)
subplot(524)
plot(time,x(:,1),'red', 'DisplayName','h')
hold on;
plot(time, h_set, '--', 'color', 'black', 'DisplayName', 'h_s')
hold off;
ylabel('h [m]')
legend()

figure(2)
subplot(526)
```

```
plot(time,x(:,2),'red', 'Displayname', 'c_A')
hold on;
plot(time, cA_set, '--', 'color', 'black', 'DisplayName', 'c_{As}')
hold off;
ylabel('cA [kmol/m3]')
legend()

figure(2)
subplot(528)
plot(time,u(:,1),'blue')
hold on;
ylabel('q_1 [m^3/min]')

figure(2)
subplot(5,2,10)
plot(time,u(:,2),'blue')
hold on;
ylabel('q_2 [m^3/min]')
xlabel('Time [min]')

IAE_h_k = IAE_h(end);
IAE_cA_k = IAE_cA(end);

%% Export figure in .eps format. This gives optimal results in Latex

set(gcf,'Position',[100 100 1200 800])                % set the figure size
                        % first 2 numbers are the coordinates on your screen
                                              % 3rd number is the figure width
                                              % 4th number is the figure hight

print('-depsc2','-r600','DisturbanceControlled.eps') % save eps.

%% Printing the IAE-values
fprintf(['For step change the setpoint of h\n' ...
    ' the IAE_h = %f, and IAE_cA = %f \n'], IAE_h_h, IAE_cA_h)
fprintf(['For step change the setpoint of cA\n' ...
    ' the IAE_h = %f, and IAE_cA = %f \n'], IAE_h_cA, IAE_cA_cA)
fprintf(['For step change in cAf\n' ...
    ' the IAE_h = %f, and IAE_cA = %f \n'], IAE_h_caf, IAE_cA_caf)
fprintf(['For step change in k\n' ...
    ' the IAE_h = %f, and IAE_cA = %f \n'], IAE_h_k, IAE_cA_k)
```

D.2 SysODE

The same MATLAB file shown Appendix A.2 was used in part 3 as well.

D.3 Bodeplot

Listing 5: The code in the file *Bodeplots.m* used to create the bodeplot in part 3

```
%% Tuning parameters for the controllers
% Collecting the gains from the output of part 2
h_gain_q1 = 0.25;
cA_gain_q2 = 0.006089743589744;
% The delay in the transfer functions as shown in the report
theta11 = 0;
theta22 = 2/34;
% Determining the closed loop time constant, as explained in the report
tauc2 = theta22*25;
tauc1 = tauc2/5;
% Calculating Kc and tauI
Kc1 = (1/h_gain_q1)*(1/(tauc1+theta11));
tauI1 = 4*(tauc1+theta11);

% The transfer function G11:

s=tf('s');
G11 = 0.25/s;
c1 = Kc1*(tauI1*s+1)/((tauI1*s));

L1 = G11*c1;
%L=set(zpk(L1), 'DisplayFormat','time constant')
margin(L1)
```

E Simulink part 3

In Figure 13, the setup for the simulink model used in part 3 is shown.

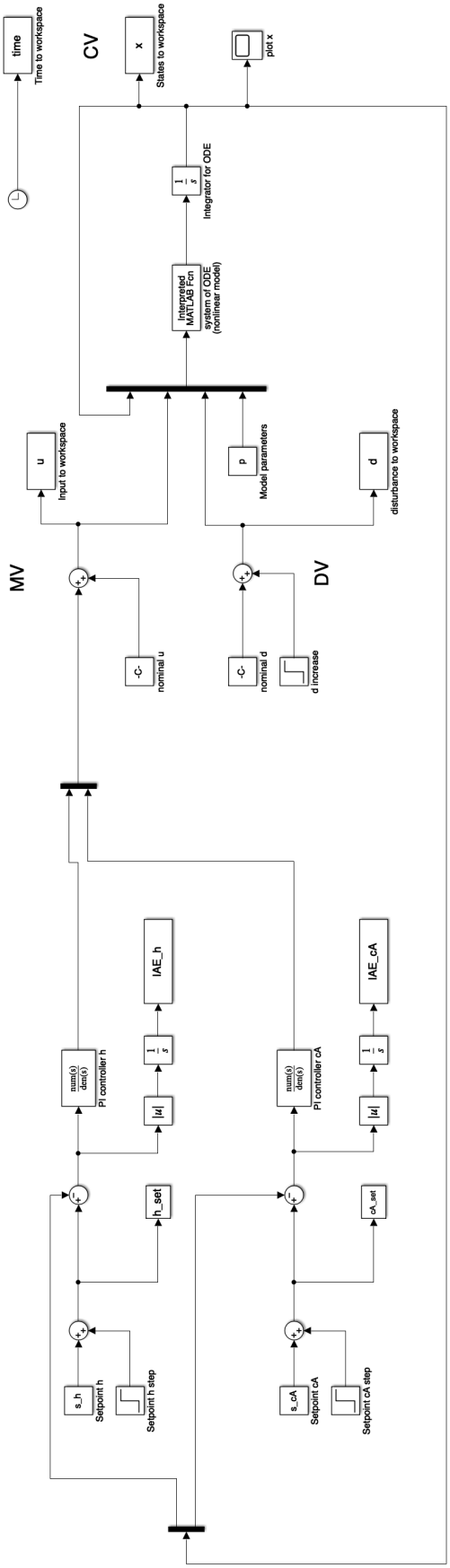

Figure 13: The simulink model used in the simulation with the implemented controllers.