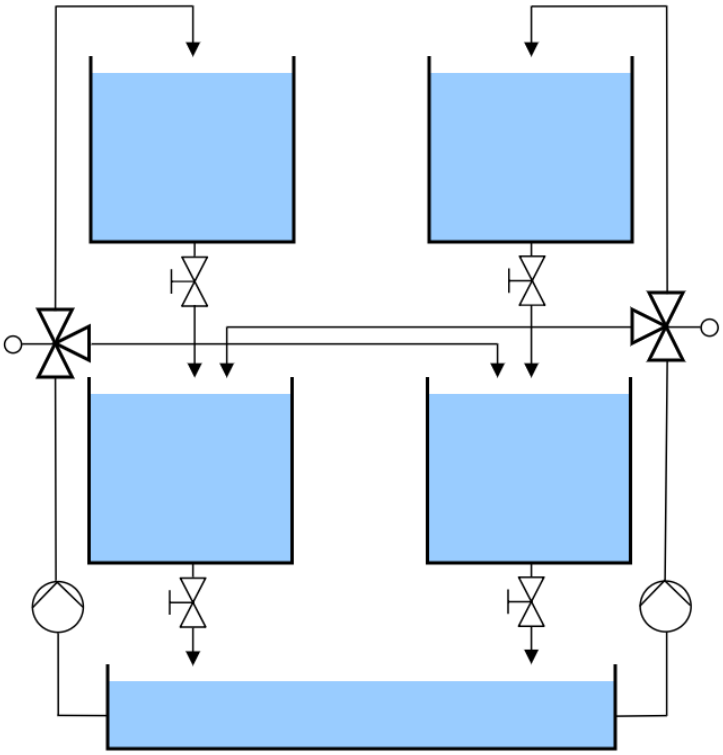
University of South-Eastern Norway

www.usn.no

FMH606 Master's Thesis 2018
Industrial IT and Automation

Experimental Evaluation of MIMO Observers and Model Based Controllers



Sondre Efjestad Fjereide

Faculty of Technology, Natural Sciences and Maritime Sciences

Campus Porsgrunn

University of South-Eastern Norway

www.usn.no

Course: FMH606 Master's Thesis 2018
Title: *Experimental Evaluation of MIMO Observers and Model Based Controllers*
Pages: *124*
Keywords: *MPC, Kalman Filter, Quadruple Tank Process, Non-minimum Phase, RGA, Model Based Control, Multi Variable*

Student: *Sondre Efjestad Fjereide*
Supervisor: *Carlos Pfeiffer and Roshan Sharma*
External partner:
Availability: *Open*

Summary:

State estimators for MIMO systems and model based controllers are important for multivariable control. Many publications have presented results of multivariable control, however most of the results are obtained from simulations. In this thesis the performance of Kalman filters and model based controllers was evaluated experimentally using the pilot sized quadruple tank process located at USN department of Porsgrunn.

Careful development and calibration of different process models was considered in order to obtain reasonable results from the experimental evaluation. The Kalman filters and model based controllers were subject to deliberate modeling errors to evaluate robustness to model mismatch. Models were obtained from first principles modeling and subspace system identification using the DS-R method for model realization. Experiments showed that the Kalman filters produce estimation error in the internal states while accurately estimating the outputs. The model based controllers showed to be robust to modeling errors due to feedback.

Preface

This thesis was written in spring 2018 as part of the final project of the master studies in ” Industrial IT and Automation” at the University of South-Eastern Norway and is titled ” Experimental Evaluation of MIMO Observers and Model Based Controllers”. The task description for the thesis can be found in appendix A.

I would like to thank my main supervisor Carlos F. Pfeifer for guidance during the project. He has been helpful in proposing tasks to perform during the project and he has been of great help during the project duration in general. I would also like to thank Roshan Sharma, Fredrik Hansen, David Di Ruscio and Hans-Petter Halvorsen for assisting with MPC development, completion of the experimental setup, system analysis and software implementation respectively.

The experimental setup used in the project is a quadruple tank process located in the process hall at University of South-Eastern Norway department of Porsgrunn. The software used during the project was MATLAB and National Instruments LabVIEW.

Porsgrunn, 15th May 2018

Sondre Efjestad Fjereide

Contents

List of Figures

List of Tables

Nomenclature

$\bar{x}$	Predicted states
$\bar{y}$	Predicted output
δp	Pressure drop over discharge valves
Δt	Time step
$\dot{m}$	Mass flow
$\dot{V}$	Volumetric flowrate
γ_i	Split factor for three-way valve number i
$\hat{x}$	Corrected states
ρ	Water density
σ_i	Variance of flow sensor i
$\tilde{A}$	Augmented system matrix A for linear MPC
$\tilde{A}_q$	Augmented system matrix A for LQ control
$\tilde{B}$	Augmented input matrix B for linear MPC
$\tilde{B}_q$	Augmented input matrix B for LQ control
$\tilde{C}$	Augmented output matrix C for linear MPC
$\tilde{C}_q$	Augmented output matrix C for LQ control
A	Tank area
A_c	Continuous time system matrix A
A_d	Discrete system matrix A

a_d	Cross-sectional area of discharge valve
A_s	Discrete time system matrix A from subspace realization
B_c	Continuous time input matrix B
B_d	Discrete input matrix B
B_s	Discrete time input matrix B from subspace realization
C_4	Controlability matrix
C_c	Discrete output matrix C
C_c	Continuous time output matrix C
c_d	Valve coefficient of discharge valve
c_i	Coefficient for discharge valve number i
C_s	Discrete time output matrix C from subspace realization
E	Weighting matrix for errors in model based controller objectives
g	Gravity constant
G_k	Process noise gain matrix
h_i	Level in tank number i
h_i^o	Operating point for level of tank number i
K_f	Kalman filter gain
K_{fs}	Kalman filter gain from DS-R
K_{pi}	Steady state gain for pump number i
O_4	Observability matrix
P	Weighting matrix for inputs in model based controller objectives
P_c	Auto-covariance of corrected states
P_p	Auto-covariance of predicted states

q_i	Flow from pump to tank number i
Q_k	State covariance matrix
R_k	Output covariance matrix
T_i	Time constant of tank number i
u_1	Input to pump number i
u_i^o	Operating point for input of pump number i
x_0	Initial states
DAQ	Data Acquisition
DS-R	Deterministic and Stochastic system identification and Realization
EKF	Extended Kalman Filter
HMI	Human Machine Interface
LQ	Linear Quadratic
MPC	Model Predictive Control
OPC	Open Platform Communications
QP	Quadratic Programming
RGA	Relative Gain Array
RMSE	Root Mean Square Error

