NTNU – Trondheim
Norwegian University of
Science and Technology

TKP4140 Process Control

Project T: Heat Exchange in Two Tanks

Meyvisch Lukas
Van Den Berghe Anke
Verdonck Marie

November 23, 2021

Contents

List of Figures

List of Tables

1 Part 1: Modeling and Simulation

1.1 Project description

Figure 1 describes heat exchange in two stirred tanks of a liquid phase. Heat Q_1 is removed from tank 1, heat Q_2 is added to tank 2. The goal is to control temperature in tank 2 (T_2). The entire system is composed of two tanks in series. The temperature of the feed to tank 1 is dependent on the amount of heat exchange happening in tank 2 before entering tank 1. In tank 1 a heat exchange system is used to control temperature in tank 2.

Heat Q_2 is a function of the temperature of the feed and the temperature in tank 2. The following equation can be used for calculations:

$$Q_2 = UA\left(\frac{(Tf + Tf_1)}{2} - T_2\right)$$

Where U is the heat transfer coefficient and A the area through which heat transfer occurs.

The objective is to control the temperature in tank 2 by adjusting the heating system in tank 1.

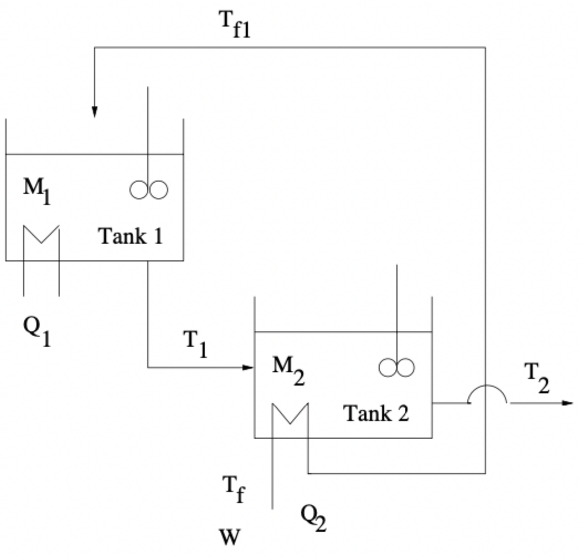


Figure 1: Sketch of the process

1.1.1 Classification of variables

Classify the variables in your model as:

- States (**x**): T_1, T_2
- Inputs (**u**): Q_1
- Outputs (measurements) (**y**): T_1, T_2
- Disturbances (**d**): T_f, W

We are manipulating input Q_1 in order to control output T_2.

1.2 Nonlinear model

In this section the nonlinear model to describe the dynamic behavior of the process will be derived, starting from some assumptions.

1.2.1 Assumptions

- Assumption 1: Constant C_p
- Assumption 2: Constant M_1 and M_2
- Assumption 3: Constant U and A
- Assumption 4: $T_{Ref} = 0$
  $H_{Ref} = 0$
- Assumption 5: H=U for liquids
  with $U = mC_p(T - T_{Ref})$
- Assumption 6: No shaft work in tanks ($W_s = 0$)
- Assumption 7: Constant volume in both tanks

1.2.2 Model

For the derivation of the non-linear model, we start from the general dynamic energy balance. This equation is then adjusted to our process. The objective is to control the temperature in tank 2. Additionally, the temperature in tank 1 is controlled. Therefore, the dynamic energy balance for both tanks is set up.

Dynamic Energy balance general form:

$$\frac{dU}{dt} = H_{in} - H_{out} + Q + W_s - p_{ex}\frac{dV}{dt}$$

Tank 1:

$$\frac{dU}{dt} = \frac{d(M_1 C_p T_1)}{dt} = wTf_1 C_p - wT_1 C_p - Q_1$$

$$M_1 C_p \frac{dT_1}{dt} = wC_p(Tf_1 - T_1) - Q_1$$

$$=> \frac{dT_1}{dt} = \frac{w}{M_1}(Tf_1 - T_1) - \frac{Q_1}{M_1 C_p}$$

Tank 2:

$$\frac{d(M_2 C_p T_2)}{dt} = wT_1 C_p - wT_2 C_p + Q_2$$

$$M_2 C_p \frac{dT_2}{dt} = wT_1 C_p - wT_2 C_p + Q_2$$

$$=> \frac{dT_2}{dt} = \frac{w}{M_2}(T_1 - T_2) + \frac{Q_2}{M_2 C_p}$$

These equations will be further used to analyze the system in Matlab and Simulink.

1.2.3 Steady state data

Table 1 contains data of the system. Steady state data is indicated by the symbol " * ". The equations under Table 1 have been used to calculate the missing data (Q_1 and UA).

Table 1: Steady state data

Variable	*Description*	Steady state value	Units
C_p	*Heat capacity*	120	J/(kg°C)
T_f	*Temperature feed*	300	°C
$T_{f,1}$	*Temperature feed into tank 1*	260	°C
T_1	*Temperature in tank 1*	200	°C
T_2	*Temperature in tank 2*	240	°C
M_1	*Mass in tank 1*	250	kg
M_2	*Mass in tank 2*	100	kg
W	*Mass flow*	50	kg/min
Q_1	*Heat added tank 1*	360 000	J/min
UA	*Product of heat transfer coefficient and surface of tank 2*	6000	J/(min*°C)
$Q2$	*Heat added tank 2*	240 000	J/min

At steady state:

$$\frac{dT_1}{dt} = 0 = w^*(Tf_1^* - T_1^*) - \frac{Q_1^*}{C_p} \qquad \frac{dT_2}{dt} = 0 = w^*(T_1^* - T_2^*) + \frac{Q_2^*}{C_p}$$

1.3 Simulation results

The nonlinear model is simulated in Simulink. A step change (a 10% increase from the nominal value) has been applied to the input variables and disturbances. The simulated plots are displayed in Figure 2. The plots include outputs T_1, T_2 and corresponding step change.

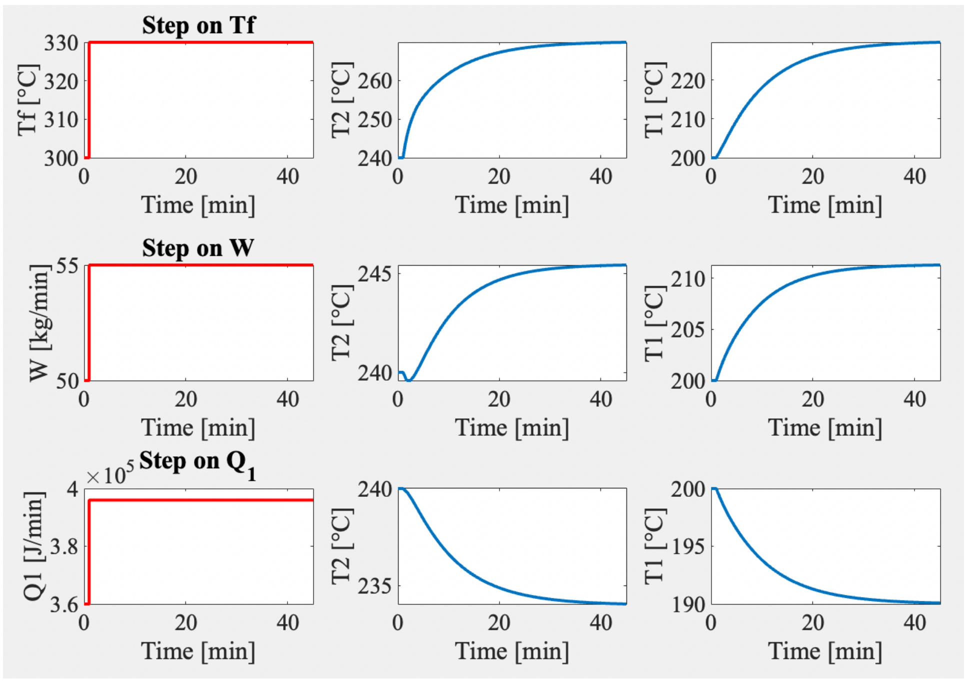


Figure 2: Simulation of nonlinear model

1.4 Steady state gains

The steady state values of the output after every step increase have been calculated from the plots. The gain can be defined as the ratio of the change in the output over the change in input responsible for this change in output.

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

The step changes on T_f, W and Q_1 are visible in Table 2. The step change equals 10% of the nominal value of the variables.

Table 2: Steady state gain after the step changes

Step on	***Step change value***	***Steady State Gain T_1***	***Steady State Gain T_2***
T_f	*30°C*	*1*	*1*
W	*5 kg/min*	*2.2614 °C*min/kg*	1.0828 °C*min/kg
Q_1	*36 kJ/min*	*$-1.6486*10^{-5}$ °C*min/J*	$2.7777*10^{-5}$ °C*min/J

1.5 Discussion

The system is stable and reaches a steady state after a step change has been applied. Steps on the different parameters (T_f, W, Q_1) give entirely different results. Every step increase will be discussed here. The plots are displayed in Figure 2 above. 45 minutes was used as the simulation time for every variable to get a clear view of as to when the value stabilized.

In general, every output has a slight time delay starting at t=0. This is caused by liquid in tank 2 needing to be flushed out of the system before a change in output temperature can be noticed, as every step increase in variables is first introduced in tank 1.

The plot for a step on T_f (disturbance) is the easiest to interpret. An increase in temperature of the feed will obviously lead to an overall increase in temperature of the system as the liquid will quite quickly heat up both tanks, and thus also the output temperature (T_2). Keep in mind that Q_1 stays the same and thus can't compensate for this added heat. The increased temperature already transfers through the heat exchange system of Q_2 and can heat up tank 2 so this change is noticed the quickest in the output. The effect of a step on T_f on T_1 requires a similar explanation; a temperature in T_f will lead to an increase in T_{f1}, which then in turn increases the temperature in tank 1. Due to the distance between the variable with a step change and the output value, a time delay was expected. This is also visible in the plots provided in Figure 2.

