Four Tank System

02619 Model Predictive Control

Author:
Nikki Christoffersen

Student number:
s145016

Professor: John Bagterp Jørgensen

Technical University of Denmark
Kongens Lyngby
December 28, 2022
DTU Electrical Engineering

Contents

Appendix

Introduction

This report is the exam in 02619 Model Predictive Control focusing on a four-tank system. The project is discussed with Gustav Stæhr Hassager, s144966 however the report is written individually.
The parameters of the system is listed in the attached *MatLab* script *"FourTankSystem SS"*. The four tanks is connected as shown in fig. 1.

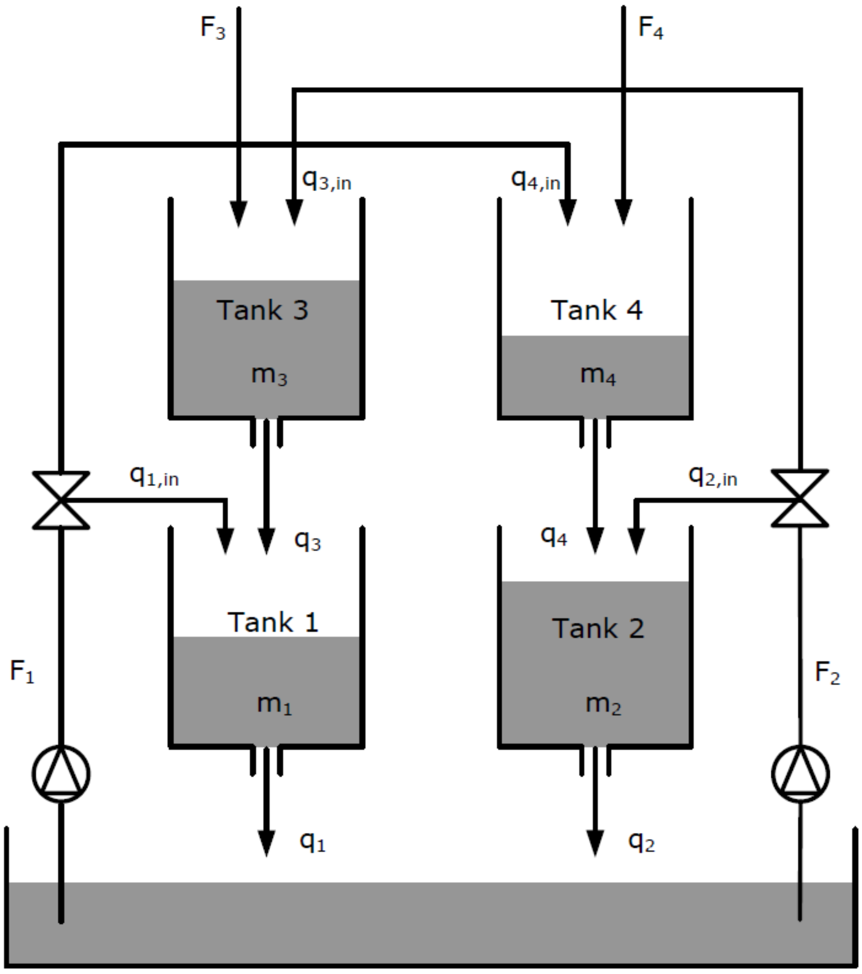

Figure 1: Schematic of Four Tank System

1 Control structure

1.1 Description

The system is a four-order system, meaning there is four states (x) which represent the mass in each tank. There is two measurements (y) which is the level of the water in tank 1 and 2. It is possible to adjust the flow through the two manipulated variables (u). There is also two disturbance variables (d) which only influence tank 3 and 4, however these are not measured and can not be manipulated. The controlled variables is the water level in each tank (z). All the above can be seen below.

$$x = \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \end{bmatrix} \quad y = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} \quad u = \begin{bmatrix} F_1 \\ F_2 \end{bmatrix} \quad d = \begin{bmatrix} F_3 \\ F_4 \end{bmatrix} \quad z = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} \tag{1}$$

1.2 Block Diagram

In fig. 2 the block diagram of the overall system including the MPC can be seen.

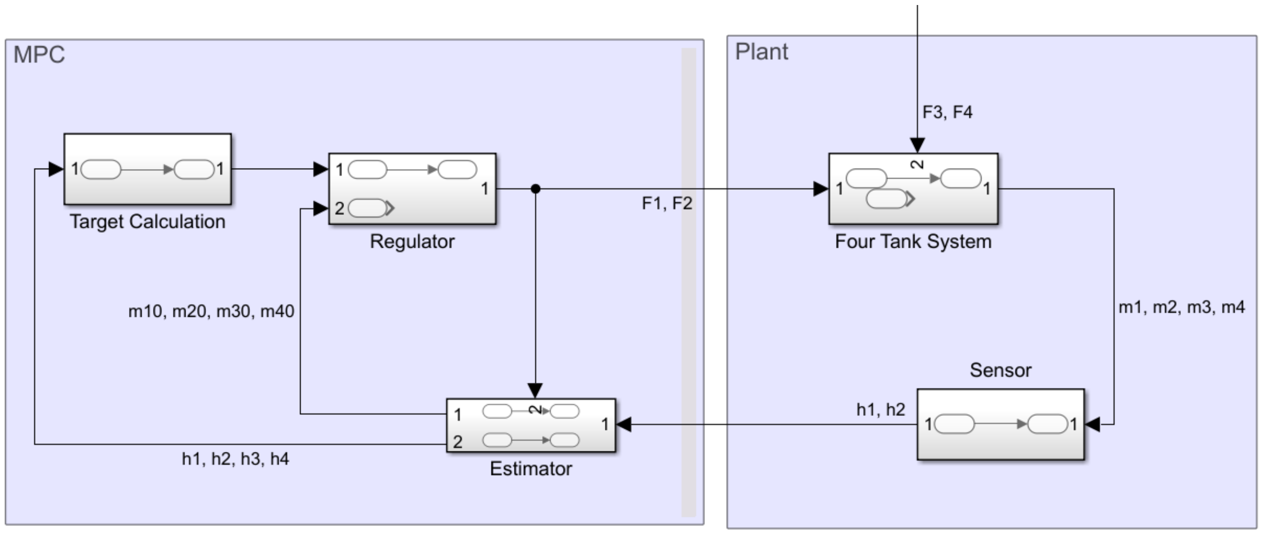

Figure 2: Block Diagram

2 Deterministic and Stochastic Nonlinear Modelling

2.1 Deterministic Nonlinear Model

2.1.1 Dynamics of the system - Point 1

The system can be described as deterministic mathematical model on the following form.

$$\dot{x}(t) = f(x(t), u(t), d(t), p) \tag{2}$$

x(t), *u(t)* and *d(t)* is the states, inputs and disturbances, respectively. p is the paramters of the system.
The states, x is the mass of the liquid in each tank. The mass in each tank can be described by a system of differential equations. Notice, the inputs and outputs are listed according to the schematic seen in fig. 1.

$$\begin{aligned} \dot{x}_1(t) &= \rho(q_{1,in}(t) + q_3(t) - q_1(t)) \\ \dot{x}_2(t) &= \rho(q_{2,in}(t) + q_4(t) - q_2(t)) \\ \dot{x}_3(t) &= \rho(q_{3,in}(t) + F_3(t) - q_3(t)) \\ \dot{x}_4(t) &= \rho(q_{4,in}(t) + F_4(t) - q_4(t)) \end{aligned} \tag{3}$$

The inputs to the tanks from the pumps is given by

$$\begin{aligned} q_{1,in}(t) &= \gamma_1\, F_1 \\ q_{2,in}(t) &= \gamma_2\, F_2 \\ q_{3,in}(t) &= (1-\gamma_2)\, F_2 \\ q_{4,in}(t) &= (1-\gamma_1)\, F_1 \end{aligned} \tag{4}$$

Where γ is a measure of the portion of water directed to the repsective tank. The water leaving the tanks is not controlled, but it is purely related to the gravity which can be described by Bernoulli's principle of constant energy. The principle is the sum of the potential and kinetic energy and the work added to the system (see eq. (5)) will converge to steady state.

$$\rho\, g\, h_{top} + \frac{1}{2}\,\rho\, v_{top}^2 + p = 0 + \frac{1}{2}\,\rho\, v_{bot}^2 + p_{bot} \tag{5}$$

The tanks are open, so the pressure at the top of the water level and bottom of each tank is equal to the atmospheric pressure, so this cancels out. Notice that the kinetic energy of the bottom is 0, since the height at the bottom is 0. It is known that the cross section area at the top of the tank is much larger than the cross section area at the bottom of the tank, which yields $v_{top} \ll v_{bot}$. The density is assumed to be constant. Now the Bernoulli principle can be rewritten into an expression determining the flow from each tank

$$\begin{aligned} q_1(t) &= a_1\sqrt{2\,g\,h_1(t)}\ ,\ h_1(t) = \frac{m_1(t)}{\rho\, A_1} \\ q_2(t) &= a_2\sqrt{2\,g\,h_2(t)}\ ,\ h_2(t) = \frac{m_2(t)}{\rho\, A_2} \\ q_3(t) &= a_3\sqrt{2\,g\,h_3(t)}\ ,\ h_3(t) = \frac{m_3(t)}{\rho\, A_3} \\ q_4(t) &= a_4\sqrt{2\,g\,h_4(t)}\ ,\ h_4(t) = \frac{m_4(t)}{\rho\, A_4} \end{aligned} \tag{6}$$

Finally the deterministic mathematical model can be determined on the above equations. Notice, the masses (m) is the states (x). The flow from the pumps $(F_1\ ,\ F_2)$ is the inputs u. The flow to tank 3 and 4 $(F_3\ ,\ F_4)$ is the disturbance.

$$\begin{aligned} \dot{x}_1(t) &= \rho\left(\gamma_1\, u_1(t) + a_3\sqrt{2\,g\,\frac{x_3(t)}{\rho\, A_3}} - a_1\sqrt{2\,g\,\frac{x_1(t)}{\rho\, A_1}}\right) \\ \dot{x}_2(t) &= \rho\left(\gamma_2\, u_2(t) + a_4\sqrt{2\,g\,\frac{x_4(t)}{\rho\, A_4}} - a_2\sqrt{2\,g\,\frac{x_2(t)}{\rho\, A_2}}\right) \\ \dot{x}_3(t) &= \rho\left((1-\gamma_2)\, u_2(t) + d_1(t) - a_3\sqrt{2\,g\,\frac{x_3(t)}{\rho\, A_3}}\right) \\ \dot{x}_4(t) &= \rho\left((1-\gamma_1)\, u_1(t) + d_2(t) - a_4\sqrt{2\,g\,\frac{x_4(t)}{\rho\, A_4}}\right) \end{aligned} \tag{7}$$

2.1.2 Sensors - Point 2

The mathematical model for the sensors (which is the measurements) reveals the correlation between the states and the measured output. By using the expressions for the dynamics as in the previous section, it is possible to determine the model on the following form.

$$y(t) = g(x(t), p) \tag{8}$$

The height in tank 1 and 2 (h) is the outputs (y).

$$\begin{aligned} y_1(t) &= \frac{x_1(t)}{\rho\, A_1} \\ y_2(t) &= \frac{x_2(t)}{\rho\, A_2} \end{aligned} \tag{9}$$

2.1.3 Outputs - Point 3

The mathematical models for the outputs is similar to the measurements, however the output is for each tank. Tt is possible to determine the model on the following form.

$$z(t) = h(x(t), p) \tag{10}$$

The height in tank 1, 2, 3 and 4 (h) is the outputs (y).

$$\begin{aligned} h_1(t) &= \frac{x_1(t)}{\rho\, A_1} \\ h_2(t) &= \frac{x_2(t)}{\rho\, A_2} \\ h_3(t) &= \frac{x_3(t)}{\rho\, A_3} \\ h_4(t) &= \frac{x_4(t)}{\rho\, A_4} \end{aligned} \tag{11}$$

2.2 Stochastic Nonlinear Model

2.2.1 Dynamics of the system - Point 1

It is desired to represent the system as mathematical model in the following form.

$$\dot{x}(t) = f(x(t), u(t), d(t), p) \ , \ d(t) = d_k(t) \, \text{for} \, t_k \leq t \leq t_{k+1} \tag{12}$$

The definition of $d_k(t)$ is related to the assumption that the disturbances F_3 and F_4 are stochastic variables but piecewise constant. Normally, a stochastic variable is modelled as Brownian motion, however this is not well defined in mathematical point of view. Therefore it is modelled as piecewise constant meaning that in a given time interval (t_k to t_{k+1}) the disturbance is constant. By this simplifying assumption, the general model described in eq. (7) in section 2.1.1 can be rewritten into the following.

$$\begin{aligned} \dot{x}_1(t) &= \rho \left(\gamma_1\, u_1(t) + a_3 \sqrt{2\, g\, \frac{x_3(t)}{\rho\, A_3}} - a_1 \sqrt{2\, g\, \frac{x_1(t)}{\rho\, A_1}} \right) \\ \dot{x}_2(t) &= \rho \left(\gamma_2\, u_2(t) + a_4 \sqrt{2\, g\, \frac{x_4(t)}{\rho\, A_4}} - a_2 \sqrt{2\, g\, \frac{x_2(t)}{\rho\, A_2}} \right) \\ \dot{x}_3(t) &= \rho \left((1-\gamma_2)\, u_2(t) + d_{k1}(t) - a_3 \sqrt{2\, g\, \frac{x_3(t)}{\rho\, A_3}} \right) \ , \ d_{k1}(t) \sim N(0, R_{QQ}(p)) \\ \dot{x}_4(t) &= \rho \left((1-\gamma_1)\, u_1(t) + d_{k2}(t) - a_4 \sqrt{2\, g\, \frac{x_4(t)}{\rho\, A_4}} \right) \ , \ d_{k2}(t) \sim N(0, R_{QQ}(p)) \end{aligned} \tag{13}$$

2.2.2 Sensors - Point 2

It is desired to model the measurements which always will be affected by some measurement noise, leading the output equation to take the following form.

$$y(t) = g(x(t), p) + v(t) \ , \ v(t) \sim N(0, R_{vv}(p)) \tag{14}$$

The noise is modelled as a normal distribution with 0 mean and the covariance $R_{vv}(p)$. The mathematical expression for each sensor is seen below.

$$\begin{aligned} y_1(t) &= \frac{x_1(t)}{\rho\, A_1} + v_1(t) \\ y_2(t) &= \frac{x_2(t)}{\rho\, A_2} + v_2(t) \end{aligned} \tag{15}$$

2.2.3 Outputs - Point 3

The mathematical model for the outputs is not affected by the noise and can therefore be modellled as the deterministic case.

$$\begin{aligned} h_1(t) &= \frac{x_1(t)}{\rho\, A_1} \\ h_2(t) &= \frac{x_2(t)}{\rho\, A_2} \\ h_3(t) &= \frac{x_3(t)}{\rho\, A_3} \\ h_4(t) &= \frac{x_4(t)}{\rho\, A_4} \end{aligned} \tag{16}$$

3 Nonlinear Simulation - Step Responses

3.1 Deterministic Model - Step Reponses

It is of interest to evaluate the step response of the system, with steps of 10, 20 and 50% on the manipulated variables (u). In the simulations, the valve setpoints is $\gamma_1 = 0.58$ and $\gamma_2 = 0.68$. The steady state inputs are $u = [300\,300]^T\,[cm^3/s]$ and the disturbances are $d = [250\,250]^T\,[cm^3/s]$. The system is initialized from 0, and after 20 minutes, the step occurs. Notice, that the step on the inputs are simulated individually.

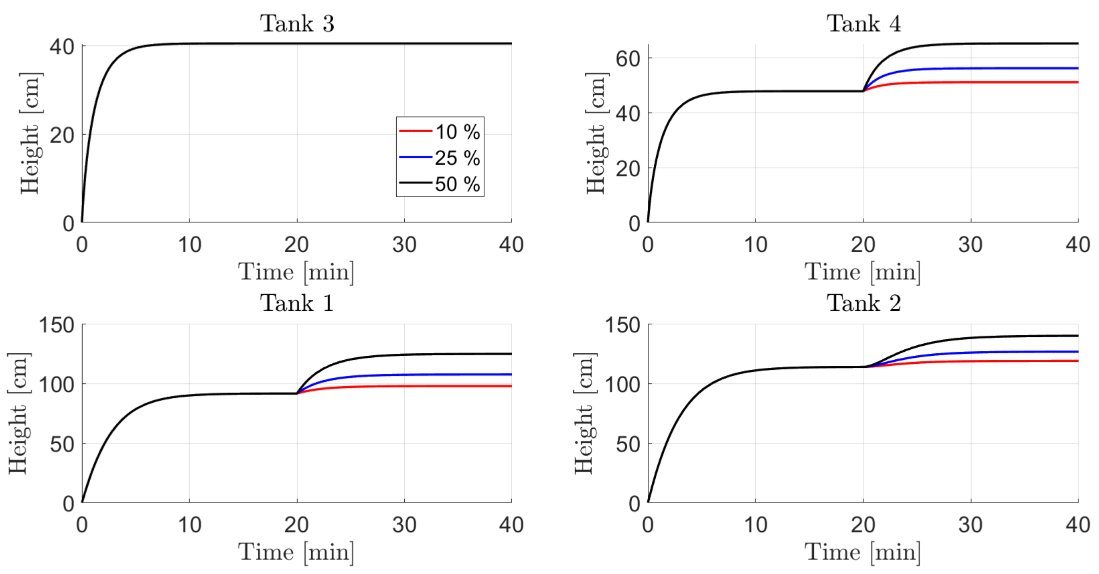

Figure 3: Step on u_1 - Deterministic model

The different valve settings is seen in regards to the steady state response of the level in tank 1 and 2, where the water level is largest at tank 2, which is consistent with the valve setting γ_1 allows more water (compared to γ_2) going to tank 4 and thereby to tank 2.
It is seen that the step on u_1 do not affect tank 3 as expected, since this is only influenced by the disturbance and on u_2

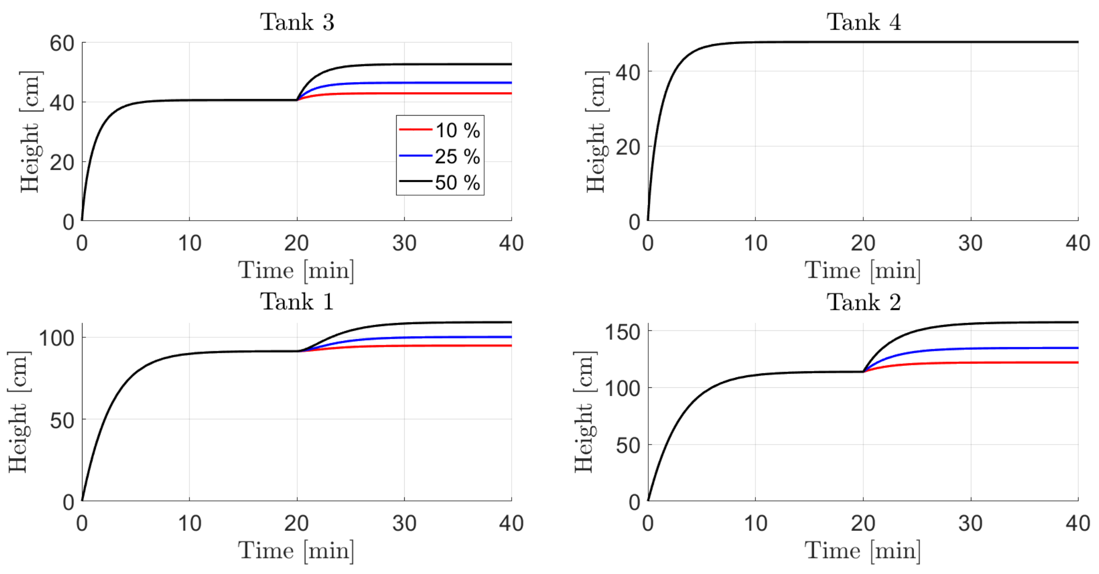

Figure 4: Step on u_2 - Deterministic model

The behaviour of the system is, as expected, opposite of when input 1 is affected.