1 Introduction

Multivariable processes with cross coupling can generally be challenging to control with single loop controllers like the conventional PID since they only receive information about one process variable. The quadruple tank process was introduced by Karl Henrik Johansson in a paper from 1998 due to an increased industrial interest in multivariable control techniques. It is a multivariable laboratory process with interconnected tanks which can induce interesting multivariable characteristics. The linearized model of the process has an adjustable multivariable zero which can be located both in the right and left hand side of the complex plane. The location of the zero can introduce dynamics that make controlling the process with single loop controllers challenging.[1]

Model based controllers are important tools for controlling multivariable processes and are often used in combination with state estimators. Many publications have been made on the topic of multivariable control and state estimation, however most of the publications presents results obtained from simulations and very few present experimental data. Simulations results of the quadruple tank process have been presented in [2]. Experimental results of the quadruple tank process have been published by e.q David Di Ruscio who presented a Model Predictive Control algorithm with integral action with experimental results obtained from the quadruple tank process in [3]. A linear discrete time state space realization of the Kalman filter used on the quadruple tank process was presented in [4]. System identification and model predictive control of the quadruple tank process model was covered in [5] and system identification has also been covered in [6].

Common for the results obtained from experimental data presented for the quadruple tank process is that measurements are only provided for the outputs and the internal states are only estimated. The actual performance of the estimation of hidden states is not documented. The lack of experimental results restricts the will for implementing multivariable control into the industry since the general perception is that there is a significant gap between theory and practice. It is therefore of interest to experimentally evaluate the performance of MIMO state estimators and model based controllers using the pilot sized quadruple tank process at USN to get presentable data on multivariable control and state estimation performance. The experimental setup at USN is equipped with level sensors for all 4 tanks which makes it possible to quantify the performance of estimation of hidden states which are usually not measured. It is of interest to investigate the robustness of model based controllers and the estimation performance of MIMO state estimators. The

state estimators and model based controllers should be evaluated using carefully developed process models, ensuring reliable results representing a best case scenario.

1.1 Objective

The main objective of this project is to experimentally evaluate the performance of state estimators and model based controllers for multiple input multiple output (MIMO) systems. The quadruple tank process with 4 states, 2 outputs and 2 inputs is considered. To get reliable results from evaluation of the MIMO state estimators and model based controllers, process models should be developed and calibrated so that they represents the physical process with reasonable accuracy. The state estimators and model based controllers should be tested on the simulation model to establish confidence that the they have been correctly implemented.

1.2 Report Structure

Chapter 2: System Overview

The system description chapter covers general information about the quadruple tank process, the hardware which is located at USN and the software used for the control system.

Chapter 3: Process Model

The process model chapter covers development of a non-linear dynamic model obtained from physical principles, a linear state space model obtained from the non-linear model and linear state space models obtained from subspace system identification. Calibration, validation and analysis of the models is also presented in this chapter.

Chapter 4: Kalman Filters

The Kalman filters chapter covers the development of a linear Kalman filter and 2 versions of the Extended Kalman filter. The 2 versions of the Extended Kalman filter use steady state Kalman gain and time varying Kalman gain.

Chapter 5: Model Based Controllers

The model based controllers chapter covers development of a linear model predictive controller (MPC) with integral action, a non-linear model predictive controller and a linear quadratic (LQ) optimal controller with integral action.

Chapter 6: Results and Discussion

The results and discussion chapter presents the results from chapter 3, 4 and 5. Observations made from the results are discussed in this chapter.

Chapter 7: Conclusion and Future Work

The conclusion chapter draws conclusions based on the work and results obtained though the project and presents recommendations for future work.

2 System Overview

This chapter gives an overview of the process which is studied though this project, the software used for the control system and the hardware for the quadruple tank process which is located at USN Porsgrunn. The process is to be controlled by various model based controllers. In order to use model based controllers the system states need to be estimated. This is done using a Kalman filter. The complete control system consists of the physical process, state estimator and model based controller. Figure 2.1 shown a rough system overview of how the quadruple tank process, model based controllers and Kalman filter are connected. Kalman filters and model based controllers are covered in later chapters.

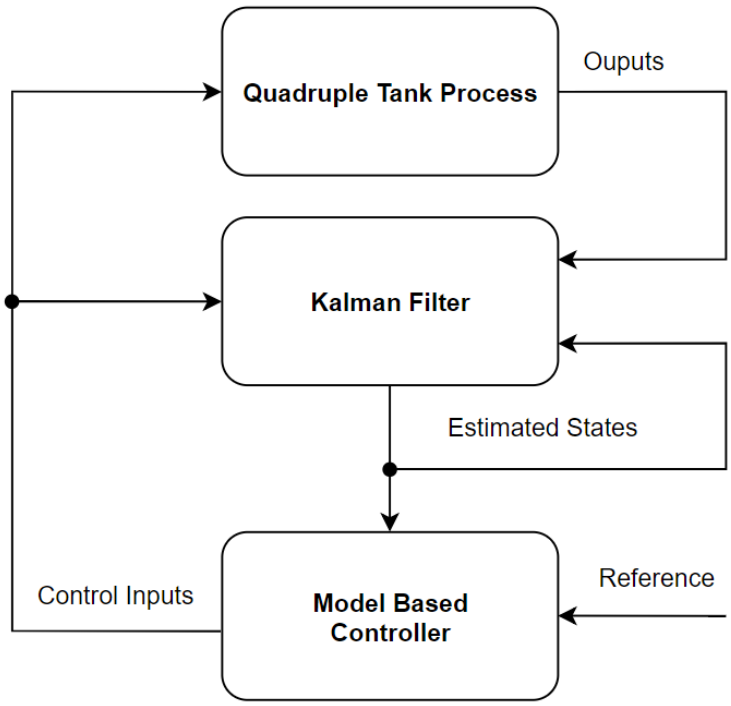


Figure 2.1: System overview

2.1 Quadruple Tank Process

The quadruple tank process is a multivariable process consisting of 4 tanks which are cross coupled. The process is controlled by 2 pumps which each pump water from the reservoir to 1 upper tank and 1 lower tank which are located diagonally on the process. The proportion of the water going to the upper tank is determined by the position of a tree-way split valve which is located on the outlet of each pump. Each tank is equipped with a discharge valve which allows water to flow out of the tanks. The water flowing out of the upper tanks flows into the lower tanks and the water from the lower tanks flows to the reservoir. This can be seen in figure 2.2 which shows the quadruple tank process.