The increase in mass flow (W, disturbance) gives quite an unusual response. On the plot of the output is visible that temperature T_2 will first give an inverse response, and thus go down, before steadily increasing again. This can be explained as follows: since we assumed the masses in the tanks to be constant, a faster feed flow will mean an increase in the flow of the entire system immediately. Before the extra liquid from the increased mass flow enters tank 1, tank 1 also will have an increased flow already. Since this liquid will have had more time to cool off by absorbing Q_1 than it will have time to heat up again in tank 2 by Q_2 due to the faster flow, the output temperature T_2 will be lower until all that liquid is flushed out of the system. After the small slump visible in the plot, we can see a steady rise in temperature before stabilizing again. The reason for this is that, after the colder liquid has left the system, the incoming liquid will be leaving tank 1 quicker which means it has less time to cool down before going down to tank 2. Since Q_2 remains the same, this means an increase in the output temperature T_2.

A step on Q_1 (input) on the other hand is, much like T_f, not too difficult to explain. We must keep in mind that Q_1 is heat being *absorbed* from the tank, not transferred, so the graph can be misleading seeing as there is a minus in the equation and must be interpreted as a decrease. A step on Q_1 therefore means more heat taken away, not the other way around. Therefore, a decrease in output T_2 makes sense.

Overall, the simulation was successful. The results, in physical terms, make sense. They allow logical interpretation and analysis.

2 Part 2: System Analysis

In this part of the project an analysis of the system will be performed, starting with linearization, followed by transfer functions, after which a comparison will be made of the nonlinear and found linearized responses. The half-rule approximation will be used to approximate the transfer functions to a first order delay model.

2.1 Linearization

Linearize the nonlinear model to get the standard state-space form:

$$x' = Ax + Bu + Ed \tag{3a}$$

$$y = Cx + Du + Wd \tag{3b}$$

Vectors x, u, d, y were defined as follows:

$$x = [\Delta\, x_1\, \Delta\, x_2]^T$$
$$u = [\Delta u_1\, \Delta\, u_2]^T$$
$$d = [\Delta\, d_1\, \Delta\, d_2]^T$$
$$y = [\Delta\, x_1\, \Delta\, x_2]^T$$

Δx_1 refers to a deviation variable defined as $x_1(t)$ - x_1^*. Other variables were defined similarly.

Which gives:

$$x' = \begin{bmatrix} \frac{d\Delta T_1}{dt} \\ \frac{d\Delta T_2}{dt} \end{bmatrix}$$

$$x = \begin{bmatrix} \Delta T_1 \\ \Delta T_2 \end{bmatrix} = y$$

$$u = [\Delta Q_1]$$

$$d = \begin{bmatrix} \Delta Tf \\ \Delta w \end{bmatrix}$$

Included below are the important steps of the linearization.

T_1 calculations

$$\frac{dT_1}{dt} = \frac{W}{M_1}(T_{f_1} - T_1) - \frac{Q_1}{C_p \cdot M_1} = f_1$$

$$= \frac{W}{M_1}T_{f_1} - \frac{W}{M_1}T_1 - \frac{Q_1}{C_p \cdot M_1}$$

$$= \frac{W}{M} \cdot \left[\frac{2W \cdot C_p \cdot T_f - UA \cdot T_f + 2T_2 \cdot UA}{UA + 2W \cdot C_p} \right] - \frac{W}{M_1}T_1 - \frac{Q_1}{C_p \cdot M_1}$$

$$\Rightarrow \frac{d\Delta T_1}{dt} \simeq \left.\frac{\partial f_1}{\partial W}\right|_* \Delta W + \left.\frac{\partial f_1}{\partial T_f}\right|_* \Delta T_f + \left.\frac{\partial f_1}{\partial T_2}\right|_* \Delta T_2 + \left.\frac{\partial f_1}{\partial T_1}\right|_* \Delta T_1 + \left.\frac{\partial f_1}{\partial Q_1}\right|_* \Delta Q_1$$

$$\left.\frac{\partial f_1}{\partial W}\right|_* = \frac{4(UA)^2}{(2C_pW^* + UA)^2} \cdot \frac{T_2^*}{M_1} + \frac{(4C_pW^* - UA)(2C_pW^* + UA) - 2C_p(2C_pW^{*2} - UA)}{(2C_pW^* + UA)^2} - \frac{T_1^*}{M_1}$$

$$= 0.347$$

$$\left.\frac{\partial f_1}{\partial T_f}\right|_* = \frac{W^*}{M_1}\left(\frac{2 \cdot C_p \cdot W^* - UA}{2 \cdot W^* \cdot C_p + UA}\right) = 1/15$$

$$\left.\frac{\partial f_1}{\partial T_2}\right|_* = \frac{W^*}{M_1}\left(\frac{2 \cdot UA}{UA + 2 \cdot W^* \cdot C_p}\right) = 2/15$$

$$\left.\frac{\partial f_1}{\partial T_1}\right|_* = -\frac{W^*}{M_1} = 1/5$$

$$\left.\frac{\partial f_1}{\partial Q_1}\right|_* = -\frac{1}{C_pM_1} = -10/3 \cdot 10^{-5}$$

So: $$\frac{d\Delta T_1}{dt} \simeq 0.347\,\Delta W + 1/15\,\Delta T_f + 2/15\,\Delta T_2 - 10/3 \cdot 10^{-5}\,\Delta Q_1$$

T_2 calculations

$$\frac{dT_2}{dt} = \frac{W}{M_2}(T_1 - T_2) + \frac{UA}{\cancel{C_p} M_2} \cdot \frac{2WC\cancel{p}}{2 \cdot W \cdot C_p + UA}(T_f - T_2)$$

$$= \frac{1}{M_2}(T_1 - T_2)W + \frac{UA}{M_2} \cdot \frac{2W}{2C_p \cdot W + UA}\,(T_f - T_2)$$

$$\frac{d\Delta T_2}{dt} \simeq \left.\frac{df_2}{dW}\right|_* \Delta W + \left.\frac{df_2}{dT_1}\right|_* \Delta T_1 + \left.\frac{df_2}{dT_2}\right|_* \Delta T_2 + \left.\frac{df_2}{dT_f}\right|_* \Delta T_f$$

$$\left.\frac{df_2}{dW}\right|_* = \frac{1}{M_2}(T_1^* - T_2^*) + \frac{UA}{M_2} \cdot \frac{2(2C_pW^* + UA) - 2W^* \cdot 2C_p}{(2C_pW^* + UA)^2}\,(T_f^* - T_2^*)$$

$$= \frac{1}{M_2}(T_1^* - T_2^*) + \frac{UA}{M_2} \cdot \frac{\cancel{4C_pW^*} + 2UA - \cancel{4C_pW^*}}{(2C_pW^* + UA)^2}\,(T_f^* - T_2^*)$$

$$= \frac{1}{M_2}(T_1^* - T_2^*) + \frac{UA}{M_2}\,\frac{2UA}{(2C_pW^* + UA)^2}\,(T_f^* - T_2^*)$$

$$= \frac{1}{M_2}(T_1^* - T_2^*) + \frac{2(UA)^2}{M_2(2C_pW^* + UA)^2}(T_f^* - T_2^*) = -\frac{4}{15}$$

$$\left.\frac{df_2}{dT_1}\right|_* = \frac{W^*}{M_2} = \frac{1}{2}$$

$$\left.\frac{df_2}{dT_f}\right|_* = \frac{UA}{M_2} \cdot \frac{2W^*}{2C_pW^* + UA} = \frac{1}{3}$$

$$\left.\frac{df_2}{dT_2}\right|_* = -\frac{W^*}{M_2} - \frac{UA}{M_2} \cdot \frac{2W^*}{2C_pW^* + UA}$$

$$= -\frac{W^*}{M_2}\left(1 + \frac{2UA}{2C_p \cdot W^* + UA}\right) = -\frac{5}{6}$$

So: $$\frac{d\Delta T_2}{dt} = -\frac{4}{15}\Delta W + \frac{1}{2}\Delta T_1 + \frac{1}{3}\Delta T_f - \frac{5}{6}\Delta T_2$$

The following matrices were obtained.

$$A = \begin{bmatrix} -0{,}2000 & 0{,}1333 \\ 0{,}5000 & -0{,}8333 \end{bmatrix} \qquad B = \begin{bmatrix} -3{,}33 \cdot 10^{-5} \\ 0 \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$E = \begin{bmatrix} 0{,}0667 & 0{,}3467 \\ 0{,}3333 & -0{,}2667 \end{bmatrix}$$

$$D = 0, \quad W = 0$$

2.2 Transfer functions

Laplace transformation was applied to the functions, after which the transfer function matrices were obtained (G and G_d: for inputs and disturbances). The transfer functions are written in a gain-time constant format, so that the time constants τ are easily identifiable.

Laplace transform was calculated as follows:

Laplace transform

$$\mathcal{L}\left\{\frac{dT_1}{dt}\right\} = \mathcal{L}\left\{A_{11}\Delta T_1 + A_{12}\Delta T_2 + E_{11}\Delta T_f + E_{12}\Delta W + B_1 \Delta Q\right\}$$

$$\Rightarrow \quad sT_1(s) = A_{11}T_1(s) + A_{12}T_2(s) + E_{11}T_f(s) + E_{12}W(s) + B_1 Q(s)$$

$$\Leftrightarrow \quad T_1(s) = \frac{1}{s-A_{11}}\left(A_{12}T_2(s) + E_{11}T_f(s) + E_{12}W(s) + B_1 Q(s)\right)$$

$$\mathcal{L}\left\{\frac{dT_2}{dt}\right\} = \mathcal{L}\left\{A_{21}\Delta T_1 + A_{22}\Delta T_2 + E_{21}\Delta T_f + E_{22}\Delta W\right\}$$

$$\Rightarrow \quad sT_2(s) = A_{21}T_1(s) + A_{22}T_2(s) + E_{21}T_f(s) + E_{22}W(s)$$

$$\Leftrightarrow \quad T_2(s) = \frac{1}{s-A_{22}}\left(A_{21}T_1(s) + E_{21}T_f(s) + E_{22}W(s)\right)$$

After that, the equations were used in a series of substitutions.