3.2 Stochastic Model - Step Response

In this exercise it is desired to simulate the non-linear model with the stochastic variables. For doing this, the chosen noise contributions can be seen below. It should be notice that the state noise is relative to the mass in gram, and is only a contribution to the unknown disturbance, as seen in eq. (13) in section 2.2, and therefore it is only a 2x2 matrix. The measurement noise is relative to the height, which is in cm. All the noise contributions is assumed to be uncorrelated.

$$
\begin{aligned}
R_{x,low} &= \begin{bmatrix} 5 & 0 \\ 0 & 10 \end{bmatrix} \quad R_{x,medium} = \begin{bmatrix} 50 & 0 \\ 0 & 100 \end{bmatrix} \quad R_{x,high} = \begin{bmatrix} 500 & 0 \\ 0 & 1000 \end{bmatrix} \\
R_{y,low} &= \begin{bmatrix} 0.2 & 0 \\ 0 & 0.4 \end{bmatrix} \quad R_{y,medium} = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \quad R_{y,high} = \begin{bmatrix} 5 & 0 \\ 0 & 10 \end{bmatrix}
\end{aligned}
\tag{17}
$$

The response of the simulation with a step of 10, 25 and 50% can be seen in fig. 5, fig. 6 and fig. 7 respectively. In these cases, the system is initialized at steady state.

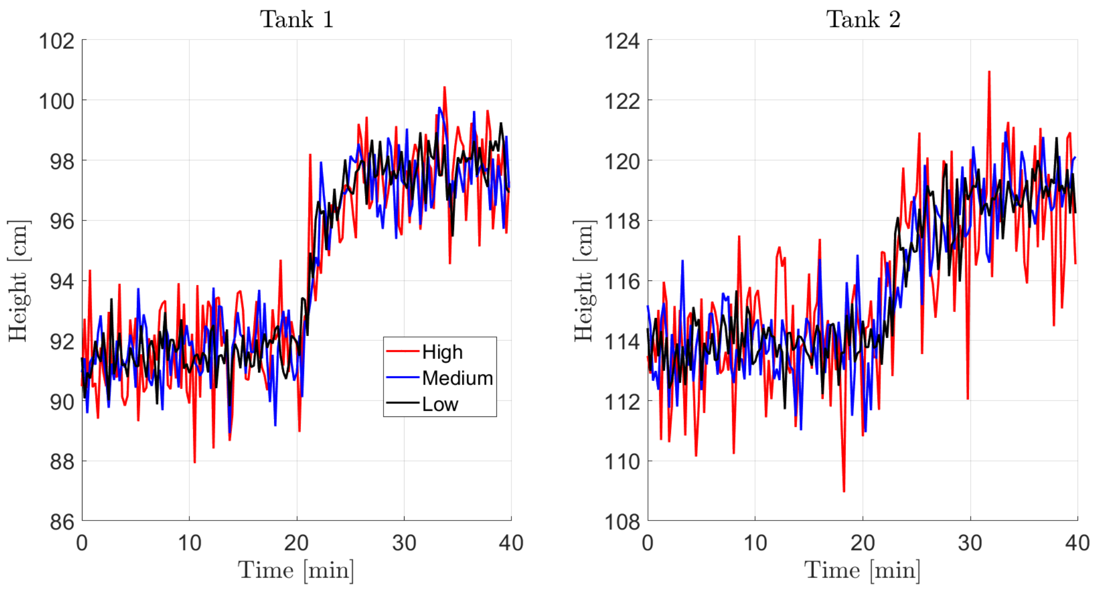

Figure 5: Step 10 % on u_1 - Stochastic model

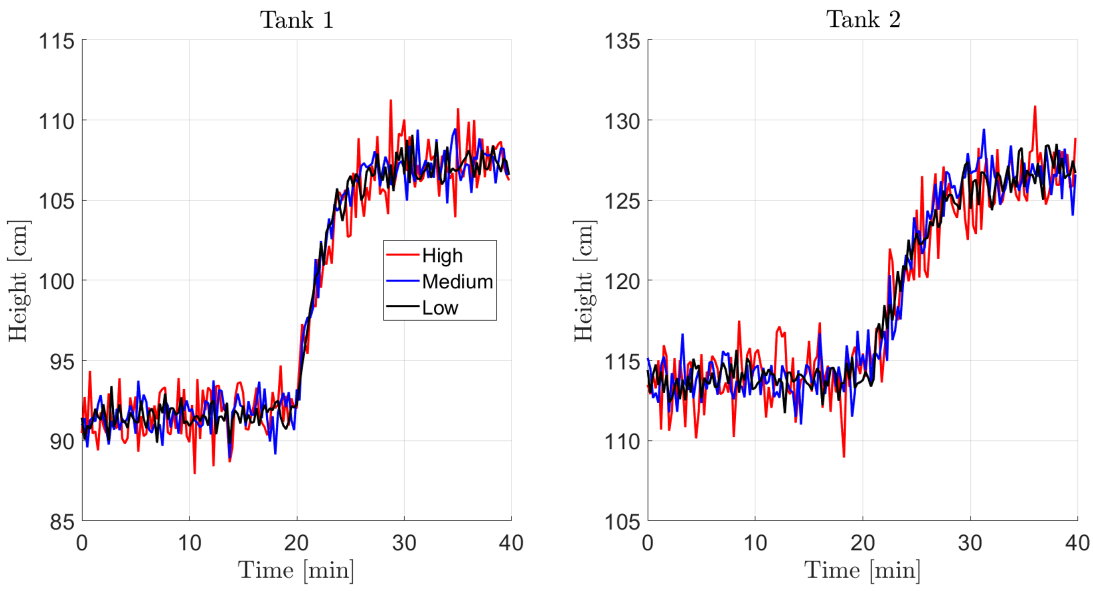

Figure 6: Step 25 % on u_1 - Stochastic model

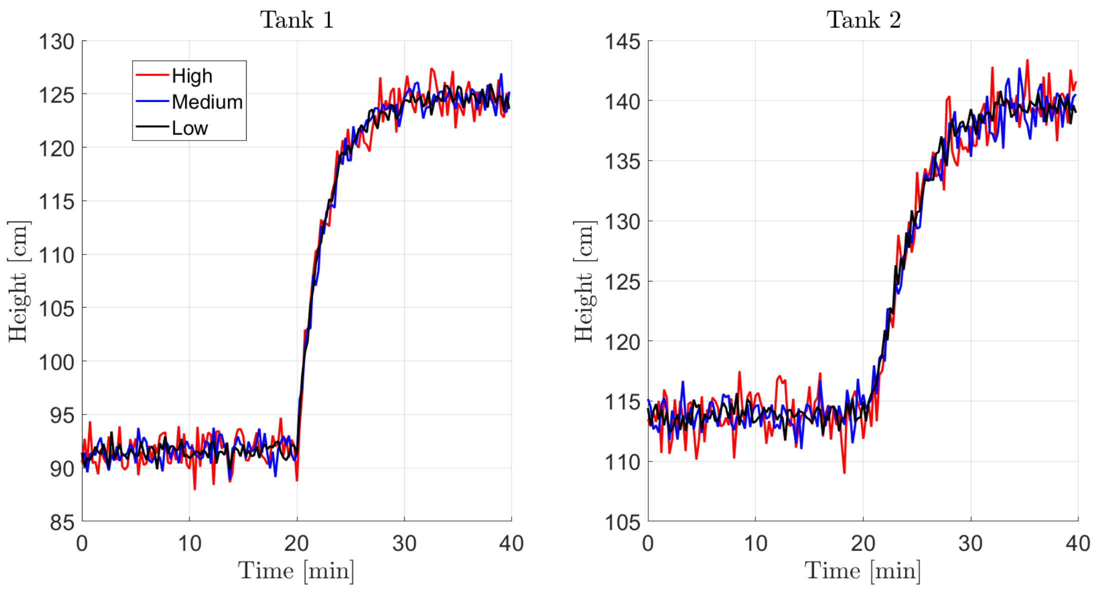

Figure 7: Step 50 % on u_1 - Stochastic model

The 3 noise affections is clearly seen for each step, and the larger the step, the noise contribution has a smaller relative effect.

3.3 Normalized Step Response

In this section, the normalized step response on the two controlled variabels (u) is seen. The step response is determined by first step on u_1 and the result is evaluated at the height in tank 1 and 2 (as seen on the left column of fig. 8) at the 3 step amplitudes (10, 25 and 50%). Afterwards the same is done for u_2 (as seen on the right column of fig. 8). During the simulation, the disturbance is kept constant at 250 $[cm^3/s]$.

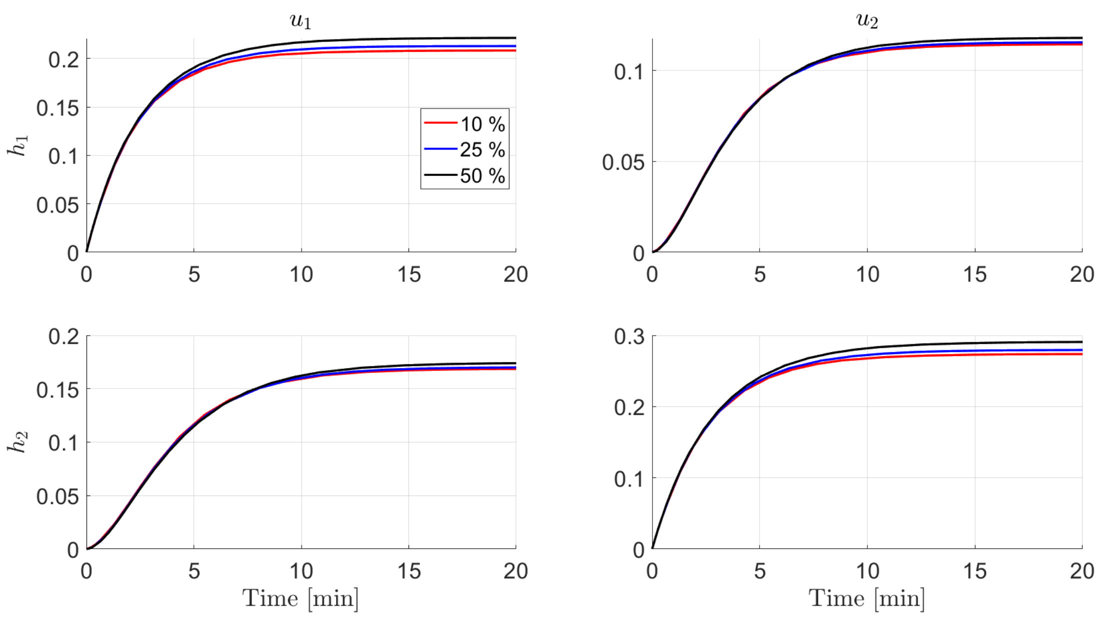

Figure 8: Normalized step response - Deterministic model

It is seen that the normalized step on each of the inputs are not the same, which is due to the non-linearity of the system. If the system was linear, the responses would be similar.
Similar to the deterministic system, the normalized step for the stochastic system is evaluated. In the given case, only the outputs relating to the input 1 is shown where the low noise level is used (see eq. (17)).

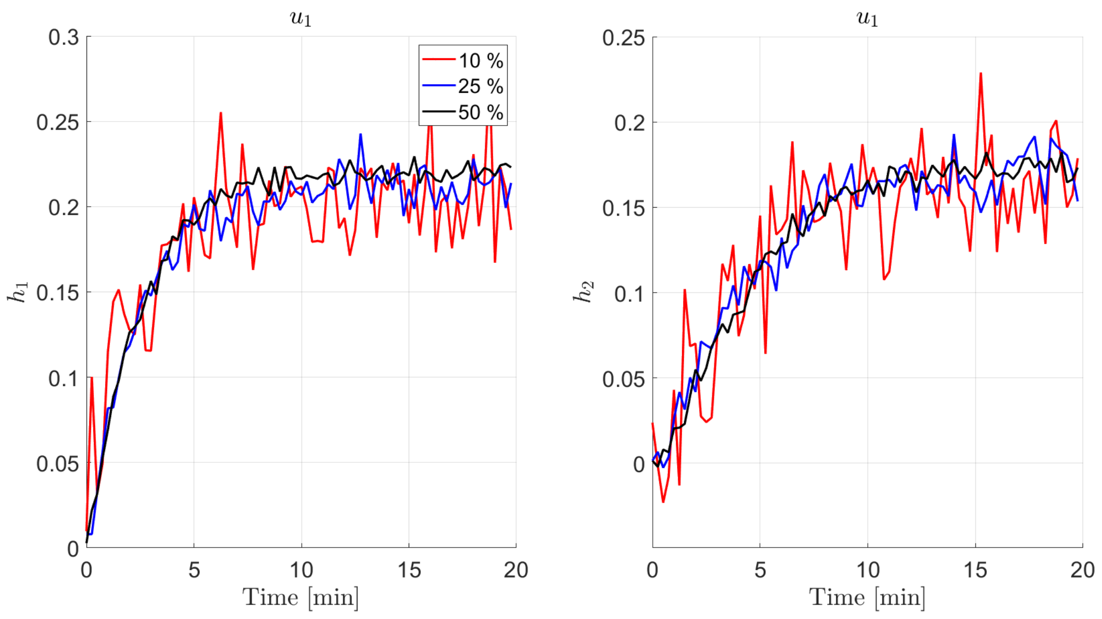

Figure 9: Normalized step response - Stochastic model

It is seen the stochastic model follows the same response as the deterministic system. However the noise is clearly present. Since the noise has a larger relative influence on the system with small steps (10%) than it has on the system with large steps (50%), the noise seems larger for the small steps.

3.4 Transfer Functions - Non-Linear System

On basis on the system response seen in the previous paragraph, it is possible to determine a transfer function for each case of the normalized step. This transfer function is an input-output model of the four tank system. Notice, that the system seen from u_1 to y_1 and from u_2 to y_2 can be described by a 1$^{\text{st}}$ order system (the general form can be seen in eq. (18)) and the system seen from u_1 to y_2 and from u_2 to y_1 is a 2$^{\text{nd}}$ order system (the general form can be seen in eq. (19)). This makes intuitive sense, since the water from u1, needs to go through tank 4, before entering tank 2, and opposite for u2 to tank 1.

$$\frac{Y(s)}{U(s)} = \frac{K\, e^{-\theta\, s}}{\tau\, s + 1} \tag{18}$$

$$\frac{Y(s)}{U(s)} = \frac{K\, e^{-\theta\, s}}{(\tau_1\, s + 1)(\tau_2\, s + 1)} \tag{19}$$

K is the gain of the system which is the steady state value seen in each normalized step. θ is the time delay, which in the given case is 0. τ is the time constant of the system.
The transfer functions describes a linear system, and therefore the model is only valid close

to the linearisation. In the following analysis, the step response of 10% is evaluated. The system is evaluated when no noise is present and the disturbance is constant at 250 $[cm^3/s]$. The estimated transfer functions is determined to be:

$$\begin{bmatrix} Y_1(s) \\ Y_2(s) \end{bmatrix} = \begin{bmatrix} \frac{0.2083}{138\,s+1} & \frac{0.1144}{(90\,s+1)(138\,s+1)} \\ \frac{0.1686}{(80\,s+1)(173\,s+1)} & \frac{0.2736}{153\,s+1} \end{bmatrix} \begin{bmatrix} U_1(s) \\ U_2(s) \end{bmatrix} \tag{20}$$

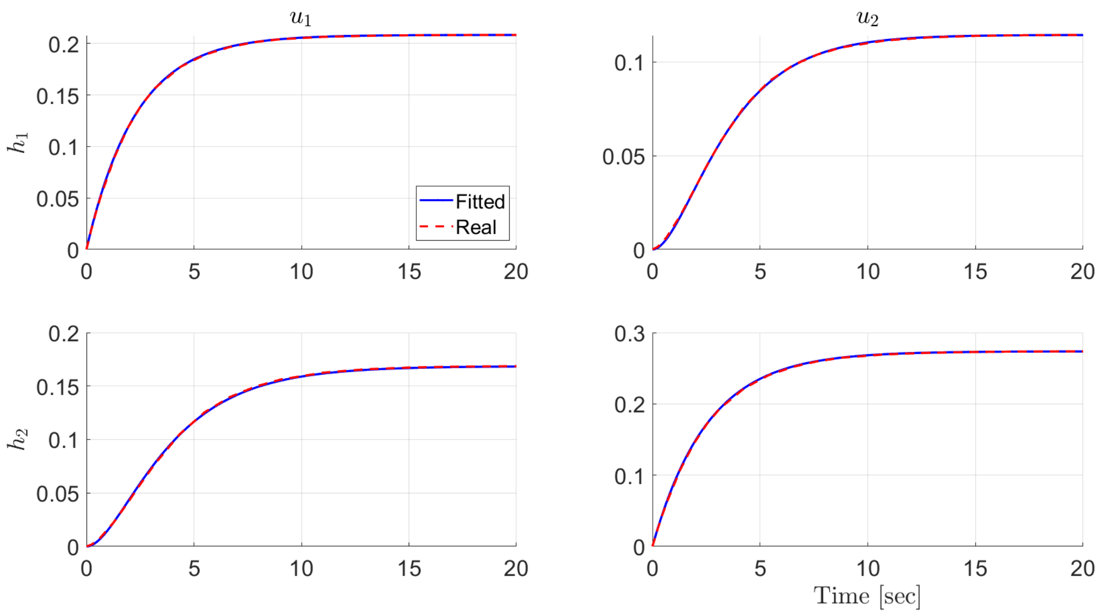

Figure 10: Comparison of transfer function and real response

3.5 Accuracy of Transfer Functions

In the previous paragraph the estimated transfer functions was shown. From the simulation of fitted and real transfer function it is seen that there is a good similarity between the "fitted" response compared "real". The error between each sample was determined, and the mean error and variance of this was determined (see)

	u_1y_1	u_2y_1	u_1y_2	u_2y_2
Mean	$2.25\,10^{-5}$	$1.34\,10^{-4}$	$4.96\,10^{-4}$	$-1.53\,10^{-4}$
Variance	$7.93\,10^{-8}$	$1.13\,10^{-7}$	$3.64\,10^{-7}$	$5.34\,10^{-8}$
Deviation [%]	0.011	0.117	0.295	-0.056

Table 1: Mean error and variance of transfer functions

It is clearly seen that the mean error is very low. The deviation in % is calculated on basis of the mean error, and the final value for the respective normalized step. As expected, the two 2nd order systems has a larger error, due to the fact that the time constants are approximated values. It is therefore concluded that the fitted transfer functions are a good approximation of the real system.