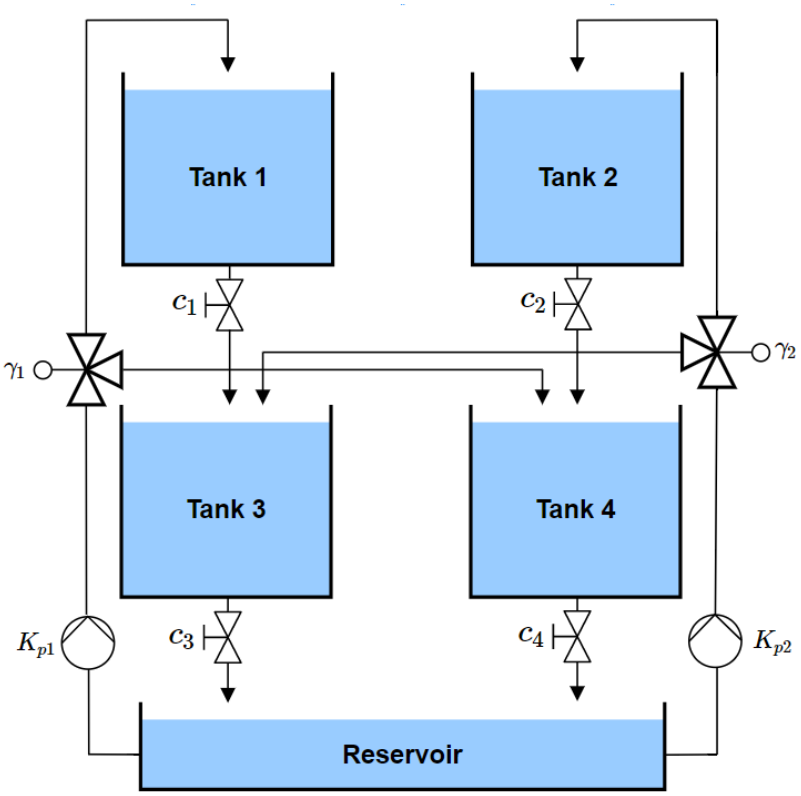


Figure 2.2: Quadrouple tank process

where c_i is the discharge coefficient of discharge valve i, K_{pi} is the pump gain for pump i and γ_i is the flow proportion of three-way valve i, where $\gamma = 1$ means that all the flow is going to the lower tank and $\gamma = 0$ means that all the flow is going to the upper tank. The value of γ for each valve is given by (2.1).

$$\gamma_1 = \frac{q_1}{q_1 + q_4}, \quad \gamma_2 = \frac{q_2}{q_2 + q_3} \tag{2.1}$$

where q_i is the flow from the pump to tank number i. The proportion of the flow going to the upper tank is therefor $0 \leq \gamma \leq 1$ and the proportion of the flow going to the lower tank is $(1-\gamma)$.

The process outputs are typically considered to be the lower tanks, thus it is the lower tanks that are subject to control and the upper tanks act as internal states. Pump 1 pumps water into tank 1 and tank 4 and pump 2 pumps water into tank 2 and tank 3. This means that the levels in the output tanks are affected by both pumps, thus there is cross coupling in the process.

2.2 Hardware

The quadruple tank process at USN is a well equipped system with 4 level sensors, 6 flow sensors, 2 pumps and 2 three-way valves. The water level in each tank is measured using ultra sonic sensors that represents the level with a voltage signal from 0V to 10V. Flow is represented by a frequency from 0Hz to 60Hz which is translated to a voltage signal between 0V and 5V using an Arduino. The pumps are controlled with an input signal from 0 to 10V and a boolean signal is required to turn the pumps on an off. When turned on, the pumps have an idle speed which is equivalent to having an input signal of 2V. The three-way valves can take an input signal from 0V to 5V to set the valve position and they give position feedback with a signal from 0V to 5V. Reading sensor signals and writing signals to the pumps and valves is done using 2 USB-6001 DAQ devices from National Instruments. Each tank has a discharge valve which can be manually operated. The valve must be adjusted so that the process can be controlled without the tanks going empty or overfill.

The process window is defined to be tank levels ranging from 10cm to 20cm. The level sensors are calibrated using linear scaling between 0.1m and 0.2m to obtain measurements in the SI unit. Since the pumps have an idle pump speed the process window of the pumps is defined to be in the range of 2V to 10V. The three-way valves are operating with γ values in the range of 0.3 to 0.7.

Figure 2.3 is a piping and instrumentation diagram of the physical process which shows the sensors, actuators and their locations on the rig. Table 2.1 contains information which explains the P&ID.