1. $T_1(s)$ in $T_2(s)$

$$T_2(s) = \frac{1}{s-A_{22}}\left(A_{21}\left[\frac{1}{s-A_{11}}\left(A_{12}T_2(s) + E_{11}T_f(s) + E_{12}W(s) + B_1 Q(s)\right)\right] + E_{21}T_f(s) + E_{22}W(s)\right)$$

2. $T_2(s)$ in $T_1(s)$

$$T_1(s) = \frac{1}{s-A_{11}}\left(A_{12}\left[\frac{1}{s-A_{22}}\left(A_{21}T_1(s) + E_{21}T_f(s) + E_{22}W(s)\right)\right] + E_{11}T_f(s) + E_{12}W(s) + B_1 Q(s)\right)$$

Matlab was used to insert and calculate the resulting transfer functions, given below.

$$G = \begin{bmatrix} \dfrac{-0{,}00027778\,(1+1{,}2s)}{(1+1{,}081s)*(1+9{,}253s)} \\ \dfrac{-0{,}00016667}{(1+1{,}081s)(1+9{,}253s)} \end{bmatrix}$$

$$G_d = \begin{bmatrix} \dfrac{1+0{,}6667s}{(1+1{,}081s)(1+9{,}253s)} & \dfrac{2{,}5333\,(1+1{,}368s)}{(1+1{,}081s)(1+9{,}253s)} \\ \dfrac{1+3{,}333s}{(1+1{,}081s)(1+9{,}253s)} & \dfrac{1{,}2\,(1-2{,}222s)}{(1+1{,}081s)(1+9{,}253s)} \end{bmatrix}$$

2.2.1 Zeros, poles and gains

For every transfer function the zeros, poles and gains are displayed in Table 3.

Table 3: Zeros, poles and gains.

Transfer function		ZEROS	POLES	GAIN
$g_{Q1(s)}$ T1	$\frac{-0{,}00027778\,(1+1{,}2s)}{(1+1{,}081s)*(1+9{,}253s)}$	$\frac{-1}{1{,}2}$	$\frac{-1}{1{,}081}$ $\frac{-1}{9{,}253}$	$-0{,}00027778\,\frac{°C\ min}{J}$
$g_{Q1(s)}$ T2	$\frac{-0{,}00016667}{(1+1{,}081s)(1+9{,}253s)}$	/	$\frac{-1}{1{,}081}$ $\frac{-1}{9{,}253}$	$-0{,}00016667\,\frac{°C\ min}{J}$
$G_{d;Tf(s)}$ T1	$\frac{1+0{,}6667s}{(1+1{,}081s)(1+9{,}253s)}$	$\frac{-1}{0{,}6667}$	$\frac{-1}{1{,}081}$ $\frac{-1}{9{,}253}$	1
$g_{d;Tf(s)}$ T2	$\frac{1+3{,}333s}{(1+1{,}081s)(1+9{,}253s)}$	$\frac{-1}{3{,}333}$	$\frac{-1}{1{,}081}$ $\frac{-1}{9{,}253}$	1
$g_{d;w(s)}$ T1	$\frac{2{,}5333\,(1+1{,}368s)}{(1+1{,}081s)(1+9{,}253s)}$	$\frac{-1}{1{,}368}$	$\frac{-1}{1{,}081}$ $\frac{-1}{9{,}253}$	$2{,}5333\,\frac{°C\ min}{kg}$
$g_{d;w(s)}$ T2	$\frac{1{,}2\,(1-2{,}222s)}{(1+1{,}081s)(1+9{,}253s)}$	$\frac{1}{2{,}222}$	$\frac{-1}{1{,}081}$ $\frac{-1}{9{,}253}$	$1{,}2\,\frac{°C\ min}{kg}$

2.3 Comparison of nonlinear and linearized responses

Simulation of the linearized model and its responses compared to the nonlinear responses (Figure 3). Only the linearized model of the step on W shows a slight deviation from the nonlinear response. For T_2 the linear model starts below its nonlinear counterpart but has a higher steady state value (by a very small margin). For T_1, the linearized model starts and stays higher than the nonlinear model. The gain is higher than for the nonlinear model.

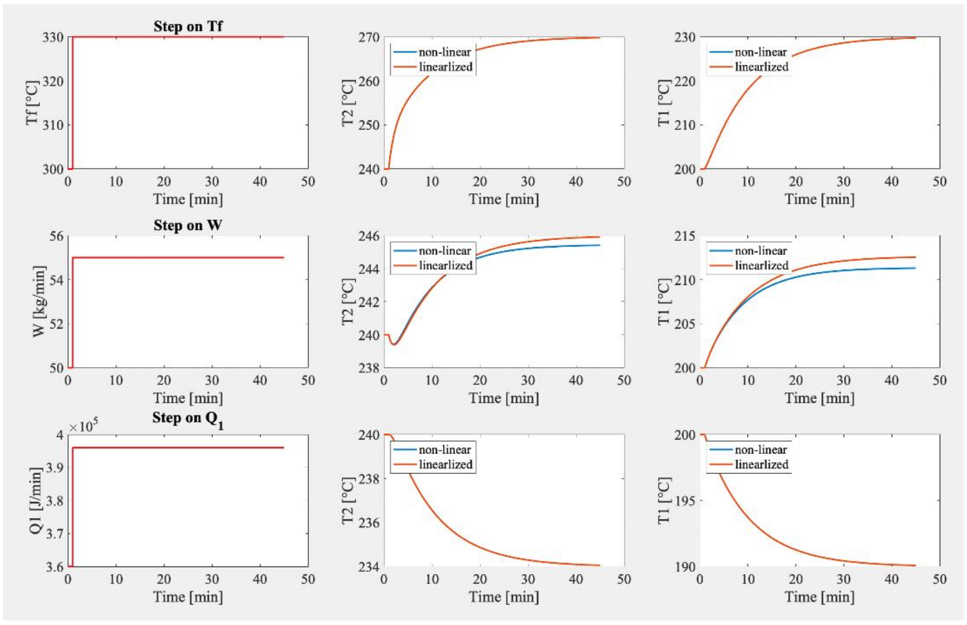


Figure 3: Simulation of linearized model compared to non-linear model. In a step on T_f and Q_1, the non-linearized models and linearized models overlap.

2.4 Half-rule approximation

The transfer functions $g_{ij}(s)$ have been approximated in the transfer function matrix (G) to a first order plus delay model using the half-rule.

For the transfer function from Q_1 to T_1, the factors (1 + 1.2s) and (1 + 1.081s) are small compared to (1 + 9.253s), so this ratio will be approximated as 1. Therefore, a first-order model without delay is achieved. The half-rule approximation doesn't have to be applied in this case.

Half rule

$$G1HF\colon \frac{-0{,}00027778}{(1+9{,}253s)}$$

$$G2HF\colon \frac{-0{,}00016667e^{-0{,}5405s}}{(1+9{,}7935s)} \quad with\ \ \theta = 0{,}5405\ and\ \tau = 9{,}7935$$

Transfer function and half-rule approximation are plotted and compared for Q_1 to T_1 (Figure 4) and Q_1 to T_2 (Figure 5).

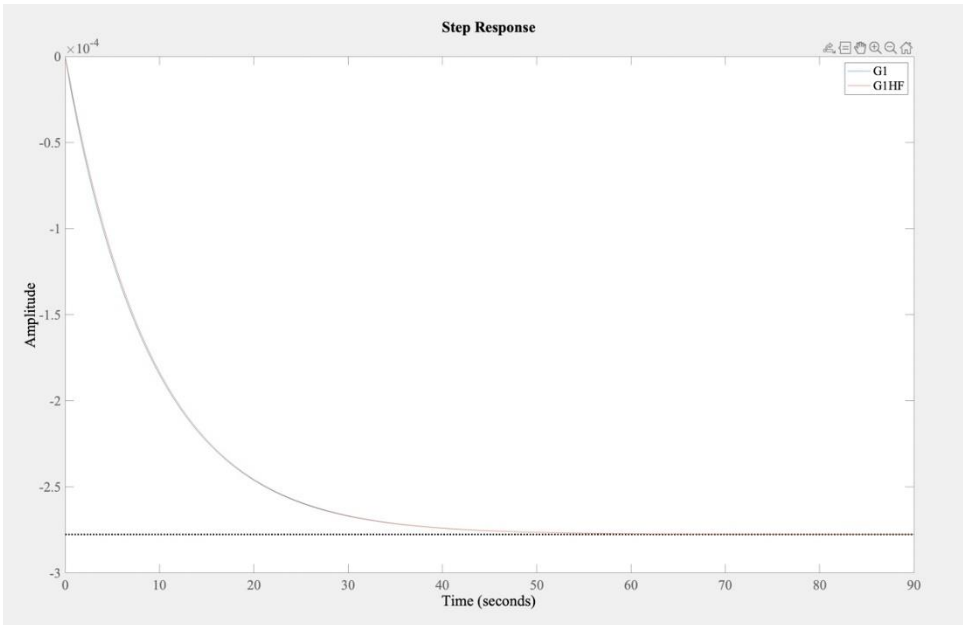


Figure 4: Unit step to transfer function G_1 with half-rule approximation G_1HF.