3.6 Impulse Response Coefficients

The Markov Parameters is determined on basis of the linearized state space model. In order to determine a state space model for the system, the SISO transfer functions determined in the previous paragraph is used. This yields the system in relation to the observer canonical form, which ensure the state space model to take the following form.

$$\begin{aligned} \dot{x}(t) &= A_{OC}\, x(t) + B_{OC}\, u(t) \\ y(t) &= C_{OC}\, x(t) + D_{OC}\, u(t) \end{aligned} \tag{21}$$

The observer canonical form matrices is determined by the normalized transfer functions for the system. It was previously determined that there was no time delay in the system, so these are ignored in the analysis.
For 1^{st} order systems (normalized) the matrices are determined as follows:

$$\frac{b_1\, s + b_0}{s + a_0} \rightarrow A_{OC} = [-a_0] \quad B_{OC} = [b_0 - a_0\, b_1] \quad C_{OC} = [1] \quad D_{OC} = [b_1] \tag{22}$$

For 2^{nd} order systems (normalized) the matrices are determined as follows

$$\frac{b_2\, s^2 + b_1\, s + b_0}{s^2 + a_1\, s + a_0} \rightarrow A_{OC} = \begin{bmatrix} -a_1 & 1 \\ -a_0 & 0 \end{bmatrix} \quad B_{OC} = \begin{bmatrix} b_1 - a_1\, b_2 \\ b_0 - a_0\, b_2 \end{bmatrix} \quad C_{OC} = \begin{bmatrix} 1 & 0 \end{bmatrix} \quad D_{OC} = [b_2] \tag{23}$$

The system is now discrtized according to the following:

$$A_k = e^{A_{OC}\, T_s} \quad B_k = \int_0^{T_s} e^{A_{OC}\, t}\, B_{OC}\, dt \quad C_k = C_{OC} \quad D_k = D_{OC} \tag{24}$$

The numerical values with a discretization time of $T_s = 15$ is determined as:

$$\begin{aligned} &A_{k11} = [0.897] \qquad B_{k11} = [0.0215] \qquad C_{k11} = [1] \qquad D_{k11} = [0] \\ &A_{k12} = \begin{bmatrix} 0.7534 & 13.0837 \\ -0.0009 & 0.9926 \end{bmatrix} \quad B_{k12} = \begin{bmatrix} 0.0013 \\ 0.0002 \end{bmatrix} \quad C_{k12} = \begin{bmatrix} 1 & 0 \end{bmatrix} \quad D_{k12} = [0] \\ &A_{k21} = \begin{bmatrix} 0.7518 & 13.0725 \\ -0.0011 & 0.9917 \end{bmatrix} \quad B_{k21} = \begin{bmatrix} 0.9456 \\ 0.1377 \end{bmatrix} \quad C_{k21} = \begin{bmatrix} 1 & 0 \end{bmatrix} \quad D_{k21} = [0] \\ &A_{k22} = [0.9066] \qquad B_{k22} = [0.0256] \qquad C_{k22} = [1] \qquad D_{k22} = [0] \end{aligned} \tag{25}$$

The response for the four transfer functions (SISO) is determined according to eq. (26), and these can be used to determine the MIMO response according to eq. (27)

$$\begin{aligned} (H_0)_{i,j} &= D_k \\ (H_k)_{i,j} &= C_k\, A_k^{k-1}\, B_k \quad k = 1, 2, \ldots \end{aligned} \tag{26}$$

$$H_k = \begin{bmatrix} (H_k)_{11} & (H_k)_{12} \\ (H_k)_{21} & (H_k)_{22} \end{bmatrix} \quad k = 1, 2, \ldots \tag{27}$$

The MIMO response coefficients is used to generate the Hankel matrix as seen below.

$$\mathcal{H}_{N,N} \triangleq \begin{bmatrix} H_1 & H_2 & \ldots & H_N \\ H_2 & H_3 & \ldots & H_{N+1} \\ \vdots & \vdots & \ddots & \vdots \\ H_N & H_{N+1} & \ldots & H_{2N-1} \end{bmatrix} \tag{28}$$

It is known that the impulse reponse $(\mathcal{H}_i)_{i=1}^{\infty}$ has a minimal relization of order n, if and only if $\text{rank}(\mathcal{H}_{n+1,j}) = \text{rank}(\mathcal{H}_{n,j})$. If this is true, it is possible to perform a Singular Value Decomposition (*SVD*) of the Hankel matrix. For the given system, $n = 3$ which yields the Hankel matrix to be the following:

$$\mathcal{H}_{3,3} = \begin{bmatrix} 0.0215 & 0.0013 & 0.0192 & 0.0033 & 0.0173 & 0.0049 \\ 0.0009 & 0.0256 & 0.0025 & 0.0232 & 0.0037 & 0.0210 \\ 0.0192 & 0.0033 & 0.0173 & 0.0049 & 0.0155 & 0.0060 \\ 0.0025 & 0.0232 & 0.0037 & 0.0210 & 0.0045 & 0.0190 \\ 0.0173 & 0.0049 & 0.0155 & 0.0060 & 0.0139 & 0.0067 \\ 0.0037 & 0.0210 & 0.0045 & 0.0190 & 0.0050 & 0.0173 \end{bmatrix} \tag{29}$$

The *SVD* is carried out on the above Hankel norm, which results in the following expression.

$$\mathcal{H}_{N,N} = K\,\Lambda\,L^T = \begin{bmatrix} K_1 & K_2 \end{bmatrix} \begin{bmatrix} \Lambda_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} L_1 \\ L_2 \end{bmatrix} = K_1\,\Lambda_1\,L_1^T \tag{30}$$

The matrices K_1, Λ_1 and L_1 is determined with the inbuild *SVD* function in *MatLab* as seen below.

$$K_1 = \begin{bmatrix} -0.3076 & 0.5710 & 0.4895 & -0.4599 \\ -0.5148 & -0.3830 & 0.5296 & 0.4291 \\ -0.3154 & 0.4748 & -0.0588 & 0.1053 \\ -0.4844 & -0.3017 & -0.0574 & -0.1151 \\ -0.3166 & 0.3941 & -0.4686 & 0.5600 \\ -0.4539 & -0.2355 & -0.5036 & -0.5161 \end{bmatrix} \qquad L_1 = \begin{bmatrix} -0.3020 & 0.5717 & -0.4768 & 0.4793 \\ -0.5181 & -0.3821 & -0.5419 & -0.4063 \\ -0.3001 & 0.4835 & 0.0414 & -0.1263 \\ -0.4942 & -0.2867 & 0.0709 & 0.0931 \\ -0.2939 & 0.4089 & 0.4342 & -0.5841 \\ -0.4687 & -0.2097 & 0.5327 & 0.4892 \end{bmatrix}$$

$$\Lambda_1 = \begin{bmatrix} 0.0712 & 0 & 0 & 0 \\ 0 & 0.0458 & 0 & 0 \\ 0 & 0 & 0.0004 & 0 \\ 0 & 0 & 0 & 0.0001 \end{bmatrix} \tag{31}$$

The Hankel matrix can also be described by the product of the extended observability matrix $\mathcal{O}_N$ and the extended controllability matrix $\mathcal{C}_N$ which is determined as seen below.

$$\mathcal{O}_N = K_1\Lambda_1^{1/2} \qquad \mathcal{C}_N = \Lambda_1^{1/2}\,L_1^T \tag{32}$$

Due to the fact that K, L, K_1 and L_1 are orthogonal matrices, it is possible to determine B from the first m columns of $\mathcal{C}_N$ and to determine C from the first p rows of $\mathcal{O}_N$. D is determined on basis eq. (26).

$$\begin{gathered} B_k = (\mathcal{C}_N)_{(:,1:m)} = \Lambda_1^{1/2}\,\left((L_1)_{(1:m,:)}\right)^T = \begin{bmatrix} -0.0806 & -0.1382 \\ 0.1224 & -0.0818 \\ -0.0098 & -0.0112 \\ 0.0051 & -0.0043 \end{bmatrix} \\ C_k = (\mathcal{O}_N)_{(1:p,:)} = (K_1)_{(1:p,:)}\,\Lambda_1^{1/2} = \begin{bmatrix} -0.0821 & 0.1223 & 0.0101 & -0.0049 \\ -0.1374 & -0.0820 & 0.0109 & 0.0046 \end{bmatrix} \\ D_k = H_0 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \end{gathered} \tag{33}$$

In order to determine the A matrix, the Hankel matrix is redefined according to the following:

$$\tilde{\mathcal{H}}_{(N+1,N+1)} = \mathcal{O}_N \, A_k \, \mathcal{C}_N \tag{34}$$

The expression for $\mathcal{O}_N$ and $\mathcal{C}_N$ can is substitued, and then A_k is isolated which yields:

$$A_k = \Lambda_1^{-1/2} \, K_1^T \, \tilde{H}_{(N+1,N+1)} \, L_1 \, \Lambda_1^{-1/2} = \begin{bmatrix} 0.9106 & -0.0302 & 0.1431 & 0.0106 \\ -0.0684 & 0.6638 & -0.0309 & 0.0951 \\ -0.1436 & 0.0283 & 0.5977 & -0.0353 \\ -0.0136 & -0.0943 & 0.0411 & 0.9010 \end{bmatrix} \tag{35}$$

With the above determine state space model it is now possible to determine the Markov parameters according to the below equation. The impulse response can be seen in fig. 11.

$$\text{Markov parameters} = \begin{cases} D & i = 0 \\ C \, A^{i-1} \, B & i = 1, 2, \ldots \end{cases} \tag{36}$$

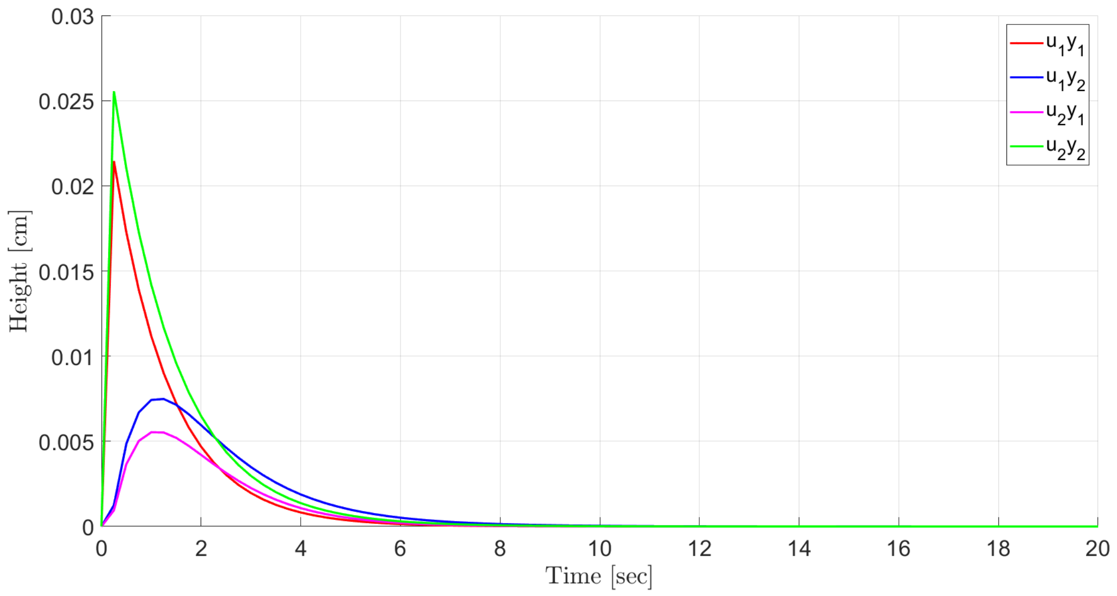

Figure 11: Impulse reponse

4 Linearization and Discretization

4.1 Continious Time State Space Models

4.1.1 Deterministic Model

The non-linear model derived in section 2.1 can be transformed into a state space model through linearization. A Linear Time Invariant (LTI) system is only valid close to the linerization point. The system in state space form is given by the below equation.

$$\begin{gathered} \dot{x}(t) = A \, X(t) + B \, U(t) + G \, D(t) \\ y(t) = C \, X(t) \\ z(t) = C_z \, X(t) \\ X(t) = x(t) - x_{ss} \quad U(t) = u(t) - u_{ss} \quad D(t) = d(t) - d_{ss} \end{gathered} \tag{37}$$

It should be noticed that the measurements (y) and the controlled variable (z) is the same for the given system. One should notice the use of the states and inputs is deviation variables, meaning the steady state (denoted ss) is subtracted from the respective state.
The matrices is determined by differentiation the respective equation with respect to the given state, input and disturbance. The derived algebraic expression for each matrix is seen below.

$$A = \begin{bmatrix} -\frac{\sqrt{2}\,a_1\,g}{2\,A_1\sqrt{\frac{g\,m_1}{A_1\,\rho}}} & 0 & \frac{\sqrt{2}\,a_3\,g}{2\,A_3\sqrt{\frac{g\,m_3}{A_3\,\rho}}} & 0 \\ 0 & -\frac{\sqrt{2}\,a_2\,g}{2\,A_2\sqrt{\frac{g\,m_2}{A_2\,\rho}}} & 0 & \frac{\sqrt{2}\,a_4\,g}{2\,A_4\sqrt{\frac{g\,m_4}{A_4\,\rho}}} \\ 0 & 0 & -\frac{\sqrt{2}\,a_3\,g}{2\,A_3\sqrt{\frac{g\,m_3}{A_3\,\rho}}} & 0 \\ 0 & 0 & 0 & -\frac{\sqrt{2}\,a_4\,g}{2\,A_4\sqrt{\frac{g\,m_4}{A_4\,\rho}}} \end{bmatrix} \quad C = \begin{bmatrix} \frac{1}{A_1\,\rho} & 0 & 0 & 0 \\ 0 & \frac{1}{A_2\,\rho} & 0 & 0 \end{bmatrix}$$
$$B = \begin{bmatrix} \gamma_1\,\rho & 0 \\ 0 & \gamma_2\,\rho \\ 0 & (1-\gamma_2)\,\rho \\ (1-\gamma_1)\,\rho & 0 \end{bmatrix} \quad G = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \rho & 0 \\ 0 & \rho \end{bmatrix} \tag{38}$$

By analyzing the A matrix, the correlation between the tanks is clearly visible. Tank 1 and 2 is depending on the water level in tank 3 and 4 respectively. The C matrix indicates that is only possible to measure the water level in tank 1 and 2 (notice, that C_z is not shown, since $C = C_z$). The B matrix indicates that it is possible to affect the water in all tanks. The G matrix indicates that the disturbance is only affecting tank 3 and 4 directly.
As mentioned, the state space model is only valid close to the linearization point, which is the steady state point. In this project, the steady state values is given by eq. (39). The corresponding state space model at the given linearization point is seen in eq. (40)

$$x_{ss} = \begin{bmatrix} 3.4778 \\ 4.3254 \\ 1.5401 \\ 1.8188 \end{bmatrix} \cdot 10^4\,[g] \quad y_{ss} = \begin{bmatrix} 91.49 \\ 113.79 \end{bmatrix}\,[cm] \quad u_{ss} = \begin{bmatrix} 300 \\ 300 \end{bmatrix}\,[cm^3/s] \quad d_{ss} = \begin{bmatrix} 250 \\ 250 \end{bmatrix}\,[cm^3/s] \tag{39}$$

$$A = \begin{bmatrix} -0.0075 & 0 & 0.0112 & 0 \\ 0 & -0.0067 & 0 & 0.0103 \\ 0 & 0 & -0.0112 & 0 \\ 0 & 0 & 0 & -0.0103 \end{bmatrix} \quad C = \begin{bmatrix} 0.0026 & 0 & 0 & 0 \\ 0 & 0.0026 & 0 & 0 \end{bmatrix}$$
$$B = \begin{bmatrix} 0.5800 & 0 \\ 0 & 0.6800 \\ 0 & 0.3200 \\ 0.4200 & 0 \end{bmatrix} \quad G = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{40}$$

4.1.2 Stochastic Model (piecewise Constant)

The stochastic model defined in section 2.2 was determined under the assumption that the disturbances is piecewise constant. This implies that the model in each time interval is deterministic, and therefore the linearized stochastic model is almost identical to the stochastic model determine the previous paragraph.

$$\begin{gathered}\dot{x}(t) = A\,X(t) + B\,U(t) + G\,D(t)\\ y(t) = C\,X(t) + I\,v(t)\\ z(t) = C_z\,X(t)\\ X(t) = x(t) - x_{ss} \quad U(t) = u(t) - u_{ss} \quad D(t) = d(t) - d_{ss} \quad v \sim N(0, R_{vv}(p)\end{gathered} \tag{41}$$

The only difference in this state space representation is the noise contribution to the output, is random, however this do not affect the system matrices.

4.2 Gains, Poles and Zeros

The poles, gains and zeros of the linearized system only depends on A, B and C and therefore they are the same for deterministic and stochastic model. The poles is the eigenvalues of the linear model.

$$\text{poles}= \begin{bmatrix} -0.0075\\ -0.0067\\ -0.0112\\ -0.0103 \end{bmatrix} \tag{42}$$

It is seen that the poles of linearized system is strictly negativ, and therefore the system is asymptotically stable.
The gains is related to the inputs affecting the outputs, and is given by:

$$K_{(y,u)} = -C_{(y,:)}\,A^{-1}\,B_{(:,u)} \rightarrow \begin{aligned} K_{(y1,u1)} &= 0.2041\\ K_{(y1,u2)} &= 0.1126\\ K_{(y2,u1)} &= 0.1648\\ K_{(y2,u2)} &= 0.2669 \end{aligned} \tag{43}$$

The zeros of the system can be determined by evaluating the system for each input with respect to each output according to the following.

$$\begin{bmatrix} A & B_{(:,u)}\\ C_{(y,:)} & 0 \end{bmatrix} \begin{bmatrix} x\\ u \end{bmatrix} = z \begin{bmatrix} I & 0\\ 0 & 0 \end{bmatrix} \begin{bmatrix} x\\ u \end{bmatrix} \tag{44}$$

The zeros which is a solution to the above and satisfy $|z| < \infty$ is determined to be

$$\begin{gathered} zeros_{(y1,u1)} \rightarrow z_1 = -0.0067\,,\ z_2 = -0.0103\,,\ z_3 = -0.0112\\ zeros_{(y1,u2)} \rightarrow z_1 = -0.0067\,,\ z_2 = -0.0103\\ zeros_{(y2,u1)} \rightarrow z_1 = -0.0075\,,\ z_2 = -0.0112\\ zeros_{(y2,u2)} \rightarrow z_1 = -0.0075\,,\ z_2 = -0.0112\,,\ z_3 = -0.0103 \end{gathered} \tag{45}$$

4.3 Transfer Function - Linear System

The system transfer functions is only dependent on A, B and C (similar to the poles, gains and zeros) and therefore the transfer functions for the deterministic and stochastic model is the same.
The transfer function can be determine in according to

$$\begin{bmatrix} Y_1(s)\\ Y_2(s) \end{bmatrix} = C\,(s\,I - A)^{-1}\,B \begin{bmatrix} U_1(s)\\ U_2(s) \end{bmatrix} \tag{46}$$

The determined transfer functions for the system is determined to be:

$$\begin{aligned} \frac{y_1}{u_1} &= \frac{0.001526}{s + 0.007475} \\ \frac{y_1}{u_2} &= \frac{9.456 \cdot 10^{-6}}{s^2 + 0.01871\, s + 8.397 \cdot 10^{-5}} \\ \frac{y_2}{u_1} &= \frac{1.142 \cdot 10^{-5}}{s^2 + 0.01704\, s + 6.928 \cdot 10^{-5}} \\ \frac{y_2}{u_2} &= \frac{0.001789}{s + 0.006703} \end{aligned} \tag{47}$$

4.4 Comparison of Linear System and Step Response

The time constants (τ) and the gains (K) from the linear model is compared with the time constants and gains determined from the transfer functions in section 3.4 as seen in table 2. The time constant for the LTI system is determined by $\tau = 1/|p_i|$ where p_i is the i^{th} pole

	K		τ	
	Linear	TF	Linear	TF
y_1/u_1	0.2041	0.2083	133.8	138
y_1/u_2	0.1126	0.1144	89 133.8	90 138
y_2/u_1	0.1648	0.1686	96.7 149.2	80 173
y_2/u_2	0.2669	0.2736	149.2	153

Table 2: Comparison of transfer function parameters

As expected, there is a difference in the gains and time constants determined from the linear model and the step responses. Due to non-linearity of the step response, the results is affected by the size of the step due to the system is "moving away" from the steady state value.

4.5 Discrete Time State Space Models

4.5.1 Deterministic Model

In order to determine the solution to the state space in discrete time, the sampling time is given by $T_s = t_{k+1} - t_k$, where k is the given iteration. We define a variable $\eta = t_{k+1} - \tau$, where τ is a time instance. This yields an expression for the discrete time system. Notice, that the discretaization process do not affect the output equation of the state space model, and it is therefore not shown.

$$x_{k+1} = e^{A\,T_s} x_k + \int_0^{T_s} e^{A\,\eta} B\, u(\eta) d\eta + \int_0^{T_s} e^{A\,\eta} G\, d(\eta) d\eta \tag{48}$$

$u(\eta)$ and $d(\eta)$ is assumed to be constant between the sampling instants, and therefore can the solution be simplified in according to

$$x_{k+1} = A_k\, x(k) + B_k\, u(k) + G_k\, d(k) \tag{49}$$

The matrices A_k, B_k and G_k can be determine in respect to eq. (48). It is also possible to determine the matrices in more a straighforward method given by

$$\exp\left(\begin{bmatrix} A_c & B_c & E_c \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} T_s\right) = \begin{bmatrix} A_k & B_k & G_k \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} \rightarrow$$

$$A_k = \begin{bmatrix} 0.8939 & 0 & 0.1465 & 0 \\ 0 & 0.9043 & 0 & 0.1365 \\ 0 & 0 & 0.8449 & 0 \\ 0 & 0 & 0 & 0.8564 \end{bmatrix} \quad B_k = \begin{bmatrix} 8.2300 & 0.3685 \\ 0.4487 & 9.7040 \\ 0 & 4.4174 \\ 5.8359 & 0 \end{bmatrix} \quad G_k = \begin{bmatrix} 1.1516 & 0 \\ 0 & 1.0685 \\ 13.8044 & 0 \\ 0 & 13.8950 \end{bmatrix} \tag{50}$$

4.5.2 Stochastic Model (piecewise Constant)

The stochastic model seen in section 2.2 was determined under the assumption that the disturbance is piecewise constant between each sampling, and therefore the discrete time model for the stochastic model is the same as seen in the previous paragraph.