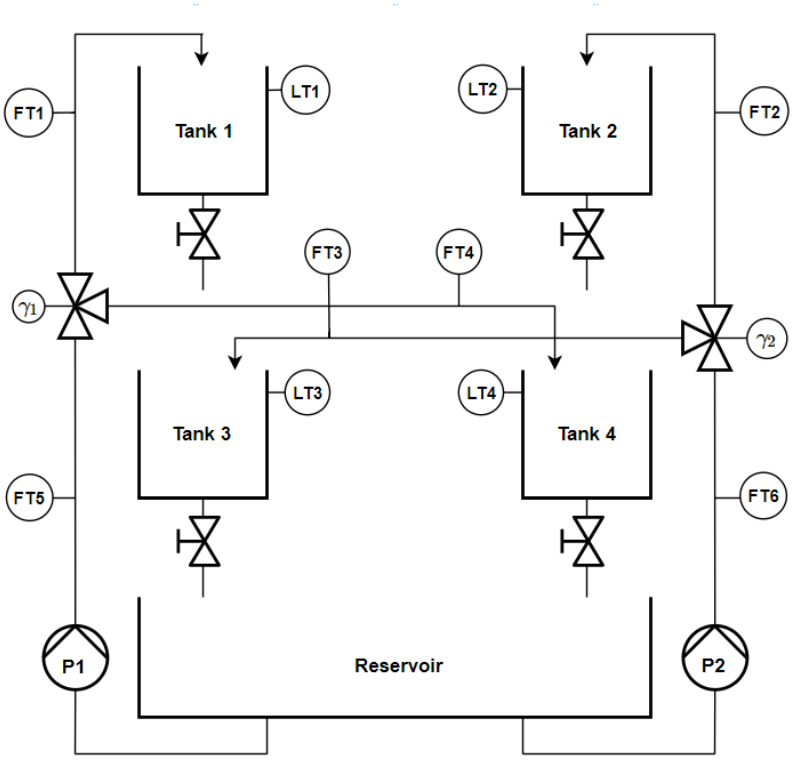


Figure 2.3: P&ID of the quadruple tank process at USN

Table 2.1: P&ID symbol list

Tag	Description	Comment
LT1	Level transmitter 1	Measures level in tank 1
LT2	Level transmitter 2	Measures level in tank 2
LT3	Level transmitter 3	Measures level in tank 3
LT4	Level transmitter 4	Measures level in tank 4
FT1	Flow transmitter 1	Measures flow to tank 1
FT2	Flow transmitter 2	Measures flow to tank 2
FT3	Flow transmitter 3	Measures flow to tank 3
FT4	Flow transmitter 4	Measures flow to tank 4
FT5	Flow transmitter 5	Measures flow from pump 1
FT6	Flow transmitter 6	Measures flow from pump 2
γ_1	Three-way valve 1	Splits flow from pump 1
γ_2	Three way valve 2	Splits flow from pump 2
P1	Pump 1	Pumps water to tank 1 and 4
P2	Pump 2	Pumps water to tank 2 and 3

2.3 Software

National Instruments LabVIEW was chosen as the software of preference for the control system. A major benefit of LabVIEW is that it is easy to make a user friendly HMI and monitor the process in real time. It also supports MATLAB Script Node which allows for running MATLAB functions in LabVIEW. This is a big advantage when solving optimization problems since MATLAB supports a number of good solvers. It also allows for implementing the controllers and Kalman filters as MATLAB functions. The control system software is developed using the state machine technique where the status of the levels and inputs are stored in a shift register [7]. Figure 2.4 shows a flow chart of the program structure for the control system implemented in LabVIEW.

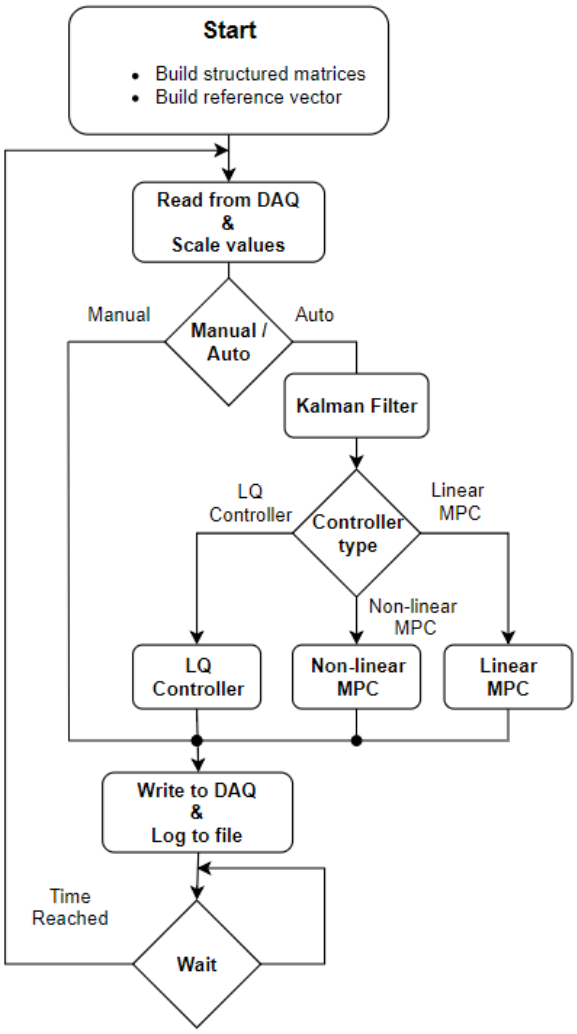


Figure 2.4: Diagram of control system

The software is structured in such a way that is initializes matrices for the controllers and Kalman filter on start up, so that they do not need to be constructed more than once. The program then moves on to read the values of the NI-6001 DAQ device and scale the values to desired units. If manual control is selected the program moves on to write the manual control input and the γ values to the shift register and moves on to write the values from the shift register to the DAQ device. If automatic control is selected, the program moves on to run the Kalman filter for state estimation. The estimated states are saved in the shift register so that they are available to the controllers. The program then checks what controller is selected and obtains a control input from the selected controller which it saves to the shift register. The control input is then written to the DAQ device. The reference vectors for the controllers are constant, and the controllers locally shift one sample in the reference vector each time step.

3 Process Model

Common for both Kalman Filters and model based controllers is that they all use a model of the process to predict the process behaviour. A model can be obtained in different ways and may vary in complexity. Once the physical process has been modeled it needs to be calibrated, and when the calibration is done it needs to be validated. This chapter will cover the development, calibration and validation of a model for the quadruple tank process described in 2.1. obtained from physical principles, also known as a first principles model, and a model identified from known input and output data gathered from the physical process, known as subspace system identification.

3.1 Model Development

This section covers development of the process models, including a non-linear model obtained from physical principles, a linear approximation of the non-linear model and a linear state space model obtained from subspace system identification.