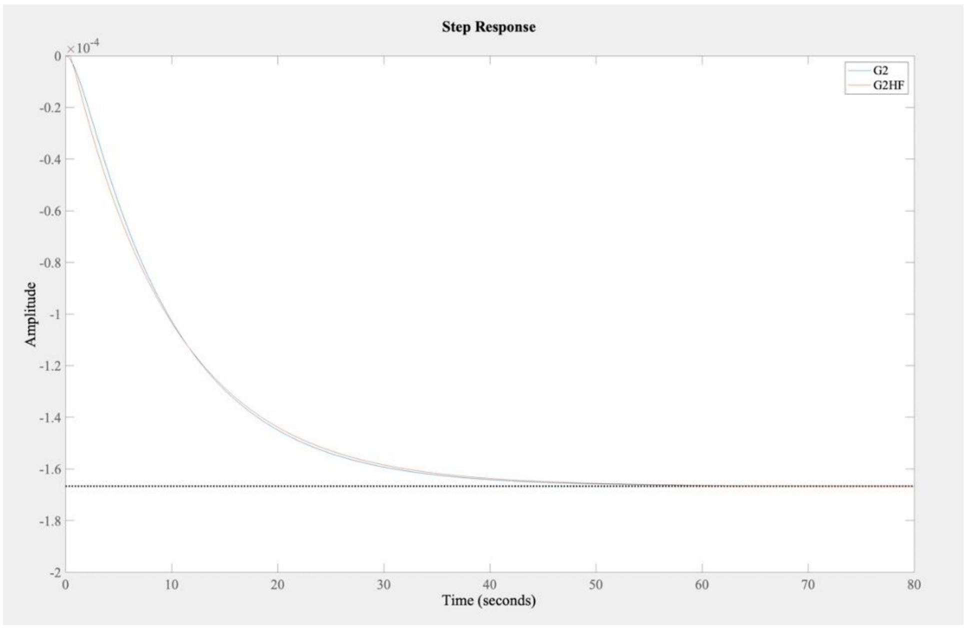


Figure 5: Unit step to transfer function G_2 with half-rule approximation G_2HF.

2.5 Discussion

2.5.1 Properties of the transfer functions

Zeros in a transfer function are responsible for the shape of the response. Only the transfer function for w(s) from T_2 has a zero in the right-hand plane (RHP). This implies an inverse response, which is problematic for feedback control. This is also visible in the simulated plots (Figure 3). All the poles are negative and real, which means they lie in the left-hand plane (LHP). This is a good thing, as poles in the RHP would give instability and if they were complex, they would cause oscillations. Once again, this is visible in Figure 3. Zeros in the left-hand plane (LHP) lift the response and often give on overshoot. This occurs when T (time constant in numerator) is greater than τ (time constant in denominator). The transfer functions with a non-zero T, have a T value smaller than τ_1, therefore no overshoot was expected. This wasn't observed and thus conforms to expectations.

All time constants are equal for the different transfer functions, therefore differences in dynamics are due to other gains and other time constants in the numerator. This is clearly visible for the step on T_f. The response on T_2 is faster than the one on T_1. The time constants in the numerator for T_2 is more than three times bigger than for T_1, which explains the obviously faster dynamics of T_2. Differences for the step on Q_1 are less pronounced. The response in T_1 is slightly faster. This is due to the time constant in the numerator of the transfer function for T_1. In the transfer function for T_2 there is no time constant in the numerator.

2.5.2 Linearization

Only the linearized model of the step on W shows a slight deviation from the nonlinear response. For T_2 the linear model starts below its nonlinear counterpart but has a higher steady state value (by a very small margin). For T_1, the linearized model starts and stays higher than the nonlinear model. The difference in steady state value shows that the nonlinear model is quite far off from linearity. If the step change on W was smaller, the gap between the steady state values would be less significant. On the other hand, if the step change was greater, so would the gap.

Overall, the linearized models compare quite well to their respective original models. This implies that the nonlinear models are not as far from linearity as anticipated.

2.5.3 Half-rule

For the transfer function from Q_1 to T_1, the transfer function G_1 and its half-rule approximation are very similar. They are barely distinguishable on the plot.

For the transfer function from Q_1 to T_2, the graphs look quite similar. The original transfer function is a second order function. The characteristic S-shape at the beginning of the response can be observed (Figure 5). No delay can be observed in this blue colored function. However, the half-rule model shows a slightly different response. First, the S-shape cannot be observed. As the half-rule model is a first order model, this sigmoid curve was not expected. Instead, a small time delay can be observed in the red colored function. This time delay was implanted to approximate the second order model. As expected, this half-rule model shows a first order response. The time constant is greater than the biggest time constant in the linearized model. The effect, a slower response after the time delay, can be observed. The blue line reaches the steady state value first (Figure 5). The initial value and steady state value of both models are equal. Initial slopes are not identical.

3 Part 3: Controller Design

This part is centered around creating different controller structures to compare their effectiveness in counteracting steps in disturbances and inputs and to eventually reach steady state.

3.1 Control structure

In this project two control structures were implemented. A simple feedback controller is compared to a more extensive control mechanism. Both g1 and. g2 have second order, slow dynamics. State T_1 is not linear for the disturbances. Especially for the variable *w*, there is a big gap between the linearized and non-linear model. Measuring temperatures is usually easy and fast, this makes T_1 a good extra variable or measurement to control T_2. Good control of the slave loop will improve the control of the outer loop. The disturbances arising in the slave loop will be corrected by the inner loop controller before they can influence T_2. Gain variations in g1, the transfer function of the process for T_1, are overcome within the slave loop and don't affect the outer loop. Furthermore, using a cascade would improve the speed of response of the outer loop. Feedforward is potentially very helpful if there is a delay in the measurement of the output. Here, we assume there is no delay as measuring temperatures is easy. Therefore, we implement feedback with cascade to compare this with the simple feedback model.

Given below are the are the feedback systems (simple feedback (Figure 6), series cascade (Figure 7)) with their respective block diagrams .

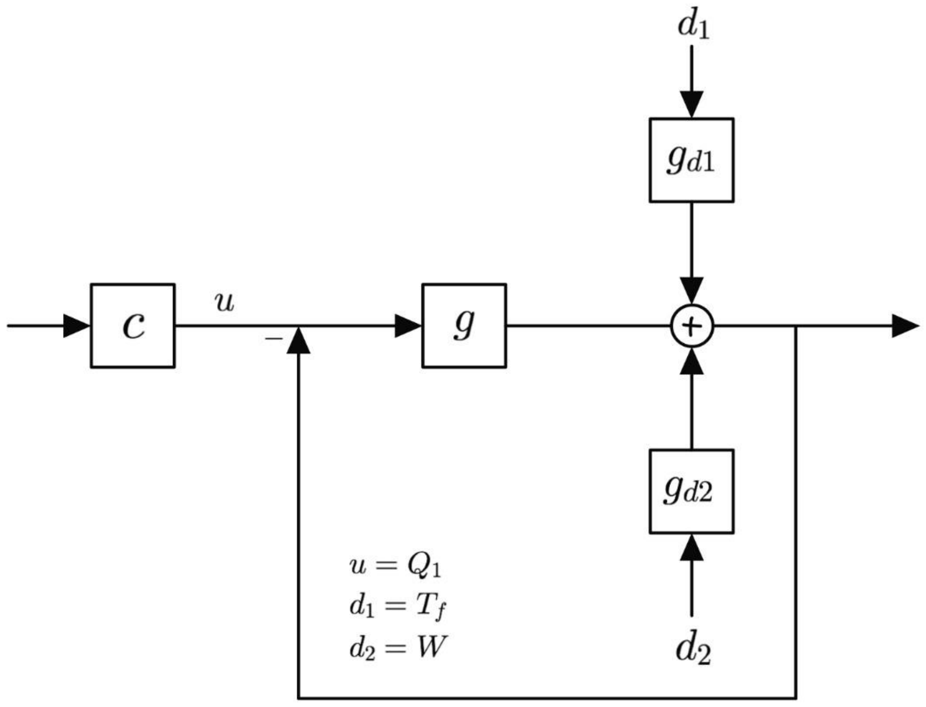


Figure 6: Simple feedback block diagram.

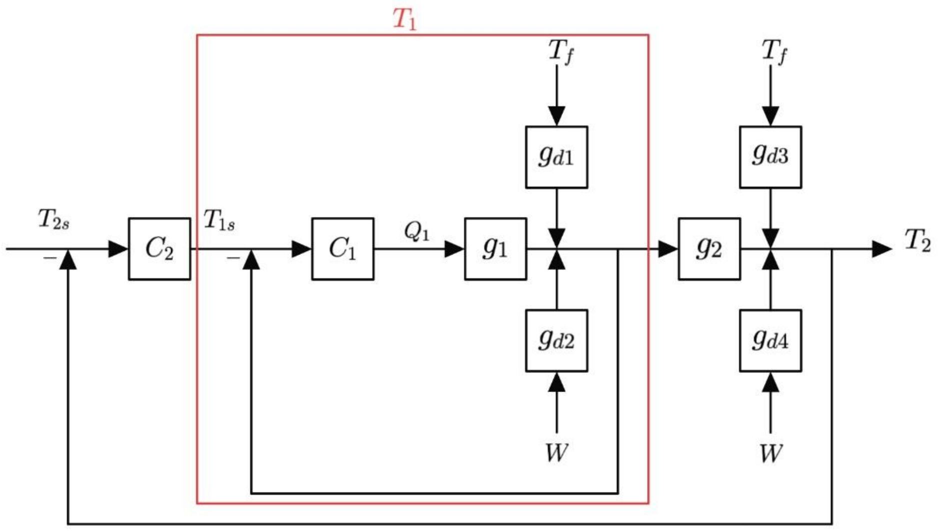
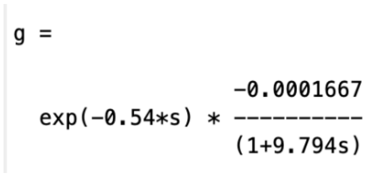


Figure 7: Series cascade block diagram

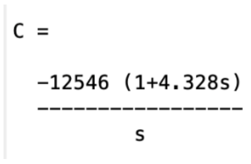
3.2 Controller tuning

For the normal feedback control, the PI controller was tuned using transfer function g. This is the transfer function from the input, Q_1, to the output, T_2. The controller was tuned using the SIMC rules with "tight control".