4.6 Impulse Response Coefficient Based on Linear Model

The discrete time state space model is now used to determine the impulse response. The response is determined as seen in eq. (36) in section 3.6. The parameters can be seen in the below figure where they are compared to the response determined on basis of the transfer function.

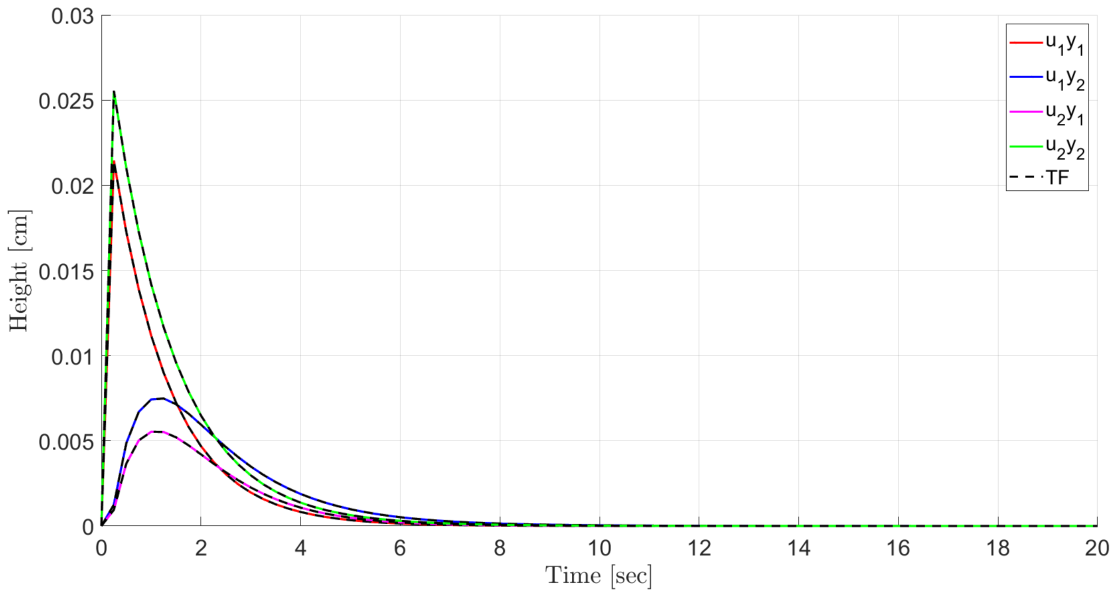

Figure 12: Impulse reponse - Comparison

The black-dotted line indicates the previous determine response based om the transfer functions. It can be seen that the difference between the response based on the discretized model and the transfer functions is more or less zero. It is expected that if the response was compared for larger step responses, a larger deviation would be expected as consequence of the linearization.

4.7 Linearization Approach

In order to determine discrete time state space model, it is required to have a exact continuous model of the system. The model is derived from the law of physics and the precision of this is determined under some assumptions. If one uses to many assumptions, in order to simplify the model, it could happen that the final model is not precise enough. Therefore the modelling of a system is often non-trivial. The advantage of the transfer functions from the step responses, is that they can be determined without any modelling, if the system exist and it is possible to do some experiments. By doing this, all the physical properties is taken into account when determining the model.

5 State Estimation for the Discrete-Time Linear System

5.1 State Space Models

The continuous time model models determined in section 4.1.1 is determined by the following:

$$A = \frac{\partial f}{\partial x}(x_{ss}, u_{ss}, d_{ss}) \quad B = \frac{\partial f}{\partial u}(x_{ss}, u_{ss}, d_{ss}) \quad G = \frac{\partial f}{\partial d}(x_{ss}, u_{ss}, d_{ss}) \quad C = \frac{\partial g}{\partial x}(x_{ss}, u_{ss}, d_{ss}) \tag{51}$$

The discrete time state space for the deterministic model is given by:

$$\begin{gathered} x(k+1) = A_k\, x(k) + B_k\, u(k) + G_k\, d(k) \qquad X(k=0) = x_0 \\ y_k = C\, x(k) \end{gathered} \tag{52}$$

Where the matrices can be determined as:

$$A_k = e^{A\,T_s} \quad B_k = \int_0^{T_s} e^{A\,T_s} B\, dt \quad G_k = \int_0^{T_s} e^{A\,T_s} G\, dt \tag{53}$$

The discrete time state space for the stochastic model where the noise is assumed to be piecewise constant is given by:

$$\begin{gathered} x(k+1) = A_k\, x(k) + B_k\, u(k) + G_k\, d(k) \qquad X(k=0) = x_0 \\ y_k = C\, x(k) + v_k \qquad v \sim N(0, R_{vv}(p)) \end{gathered} \tag{54}$$

Due to the fact that the noise is piecewise constant, the matrices for the deterministic and stochastic model is the same.

5.2 Static and Dynamic Kalman Filter

First, the general static and dynamic Kalman filter will be derived. The one step prediction of the Kalman filter is given by

$$\begin{gathered} R_{e,k} = C\, P_{k|k-1}\, C^T + R \\ K_{fx,k} = P_{k|k-1}\, C^T\, R_{e,k}^{-1} \\ P_{k+1|k} = A\, P_{k|k}\, A^T + G\, Q_{k|k}\, G^T - K_{fx,k}\, R_{e,k}\, K_{fx,k}^T \end{gathered} \tag{55}$$

Where R_e is the covariance of the measurement noise, K_{fx} is the Kalman filter gain, P is the covariance of the states and $Q_{k|k}$ can be determined by

$$Q_{k|k} = Q - S\, R_{e,k}^{-1}\, R_{e,k}\, (S\, R_{e,k}^{-1})^T \tag{56}$$

Q is the state noise, and is determined by eq. (57) and S is the correlation between the states. In this assignment, the correlation is assumed to be 0, causing $S = 0$. It is clearly then seen that $Q_{k|k} = Q$, which simplified is determined as.

$$\begin{bmatrix} \phi_{11} & \phi_{12} \\ \phi_{21} & \phi_{22} \end{bmatrix} = \exp\left(\begin{bmatrix} -A_c & G_c G_c^T \\ 0 & A_c^T \end{bmatrix} T_s\right) \qquad Q = \phi_{22}^T \phi_{12} \tag{57}$$

$P_{k+1|k}$ converges to the matrix P (as $t \to \infty$) which is the solution to the discrete algebraic Riccati equation ($DARE$). The finite elements in the one step prediction can be determined as:

$$\begin{aligned} R_e &= C\,P\,C^T + R \\ K_{fx} &= P\,C^T\,R_e^{-1} \\ P &= A\,P\,A^T + G\,Q\,G^T - K_{fx}\,R_e\,K_{fx}^T \end{aligned} \tag{58}$$

The general dynamic and static Kalman filter algorithm can be seen in table 3. Notice this algorithm is adjusted relating to the assumption of $S = 0$, resulting in the filter gain to become 0.
Since the disturbance is unknown, the system should be augmented in order to properly estimate the states. This leads the state space model of the system to take the following form.

$$\begin{aligned} x(k+1) &= A_{aug}\,x(k) + B_{aug}\,u(k) + G_{aug}\,d(k) \\ y(k) &= C_{aug}\,x(k) + v(k) \end{aligned} \tag{59}$$

It is seen, that the unknown disturbances are stochastic variables $d(k) = N(0, \sqrt{Q})$. The augmented matrices is determined according to eq. (60). Notice, that also the state noise matrix Q likewise is augmented.

$$A_{aug} = \begin{bmatrix} A_k & E_k \\ 0 & I \end{bmatrix} \quad B_{aug} = \begin{bmatrix} B_k \\ 0 \end{bmatrix} \quad G_{aug} = \begin{bmatrix} G_k \\ 0 \end{bmatrix} \quad C_{aug} = \begin{bmatrix} C & 0 \end{bmatrix} \quad Q_{aug} = \begin{bmatrix} Q & 0 \\ 0 & I \end{bmatrix} \tag{60}$$

The augmented states is used in the Kalman algorithm. Notice that some of the steps in the algorithm is similar for both the static and dynamic filter.

	Dynamic	Static
Measurement noise	$R_{e,k} = C_{aug}\,P_{k\|k+1}\,C_{aug}^T + R$	$R_e = C_{aug}\,P\,C_{aug}^T + R$
Filter gain	$K_{fx,k} = P_{k\|k+1}\,C_{aug}^T\,R_{e,k}^{-1}$	$K_{fx} = P\,C_{aug}^T\,R_e^{-1}$
Measurement update	$y_{k\|k-1} = C\,x_{k\|k-1} + v_k$	
Estimated measurement	$\hat{y}_{k\|k-1} = C_{aug}\,\hat{x}_{k\|k-1} + v_k$	
Error	$e_k = y_k - \hat{y}_{k\|k-1}$	
Estimated states	$\hat{x}_{k\|k} = \hat{x}_{k\|k-1} + K_{fx,k}\,e_k$	$\hat{x}_{k\|k} = \hat{x}_{k\|k-1} + K_{fx}\,e_k$
One step prediction	$\hat{x}_{k+1\|k} = A_{aug}\,\hat{x}_{k\|k} + B_{aug}\,u_{k\|k} + G_{aug}\,d_{k\|k}$	
Filtered state	$P_{k\|k} = P_{k\|k-1} - K_{fx,k}\,R_{e,k}\,K_{fx,k}$	$P_\infty = P - K_{fx}\,R_e\,K_{fx}$
Covariance prediction step	$P_{k+1\|k} = A_{aug}\,P_{k\|k}\,A_{aug}^T + G_{aug}Q_{aug}G_{aug}^T$	P

Table 3: Generel Kalman filter algorithm

Further more, it should be noted that the filter gain in order to minimize the noise ($K_{fw,k}$) is not shown, since it is 0, due to the fact that $S = 0$ (no correlation of the state noise).

5.2.1 Deterministic model

For the deterministic model, the proces noise and measurement noise is 0, however the covariance of the measurement noise shall be $R > 0$, in order to find a solution to the *DARE*.

	Dynamic	Static
Measurement noise	$R_{e,k} = C_{aug}\, P_{k\|k+1}\, C_{aug}^T + R$	$R_e = R$
Filter gain	$K_{fx,k} = P_{k\|k+1}\, C_{aug}^T\, R_{e,k}^{-1}$	$K_{fx} = 0$
Measurement update	$y_{k\|k-1} = C\, x_{k\|k-1}$	
Estimated measurement	$\hat{y}_{k\|k-1} = C_{aug}\, \hat{x}_{k\|k-1}$	
Error	$e_k = y_k - \hat{y}_{k\|k-1}$	
Estimated states	$\hat{x}_{k\|k} = \hat{x}_{k\|k-1}$	
One step prediction	$\hat{x}_{k+1\|k} = A_{aug}\, \hat{x}_{k\|k} + B_{aug}\, u_{k\|k} + G_{aug}\, d_{k\|k}$	
Filtered state	$P_{k\|k} = P_{k\|k-1} - K_{fx,k}\, R_{e,k}\, K_{fx,k}$	$P_\infty = 0$
Covariance prediction step	$P_{k+1\|k} = A_{aug}\, P_{k\|k}\, A_{aug}$	0

Table 4: Kalman filter algorithm deterministic system

5.2.2 Stochastics model (piecewise constant)

The algorithm for the dynamic and static Kalman filter is seen below.

	Dynamic	Static
Measurement noise	$R_{e,k} = C_{aug}\, P_{k\|k+1}\, C_{aug}^T + R$	$R_e = C_{aug}\, P\, C_{aug}^T + R$
Filter gain	$K_{fx,k} = P_{k\|k+1}\, C_{aug}^T\, R_{e,k}^{-1}$	$K_{fx} = P\, C_{aug}^T\, R_e^{-1}$
Measurement update	$y_{k\|k-1} = C\, x_{k\|k-1} + v_k$	
Estimated measurement	$\hat{y}_{k\|k-1} = C_{aug}\, \hat{x}_{k\|k-1} + v_k$	
Error	$e_k = y_k - \hat{y}_{k\|k-1}$	
Estimated states	$\hat{x}_{k\|k} = \hat{x}_{k\|k-1} + K_{fx,k}\, e_k$	$\hat{x}_{k\|k} = \hat{x}_{k\|k-1} + K_{fx}\, e_k$
One step prediction	$\hat{x}_{k+1\|k} = A_{aug}\, \hat{x}_{k\|k} + B_{aug}\, u_{k\|k} + G_{aug}\, d_{k\|k}$	
Filtered state	$P_{k\|k} = P_{k\|k-1} - K_{fx,k}\, R_{e,k}\, K_{fx,k}$	$P_\infty = P - K_{fx}\, R_e\, K_{fx}$
Covariance prediction step	$P_{k+1\|k} = A_{aug}\, P_{k\|k}\, A_{aug}^T + G_{aug} Q_{aug} G_{aug}^T$	P

Table 5: Kalman filter algorithm stochastic system

5.2.3 Simulation of Kalman Filter

A simulation of the Kalman filter performance can be seen below. For the simulation, the system is initialized from the steady state where $u = \begin{bmatrix}300 & 300\end{bmatrix}^T$ and $d = \begin{bmatrix}250 & 250\end{bmatrix}^T$, with a step on the input of 10% and the noise for the stochastic model is modelled as piecewise constant, where the covariance Q, the measurement noise covariance R and the stochastic disturbance covariance R_G has the following values (with $T_s = 15$[sec]).

$$Q = \begin{bmatrix} 0.1152 & 0 & 1.0314 & 0 \\ 0 & 0.0994 & 0 & 0.9653 \\ 1.0314 & 0 & 12.7341 & 0 \\ 0 & 0.9653 & 0 & 12.8971 \end{bmatrix} \qquad R = 0.5 \qquad R_G = 5 \tag{61}$$

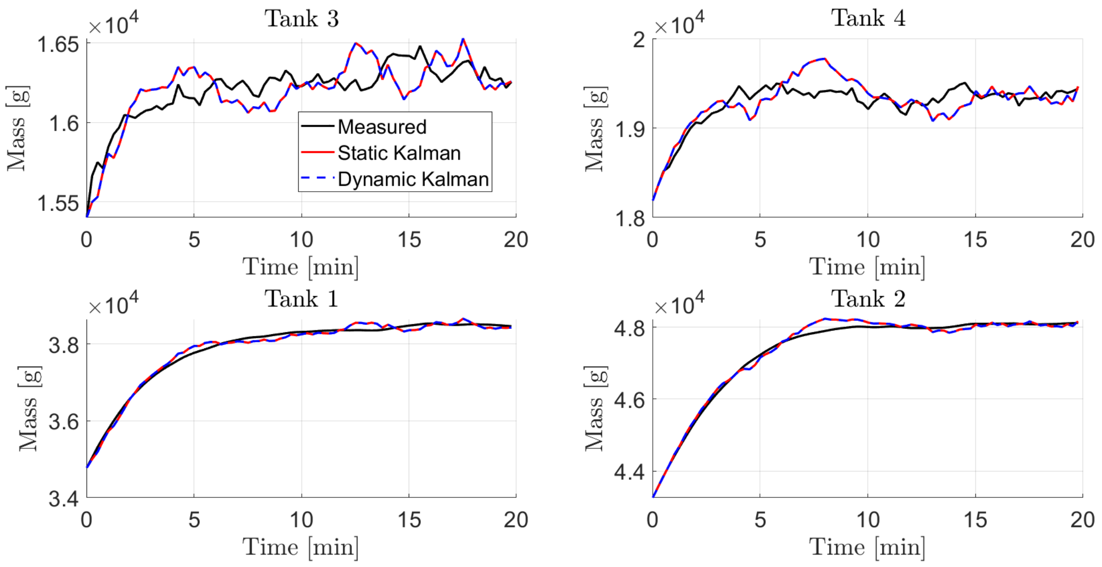

Figure 13: Kalman filter - Stochastic model states

In the above figure, the states (mass in each tank) is shown. Notice, that the states is not affected directly by the measurement noise and therefore it is not shown here. It is clearly seen, that the estimate of tank 3 and 4 is poor in the beginning compared to tank 1 and 2. The reason for this, is that the system do not have a measurement of tank 3 and 4, as for tank 1 and 2 which is used to correct the estimates. The response of the dynamic and static Kalman filter is more less the same.

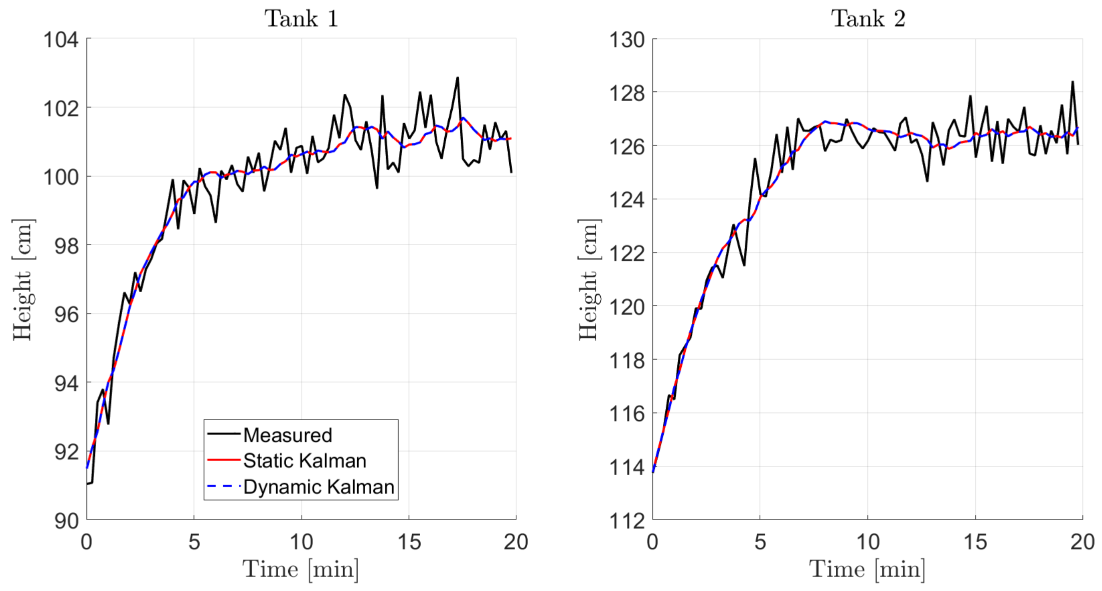

Figure 14: Kalman filter - Stochastic model outputs

In the above figure, the measured output and the estimated outputs is shown. Is is clearly seen that the measured values are corrupted by the measurements noise. The Kalman filter is able to estimate the output relatively well despite the noise.

5.3 Static and Dynamic Kalman Filter - Step on Disturbance

In this section it is desired to evaluate the performance of the Kalman filter, where a step change occurs in the unknown disturbance. In the previous paragraph, the system was augmented in order to have a proper estimation of the unknown disturbance. This model, is the same used in this paragraph. The consequence of the system not to be augmented, would be that the influence of step change of the disturbance, would move the system away from the linearization point, which is the foundation for the Kalman filter gains. Therefore would the original gains, not be sufficient to drive the difference between the output of the Kalman filter and the true output to be 0. The simulation is carried out as described in the previous paragraph. After 20 minutes, a step change of 10% occurs in the unknown disturbance.