3.1.1 First Principles Non-linear Model

The first principles model is obtained from the law of conservation of mass. The accumulated mass in each tank is the sum of the mass flow into the tank minus the sum of the mass flow out of the tank. The density of water is constant so the water level in each tank is dependent on accumulated mass and the mass balance equation (3.1) is therefore used to model the tank system.

$$\frac{dm}{dt} = \dot{m}_{in} - \dot{m}_{out} \tag{3.1}$$

Where $\dot{m}_{in}$ is mass flow into the tank and $\dot{m}_{out}$ is the mass flow out of the tank. Differential equations for the levels are obtained by rearranging the mass balance equation as shown in (3.2).

$$\begin{aligned} \rho \frac{dV}{dt} &= \dot{V}_{in}\rho - \dot{V}_{out}\rho \\ A\frac{dh}{dt} &= \dot{V}_{in} - \dot{V}_{out} \\ \frac{dh}{dt} &= \frac{\dot{V}_{in} - \dot{V}_{out}}{A} \end{aligned} \tag{3.2}$$

where h is the tank water level, $\dot{V}_{in}$ is the volumetric flow rate going into the tank, $\dot{V}_{out}$ is the volumetric flow rate going out of the tank and A is the area of the tank. The four tanks of the process at USN all have the same area so that $A_1 = A_2 = A_3 = A_4 = A$.

It is necessary to model the flow into the tanks and the flow out of the tanks. For the upper tanks the flow in is the ratio of the pump flow directed upwards from the tree-way valve and the flow out is the flow running through the discharge valve. For the lower tanks the flow in is the ratio of the pump flow directed downwards by the three-way valve and the flow flowing through the discharge valve of the tank above. The flow out of the lower tanks is the flow running through the discharge valve. It is therefore necessary to model the discharge valves and the pumps including the three-way valves.

Flow running through the discharge valve is modelled using Bernoullis equation (3.3) for flow through an orifice [8].

$$\dot{V} = a_d c_d \sqrt{\frac{2}{\rho}\delta p} \tag{3.3}$$

where $\dot{V}$ is the volumetric flow rate through the valve, a_d is the cross-sectional area of the valve, c_d is the valve coefficient and δp is pressure drop over the valve. The pressure drop is equal to the hydro static pressure at the bottom of the tank due to having atmospheric pressure above the water column and on the valve outlet. This is used to simplify the valve equation, collecting all the constant terms and resulting in the equation (3.4) where the flow is only dependent on the level.

$$\dot{V} = c\sqrt{h} \tag{3.4}$$

where $c = a_d c_d \sqrt{2g}$ and g is the gravity constant.

When modeling the pumps one must evaluate what level of complexity is needed. The pumps are not infinitely fast, meaning that they have some dynamics which affect the flow. Figure 3.1 shows the step response of the pumps and the lower tanks performed on the physical process. The step is from 5V to 10V.

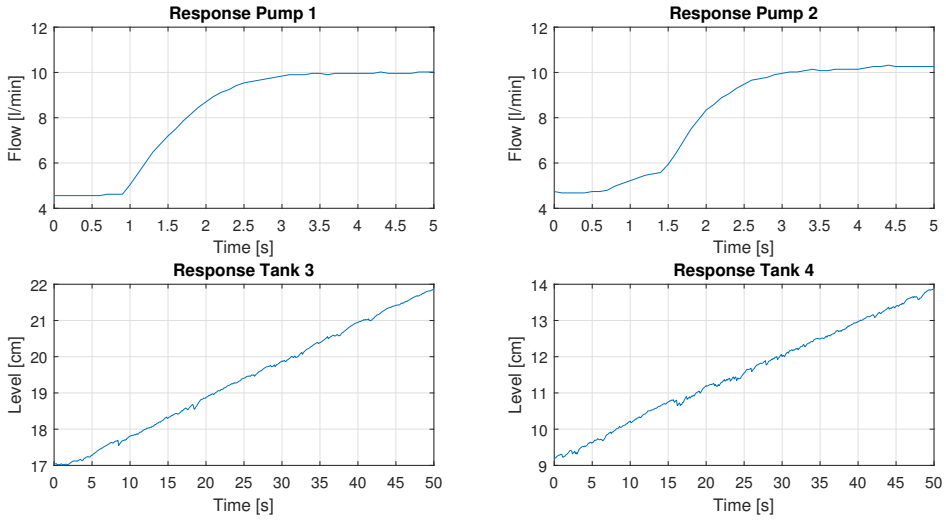


Figure 3.1: Pump and tank step response of the physical process

It can be seen that the dynamics of the pumps are much faster than the dynamics of the tanks. The pumps reach steady state after approximately 3 seconds, while the tanks have not even begun to settle after 50 seconds. In a paper from 1983, Wlodzimierz Klonowski showed that states with much faster time constants than the states of interest can be steady-state approximated [9]. This leads to the decision to neglect the pump dynamics to reduce model complexity, as taking the pump dynamics into account would introduce 2 extra states in the model.

To see how the flow from the pumps should be modeled, the steady state pump flows were measured at different pump inputs. Figure 3.2 shows a plot of measured steady-state pump flows against the pump inputs, and shows that the relationship between input and output is close to linear. The pump response is therefore modeled with a constant pump gain.

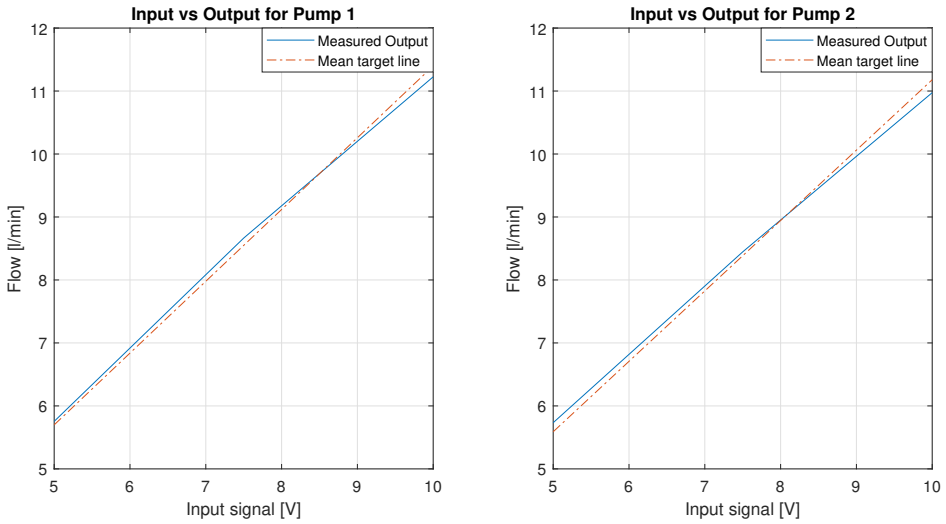


Figure 3.2: Steady-state pump response

Each tank is modeled separately and the expressions are combined into a set of differential equations describing the tank dynamics. The result (3.5) is a continuous time model.