This gives a Kc = -5.4300e+04, and τ_I = 4.3280. This gives function C for the controller.



```
g =

                   -0.0001667
  exp(-0.54*s) * ----------
                   (1+9.794s)
```



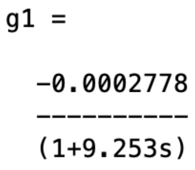
```
C =

  -12546 (1+4.328s)
  -----------------
          s
```

The transfer functions of the disturbances to output T_2 are given below.

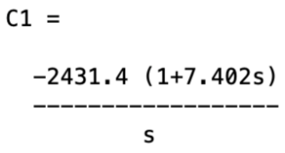
$$gd1 = \frac{1 + 3{,}333s}{(1 + 1{,}081s)(1 + 9{,}253s)} \qquad gd2 = \frac{1{,}2\,(1 - 2{,}222s)}{(1 + 1{,}081s)(1 + 9{,}253s)}$$

In the feedback loop with cascade, two controllers are used. C2 is the controller for the entire process, containing the primary and secondary loop. The slave loop is controlled by C1. In the slave loop, T_1 is controlled as an extra measurement to improve the control of T_2.

C1 is tuned using g1, the half rule approximation of the transfer function from Q_1 to T_1. Function g1 has a time delay of 0 seconds. τ = 9,253. Therefore, τ_C is set to $\tau/5$. This gives a τ_C value of 1,8506. Using the SIMC rules, this gives a Kc = -1,7999e+04 and a τ_I = 7,4024. The PI controller has function C1.



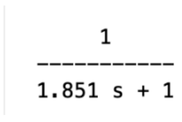
```
g1 =

  -0.0002778
  ----------
  (1+9.253s)
```



```
C1 =

  -2431.4 (1+7.402s)
  ------------------
          s
```

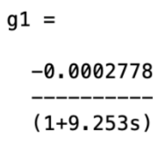
To tune controller 2, the transfer function for the 'new' process from T_{1S} to T_2 was obtained. This function, g2' was defined as $g_2' = L1*(g_2/g_1)$. Where L1 is the closed loop transfer function from T_{1S} to T_1. L1 was approximated to a first order with time delay process. This is used to tune controller 2. To approximate L1, function g1 was used. τ_C is set to $\tau/5$.

$$L1 \approx \frac{e^{-\theta s}}{\tau_c s + 1} =$$

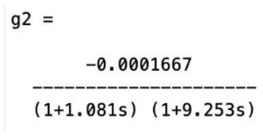


```
      1
 -----------
 1.851 s + 1
```

With g_1 and g_2,

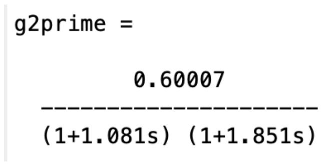


```
g1 =

  -0.0002778
  ----------
  (1+9.253s)
```

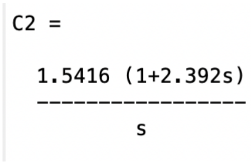


```
g2 =

      -0.0001667
  ---------------------
  (1+1.081s) (1+9.253s)
```

This gives a g_2' and C2 of



```
g2prime =

         0.60007
  ---------------------
  (1+1.081s) (1+1.851s)
```

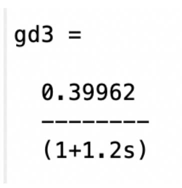


```
C2 =

  1.5416 (1+2.392s)
  -----------------
          s
```

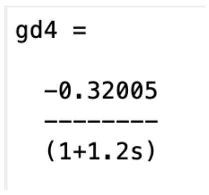
The transfer functions of the disturbances to T_1 are given below.

$$gd1 = \frac{1 + 0{,}6667s}{(1 + 1{,}081s)(1 + 9{,}253s)} \qquad gd2 = \frac{2{,}5333\,(1 + 1{,}368s)}{(1 + 1{,}081s)(1 + 9{,}253s)}$$

The transfer functions for the disturbances in the outer loop are given below.



```
gd3 =

  0.39962
  --------
  (1+1.2s)
```



```
gd4 =

  -0.32005
  --------
  (1+1.2s)
```

3.3 Simulations

Given below are the feedback systems (simple feedback (Figure 8), series cascade (Figure 9)) with their simulations. The closed-loop behavior was studied whilst applying step changes in the disturbances and set points. Just like in previous parts, the value was increased by 10% of the nominal value.

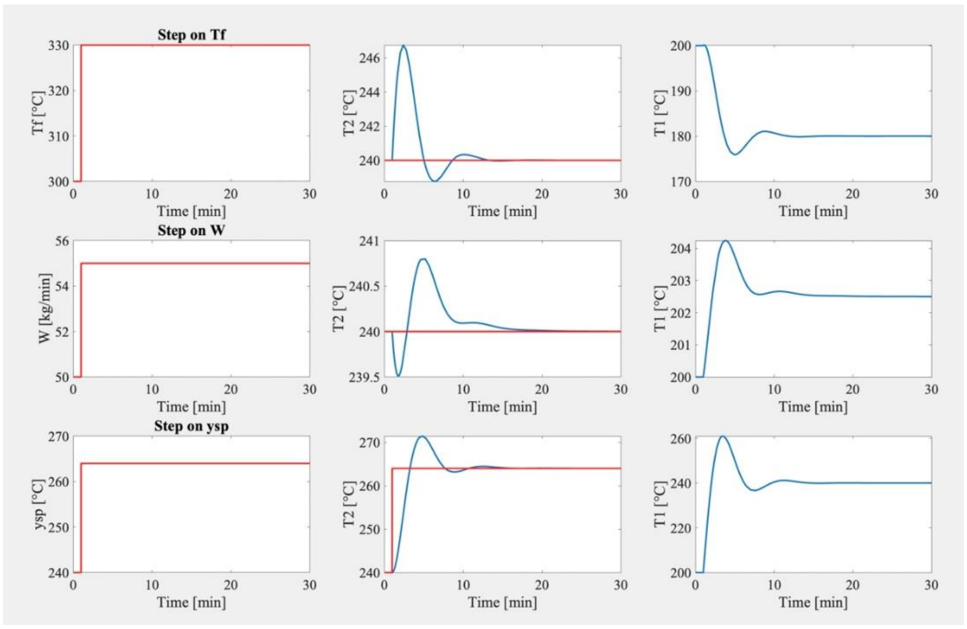


Figure 8: Simple feedback simulation.

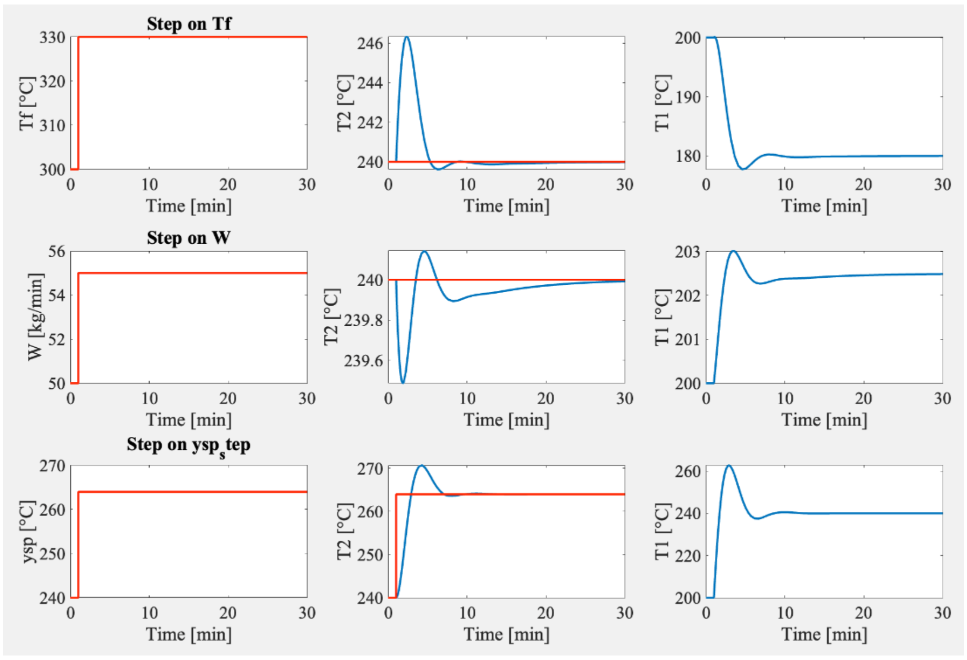


Figure 9: Series cascade simulation.

3.3.1 Integral Absolute Error

In order to analyze the implemented controls structures quantitatively, the IAE (Integral Absolute Error) performance index was calculated.

The IAE implementation in MATLAB returned a vector. The IAE corresponds to the last value of that vector, which is the integration of the error until the error is zero (when you reach the setpoint). A table of the IAE values is given below.

The IAE indicates the value of the amount of error present for each system. Therefore, the series cascade system corrects errors better, as there is a lower value of IAE.

Table 4: IAE of Simple and Cascade system

	Simple	**Cascade**
IAE (°C)	53,2684	42,4243

3.4 Bode plot

A Bode plot for the closed loop function of the simple feedback (Q_1 to T_2) was created (Figure 10). The slopes, poles and zeros, and phase margin (PM) are indicated on the plot.