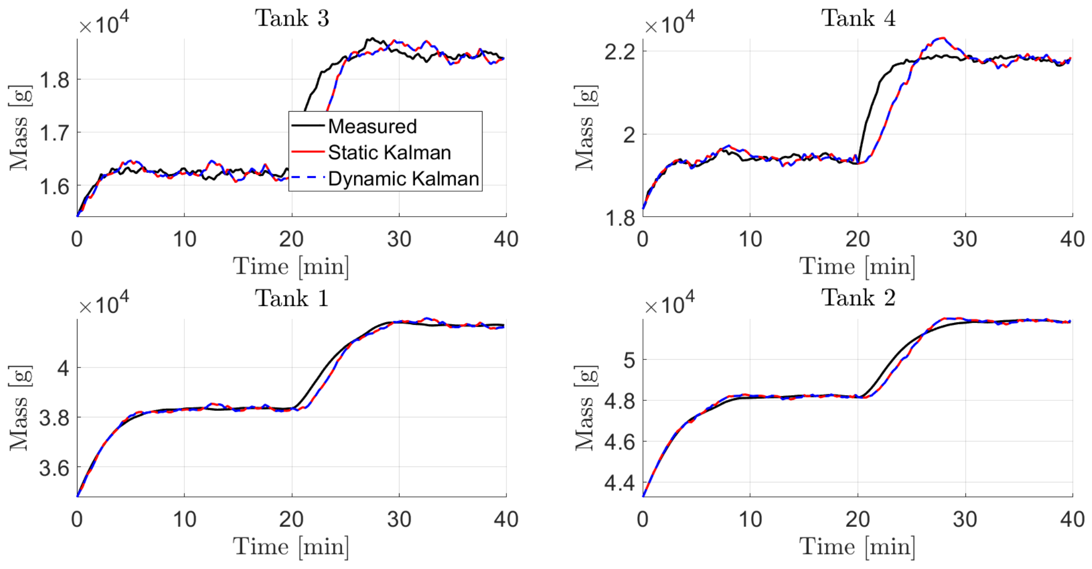

Figure 15: Kalman filter - Stochastic model states

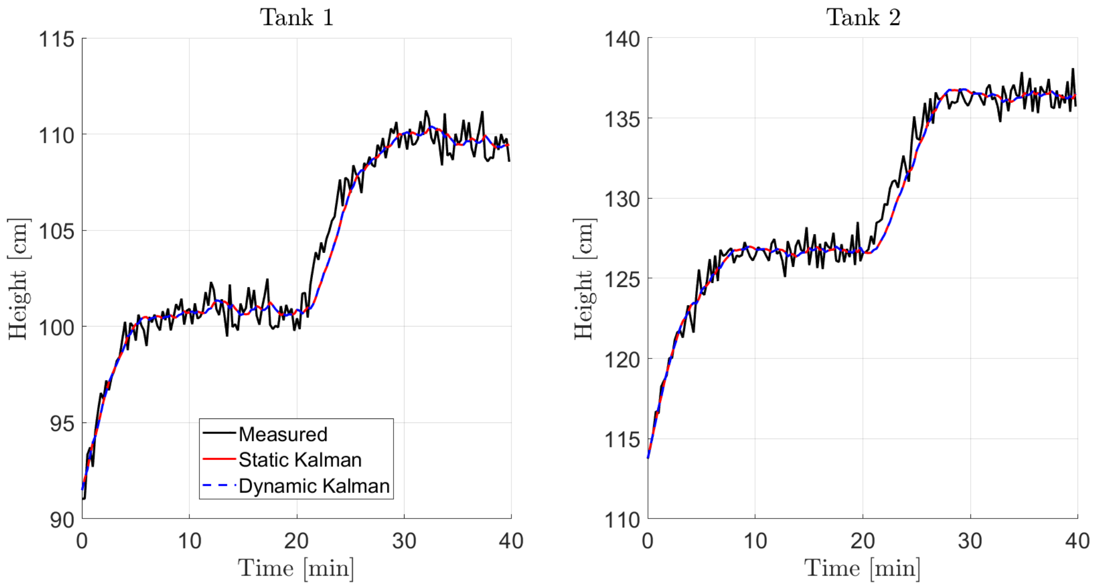

Figure 16: Kalman filter - Stochastic model outputs

It is seen that the Kalman filter is able to estimate the states when the system is perturbed with a step change in the disturbance of 10%. The Kalman filter also filters the noise as seen on the output. From the estimated states in tank 3 and 4, it can be seen that when a step occurs in the unknown disturbance, there is delay before the Kalman filter is able to adjust the estimate. This is corresponding to the delay from when the step occurs, until the water level raises in tank 1 and 2, so the estimates can use the measurements to adjust.

5.4 Kalman Filter Performance - Non Linear Model

In this section, the Kalman filter is implemented on the non linear system based on the same inputs etc. as for the linear model.

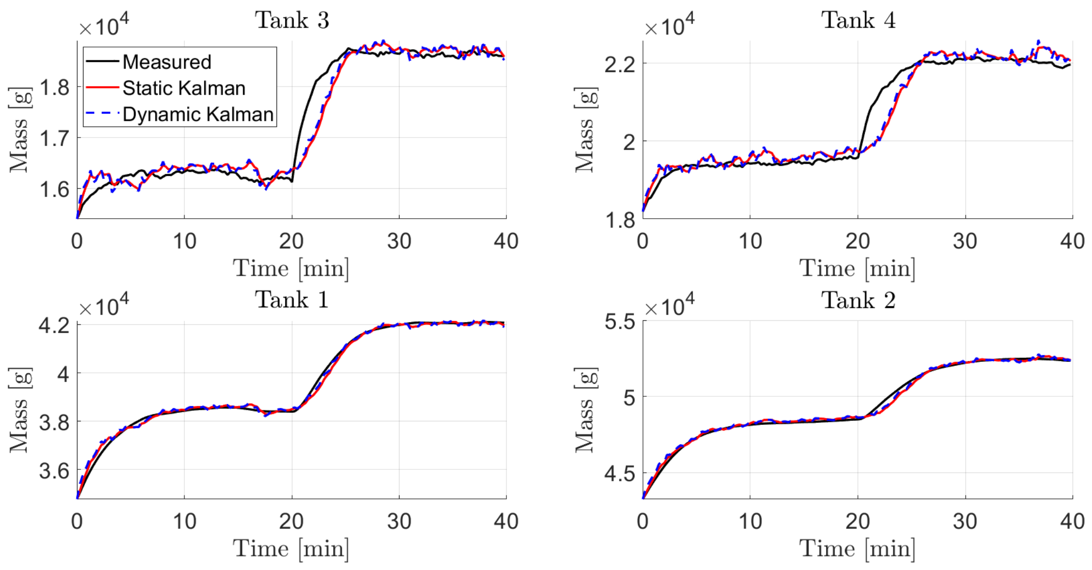

Figure 17: Kalman filter - Non-linear system - Stochastic model states

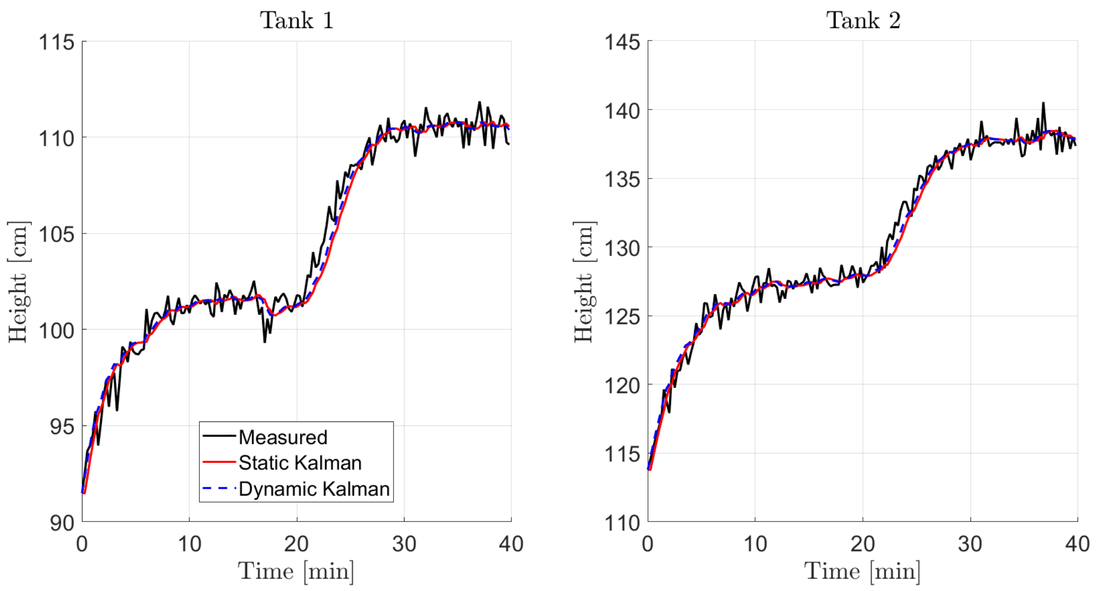

Figure 18: Kalman filter - Non-linear system - Stochastic model outputs

As seen, the difference between the Kalman filter performance on the linear and non-linear model is quite similar. However when the step on the disturbance have occurred, a constant offset (in the states) is seen which is due to the non-linearity's.

6 QP solver interface

In this section, the implementation of the *QP Solver* will be explained.
The *QPSolver* function is used for the solution of the convex quadratic program. The *QPSolver* uses the inbuild *quadprog* function from *MatLab*, which finds the minimum for a problem, specified in eq. (62) (left side).

$$\begin{array}{cc} \underset{x\in R^n}{min} & \phi = \frac{1}{2}x^THx + g^Tx \\ s.t. & l \leq x \leq u \\ & b_l \leq Ax \leq b_u \end{array} \quad \rightarrow \quad \begin{array}{cc} \underset{x\in R^n}{min} & \phi = \frac{1}{2}x^THx + g^Tx \\ s.t. & l \leq x \leq u \\ & Ax \leq b_u \\ & -Ax \leq -b_l \end{array} \tag{62}$$

The reasoning for this rephrasing of the *QP Solver* is due to the structure of *quadprog* function. The elements will be described in the specific MPC functions in the following paragraphs.

7 Unconstrained MPC

7.1 Design of Unconstrained MPC - Point 1, 2 & 3

The objective of the MPC is to be minimized, which can be described as a least square problem (for unconstrained MPC), which for a discrete time state space is given by

$$\underset{x\in R^n}{f(x)} = \frac{1}{2} = \|Ax - b\|_2^2 = \frac{1}{2}x^TA^TAx - (A^Tb)^Tx + \frac{1}{2}b^Tb \tag{63}$$

Where the expression can be rewritten in according to $H = A^TA$, $g = -A^Tb$ and $\rho = \frac{1}{2}b^Tb$. The unconstrained optimization problem is defined for the vector case as seen below

$$\underset{x\in R^n}{f(x)} = \frac{1}{2}x^THx + g^Tx \tag{64}$$

Where $\nabla f(x) = Hx + g = 0$ and $\nabla^2 f(x) = H \geq 0$ is necessary and sufficient optimality conditions. H is positive definite and the solution is unique.
The MPC aims to find e.g. an input value, for which a predetermined function has a minimum. The minimization is determine based on weights S and Q (which is the tuning parameters). The MPC is designed for the linear discrete time model determined in section 4.5. It is desired that the control input variation and the error on the output should be minimized, which yields the following least square problem.

$$\min \quad \phi = \frac{1}{2}\sum_{k=0}^{N}\|z(k) - r_k\|_{Q_z}^2 + \frac{1}{2}\sum_{k=0}^{N-1}\|\Delta u_k\|_S^2 \tag{65}$$

Where N is the prediction horizon, $z(k)$ is the output vector (which can be determine in according to eq. (66)), $r(k)$ is the reference vector, Q_z and S is the weight matrices and

$\Delta U_k = U_k - U_{k-1}$.
First the minimized output (1st part of the equation) is analyzed.

$$Z = \phi\, x_0 + \Gamma\, U + \Gamma_d\, D \tag{66}$$

Notice that since the disturbance is unknown, D is unknown. Hence, the matrix Γ_d is not used in the simulation. Each element is determined as

$$Z = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_N \end{bmatrix} \quad \phi = \begin{bmatrix} CA \\ CA^2 \\ CA^3 \\ \vdots \\ CA^N \end{bmatrix} \quad \Gamma = \begin{bmatrix} H_1 & 0 & 0 & \dots & 0 \\ H_2 & H_1 & 0 & \dots & 0 \\ H_3 & H_2 & H_1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ H_N & H_{N-1} & H_{N-2} & \dots & H_1 \end{bmatrix} \tag{67}$$
$$H_N = CA^{i-1}B$$

The expression of $z(k)$ is now substituted into the first part of the least square problem (containing the output).

$$\phi_z = \frac{1}{2}\|\phi\, x_0 + \Gamma\, U - R\|^2_{Q_z} \tag{68}$$

It is desired to have the form seen in eq. (64), why the above expression is rewritten by using linear algebra, which yields

$$\phi_z = \frac{1}{2}U^T \underbrace{(\Gamma^T Q_z \Gamma)}_{H} U + \underbrace{(-\Gamma^T Q_z K)^T}_{g} U + \underbrace{\frac{1}{2}K^T Q_z K}_{\rho} \quad , \qquad K = R - \phi x_0 \tag{69}$$

Now the minimized control input variation (2nd part) is analyzed.

$$\phi_{\Delta_u} = \frac{1}{2}\, U^T H_s U + (M_{u1}\, u_{-1})^T U \tag{70}$$

Where each element matrix is determined as

$$U = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_N \end{bmatrix} \qquad H_s = \begin{bmatrix} 2S & -S & 0 & \dots & 0 \\ -S & 2S & -S & \dots & 0 \\ 0 & -S & 2S & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & -S & S \end{bmatrix} \qquad M_{u1} = -\begin{bmatrix} S \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Now the two optimizations objects is added together, so the complete minimization is

$$\underset{U}{min}\, \phi = \phi_z + \phi_{\Delta u} = \frac{1}{2}U^T H U + g^T U \tag{71}$$

Where

$$\begin{aligned} H &= H_z + H_s = \Gamma^T Q_z \Gamma + H_s \\ g &= M_{x0}\, x_0 + M_R\, R + M_{u1}\, u_{-1} \\ M_{x0} &= \Gamma^T Q_z \phi \quad M_R = -\Gamma^T Q_z \end{aligned} \tag{72}$$

The implementation in *MatLab* is carried out through three functions. Prior to the simulation, the MPC design i carried out (see appendix A.I) where, H, H_s and g is determined. In order to do this, the function calls a seperate function (see appendix A.II) where ϕ and Γ is determined. Finally, the qpsolver is called. Notice, that since the simulation is unconstrained, the only inputs are H, H_s and g (see appendix A.III)

8 Input Constrained MPC

8.1 Design of Input Constrained MPC - Point 1, 2 & 3

In this section, the MPC is designed with constraints relating to the input. The model uses the state space model and least square problem, is the same as seen in eq. (65) in the previous paragraph. The MPC is allowed to determine a value, which satisfies the control input to be between u_{min} and u_{max} for all steps (N) in the prediction horizon. Besides the constrain of the minimum and maximum of the control input, also the variation between each step is limited between Δu_{min} and Δu_{max}.

$$u_{min} \leq u_k \leq u_{max} \qquad \Delta u_{min} \leq u_k \leq \Delta u_{max} \qquad k = 0, 1, \ldots, N-1 \tag{73}$$

$$\begin{bmatrix} u_{min} \\ u_{min} \\ \vdots \\ u_{min} \end{bmatrix} \leq \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix} \leq \begin{bmatrix} u_{max} \\ u_{max} \\ \vdots \\ u_{max} \end{bmatrix} \qquad \begin{bmatrix} \Delta u_{min} \\ \Delta u_{min} \\ \vdots \\ \Delta u_{min} \end{bmatrix} \leq \begin{bmatrix} u_0 - u_{-1} \\ u_1 - u_0 \\ \vdots \\ u_{N-2} - u_{N-1} \end{bmatrix} \leq \begin{bmatrix} \Delta u_{max} \\ \Delta u_{max} \\ \vdots \\ \Delta u_{max} \end{bmatrix} \tag{74}$$

For the deviation constraints (Δu) it is clearly seen that it is not straight forward to implement in the *QP Solver*. Therefore a row vector I_0 is defined which consists of N-number of rows, where the top row is the identity and the remaining elements is 0. This yields the deviation constraints on the input to take the following form:

$$\begin{bmatrix} \Delta u_{min} + u_{-1} \\ \Delta u_{min} \\ \Delta u_{min} \\ \vdots \\ \Delta u_{min} \end{bmatrix} \leq \underbrace{\begin{bmatrix} I & 0 & 0 & \ldots & 0 \\ -I & I & 0 & \ldots & 0 \\ 0 & -I & I & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \ldots & 0 & -I & I \end{bmatrix}}_{\Lambda} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{N-1} \end{bmatrix} \leq \begin{bmatrix} \Delta u_{max} + u_{-1} \\ \Delta u_{max} \\ \Delta u_{max} \\ \vdots \\ \Delta u_{max} \end{bmatrix} \tag{75}$$

With this, it is possible to take the minimization object seen in the previous paragraph, where now U is restricted by eq. (77)

$$\min_{U} \phi = \phi_z + \phi_{\Delta u} = \frac{1}{2} U^T H U + g^T U \tag{76}$$

$$\begin{gathered} U_{min} \leq U \leq U_{max} \\ \Delta u_{min} + I_0 u_{-1} \leq \Lambda U \leq \Delta u_{max} + I_0 u_{-1} \end{gathered} \tag{77}$$

The implementation in *MatLab* is carried out through three functions. Since the objective functions is the same as in the previous paragraph, the MPC design is identical (see appendix A.I) and the constant are the same (see appendix A.II). However, the qpsolver is not the same (see appendix B.I).

9 MPC with Input Constraints and Soft Output Constraints

9.1 Design of Input Constrained and Soft Output Constrained MPC - Point 1, 2 & 3

In the previous paragraph the MPC was implemented with hard constraints on the input, meaning that the solution will never determine a value which was outside these constraints. When adding constraints on the output, they are determined as soft constraints in order to make the solution more robust. The solution is once again build on the discrete time linear model, however the objective function will be changed to the following:

$$\min \quad \phi = \phi_z + \phi_{\Delta u} + \phi_u + \phi_s + \phi_t \tag{78}$$

Each of the elements will be described below. Since the qp solver has a specific interface as seen in section 6 eq. (62) the general expression for each of the elements of ϕ will be rewritten, in order to setup a complete expression for H, A, l, u, b_u and b_l.