$$\begin{aligned}
\frac{dh_1(t)}{dt} &= \frac{1}{A}\left[K_{p1}(1-\gamma_1)u_1(t) - c_1\sqrt{h_1(t)}\right] \\
\frac{dh_2(t)}{dt} &= \frac{1}{A}\left[K_{p2}(1-\gamma_2)u_2(t) - c_2\sqrt{h_2(t)}\right] \\
\frac{dh_3(t)}{dt} &= \frac{1}{A}\left[K_{p2}\gamma_2 u_2(t) + c_1\sqrt{h_1(t)} - c_3\sqrt{h_3(t)}\right] \\
\frac{dh_4(t)}{dt} &= \frac{1}{A}\left[K_{p1}\gamma_1 u_1(t) + c_2\sqrt{h_2(t)} - c_4\sqrt{h_4(t)}\right]
\end{aligned} \tag{3.5}$$

Since the model is to be implemented in a computer it needs to be represented on discrete form. The continuous time model in (3.5) is discretized using the Forward Euler approximation (3.6).

$$\frac{dh(t)}{dt} = \frac{h_{(k+1)} - h_{(k)}}{\Delta t} \tag{3.6}$$

where $h_{(k+1)}$ is the level of the next time step, $h_{(k)}$ is the level of the current time step and Δt is the step size. Applying (3.6) to all the of the differential equations in (3.5) results in the discrete time non-linear model seen in (3.7).

$$\begin{aligned}
h_{1(k+1)} &= h_{1(k)} + \frac{\Delta t}{A}\left[K_{p1}(1-\gamma_1)u_{1(k)} - c_1\sqrt{h_{1(k)}}\right] \\
h_{2(k+1)} &= h_{2(k)} + \frac{\Delta t}{A}\left[K_{p2}(1-\gamma_2)u_{2(k)} - c_2\sqrt{h_{2(k)}}\right] \\
h_{3(k+1)} &= h_{3(k)} + \frac{\Delta t}{A}\left[K_{p2}\gamma_2 u_{2(k)} + c_1\sqrt{h_{1(k)}} - c_3\sqrt{h_{3(k)}}\right] \\
h_{4(k+1)} &= h_{4(k)} + \frac{\Delta t}{A}\left(K_{p1}\gamma_1 u_{1(k)} + c_2\sqrt{h_{2(k)}} - c_4\sqrt{h_{4(k)}}\right]
\end{aligned} \tag{3.7}$$

3.1.2 First Principles Linear Model

The model developed in 3.1.1 is non-linear due to the states being under a square root. In order to develop a linear Kalman Filter and linear MPC it is necessary to obtain a linear approximation of the non-linear model and formulate it as a discrete time linear state space model. In this case the model is first linearized and then discretized, however the procedure could be done in the opposite order. Linearization is done by approximating the non-linear continuous time differential equations as linear continuous time differential equations using the first two term of the Taylor series expansion (3.8) [10].

$$\frac{dh(t)}{dt} = f(h^o,u^o) + \frac{\partial f}{\partial h}\Big|_{(h^o,u^o)}(h(t)-h^o) + \frac{\partial f}{\partial u}\Big|_{(h^o,u^o)}(u(t)-u^o) \tag{3.8}$$

where h^0 and u^o are some nominal values of the states h and inputs u. These nominal values will be referred to as operating points. By defining $h(t) = (\delta h(t) + h^o)$ and $u(t) = (\delta u(t) + u^o)$, equation (3.8) can be rewritten as shown in (3.9)

$$\frac{d\delta h(t)}{dt} = \frac{\partial f}{\partial h}\Big|_{(h^o,u^o)}\delta h(t) + \frac{\partial f}{\partial u}\Big|_{(h^o,u^o)}\delta u(t) \tag{3.9}$$

The linear model is obtained by approximating each of the differential equations of the non-linear model using (3.9). The resulting model can be seen in (3.10).

$$
\begin{aligned}
\frac{d\delta h_1(t)}{dt} &= \frac{1}{A}\left[K_{p1}(1-\gamma_1)\delta u_1(t) - \frac{c_1}{2\sqrt{h_1^o}}\delta h_1(t)\right] \\
\frac{d\delta h_2(t)}{dt} &= \frac{1}{A}\left[K_{p2}(1-\gamma_2)\delta u_2(t) - \frac{c_2}{2\sqrt{h_2^o}}\delta h_2(t)\right] \\
\frac{d\delta h_3(t}{dt} &= \frac{1}{A}\left[K_{p2}u_2\gamma_2\delta u_2(t) + \frac{c_1}{2\sqrt{h_1^o}}\delta h_1(t) - \frac{c_3}{2\sqrt{h_3^o}}\delta h_3(t)\right] \\
\frac{d\delta h_4(t)}{dt} &= \frac{1}{A}\left[K_{p1}\gamma_1\delta u_1(t) + \frac{c_2}{2\sqrt{h_2^o}}\delta h_2(t) - \frac{c_4}{2\sqrt{h_4^o}}\delta h_4(t)\right]
\end{aligned}
\tag{3.10}
$$

To be represented as a linear state space model, the derivatives, states, inputs and outputs are combined into vectors and the coefficients are combined into matrices. The resulting continuous time state space model can be seen in (3.11).