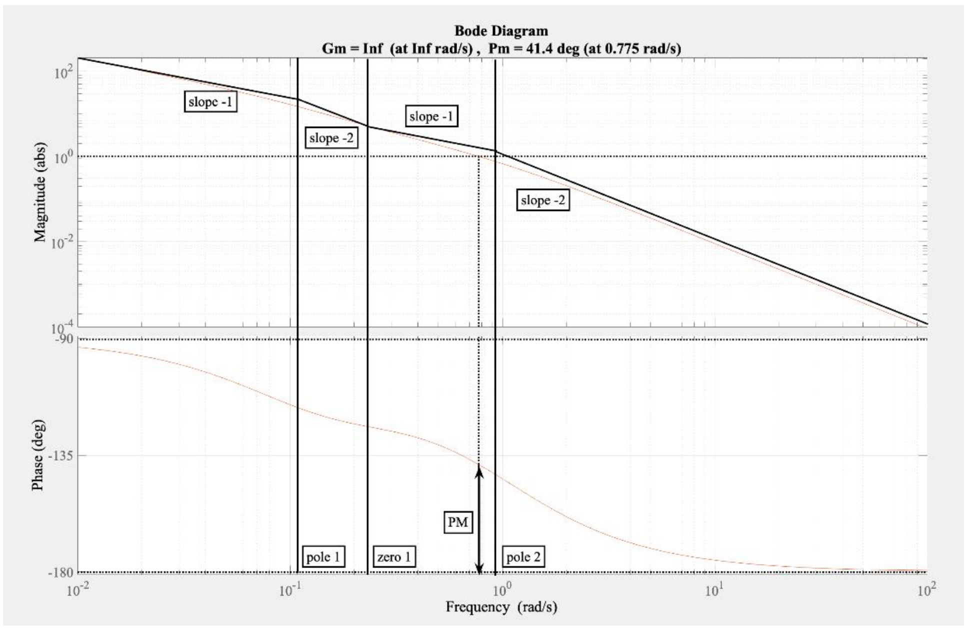


Figure 10: Bode Plot of the closed loop function (L) of the simple feedback

3.4.1 Margins

- Gain margin = Infinite
- Phase margin = 41,4 degrees = 0,723 rad
- Time delay margin = PM/ω_c = 0,723 rad/ 0,772 rad/s = 0,937 s

These margins were calculated from Figure 10. The phase margin is indicated on the plot. The gain margin is infinite and therefore not indicated on the plot.

3.5 Discussion

As visible in Figure 8 and 9 that display the simulation results for both feedback structures, the cascade controller is more robust than the simple feedback controller. This is because it performs better in the correction of the applied step changes in the set point and disturbances. Proof of this is the value of the IAE. The additional effectiveness is a result of the inclusion of the slave loop (which has faster control): before the disturbances within the loop influence output T_2, they are corrected by the slave controller. Gain variations in the control of T_1 are overcome within the slave loop, so they do not influence output T_2.

Overall, the quality of the control was good. Additional attention to the effect of W on T_2 might be required for better control. A good understanding of this disturbance is necessary. Adding the cascade to the feedback control already resulted in a better performance.

The gain margin is equal to infinity, which means the system will never become unstable, no matter how much the gain is increased. The phase margin is 0,723 rad. The grater the phase margin, the greater the stability of the system. The phase margin could be improved. A time delay margin of 0,937 s was obtained. This could also be improved and makes sense as time delay margin is a function of phase margin. In the future, a higher phase margin, and therefore a higher time delay margin, would be desired.

4 Final Discussion

In this project the process of heat exchange in two tanks was analyzed. The original nonlinear model was successfully simulated and showed logical results. To analyze the system, this model was than linearized, after which the transfer functions were calculated. Only for disturbance W, real difference between the nonlinear and linear model was observed. This indicates that the relation between both T_f and Q_1 with T_1 and T_2 is close to linear. The transfer functions were plotted and the expected trends were observed. The transfer functions from Q_1 to T_1 and T_2 were approximated by the half-rule. The obtained first order models did not differ much from the original functions.

A simple feedback was compared to feedback with cascade control structure. As expected the addition of the cascade improved control. This was confirmed by simulations and IAE values. From the bode plot however, it was concluded that the phase margin and the time delay margin could have been improved.

4.1 Additional comments

We would like to suggest that teaching assistants agree on how to correct, because we got into a time crunch due to the disagreement of two different assistants as our part 1 was corrected incorrectly. Because of this we had to include handwritten calculations, which unfortunately makes the report less “professional”.

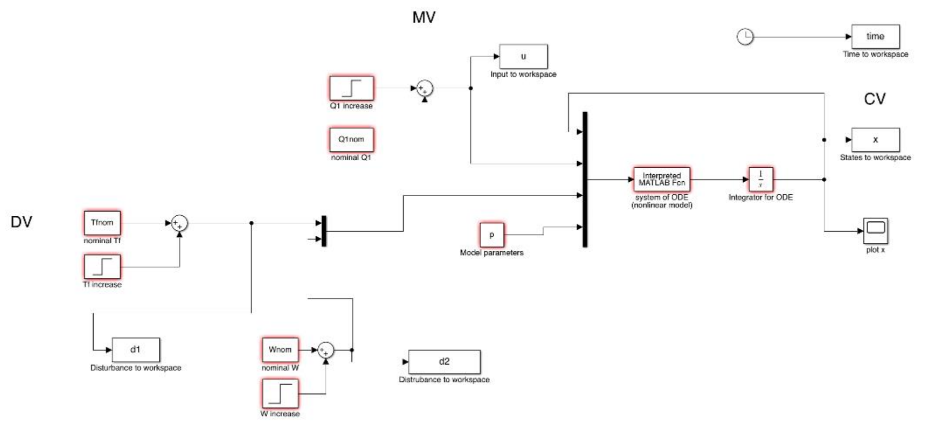
Bibliography

[1] S. Skogestad and C. Grimholt. The SIMC Method for Smooth PID Controller Tuning. In *PID Control in the Third Millennium*, pages 147– 175. Springer, 2012.

A Simulink

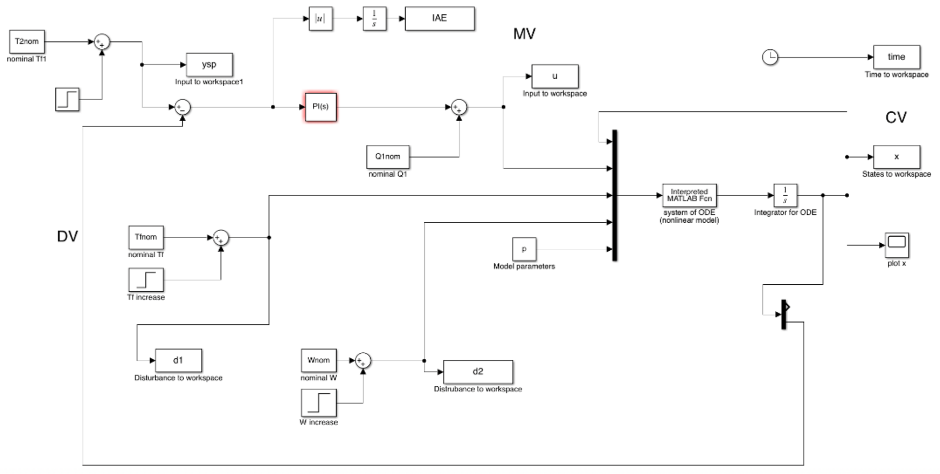
In this section, Simulink screenshots should be included.