Objective term relating to the output (Φ_z)

ϕ_z is determined as:

$$\phi_z = \frac{1}{2}\sum_{k=0}^{N} \|W_z(z_k - r_k)\|_2^2 \tag{79}$$

Where $Q_z = W_z^T W_z$, hence W_z is the tuning parameter. $r(k)$ is the reference signal and $z(k)$ is defined as

$$z(k) = \underbrace{\phi\, x_k + \phi_w\, w_k}_{b_k} + \Gamma\, U \tag{80}$$

Where ϕ and Γ was determined in section 7.1, and the remaining elements is determined as

$$Z_k = \begin{bmatrix} z_{k+1} \\ z_{k+2} \\ z_{k+3} \\ \vdots \\ z_{k+N} \end{bmatrix} \quad U_k = \begin{bmatrix} u_{k+1} \\ u_{k+2} \\ u_{k+3} \\ \vdots \\ u_{k+N} \end{bmatrix} \quad \phi_w = \begin{bmatrix} CE \\ CAE \\ CA^2E \\ \vdots \\ CA^{N-1}E \end{bmatrix}$$

The objective function is now rewritten to take the form used in the qp solver.

$$\phi_z = \frac{1}{2} U_k^T \underbrace{((\bar{W}_z\,\Gamma)^T \bar{W}_z\,\Gamma)}_{H_z} U_k + \underbrace{(-(\bar{W}_z\,\Gamma)^T \bar{W}_z\, c_k)}_{g_z}{}^T U_k + \underbrace{\frac{1}{2}(c_k \bar{W}_z)^T \bar{W}_z\, c_k}_{\rho_z} \tag{81}$$

Where

$$\bar{W}_z = I_N \otimes W_z \qquad c_k = R_k - b_k \tag{82}$$

Objective term relating to the input variation ($\Phi_{\Delta u}$)

$\phi_{\Delta u}$ is determined as.

$$\phi_{\Delta u} = \frac{1}{2}\sum_{k=0}^{N-1} \|W_{\Delta u}(\Delta u_k)\|_2^2 \tag{83}$$

Where $S = W_{\Delta u}^T W_{\Delta u}$, hence $W_{\Delta u}$ is the tuning parameter. The objective function is now rewritten to take the form used in the qp solver.

$$\phi_{\Delta u} = \frac{1}{2} U_k^T \underbrace{\left((\bar{W}_{\Delta u}\Lambda)^T \bar{W}_{\Delta u}\Lambda\right)}_{H_{\Delta u}} U_k + \underbrace{\left(-(\bar{W}_{\Delta u}\Lambda)^T \bar{W}_{\Delta u} I_0 u_{-1}\right)^T}_{g_{\Delta u}} U_k + \underbrace{\frac{1}{2}\left(\bar{W}_{\Delta u} I_0 u_{-1}\right)^T \bar{W}_{\Delta u} I_0 u_{-1}}_{\rho_{\Delta u}} \tag{84}$$

Where Λ was determined in section 8.1 and $\bar{W}_{\Delta u}$ is given by

$$\bar{W}_{\Delta u} = I_N \otimes W_{\Delta u} \tag{85}$$

Objective term relating to the input distance from a predetermined value (Φ_u)

ϕ_u allows the controller to penalize the input on basis of a predetermined value $\bar{u}_k$

$$\begin{aligned} \phi_u &= \tfrac{1}{2}\sum_{k=0}^{N-1} \left\|\bar{W}_u(u_k - \bar{u}_k)\right\|_2^2 \\ \phi_u &= \tfrac{1}{2} U_k^T \underbrace{\left(\bar{W}_u^T \bar{W}_u\right)}_{H_u} U_k + \underbrace{\left(-\bar{W}_u^T \bar{W}_u \bar{U}_k\right)^T}_{g_u} U_k + \underbrace{\frac{1}{2}\bar{U}_k \bar{W}_u^T \bar{W}_u \bar{U}_k}_{\rho_u} \end{aligned} \tag{86}$$

where

$$\bar{W}_u = I_N \otimes W_u \tag{87}$$

and W_u is a tuning parameter.

Objective term relating to the lower soft constraints (Φ_s)

The idea of implementing soft constraints on the output, is to allow the algorithm to violate these constraints, at high costs, in order to ensure the controller can find a optimal solution. This is done by using slack varaibles denoted S_k. ϕ_s is given by

$$\phi_s = \frac{1}{2}\sum_{k=0}^{N-1} \|W_{s,2}(s_k)\|_2^2 + \frac{1}{2}\sum_{k=0}^{N-1} \|W_{s,1}(s_k)\|_1 \quad , \quad \begin{matrix} Z_k \geq R_{min,k} - S_k \\ S_k \geq 0 \end{matrix} \tag{88}$$

Where $W_{s,1}$ and $W_{s,2}$ is tuning parameters. $R_{min,k}$ is the constraint the output should be larger than at all times, unless it is not possible. In the case where the output becomes smaller than R_{min}, S_k (see eq. (89)) becomes non zero. As mentioned, it is not desirable to violate the soft constraints, and therefore is the tuning $W_{s,1}$ and $W_{s,2}$ are often chosen to be relative high.

$$S_k = \begin{bmatrix} s_{k+1} \\ s_{k+2} \\ \vdots \\ s_{k+N} \end{bmatrix} \tag{89}$$

The objective term is rewritten into

$$\phi_s = \frac{1}{2} S_k^T \underbrace{\left(\bar{W}_{s,2}^T \bar{W}_{s,2}\right)}_{H_s} S_k + \underbrace{\left(\bar{W}_{s,1} e\right)^T}_{g_s} S_k \tag{90}$$

Where

$$\begin{aligned} \bar{W}_{s,1} &= I_N \otimes W_{s,1} \\ \bar{W}_{s,2} &= I_N \otimes W_{s,2} \\ e &= \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}^T \end{aligned} \tag{91}$$

Objective term relating to the upper soft constraints (Φ_t)

The idea of implementing a soft upper constraint is similar to the soft lower constraint. ϕ_t is given by

$$\phi_t = \frac{1}{2}\sum_{k=0}^{N-1}\|W_{t,2}(t(k))\|_2^2 + \frac{1}{2}\sum_{k=0}^{N-1}\|W_{t,1}(t(k))\|_1 \qquad , \qquad \begin{matrix} Z_k \leq R_{max,k} + T_k \\ T_k \geq 0 \end{matrix} \tag{92}$$

Where $W_{t,1}$ and $W_{t,2}$ is tuning parameters. Similar to the lower bound, thse should be chosen to be high. $R_{max,k}$ is the upper soft constraint and T_k is determined as.

$$T_k = \begin{bmatrix} t_{k+1} \\ t_{k+2} \\ \vdots \\ t_{k+N} \end{bmatrix} \tag{93}$$

The objective term is rewritten into

$$\phi_t = \frac{1}{2}T_k^T \underbrace{\left(\bar{W}_{t,2}^T \bar{W}_{t,2}\right)}_{H_t} T_k + \underbrace{\left(\bar{W}_{t,1} e\right)}_{g_t}{}^T T_k \tag{94}$$

Where

$$\begin{aligned} \bar{W}_{t,1} &= I_N \otimes W_{t,1} \\ \bar{W}_{t,2} &= I_N \otimes W_{t,2} \end{aligned} \tag{95}$$

Combination of objective function

Since all the objective functions has been determined individually, it is now combined. It is of interest to minimize, U_k, S_k and T_k, which yields

$$\min_{U_k,S_k,T_k} \quad \frac{1}{2}\begin{bmatrix} U_k \\ S_k \\ T_k \end{bmatrix}^T \underbrace{\begin{bmatrix} H & 0 & 0 \\ 0 & H_s & 0 \\ 0 & 0 & H_t \end{bmatrix}}_{H} \begin{bmatrix} U_k \\ S_k \\ T_k \end{bmatrix} + \underbrace{\begin{bmatrix} g \\ g_s \\ g_t \end{bmatrix}}_{g}{}^T \begin{bmatrix} U_k \\ S_k \\ T_k \end{bmatrix} + \rho \tag{96}$$

The remaining inputs to the qp solver is given by.

$$\begin{gathered} \underbrace{\begin{bmatrix} U_{min,k} \\ 0 \\ 0 \end{bmatrix}}_{l} \leq \begin{bmatrix} U_k \\ S_k \\ T_k \end{bmatrix} \leq \underbrace{\begin{bmatrix} U_{max,k} \\ \infty \\ \infty \end{bmatrix}}_{u} \\ \underbrace{\begin{bmatrix} \Delta U_{min} + I_0 u_{-1} \\ R_{min,k} - b_k \\ -\infty \end{bmatrix}}_{b_l} \leq \underbrace{\begin{bmatrix} \Lambda & 0 & 0 \\ \Gamma & I & 0 \\ \Gamma & 0 & -I \end{bmatrix}}_{A} \begin{bmatrix} U_k \\ S_k \\ T_k \end{bmatrix} \leq \underbrace{\begin{bmatrix} \Delta U_{max} + I_0 u_{-1} \\ \infty \\ R_{max,k} - b_k \end{bmatrix}}_{b_u} \end{gathered} \tag{97}$$

The implementation in *MatLab* is carried out through three functions. Prior to the simulation, the MPC design i carried out (see appendix C.I) where, the matrices for all H, g and M is determined. In order to do this, the function calls the previous function (see appendix A.II) where ϕ and Γ is determined. Finally, the qpsolver is called. (see appendix C.II)

10 Closed-Loop Simulations

In this section the MPC's determined in section 7.1, 8.1 and 9.1 will be simulated on both the linear models and the non-linear models.
The reference signal is modelled as shown in table 6. The system is initialized from steady state, where $u = \begin{bmatrix}300 & 300\end{bmatrix}^T$. The disturbance is set be stochastic variable which follows a normal distribution given by: $d = N(250, \sqrt{10})$. The measurement noise is modelled as normal distributed according to $v_k = N(0, \sqrt{20})$.

Time [min]	Δr_1 [m]	Δr_2 [m]
0	0	0
10	10	10
20	20	-10

Table 6: Reference signal

For all simulations, the overall concept is done by the MPC predicts 2*N (where $N = 40$) number of inputs ahead. Then the first coming two rows in the prediction (due to the fact that we have two inputs) is saved as the input for the k$^{\text{th}}$ iteration. This is used in both the linear model (as a deviation variable) and in the non-linear model (as absolute value). Then the non-linear and linear states is determined and finally the measurement is done. The measurement are then used as inputs to the Kalman filter in order to estimate the states, and filter out measurement noise.

10.1 Unconstrained MPC - Closed Loop Simulation

In this section the unconstrained MPC is simulated on both the linear and non-linear system. For this simulation, the tuning is set to $Q = 100$ and $S = 0.01$. It should be noticed that the values shown in the plot is absolute values.

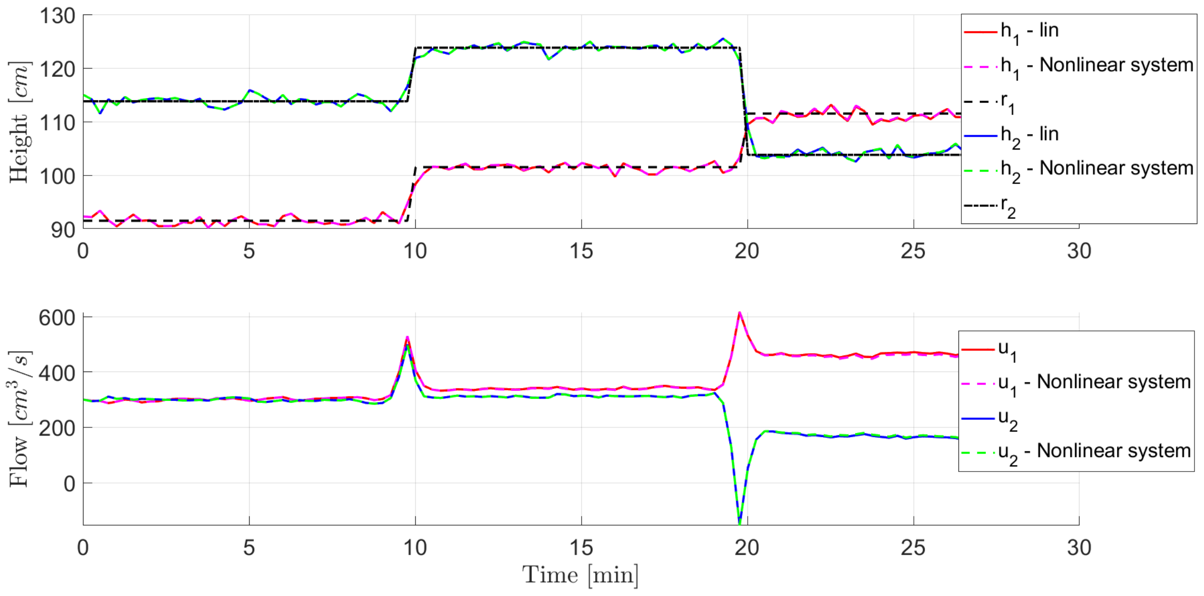

Figure 19: Unconstrained MPC - Simulation on linear and non-linear model

It can be seen that the response of the linear and non-linear system with the linear unconstrained MPC is quite similar. However, if one zooms in minor deviations can be seen, specially close to the steps (I would have expected to have a larger deviation, but could not figure out why it didn't appear). The MPC is able to track the reference signal fast, however it is clear that the inputs are relatively high and fast changing. Wheen evaluating u_2 at $t = 20\,[min]$ it is seen that the at negative flow is of course not realistic, but it is a consequence of the lack of constraints.

10.2 Input Constrained MPC - Closed Loop Simulation

In this section the input constrained MPC is simulated on both the linear and non-linear system. For this simulation, the tuning is set to $Q = 100$ and $S = 0.01$. The constraints is set to

$$u_{min} = 0 \qquad u_{max} = 450 \qquad \Delta u_{min} = -10 \qquad \Delta u_{max} = 10 \tag{98}$$

It should be noticed that the values shown in the plot is absolute values.

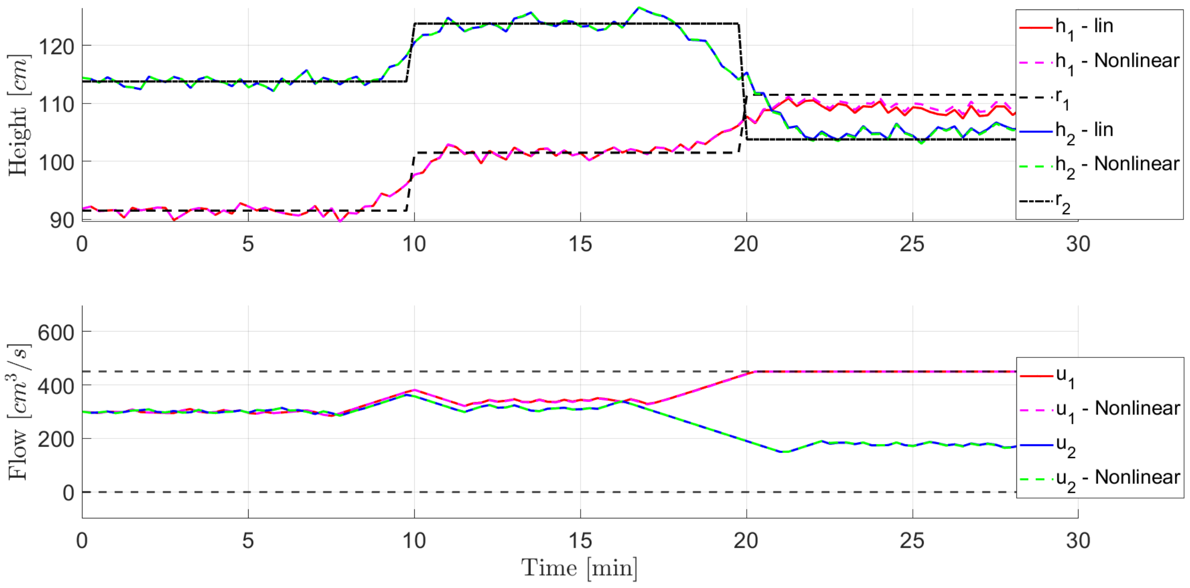

Figure 20: Input constrained MPC - Simulation on linear and non-linear model

As for the unconstrained MPC, the deviation between the linear and the non-linear system is very small. In the case of input constrained MPC, the settling time (in case of step changes in the reference) is longer which is intuitive when the inputs are constrained. It is also seen that the MPC try to regulate earlier (in relation to at which time the reference occurs). In case the constraints on Δu was lower, the regulation will begin earlier. It can be seen that the input 1 is reaching its maximum limit, which causes the reference in tank 1 to be unreachable (however very close).

10.3 Input Constrained and Soft Output Constrained MPC - Closed Loop Simulation

In this section the hard input and soft output constrained MPC is simulated on both the linear and non-linear system. For this simulation, the tuning is set as:

$$\begin{gathered} W_z = 100 \quad W_u = 0 \quad W_{du} = 0 \\ W_{t1} = 10000 \quad W_{t2} = 10000 \quad W_{s1} = 10000 \quad W_{s2} = 10000 \end{gathered} \tag{99}$$

The constraints is set to

$$u_{min} = 0 \quad u_{max} = 450 \quad \Delta u_{min} = -10 \quad \Delta u_{max} = 10 \quad Z_{min} = \begin{bmatrix} 85 \\ 105 \end{bmatrix} \quad Z_{max} = \begin{bmatrix} 110 \\ 120 \end{bmatrix} \tag{100}$$

It should be noticed that the values shown in the plot is absolute values.

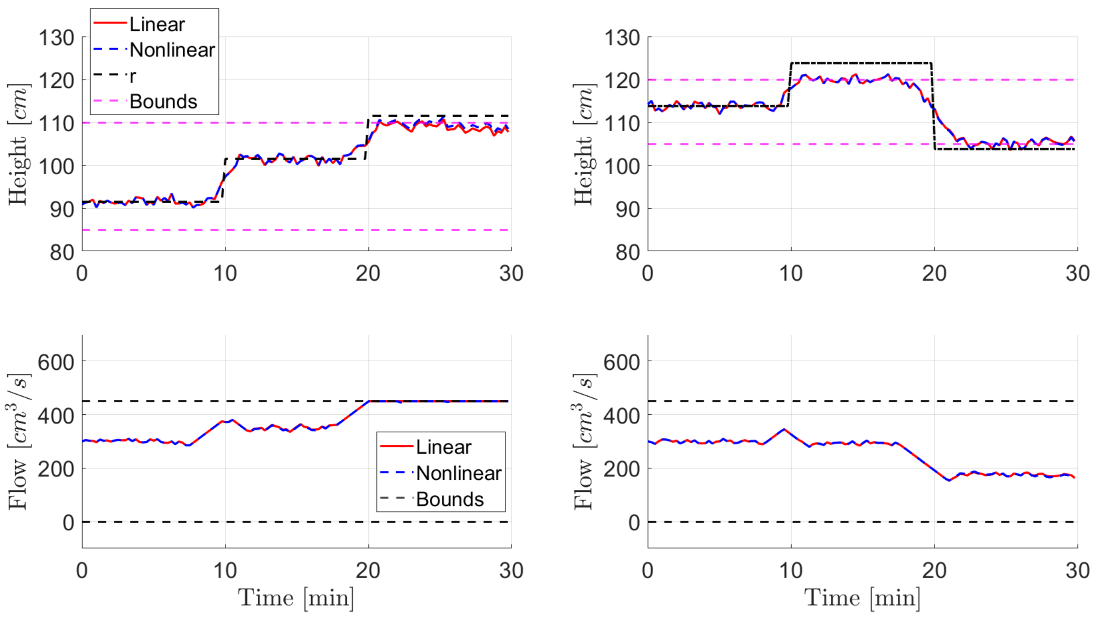

Figure 21: Input constrained output constrained MPC - Simulation on linear and non-linear model

By first analyzing the output in tank 1, it is seen that at $T = 20\,[min]$ the reference changes. The system is both limited by the input which is a hard constraint, and therefore the system is not able to reach desired output. For tank 2, the input constraints are far from the operation, so these will not limit the system. The output constraint is clearly seen from $T = 10 - 20\,[min]$ where the output is limited. The idea of a soft constraint is clearly seen since the controller allows system to trespass the value (which is caused by the present of noise). At $T = 20\,[min]$ the reference changes again, and here the lower output constraint limits the output. Once again, the presence of noise is causing the output to become lower than the limit in some small periods.

Due to the structure of the system, the output constraint on e.g. tank 2 can also affect tank 1. If the water level reached the constraint, this will limit the flow input. But since the flow from the pump affects both tanks, the dynamics in tank 1 can be affected.