$$
\begin{aligned}
\dot{x} &= A_c x + B_c u \\
y &= C_c x
\end{aligned}
\tag{3.11}
$$

where

$$
\dot{x} = \begin{bmatrix} \frac{d\delta h_1(t)}{dt} \\ \frac{d\delta h_2(t)}{dt} \\ \frac{d\delta h_3(t}{dt} \\ \frac{d\delta h_4(t)}{dt} \end{bmatrix}, \quad x = \begin{bmatrix} \delta h_1(t) \\ \delta h_2(t) \\ \delta h_3(t) \\ \delta h_4(t) \end{bmatrix}, \quad u = \begin{bmatrix} \delta u_1(t) \\ \delta u_2(t) \end{bmatrix}, \quad y = \begin{bmatrix} \delta y_1(t) \\ \delta y_2(t) \end{bmatrix}, \quad C_c = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
A_c = \begin{bmatrix} -\frac{c_1}{2A\sqrt{h_1^o}} & 0 & 0 & 0 \\ 0 & -\frac{c_2}{2A\sqrt{h_2^o}} & 0 & 0 \\ \frac{c_1}{2A\sqrt{h_1^o}} & 0 & -\frac{c_3}{2A\sqrt{h_3^o}} & 0 \\ 0 & \frac{c_2}{2A\sqrt{h_2^o}} & 0 & -\frac{c_4}{2A\sqrt{h_4^o}} \end{bmatrix}, \quad B_c = \begin{bmatrix} \frac{K_{p1}(1-\gamma_1)}{A} & 0 \\ 0 & \frac{K_{p2}(1-\gamma_2)}{A} \\ 0 & \frac{K_{p2}\gamma_2}{A} \\ \frac{K_{p1}\gamma_1}{A} & 0 \end{bmatrix}
$$

The state space model is also discretized using the Forward Euler approximation as seen in (3.12).

$$\begin{aligned} x_{(k+1)} &= x_{(k)} + \Delta t\left[A_c x_{(k)} + B_c u_{(k)}\right] \\ y_{(k)} &= C_c x_{(k)} \end{aligned}$$
$$\begin{aligned} x_{(k+1)} &= \left(I + \Delta t A_c\right) x_{(k)} + \Delta t B_c u_{(k)} \\ y_{(k)} &= C_c x_{(k)} \end{aligned} \tag{3.12}$$

Reformulating the discrete model in (3.12) results in a discrete time state space model on standard form as seen in (3.13).

$$\begin{aligned} x_{(k+1)} &= A_d x_{(k)} + B_d u_{(k)} \\ y_{(k)} &= C_d x_{(k)} \end{aligned} \tag{3.13}$$

where $A_d = \left(I + \Delta t A_c\right)$, $B_d = \Delta t B_c$ and $C_d = C_c$.

This linear state space model is on deviation form, which means that the model only expresses the deviation from the operating points that the model was linearized around. The operating points are chosen to be some steady state values of the levels and inputs within the process window. Steady state levels for different combinations of inputs spanning from 4V to 10V were found by simulating the non-linear model in MATLAB. Figure 3.3 shows the steady state levels plotted against the corresponding inputs. The plot shows the steady state values for a set of model parameters, however whenever the parameters are changed the operating points will also change, so the simulation must be repeated when changing the parameters.

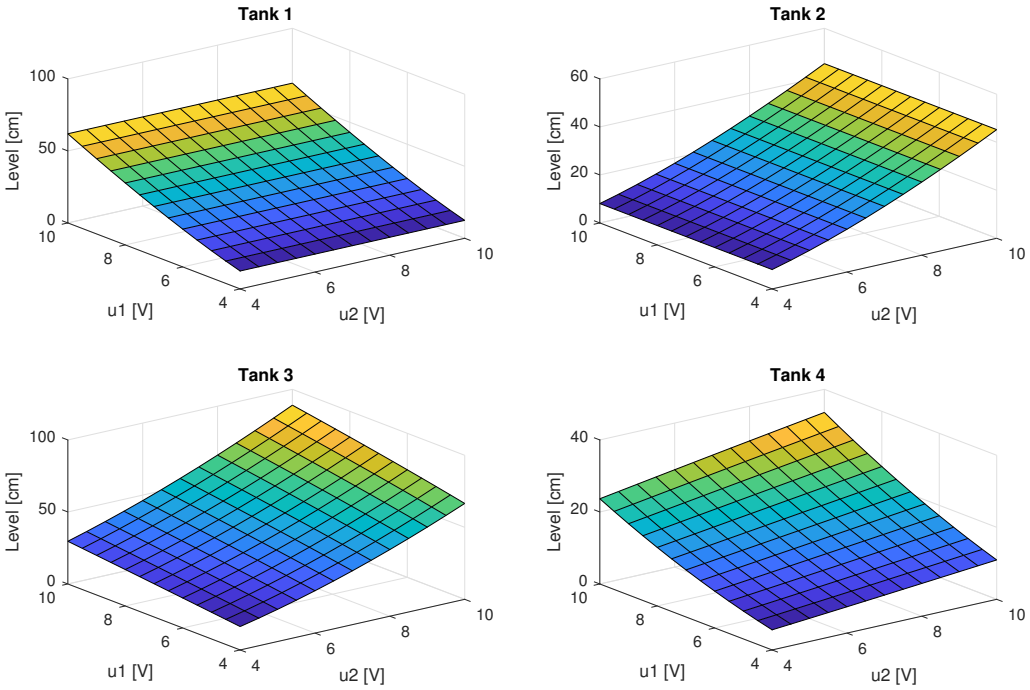


Figure 3.3: Steady state levels for different combinations of inputs

3.1.3 Subspace Model

If a process is complex it may be hard to model it mathematically. The modeling may take a lot of time and it may be hard to capture the necessary complexity of the physical process to get a model with the required accuracy. In such a scenario it may be beneficial to use subspace system identification, where the model is obtained from a set of observed data . It is of interest to look at the performance of models obtained from subspace system identification, and the DS-R toolbox in MATLAB was chosen for this. [11]

A big advantage of using subspace identification is that there is no need for mathematical modeling, however there are some downsides. A first principles model can easily be modified by changing the values of parameters and updating the model. A subspace model needs to be re-identified from a new set of data if the parameters are changed. This requires running the calibration experiment all over again and may be inconvenient when working with the quadruple tank process, where it may be of interest to change the parameters γ_1 and γ_2. The resulting discrete state space model from subspace identification describes a discrete model with a time step Δt equal to the time step of the logged data. Therefore the data must be logged with a time step equal to the desired time step of the application which the model should be used for. Calibration data sets were obtained from open loop experiments carried out on the physical process. The experiments were carried out by applying of a number of step inputs with varying lengths to the pumps, while making sure that the levels of tank 3 and 4 stayed within the process window. Figure 3.4 shows the inputs used in a calibration experiment with time step $\Delta t = 1s$. Figure 3.5 shows the resulting levels in tank 3 and 4.