A.1 Simulink Part 1

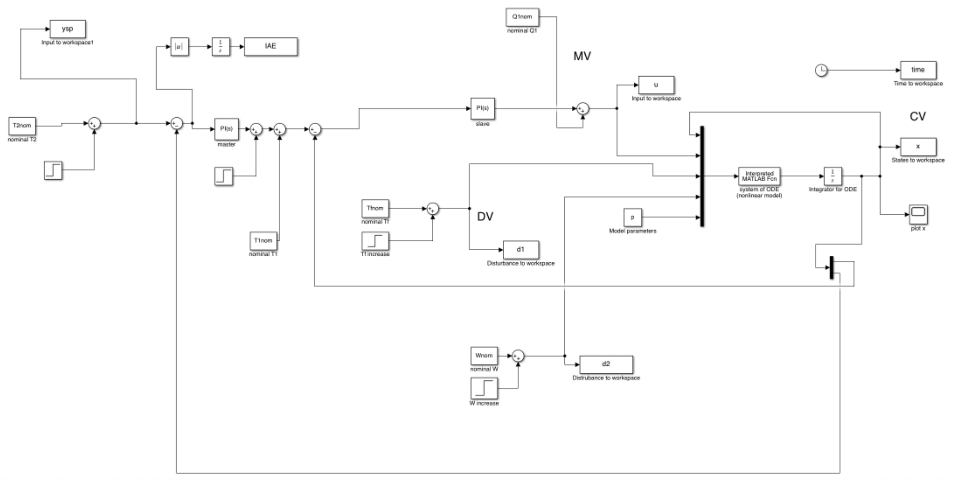


A.2 Simulink Part 3

Simple feedback



Cascade feedback



B Matlab codes

SCRIPT PART 1

```
% TKP4140 Process Control Project

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)

%% Initializing

%Model parameters

Cp=120; %[J/kg°C]
M1=250; %[kg]
M2=100; %[kg]
UA= 6000; % G = UA [J/min*°C]
%Tf1 = 260; %[°C]

p = [Cp; M1; M2; UA];

% Nominal  values for  Q1 and Tf and W.
Tfnom=300;
Wnom=50;
Q1nom=360000;

% Set steps to 0 - Operating at nominal point
Q1_step=0;
Tf_step=0;
W_step=0;

% Define the number of variales and model parameters
Nx = 2; %--number of states--NUMBER OF DIFFERENTIAL EQUATIONS
Nu = 1; %--number of inputs--NUMBER OF MVS
Nd = 2; %--number of disturbances--NUMBER OF DVS
Np = 4; %--number of model parameters--CONSTANT VALUES
Ny = Nx;%--number of outputs---equal to number of states

%INITIAL CONDITIONS FOR THE STATES
x0 = [200; 240];

%% Simulate a step increase on Tf (Disturbance)
% Set the value for Step on q0 (Disturbance)
Tf_step=(0.1*Tfnom);
W_step=0;
Q1_step=0;

% Set the time for giving the Step on Tf
Tf_t_step = 1;
W_t_step=1;
Q1_t_step = 1;

%run Simulink model to test the step in Tf
tsim = 45;                  % 30 minutes
sim('ProjectT_2')
```

```
%We are using subplots to plot the result.
figure(1)
subplot(331)
plot(time,d1,'red')                %the disturbance d1 corresponds to Tf
title('Step on Tf')
ylabel('Tf [°C]')
xlabel('Time [min]')


subplot(332)
plot(time,x(:,2))        % the state x corresponds to the output y = T2
ylabel('T2 [°C]')
xlabel('Time [min]')

subplot(333)
plot(time,x(:,1))        % the state x corresponds to the output y = T1
ylabel('T1 [°C]')
xlabel('Time [min]')

%Simulate a step increase on W (Disturbance)
Tf_step=0;
W_step=(0.1*Wnom);
Q1_step=0;

% Set the time for giving the Step on q0
Tf_t_step = 1;
W_t_step=1;
Q1_t_step = 1;

%run Simulink model to test the step in q0
tsim = 45;                  % 30 minutes
sim('ProjectT_2')

%We are using subplots to plot the result.
figure(1)
subplot(334)
plot(time,d2,'red')%the disturbance d2 corresponds to W
title('Step on W')
ylabel('W [kg/min]')
xlabel('Time [min]')


subplot(335)
plot(time,x(:,2))           % the state x corresponds to the output y = T2
ylabel('T2 [°C]')
xlabel('Time [min]')

subplot(336)
plot(time,x(:,1))        % the state x corresponds to the output y = T1
ylabel('T1 [°C]')
xlabel('Time [min]')


%% Simulate a step increase on q1 (Manipulated variable)
Q1_step=(0.1*Q1nom);
Tf_step=0;
W_step=0;

% Run Simulink model to test the effect of a step in Q1.
sim('ProjectT_2')

figure(1)
```

```
subplot(337)
plot(time,u,'red')
title('Step on Q_1')
ylabel('Q1 [J/min]')
xlabel('Time [min]')


subplot(338)
plot(time,x(:,2))
ylabel('T2 [°C]')
xlabel('Time [min]')

subplot(339)
plot(time,x(:,1))        % the state x corresponds to the output y = T1
ylabel('T1 [°C]')
xlabel('Time [min]')

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

set(gcf,'Position',[100 100 750 500])              % 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','ExamplePart1.eps') % save eps.
```

SCRIPT PART 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

% Hint:
% If you have more than one MV, CV, or DV, be careful with the order of the
% states, inputs, and disturbances when connecting the Simulink diagram.
% The order must correspond to the order you use in the
% computeTransferFunctions function. It is also very recommendable to be
% consistent with the order you used in the nonlinear model in part 1. In
% other words, if h was x1 and T was x2, you should always keep it so, and
% don't use T as x1 h as x2.

%Model parameters
Cp = 120; % [J/kg*°C]
M1 = 250; % [kg]
M2 = 100; % [kg]
UA = 6000; % [J/min*°C]
Tf1 = 260;

p = [Cp; M1; M2; UA; Tf1]; % vertical vector with model parameters. This is used in↙
Simulink

% Nominal  values for  Q1 and Tf and W.
Tfnom=300;
Wnom=50;
Q1nom=360000;

% Set steps to 0 - Operating at nominal point
Q1_step=0;
Tf_step=0;
W_step=0;

% Define the number of variales and model parameters
Nx = 2; %---CHANGE HERE----- number of  states----NUMBER OF DIFFERENTIAL↙
EQUATIONS------
Nu = 1; %---CHANGE HERE----- number of inputs-----NUMBER OF MVS------
Nd = 2; %---CHANGE HERE----- number of disturbances----NUMBER OF DVS------
Np = 5;  %---CHANGE HERE-----number of model parameters------CONSTANT VALUES------
Ny = Nx; % Do  not change----number of outputs---The number of outputs=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.
%Remember that all states are "x"
x0 = [200; 240]; % x0 is vertical; e.g. 2 states: x0 = [0; 50];

%% Check what happens when there is a disturbance
% Set the value for Step on q0 (Disturbance)
Tf_step=(0.1*Tfnom);
W_step=0;
Q1_step=0;

% Set the time for giving the Step on q0
Tf_t_step = 1;
W_t_step=1;
Q1_t_step = 1;

% Tip to check your model: If you set q_0 step=0 (no disturbance); and run the model,
%  h should remain constant. If with no step in the input (MV) or your
%  disturbance (DV) your output (CV) changes, then there's a problem with
%  your model. Check your equations and the initial values (x0 in system).

% Tip 2: If you have several disturbances, you have simulate one disturbance at a↙
time.
% To do this: Set q0_step=0,
% then set the step for the second disturbance, (e.g. d2_step=1)
% Then do a new simulation

%run Simulink model to test the step in q0
tsim = 45; % set simulation time. same units as your model
sim('SimulinkPart2T')

%We are using subplots to plot the result.
figure(1)
subplot(331)
plot(time,d1,'red')
title('Step on Tf')
ylabel('Tf [°C]')
xlabel('Time [min]')

subplot(332)
plot(time,x(:,2))
hold on
plot(time,ylinear(:,2),'-')
legend('non-linear','linearlized','Location','northwest')
ylabel('T2 [°C]')
xlabel('Time [min]')

subplot(333)
plot(time,x(:,1))
hold on
plot(time,ylinear(:,1),'-')
legend('non-linear','linearlized','Location','northwest')
ylabel('T1 [°C]')
xlabel('Time [min]')
%%
%Step on W (Manipulated variable)
Tf_step=0;
W_step=(0.1*Wnom);
Q1_step=0;
```

```
Tf_t_step = 1;
W_t_step=1;
Q1_t_step = 1;

%run Simulink model to test the effect of a step in W.
sim('SimulinkPart2T')

figure(1)
subplot(334)
plot(time,d2,'red') %the disturbance d2 corresponds to W
title('Step on W')
ylabel('W [kg/min]')
xlabel('Time [min]')

subplot(335)
plot(time,x(:,2))
hold on
plot(time,ylinear(:,2),'-')
legend('non-linear','linearlized','Location','northwest')
ylabel('T2 [°C]')
xlabel('Time [min]')

subplot(336)
plot(time,x(:,1))
hold on
plot(time,ylinear(:,1),'-')
legend('non-linear','linearlized','Location','northwest')
ylabel('T1 [°C]')
xlabel('Time [min]')

%% Simulate a step increase on Q1 (Manipulated variable)
Q1_step=(0.1*Q1nom);
Tf_step=0;
W_step=0;

% Set the time for giving the Step on Q1
Tf_t_step = 1;
W_t_step=1;
Q1_t_step = 1;


%run Simulink model to test the step in q0
tsim = 45;                   % 30 minutes
sim('SimulinkPart2T')

figure(1)
subplot(337)
plot(time,u,'red')
title('Step on Q_1')
ylabel('Q1 [J/min]')
xlabel('Time [min]')

subplot(338)
plot(time,x(:,2))
hold on
plot(time,ylinear(:,2),'-')
legend('non-linear','linearlized','Location','northwest')
ylabel('T2 [°C]')
xlabel('Time [min]')

subplot(339)
plot(time,x(:,1))
```

```
hold on
plot(time,ylinear(:,1),'-')
legend('non-linear','linearlized','Location','northwest')
ylabel('T1 [°C]')
xlabel('Time [min]')

%% Export figure in .eps format.
print('-depsc2','-r600','ExamplePart2.eps') % save figure as eps
```

SCRIPT PART 3 - SIMPEL FEEDBACK

```
% TKP4140 Process Control Project

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)

%% SIMC

% Normal Feedback - T2
tau = 9.794;
theta = 0.541;
tauI=4.328;
k = -0.0001667;
kc= (1/k)*(tau/(2*theta));
s=tf('s');
C = (kc*((tauI*s)+1)/(tauI*s));
g = (-0.0001667*exp(-0.541*s))/((9.794*s)+1);

%% Initializing

%Model parameters

Cp=120; %[J/kg°C]
M1=250; %[kg]
M2=100; %[kg]
UA= 6000; % G = UA [J/min*°C]
%Tf1 = 260; %[°C]

p = [Cp; M1; M2; UA];

% Nominal  values for  Q1 and Tf and W.
Tfnom=300;
Wnom=50;
Q1nom=360000;

% Set steps to 0 - Operating at nominal point
Q1_step=0;
Tf_step=0;
W_step=0;

% Define the number of variales and model parameters
Nx = 2; %--number of states--NUMBER OF DIFFERENTIAL EQUATIONS
Nu = 1; %--number of inputs--NUMBER OF MVS
Nd = 2; %--number of disturbances--NUMBER OF DVS
Np = 4; %--number of model parameters--CONSTANT VALUES
Ny = Nx;%--number of outputs---equal to number of states

%INITIAL CONDITIONS FOR THE STATES
x0 = [200; 240];
T2nom= 240;
T1nom=200;

%% Simulate a step increase on Tf (Disturbance)
% Set the value for Step on q0 (Disturbance)
```

```
Tf_step=(0.1*Tfnom);
W_step=0;
ysp_step=0;

% Set the time for giving the Step on Tf
Tf_t_step = 1;
W_t_step=1;
ysp_t_step = 1;

%run Simulink model to test the step in Tf
tsim = 30;                  % 30 minutes
sim('ProjectTdownload')

%We are using subplots to plot the result.
figure(1)
subplot(331)
plot(time,d1,'red')              %the disturbance d1 corresponds to Tf
title('Step on Tf')
ylabel('Tf [°C]')
xlabel('Time [min]')


subplot(332)
plot(time,x(:,2))        % the state x corresponds to the output y = T2
hold on
plot(time,ysp,'red')
ylabel('T2 [°C]')
xlabel('Time [min]')

subplot(333)
plot(time,x(:,1))        % the state x corresponds to the output y = T1
ylabel('T1 [°C]')
xlabel('Time [min]')

%Simulate a step increase on W (Disturbance)
Tf_step=0;
W_step=(0.1*Wnom);
ysp_step=0;

% Set the time for giving the Step on q0
Tf_t_step = 1;
W_t_step=1;
ysp_t_step = 1;

%run Simulink model to test the step in q0
tsim = 30;                  % 30 minutes
sim('ProjectTdownload')