11 Non-Linear MPC

11.1 Continious-Discrete Mathematical Model

The system can be represented as continious-discrete mathematical model according to the following:

$$
\begin{aligned}
x(t_0) &= \hat{x}_0 \qquad \hat{x}_0 \sim N(\hat{x}_0, \hat{P}_0) \\
dx(t) &= f\Big(x(t), u(t), d(t), \theta\Big)dt + \sigma\Big(x(t), u(t), d(t), \theta\Big)d\omega(t) \qquad d\omega(t) \sim N_{iid}(0, I\,dt) \\
y(t_k) &= g(x(t_k), \theta) + v(t_k) \qquad v(t_k) \sim N_{iid}(0, R(\theta))
\end{aligned}
\tag{101}
$$

Where the function f describes the drift and function σ describes the diffusion. The states and the outputs is given by

$$
\begin{aligned}
dx_1(t) &= \rho\left(\gamma_1\, u_1(t) + a_3\sqrt{2\,g\,\frac{x_3(t)}{\rho\,A_3}} - a_1\sqrt{2\,g\,\frac{x_1(t)}{\rho\,A_1}}\right)dt \\
dx_2(t) &= \rho\left(\gamma_2\, u_2(t) + a_4\sqrt{2\,g\,\frac{x_4(t)}{\rho\,A_4}} - a_2\sqrt{2\,g\,\frac{x_2(t)}{\rho\,A_2}}\right)dt \\
dx_3(t) &= \rho\left((1-\gamma_2)\, u_2(t) + d_1 + \omega(t) - a_3\sqrt{2\,g\,\frac{x_3(t)}{\rho\,A_3}}\right)dt \\
dx_4(t) &= \rho\left((1-\gamma_1)\, u_1(t) + d_1 + \omega(t) - a_4\sqrt{2\,g\,\frac{x_4(t)}{\rho\,A_4}}\right)dt \\
y_1(t_k) &= \frac{x_1(t_k)}{\rho\,A_1} + v_1(t_k) \\
y_2(t_k) &= \frac{x_2(t_k)}{\rho\,A_2} + v_2(t_k)
\end{aligned}
\tag{102}
$$

11.2 Continuous-Discrete Extended Kalman Filter

The Continuous-Discrete Extended Kalman Filter (*CDEKF*) consists of a time update (eq. (103) and a measurement update (eq. (104)). The advantage of using a *CDEKF* is the fact that the filter uses the non-linear model to estimate the states, where the clasical Kalman filter uses the linearized model.
Time update:

$$
\begin{aligned}
\hat{y}_{k|k-1} &= g(\hat{x}_{k|k-1}, \theta) \qquad C_k = \frac{\partial g}{\partial x}(\hat{x}_{k|k-1}, \theta) \\
e_l &= y_k - \hat{y}_{l|k-1} \\
R_{e,k} &= C_k P_{k|k-1} C_k^T + R_k \\
\hat{x}_{k|k} &= \hat{x}_{k|k-1} + K_k e_k \\
K_k &= P_{k|k-1} C_k^T R_{e,k}^{-1} \\
P_{k|k} &= P_{k|k-1]} - K_k R_{e,k} K_k^T
\end{aligned}
\tag{103}
$$

Measurement update:

$$
\begin{aligned}
\frac{dx(t)}{dt} &= f\Big(\hat{x}_k(t), u_k, d_k, \theta\Big) \qquad \hat{x}_k(t_k) = \hat{x}_{k|k} \\
\frac{dP_k(t)}{dt} &= A_k(t)P_k(t) + P_k(t)A_k(t)^T\sigma_k(t)\sigma_k(t)^T \qquad P_k(t_k) = P_{k|k} \\
A_k(t) &= \frac{\partial f}{\partial x}\Big(\hat{x}_k(t), u_k, d_k, \theta\Big) \\
\sigma_k(t) &= \sigma\Big(\hat{x}_k(t), u_k, d_k, \theta\Big)
\end{aligned}
\tag{104}
$$

The implementation of the *CDEKF* is implemented as seen in appendix D.I. The simulation of the system can be seen in the below figure. Notice that it is the states (hence the units is in gram), since these clearly visualize the non-linearity of the system where outputs in cm is purely a scaling. The system is initialized from steady state and a step change occurs in the input at time 10 and 20 minutes.

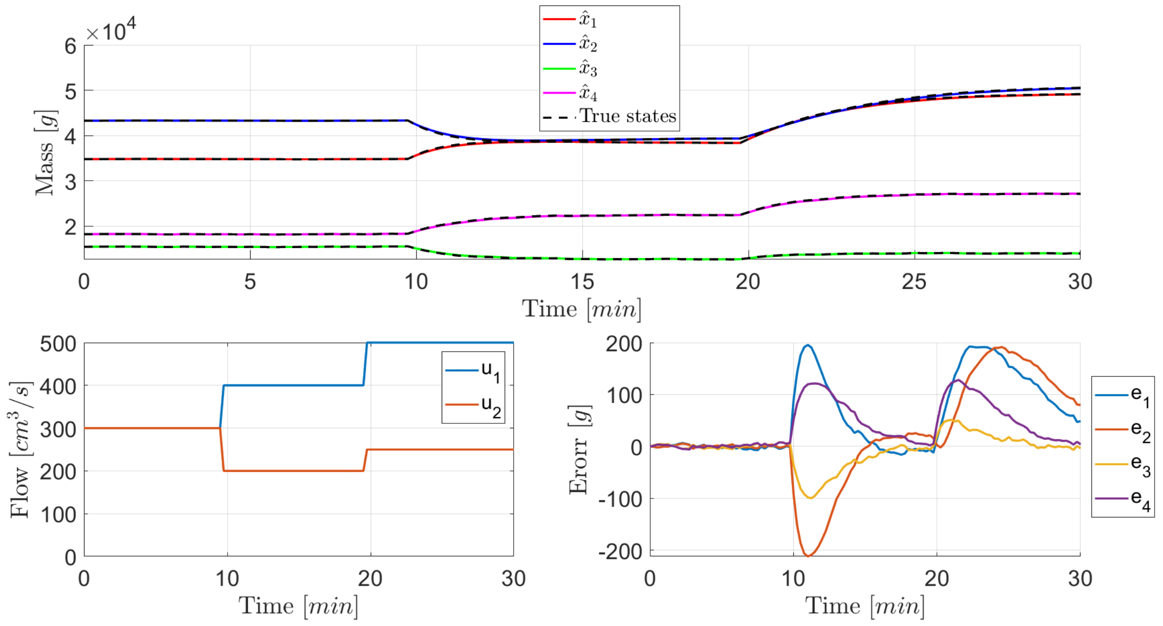

Figure 22: CDEKF implementation

It can be seen that the error between the true state and the estimated states are relatively small, even though the steps forces the system to move far from the steady state point.

11.3 Prediction-Error Method

The designed Kalman Filter in the previous paragraph, can be used to estimate unknown parameters. By using the the estimated outputs and and the error and covariance of this, it is possible to estimate parameters such as cross section area of the tanks by using the following method.

$$
e_k = y_k - \hat{y}_k(A_1, A_2) \quad \hat{y}_k(A_1, A_2) = \begin{bmatrix} \frac{1}{\rho A1}x_1 & \frac{1}{\rho A2}x_2 \end{bmatrix}^T \tag{105}
$$

The best estimation of A_1 and A_2 from the above equation can be used to minimize the following function.

$$
\underset{A1,A2}{min} \qquad V(A_1, A_2) = \sum_{0}^{k-1} \frac{1}{2}\ln\left(det(R_{e,k}) + \frac{1}{2}\Big(y_k - \hat{y}_k(A_1, A_2)\Big)^T R_{e,k}^{-1}\Big(y_k - \hat{y}_k(A_1, A_2)\Big)\right. \tag{106}
$$

By doing so, the minimization problem will be able to reconstruct the parameters with a relative good precision.

11.4 Bound-Constrained NMPC

Based on the determined non-linear stochastic model the bound-constrained Non-Linear MPC (*NMPC*) is determined by the following objective function. This is a optimal control problem, with multiple-shooting (with sensitivities).

$$\underset{x,u}{min} \qquad \phi_k = \phi_{z,k} + \phi_{u,k} + \phi_{\Delta u,k} \tag{107}$$

Where each part is determined as.

$$\begin{aligned} \phi_{z,k} &= \frac{1}{2}\int_{t_k}^{t_k+N*T_s} ||z(t) - \bar{z}(t)||^2_{Q_t} dt \\ \phi_{u,k} &= \frac{1}{2}\int_{t_k}^{t_k+N*T_s} ||u(t) - \bar{u}(t)||^2_{Q_u} dt \\ \phi_{\Delta u,k} &= \frac{1}{2}\int_{t_k}^{t_k+N*T_s} ||\Delta u_{k+j}||^2_{Q_{\Delta u}} \end{aligned} \tag{108}$$

The implementation for the *NMPC* is given by

$$\begin{gathered} x(t_k) = \hat{x}_{k|k} \\ \dot{x}(t) = f(x(t), u(t), \theta), \qquad t_k \leq t_k + T_p \\ z(t) = \dot{h}(x(t), \theta), \qquad t_k \leq t \leq t_k + T_p \\ u(t) = u_{k+j|k}, \qquad j \in N,\ t_{k+j} \leq t < t_{k+j+1} \\ u_{min} \leq u_{k+j|k} \leq u_{max}, \qquad j \in N \\ \Delta u_{min} \leq \Delta u_{k+j|k} \leq \Delta u_{max}, \qquad j \in N \end{gathered} \tag{109}$$

11.5 Comparison of Ordinary MPC and NMPC

In this section, the Non-linear MPC is tested. The files for the simulation is handed out from the Professor in the course.

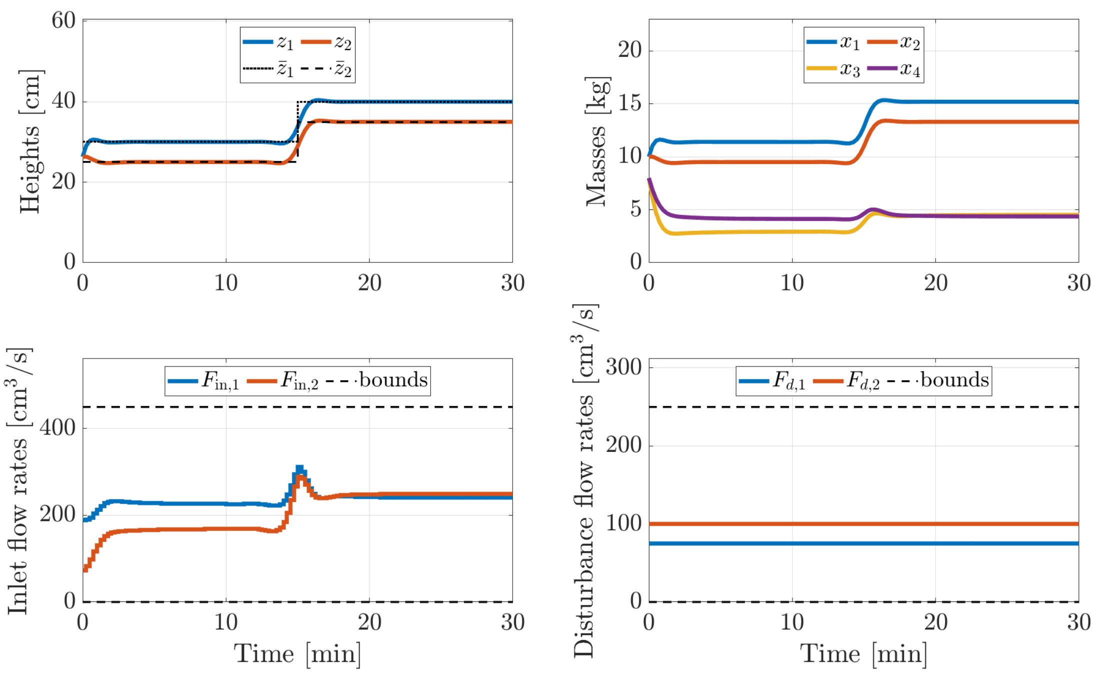

Figure 23: Non-linear MPC - Simulation

It was previously determined that when the unconstrained *MPC* was tested on the non-linear system the deviation from the linearization point was influencing the performance due to non-linearities. As expected, a *NMCP* takes the non-linearities into acocunt thus a better performance is achieved.

12 Economic Linear MPC

The idea of the linear economic MPC is to perform control of system only based on requirements for the inputs and the outputs, and the economic aspect. In this project, the cost is directly linked to the amount of water fed from the pumps. It is desired to keep the water level over a certain value in the tanks.

12.1 Optimal Control Problem

$$\min_{u_k,v_k} = \phi = \sum_{k=1}^{N} \rho v + \sum_{k=0}^{N-1} c_k^T u_k \tag{110}$$

Where ρv is slack variables, which ensures demand is met whenever is possible (v is determined as in eq. (112). The constraints on the input is determined similar to the input constrained MPC seen in eq. (75).

$$\begin{bmatrix} \Delta u_{min} + u_{-1} \\ \Delta u_{min} \\ \Delta u_{min} \\ \vdots \\ \Delta u_{min} \end{bmatrix} \leq \underbrace{\begin{bmatrix} I & 0 & 0 & \dots & 0 \\ -I & I & 0 & \dots & 0 \\ 0 & -I & I & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & -I & I \end{bmatrix}}_{\Lambda} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{N-1} \end{bmatrix} \leq \begin{bmatrix} \Delta u_{max} + u_{-1} \\ \Delta u_{max} \\ \Delta u_{max} \\ \vdots \\ \Delta u_{max} \end{bmatrix} \tag{111}$$

$$\begin{bmatrix}0\\0\\0\\\vdots\\0\end{bmatrix} \leq \begin{bmatrix}v_0\\v_1\\v_2\\\vdots\\v_{N-1}\end{bmatrix} \leq \begin{bmatrix}\infty\\\infty\\\infty\\\vdots\\\infty\end{bmatrix} \tag{112}$$

The two sets of constraints can be but on a vector form defined as

$$\begin{bmatrix}U_{min}\\0\end{bmatrix} \leq \begin{bmatrix}U\\V\end{bmatrix} \leq \begin{bmatrix}U_{max}\\\infty\end{bmatrix} \tag{113}$$

The soft constrains is similar to what was seen for the input constrained MPC seen in section 8.1. Notice, that the disturbance is still unknown and therefore Γ_d is 0.

$$Z = \phi x_0 + \Gamma U + \Gamma_d D \geq R - V \rightarrow \Gamma U + V \geq R - \phi x_0 \tag{114}$$

Γ, ϕ, and R is the same a previously and can be seen in eq. (67). These can be used as input to the *QPsolver* by defining each element as

$$\begin{gathered} H = 0 \qquad g = \begin{bmatrix}g_u\\g_v\end{bmatrix} \qquad x = \begin{bmatrix}U\\V\end{bmatrix} \\ A = \begin{bmatrix}\Lambda & 0\\\Gamma_u & I\end{bmatrix} \qquad u = \begin{bmatrix}U_{max}\\\infty\end{bmatrix} \qquad l = \begin{bmatrix}U_{min}\\\bar{R}\end{bmatrix} \end{gathered} \tag{115}$$

Where $\bar{R} = R - \phi x_0$.
The implementation of the economic MPC is carried out in according to the following:

$$\begin{gathered} x_{k+1} = Ax_k + Bu_k + Ed_k \qquad k = 0, 1, \dots, N-1 \\ y = Cx_k \qquad k = 1, 2, \dots, N \\ u_{min} \leq u \leq u_{max} \qquad k = 1, 2, \dots, N \\ \Delta u_{min} \leq \Delta u \leq \Delta u_{max} \qquad k = 0, 1, \dots, N-1 \\ y_k \geq r_k - v_k \qquad k = 1, 2, \dots, N \\ v_k \geq 0 \qquad k = 1, 2, \dots, N \end{gathered} \tag{116}$$

12.2 Implementation and Simulation

The implementation of the economic MPC can be seen in appendix E.I.
In the simulation, the cost and the slack has been chosen to be the same for both pumps and outputs.

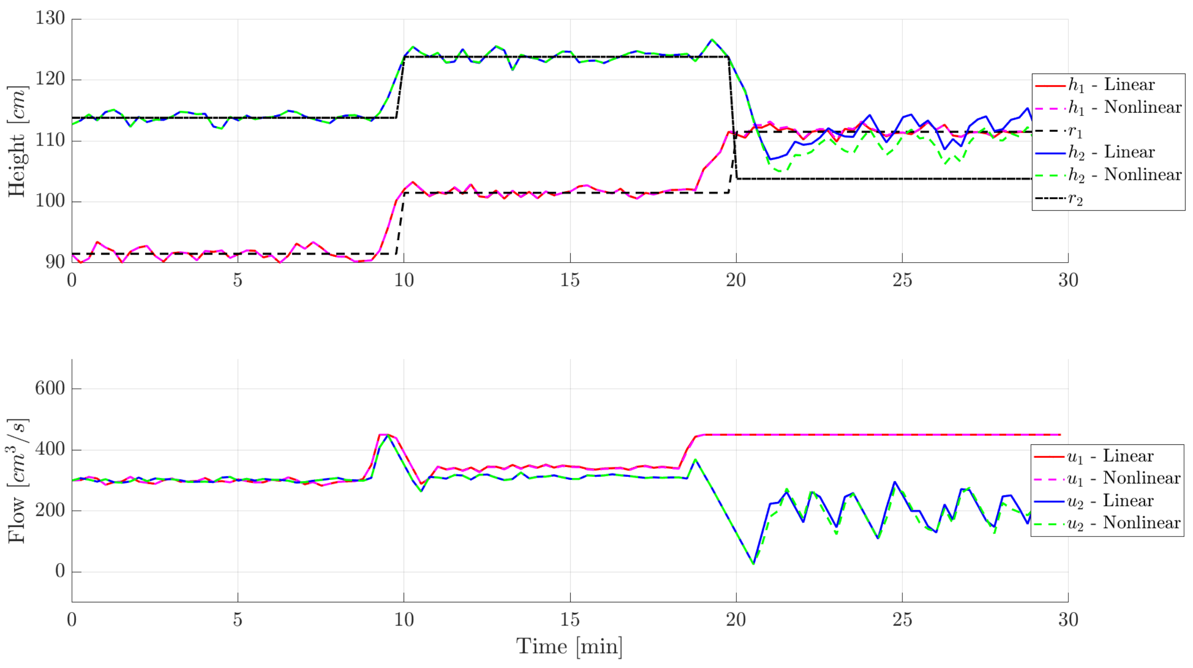

Figure 24: Economic MPC - Simulation

After 20 minutes, where the final step change in the reference occurs, the impact of the economic MPC can clearly be seen. The level in tank 2 is sufficiently close to the reference, and therefore the "on/off" effect on the pump occurs causing the input flow to vary.
The economic MPC differs significantly from the remaining MPC's designed in this project, since the economic MPC is a linear problem (where the remaining is a quadratic). If the cost of each pump is the same, the total flow is minimized, however if one of the pumps is evaluated to have a higher cost, the lowest cost pump could saturate. In these cases, the slack variable described previously would be different from 0.

13 PID Control

13.1 Pairing of Inputs and Outputs

It is known that u_1 affects the water level in tank 1 and 4 (due to the valve), which then influence tank 2. The valve is operated such that 58% of the water from pump 1 is lead to tank 1, and the remaining 42% is lead to tank 4 (hence tank 2). Opposite, the second valve is set to lead 68% of the water from pump 2 into tank 2, and the remaining 32% is led to tank 3. This correlation will mean that classical controllers (*P*, *PI* and *PID*) will have a limited performance.

13.2 Implementation of P, PI and PID Controller

The P, PI and PID Controller was implemented as functions in *MatLab* and can be seen in appendix F.I, appendix F.II and appendix F.III respectively.

13.3 Closed Loop Simulation of P, PI and PID Controllers

The desinged controllers have been tested in closed loop simulation on the linear model. The controllers have not been tuned persistently, but simply implemented in order to compare the differences between MPC and classical controllers in order to evaluate the main differences which will be described in the following paragraph. Notice, that the time between each step is 20 minutes, compared to previously where it was 10 minutes.

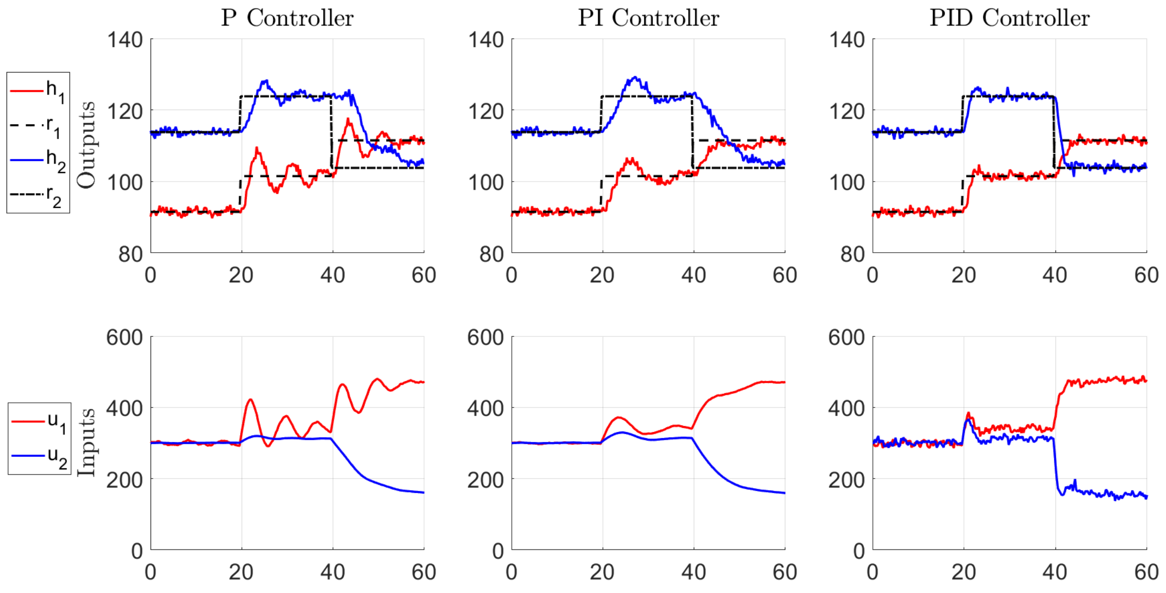

Figure 25: Closed loop simulation with P, PI and PID controller

13.4 P, PI and PID Controllers Compared to MPC

As seen in the previous paragraph the classical controllers are able to track a given reference signal, however there is minor overshoots and settling time is increased (compared to the closed loop with MPC), however this could probably be accounted for by better tuning.
In the controller implementation, it is possible to limit the inputs of the system to a certain minimum and maximum value (in this case u range from 0-500). However this only works as a saturation, and is not used as input to find an optimal input signal. It is not possible to set a limit on how fast the inputs changes, which of course in real life implementation can be troublesome.

14 Discussion and Conclusion

14.1 Comparison of MPC's

An advantage of the classical controllers is that they are relatively easy to implement and it required a low computational power while the system is online. However, they do not have the ability to "predict" ahead of time and therefore adjust prior to changes. They have no knowledge of unknown inputs, such as disturbances, which is used together with classic MPC (in case of a Kalman filter).

The main advantage of MPC is the ability to predict future reference changes. This allows the system to begin the control action in advance, which reduces settling time. Also the ability to design the MPC with knowledge about the input constraints (which can be used to ensure saturation of actuators do not occur), the outputs constraints and the rate of movement in the input. However the cons of these MPC is they are relatively difficult to implement and require more processing power (the more constraint, the more demanding).
The MPC can be implemented with an economic perspective by using the Economic MPC, which allows the design to take into account the cost of the performance instead of evaluating the control only with respect to performance. This case can roughly be described as a special case of the soft output constrained MPC.

14.2 Conclusion

In this report the advantage of different MPC's have been showed. The MPC works well with stochastic systems, where there is unknown inputs. Eventhough the MPC is build on a linear model, the performance is well for non-linear systems. The advantage of constraints on the MPC has been shown useful (especially in relation to real life implementation).

Appendix

A MatLab Implementation Unconstrained MPC

A.I MPC Design

```
function MPC_sys = MPCDesign(sys,Q,S,N)
    [phi,phi_w,Gamma,Gamma_d] = MPC_Constants(sys,N);
    % Determine Qz weight
    Q_z = eye(size(Gamma))*Q;

    % Generate Hs
    Hs = zeros(N*2);
    % Create arrow for matrix dimensions
    pil_Hs_row = [1:size(S,1)]; pil_Hs_col = pil_Hs_row;
    for i = 1:N
        while (pil_Hs_row(end) < N*2)
            Hs(pil_Hs_row,pil_Hs_col) = S*2; % Set S in diagonal
            Hs(pil_Hs_row,pil_Hs_col+size(S,1)) = -S;
            Hs(pil_Hs_row+size(S,1),pil_Hs_col) = -S;
            % Update arrows
            pil_Hs_row = pil_Hs_row + size(S,1); pil_Hs_col =
               pil_Hs_row;
        end % while
        if i == N
            Hs(pil_Hs_row,pil_Hs_col) = S;
        end % if i == N
        % Update arrows
        pil_Hs_row = [1:size(S,1)] + size(S,1)*i; pil_Hs_col =
           pil_Hs_row;
    end % i

    % Determine H
    H = transpose(Gamma)*Q_z*Gamma + Hs;

    % Now determine elements for g
    M_x0 = transpose(Gamma)*Q_z*phi;
    M_r = -transpose(Gamma)*Q_z;
    M_d  = transpose(Gamma)*Q_z*Gamma_d;
    M_u1 = zeros(size(M_d,1),size(S,1)); M_u1(1:size(S,1),1:size(S
       ,1)) = -S;

    % Now we determine the bounds on the input according to
    % "Lecture_07C_MPC" slide 18
    Lambda = eye(N);
    Lambda = kron(eye(2),Lambda);
    for i=1:N*2
        Lambda(i+2:i+3,i:i+1) = -eye(2);
    end
    Lambda = Lambda(1:end-3,1:end-1);
    I_0 = zeros(N*size(S,1),size(S,2));
    I_0(1:size(S,1),1:size(S,2)) = eye(size(S,1));

    MPC_sys = {phi,phi_w,Gamma,Gamma_d,Q_z,M_x0,M_r,M_d,M_u1,Hs,H,
       Lambda,I_0};
end % function
```

A.II MPC Constants

```
function [phi,phi_w,Gamma,Gamma_d] = MPC_Constants(sys,N)
    % Determine matrice
    A = sys.A; B = sys.B(:,1:2); E = sys.B(:,3:4);C = sys.C;

    % First create arrows to ensure matrix dimensions
    pil_phi = [1:size(C,1)];
    Nul_mat = zeros(size(C,1),2);
    pil_Gamma_row = [1:size(C*A*B,1)];
    pil_Gamma_col = [1:size(C*A*B,2)];
    pil_Gamma_d_row = [1:size(C*A*E,1)];
    pil_Gamma_d_col = [1:size(C*A*E,2)];

    for i = 1:N
        % Determine phi for N iterations.
        phi(pil_phi,:) = C*A^i;
        phi_w(pil_phi,:) = C*A^i*E;
        pil_phi = pil_phi+size(C,1);

        % Determine Gamma for N iterations
        while(pil_Gamma_row(end)< N*size(Nul_mat,2)+1)
            Gamma(pil_Gamma_row,pil_Gamma_col) = [C*A^(i-1)*B];
            pil_Gamma_row = pil_Gamma_row + size(Nul_mat,2);
            pil_Gamma_col = pil_Gamma_col + size(Nul_mat,1);
        end% while
        % Shift row and columns
        pil_Gamma_row = [1:size(Nul_mat,1)] + size(Nul_mat,1)*i;
        pil_Gamma_col = [1:size(Nul_mat,2)];

        % Determine Gamma_d for N iterations
        while(pil_Gamma_d_row(end)< N*size(Nul_mat,2)+1)
            Gamma_d(pil_Gamma_d_row,pil_Gamma_d_col) = [C*A^(i-1)*E
            ];
          pil_Gamma_d_row = pil_Gamma_d_row + size(Nul_mat,2);
          pil_Gamma_d_col = pil_Gamma_d_col + size(Nul_mat,1);
        end % while
        % Shift row and columns
        pil_Gamma_d_row = [1:size(Nul_mat,1)] + size(Nul_mat,1)*i;
        pil_Gamma_d_col = [1:size(Nul_mat,2)];
    end % i
end
```

A.III Unconstrained MPC

```
function u_mpc = Uncon_MPC(MPC_sys,x,R,u_prev)
    % Extract data for design
    M_x0 = cell2mat(MPC_sys(1,6)); M_r = cell2mat(MPC_sys(1,7));
       M_u1 = cell2mat(MPC_sys(1,9));
    Hs = cell2mat(MPC_sys(1,10)); H = cell2mat(MPC_sys(1,11));
    g = M_x0*x + M_r*R + M_u1*u_prev;
    [u_mpc] = qpsolver(H+Hs,g,[],[],[],[],[],[]);
end % function
```

B MatLab Implementation Input Constrained MPC

B.I Input Constrained MPC

```
function u_mpc = U_con_MPC(MPC_sys,x,R,u_prev,cons);
    % Extract data for design
    M_x0 = cell2mat(MPC_sys(1,6)); M_r = cell2mat(MPC_sys(1,7));
       M_u1 = cell2mat(MPC_sys(1,9));
    Hs = cell2mat(MPC_sys(1,10)); H = cell2mat(MPC_sys(1,11));
       Lambda = cell2mat(MPC_sys(1,12)); I_0 = cell2mat(MPC_sys
       (1,13));
    % Extract data for constraints
    lb = cell2mat(cons(1,1)); ub = cell2mat(cons(1,2)); Delta_min =
       cell2mat(cons(1,3)); Delta_max = cell2mat(cons(1,4));

    ub_Delta = Delta_max+I_0*u_prev;
    lb_Delta = Delta_min+I_0*u_prev;
    g = M_x0*x + M_r*R + M_u1*u_prev;
    [u_mpc] = qpsolver(H+Hs,g,lb,ub,Lambda,lb_Delta,ub_Delta,[]);
end % function
```

C MatLab Implementation Input Output Constrained MPC

C.I MPC Design for Input Output Constraints

```
function MPC_sys_soft = Soft_MPCDesign(sys,MPC_sys,N,W_z,W_u,W_du,
    soft_cons)
    % Extract data
    W_t1 = cell2mat(soft_cons(1,1)); W_t2 = cell2mat(soft_cons(1,2))
        ; W_s1 = cell2mat(soft_cons(1,3)); W_s2 = cell2mat(soft_cons
        (1,4));
    Gamma = cell2mat(MPC_sys(1,3)); Lambda = cell2mat(MPC_sys(1,12))
        ; I_0 = cell2mat(MPC_sys(1,13));

    % Determine Ht
    W_t2_bar=kron(eye(N),W_t2);
    H_t=W_t2_bar*W_t2_bar;

    % Determine gt
    W_t1_bar=kron(eye(N),W_t1);
    g_t=W_t1_bar*ones(size(W_t1_bar,2),1);

    % Set point
    W_z_bar=kron(eye(N),W_z);
    H_z=(W_z_bar*Gamma)'*(W_z_bar*Gamma);
    M_z=-(W_z_bar*Gamma)'*W_z_bar;

    % Input to reference input
    W_u_bar=kron(eye(N),W_u);
    H_u=W_u_bar'*W_u_bar;
    M_u=-H_u;

    % Input variations
    W_du_bar=kron(eye(N),W_du);
    H_du=(W_du_bar*Lambda)'*(W_du_bar*Lambda);
    M_du=-(W_du_bar*Lambda)'*W_du_bar*I_0;

    % Determine H total
    H_top=H_z+H_u+H_u;

    % lower bound
    W_s2_bar=kron(eye(N),W_s2);
    W_s1_bar=kron(eye(N),W_s1);
    H_s=W_s2_bar'*W_s2_bar;
    g_s=W_s1_bar*ones(size(W_s1_bar,2),1);

    MPC_sys_soft = {H_t,H_u,H_du,H_top,H_s,g_t,g_s,M_z,M_u,M_du};
```

```
end
```

C.II Input Output Constrained MPC

```
function u_mpc = InOut_MPC(MPC_sys,MPC_sys_soft,x,R,u_prev,
    cons_InOut,N);
    % Extract data
    H_t = cell2mat(MPC_sys_soft(1,1)); H_top= cell2mat(MPC_sys_soft
        (1,4)); H_s = cell2mat(MPC_sys_soft(1,5));
    g_t = cell2mat(MPC_sys_soft(1,6));   g_s = cell2mat(MPC_sys_soft
        (1,7));
    M_z = cell2mat(MPC_sys_soft(1,8)); M_u = cell2mat(MPC_sys_soft
        (1,9)); M_du = cell2mat(MPC_sys_soft(1,10));

    phi = cell2mat(MPC_sys(1,1)); Gamma = cell2mat(MPC_sys(1,3));
        Lambda = cell2mat(MPC_sys(1,12)); I_0 = cell2mat(MPC_sys
        (1,13));

    U_bar = cell2mat(cons_InOut(1,1)); U_min = cell2mat(cons_InOut
        (1,2)); U_max = cell2mat(cons_InOut(1,3));
    D_u_min = cell2mat(cons_InOut(1,4)); D_u_max = cell2mat(
        cons_InOut(1,5));
    Z_max = cell2mat(cons_InOut(1,6)); Z_min = cell2mat(cons_InOut
        (1,7));

    % Hbar is H matrix quadprog
    H_bar = kron(diag([1 0 0]),H_top)+kron(diag([0 1 0]),H_s)+kron(
        diag([0 0 1]),H_t);
    b_k = phi*x;
    c_k = R-b_k;
    g = M_z*c_k+M_u*U_bar+M_du*u_prev;

    % gbar is g quadprog
    g_bar = [g ; g_s ; g_t];

    % Determine A matrix
    A_bar = kron(diag([1 0 0]),Lambda)+kron(diag([0 1 -1]) , eye(2*N
        ))+kron([0 0 0; 1 0 0; 1 0 0] , Gamma);

    % boundary values on states
    u_low = [U_min ; zeros(size(U_min)) ; zeros(size(U_min))];

    u_up = [U_max;Inf*ones(size(U_max));inf(1)*ones(size(U_max))];

    b_l_bar = [D_u_min+I_0*u_prev;Z_min-b_k;-Inf*ones(size(D_u_min))
```

```
        ];

    b_u_bar = [D_u_max+I_0*u_prev;inf(1)*ones(size(D_u_max));Z_max-
        b_k];

    u_mpc = qpsolver(H_bar,g_bar,u_low,u_up,A_bar,b_l_bar,b_u_bar
        ,[]);

end
```

D MatLab Implementation of CDEKF

D.I CDEKF

```
function [x_k,P_k,x_k1,P_k1,e_k,Re_k,y_k1] = CDEKF(sigma,P_func,p,
   x_k1,u,d,Pk1,y_k,w_k,t,i,C)
    A1 = p(5,1); rho = p(12,1);
    y_k1 = x_k1(1:2,:) / (rho*A1);
    e_k = y_k1 - y_k;
    Re_k = C*Pk1*C' + w_k;
    K_k = Pk1*C'*inv(Re_k);
    x_k = x_k1 + K_k*e_k;
    P_k = (eye(size(K_k*C,1))-K_k*C) * Pk1 * (eye(size(K_k*C,1))-K_k
        *C)' + K_k*w_k*K_k';

    [~,P] = ode15s(P_func,[t(i) t(i+1)],[P_k(:,1) ; P_k(:,2) ; P_k
        (:,3) ; P_k(:,4) ; x_k],[],sigma,[u ; d],p);
    P_k1 = [P(end,1:4)' ; P(end,5:8)' ; P(end,9:12)' ; P(end,13:16)
        '];
    x_k1 = [P(end,17:20)'];
end

function [P_dot]=P_func(t,P,sigma,in,p)
    a1 = p(1,1); a2 = p(2,1); a3 = p(3,1); a4 = p(4,1); A1 = p(5,1);
         A2 = p(6,1); A3 = p(7,1); A4 = p(8,1); gamma1 = p(9,1);
        gamma2 = p(10,1); g = p(11,1); rho = p(12,1);
    u = in(1:2,1); d = in(3:4,1);

    x_dot1=-rho*a1*sqrt(2*g*P(17)/(rho*A1))+rho*a3*sqrt(2*g*P(19)/(
        rho*A3))+rho*gamma1*u(1,1);
    x_dot2=-rho*a2*sqrt(2*g*P(18)/(rho*A2))+rho*a4*sqrt(2*g*P(20)/(
        rho*A4))+rho*gamma2*u(2,1);
    x_dot3=-rho*a3*sqrt(2*g*P(19)/(rho*A3))+rho*(1-gamma2)*u(2,1)+
        rho*d(1,1);
    x_dot4=-rho*a4*sqrt(2*g*P(20)/(rho*A4))+rho*(1-gamma1)*u(1,1)+
        rho*d(1,1);

    A = [-a1/A1*sqrt(g/(2*P(17)/(rho*A1))), 0, a3/A3*sqrt(g/(2*P(19)
        /(rho*A3))),0;
          0, -a2/A2*sqrt(g/(2*P(18)/(rho*A2))), 0, a4/A4*sqrt(g/(2*P
              (20)/(rho*A4)));
          0, 0 ,-a3/A3*sqrt(g/(2*P(19)/(rho*A3))),0;
          0, 0, 0, -a4/A4*sqrt(g/(2*P(20)/(rho*A4)))];

    P_1=[P(1:4,1),P(5:8,1),P(9:12,1),P(13:16,1)];
    P_dot1=A*P_1+P_1*(A)'+sigma*(sigma)';
    P_dot=[P_dot1(:,1);P_dot1(:,2);P_dot1(:,3);P_dot1(:,4);[x_dot1;
        x_dot2;x_dot3;x_dot4]];
end
```

E MatLab Implementation of Economic MPC

E.I Economic

```
function u_mpc = Econ_MPC(MPC_sys,x,R,u_prev,cons,cons_Econ)
    % Unpack
    phi = cell2mat(MPC_sys(1,1)); Gamma = cell2mat(MPC_sys(1,3));
        Lambda = cell2mat(MPC_sys(1,12)); I_0 = cell2mat(MPC_sys
        (1,13));
    lb = cell2mat(cons(1,1)); ub = cell2mat(cons(1,2)); Delta_min =
        cell2mat(cons(1,3)); Delta_max = cell2mat(cons(1,4));
    g_u = cons_Econ(:,1); g_v = cons_Econ(:,2);

    N = length(g_u)/2;

    R_bar = R-phi*x;

    ub = [ub ; inf(N*2,1)];
    lb = [lb ; zeros(N*2,1)];
    ub_Delta = [Delta_max+I_0*u_prev ; inf(N*2,1)];
    lb_Delta = [Delta_min+I_0*u_prev ; R_bar];

    A = [Lambda , zeros(N*2,N*2) ; Gamma, eye(N*2)];

    g = [g_u ; g_v];
    u_mpc = qpsolver([],g,lb,ub,A,lb_Delta,ub_Delta,[]);
end
```

F P, PI, PID Controller Implementation

F.I P Controller Implementation

```
function [u e] = P_con(r,y,u_prev,u_min,u_max,K_P)
    e = r-y;
    u = u_prev + K_P*e ;
    if u(1,:)>u_max
        u(1,:)=u_max;
    elseif u(1,:)<u_min
        u(1,:)=u_min;
    end % end if
    if u(2,:)>u_max
        u(2,:)=u_max;
    elseif u(2,:)<u_min
        u(2,:)=u_min;
    end % end if
end
```

F.II PI Controller Implementation

```
function [u e] = PI_con(r,y,u_prev,u_min,u_max,K_PI,I,Ts,Ti)
    e = r-y;
    u = u_prev + K_PI*e + I;
    if u(1,:)>u_max
        u(1,:)=u_max;
    elseif u(1,:)<u_min
        u(1,:)=u_min;
    else
        I(1,:) = I(1,:) + (K_PI(1,1)*Ts/Ti*e(1,:));
    end % end if
    if u(2,:)>u_max
        u(2,:)=u_max;
    elseif u(2,:)<u_min
        u(2,:)=u_min;
    else
        I(2,:) = I(2,:) + (K_PI(2,2)*Ts/Ti*e(2))
    end % end if
end
```

F.III PID Controller Implementation

```
function [u e] = PID_con(r,u_prev,y,y_prev,u_min,u_max,K_P,I,K_I,K_D
    ,dt)
    e = r-y;
    P = K_P*e;
    D = -K_D*(y-y_prev)/dt;
    u = u_prev + P + I + D;

    if (u(1,:) >= u_max)
        u(1,:) = u_max;
    elseif (u(1,:) <= u_min)
       u(1,:) = u_min;
    else
        I(1,:) = I(1,:)+ K_I(1,1)*e(1,:)*dt;
    end

    if (u(2,:) >= u_max)
        u(2,:) = u_max;
    elseif (u(2,:) <= u_min)
        u(2,:) = u_min;
    else
        I(2,:) = I(2,:)+ K_I(2,2)*e(2,:)*dt;
    end
end
```