%We are using subplots to plot the result.
figure(1)
subplot(334)
plot(time,d2,'red') %the disturbance d2 corresponds to W
title('Step on W')
ylabel('W [kg/min]')
xlabel('Time [min]')


subplot(335)
plot(time,x(:,2))           % the state x corresponds to the output y = T2
hold on
plot(time,ysp,'red')
ylabel('T2 [°C]')
```

```
xlabel('Time [min]')

subplot(336)
plot(time,x(:,1))        % the state x corresponds to the output y = T1
ylabel('T1 [°C]')
xlabel('Time [min]')


%% Simulate a step increase on q1 (Manipulated variable)
ysp_step=(0.1*T2nom);
Tf_step=0;
W_step=0;

% Run Simulink model to test the effect of a step in Q1.
sim('ProjectTdownload')

figure(1)
subplot(337)
plot(time,ysp,'red')
title('Step on ysp')
ylabel('ysp [°C]')
xlabel('Time [min]')


subplot(338)
plot(time,x(:,2))
hold on
plot(time,ysp,'red')
ylabel('T2 [°C]')
xlabel('Time [min]')

subplot(339)
plot(time,x(:,1))        % the state x corresponds to the output y = T1
ylabel('T1 [°C]')
xlabel('Time [min]')

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

set(gcf,'Position',[100 100 750 500])              % 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','ExamplePart1.eps') % save eps.
```

SCRIPT PART 3 - FEEDBACK WITH CASCADE

```matlab
% TKP4140 Process Control Project

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)

%% SIMC

%Slave loop - T1
s= tf('s');
tau1= 9.253;
tauC1= tau1/5;
theta1 = 0;
k1 = -0.0002778;
kc1 = (1/k1)*(tau1/(tauC1 + theta1));
tauI1 = min(tau1, 4*(tauC1 + theta1));
g1 = (k1)/((tau1*s)+1);
C1 = (kc1*((tauI1*s)+1)/(tauI1*s));
L1 = exp(-theta1*s)/(tauC1*s + 1);

%'New process'
s=tf('s');
g2 = (-0.0001667)/((1.081*s + 1)*(9.253*s+1));
g2prime = L1*(g2/g1);
g2prime = minreal(g2prime);
g2prime=set(zpk(g2prime), 'DisplayFormat','time constant');

k2 = 0.60007;
tau1_2=2.3915;
theta2=0.5405;
tauC2=theta2;
kc2 = (1/k2)*(tau1_2/(tauC2 + theta2));
tauI2 = min(tau1_2, 4*(tauC2 + theta2));
%% Initializing

%Model parameters

Cp=120; %[J/kg°C]
M1=250; %[kg]
M2=100; %[kg]
UA= 6000; % G = UA [J/min*°C]
%Tf1 = 260; %[°C]

p = [Cp; M1; M2; UA];

% Nominal  values for  Q1 and Tf and W.
Tfnom=300;
Wnom=50;
Q1nom=360*1e3;

% Set steps to 0 - Operating at nominal point
ysp_step=0;
Tf_step=0;
W_step=0;
```

```
% Define the number of variales and model parameters
Nx = 2; %--number of states--NUMBER OF DIFFERENTIAL EQUATIONS
Nu = 1; %--number of inputs--NUMBER OF MVS
Nd = 2; %--number of disturbances--NUMBER OF DVS
Np = 4; %--number of model parameters--CONSTANT VALUES
Ny = Nx;%--number of outputs---equal to number of states

%INITIAL CONDITIONS FOR THE STATES
x0 = [200; 240];
T2nom= 240;
T1nom=200;

%% Simulate a step increase on Tf (Disturbance)
% Set the value for Step on Tf (Disturbance)
Tf_step=(0.1*Tfnom);
W_step=0;
ysp_step=0;

% Set the time for giving the Step on Tf
Tf_t_step = 1;
W_t_step=1;
ysp_t_step = 1;

%run Simulink model to test the step in Tf
tsim = 30;                   % 30 minutes
sim('ProjectTcascade')

%We are using subplots to plot the result.
figure(2)
subplot(331)
plot(time,d1,'red')               %the disturbance d1 corresponds to Tf
title('Step on Tf')
ylabel('Tf [°C]')
xlabel('Time [min]')


subplot(332)
plot(time,x(:,2))
hold on
plot(time,ysp,'red')% the state x corresponds to the output y = T2
ylabel('T2 [°C]')
xlabel('Time [min]')

subplot(333)
plot(time,x(:,1))        % the state x corresponds to the output y = T1
ylabel('T1 [°C]')
xlabel('Time [min]')
%%
%Simulate a step increase on W (Disturbance)
Tf_step=0;
W_step=(0.1*Wnom);
ysp_step=0;

% Set the time for giving the Step on q0
Tf_t_step = 1;
W_t_step=1;
ysp_t_step = 1;

%run Simulink model to test the step in q0
tsim = 30;                   % 30 minutes
sim('ProjectTcascade')
```

```
%We are using subplots to plot the result.
figure(2)
subplot(334)
plot(time,d2,'red')%the disturbance d2 corresponds to W
title('Step on W')
ylabel('W [kg/min]')
xlabel('Time [min]')


subplot(335)
plot(time,x(:,2))
hold on
plot(time,ysp,'red')% the state x corresponds to the output y = T2
ylabel('T2 [°C]')
xlabel('Time [min]')

subplot(336)
plot(time,x(:,1))        % the state x corresponds to the output y = T1
ylabel('T1 [°C]')
xlabel('Time [min]')


%% Simulate a step increase on ysp_step (Manipulated variable)
ysp_step=(0.1*T2nom);
Tf_step=0;
W_step=0;

Tf_t_step = 1;
W_t_step=1;
ysp_t_step = 1;

% Run Simulink model to test the effect of a step in Q1.
sim('ProjectTcascade')

figure(2)
subplot(337)
plot(time,ysp,'red')
title('Step on ysp_step')
ylabel('ysp [°C]')
xlabel('Time [min]')


subplot(338)
plot(time,x(:,2))
hold on
plot(time,ysp,'red')
ylabel('T2 [°C]')
xlabel('Time [min]')

subplot(339)
plot(time,x(:,1))        % the state x corresponds to the output y = T1
ylabel('T1 [°C]')
xlabel('Time [min]')

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

set(gcf,'Position',[100 100 750 500])              % 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','ExamplePart1.eps') % save eps.
```

SCRIPT PART 3 - BODE PLOT

```
%% BODE
clear all

s=tf('s')
tau = 9.794;
theta = 0.541;
tauI=4.328;
k = -0.0001667;
kc= (1/k)*(tau/(2*theta));
C2=(kc*((tauI*s)+1)/(tauI*s));
g2=(-0.0001667)/((1+(1.081*s))*((9.253*s)+1));


L = C2*g2;

% T2
figure(3), bode(L)
w=logspace(-3,1,1000);
[mag,phase]=bode(L,w);

hold on
xline(0.1081)
hold on
xline(0.9251)
hold on
xline(0.2311)

margin(L)
[Gm,Pm,Wcg,Wcp] = margin(L);

hold off

desiredX = interp1(w,L,1)
```

SCRIPT - SYS_ODE

```
% 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============================
% Pass the model parameters from the exampleMatlab script
% the parameters are given in vector p (function argument)
% In this case it is the area, you can add more parameters.
% Use exact values (try to not round-up/down).
% Be careful with unit consistency between parameters and equations.

Cp = p(1);
M1 = p(2);
M2 = p(3);
UA = p(4);

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

% EXTRACT INFORMATION
% inputs of the model are found in u
% If you have more inputs, the order of the u's (u(1), u(2), etc.)
% corresponds to the order in which you connect the input signals
% to the Interpreted MATLAB Fcn block.
Q1=u(1);  %The first signal in u, corresponding to q1

% disturbances of the model are found in d
% If you have more disturbances, the order of the u's (u(1), u(2), etc.)
% corresponds to the order in which you connect the input signals
% to the Interpreted MATLAB Fcn block.

Tf = d(1); %The first signal in d, corresponding to q0
W = d(2);

% Extract information: the states are given in x
% Define your states. You may need these to write the differential equations
T1 = x(1);
T2 = x(2);

%In your project, you might want to use this information to write
%some algebraic equations. For example, if you wanted to use the
%height of the tank to calculate the volume in the tank, then you
%would write:
%V=A*h;
%Then, you can use the volume (V) in your differential equations (if you need it).


%Write the differential equations
%The order of the differential equations (dxdt(i)) corresponds to
%the order in which you defined your states.
dxdt(1) = 1/M1*(w(Tf1 - T1) - Q1/Cp);
dxdt(2) = 1/M2*(w(T1 - T2) + UA/Cp((Tf + Tf1)/2 - T2));

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

end
```

SCRIPT - COMPUTE TRANSFER FUNCTIONS

```
%% 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 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]=computeTransferFunctions


%%Definition of differential equations

% Define/create symbolic variables for your MV, DV, CV
syms Q1;
syms T_f;
syms W;
syms T1;
syms T2;

% 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(1) = T_f;   %deviation variables
d(2) = W;
u = Q1;
x(1) = T1;
x(2) = T2;
y(1) = T1;
y(2) = T2;

%Parameters that will be used in the differential equations
Cp = 120; % [J/kg*°C]
M1 = 250; % [kg]
M2 = 100 ; % [kg]
UA = 6000 ; % [J/min*°C]
%Tf1 = 260;

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

%Differential equations
Tf1 = (2*W*Cp*T_f - UA*T_f + 2*T2*UA)/(UA + 2*W*Cp);
f(1,1) = ((W/M1)*(Tf1-T1))-(Q1/(M1*Cp)) ;  % This corresponds to the first state.
f(1,2)= ((W/M2)*(T1-T2))+(UA/(M2*Cp))*(((T_f+Tf1)/2)-T2);

%For more than 1 state: f(1,1)=...; f(1,2)=...;
```

```
%% 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
T_f=300;
W=50;
Q1=360000;
T1=200;
T2=240;

%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
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');

%
```