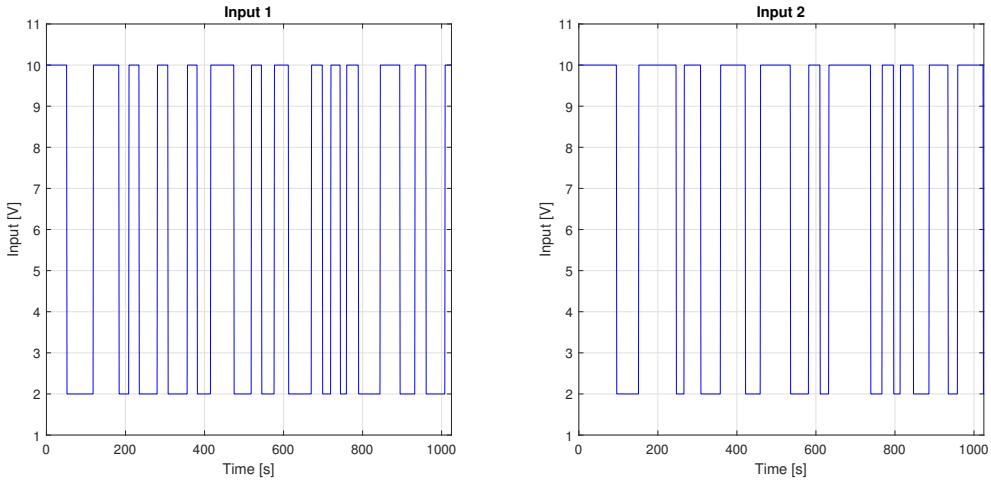


Figure 3.4: Pump inputs for experiment for calibrating subspace model

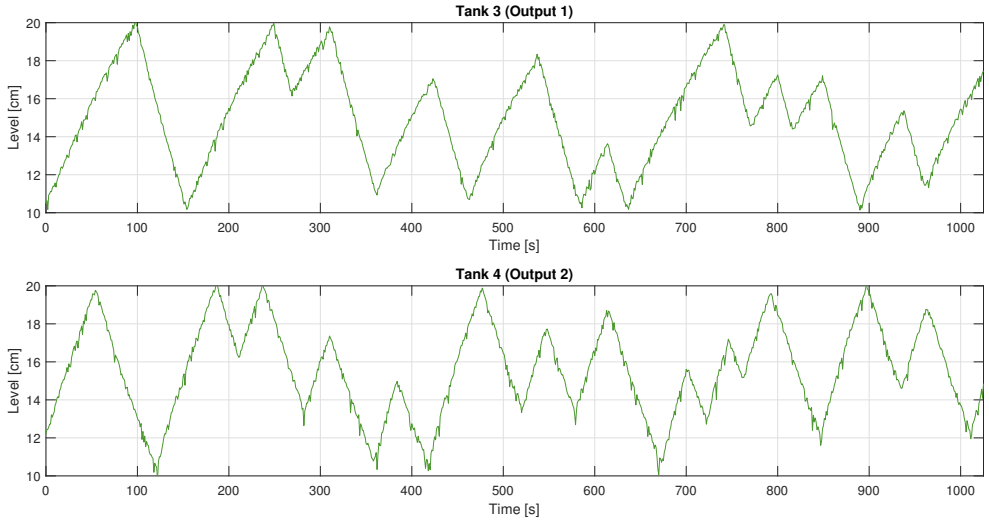


Figure 3.5: Levels of tank 3 and tank 4 during experiment for calibrating subspace model

The DS-R toolbox contains a number of functions, but *dsr.m* is designed for data without feedback and was therefore chosen for the identification.This function returns a linear discrete state space model (A,B,C,D), the initial states x_0 and the Kalman filter gain matrix K_{fs}. The input arguments may vary depending on preference as some of the arguments are optional. The function is called as seen in (3.14).

$$\left[A_s,B_s,C_s,D_s,K_{fs}F,F,x_0\right] = dsr(Y,U,L,g,J,M,n) \tag{3.14}$$

where Y is the logged output data, U is the logged input data, L is the number of block rows in the extended observability matrix, g is set to 0 to force the matrix D to be 0, J is set equal to L, $M = 1$ is default and n is the system order. The Kalman filter gain matrix is obtained from $K_{fs}F$ by multiplying with the inverse of F. The levels from tank 3 and 4 were logged and sorted into a matrix Y and the inputs were sorted into a matrix U. The resulting state space model (A,B,C), initial states x_0 and Kalman filter gain K_{fs} from the experiment with $\Delta t = 1s$ can be seen in (3.15).

$$
A_s = \begin{bmatrix} 0.9869 & -0.0004 & 0.5662 & -0.3346 \\ 0.0006 & 0.9970 & 0.2810 & 0.5932 \\ -0.0009 & -0.0018 & 0.4871 & -0.0670 \\ 0.0005 & -0.0022 & -0.0590 & 0.5298 \end{bmatrix}
$$

$$
B_s = \begin{bmatrix} -0.0002649 & -0.0002353 \\ -0.0002570 & 0.0001926 \\ -0.0003083 & -0.0000348 \\ -0.0001335 & 0.0002427 \end{bmatrix}
$$

$$
C_s = \begin{bmatrix} -0.3690 & 0.3455 & 0.2330 & -0.7204 \\ -0.3511 & -0.3624 & 0.7203 & 0.2766 \end{bmatrix} \tag{3.15}
$$

$$
x_0 = \begin{bmatrix} -0.3201 \\ -0.0337 \\ -0.0032 \\ 0.0012 \end{bmatrix}
$$

$$
K_{fs} = \begin{bmatrix} -0.5834 & -0.4434 \\ 0.4074 & -0.3711 \\ -0.0405 & 0.0021 \\ 0.0130 & 0.0101 \end{bmatrix}
$$

The DS-R subspace identification return a discrete time state space realization of the physical process. The states of this realization are not located in the same coordinate system as the measured states. It can be seen from the resulting matrices in (3.15) that they do not resemble the state space matrices obtained from first principles modeling. The output matrix C_s convert the state information into outputs that are in the same coordinate system as the measured states, however the internal states are not converted. This means that the model provides information about the levels in the lower tanks through the output matrix C_s while the information about the levels in upper tanks is not directly intuitive. [11]

3.2 Model Parameter Calibration

Model calibration is the procedure of identifying the model parameters from experimental data gathered from the physical process. The parameters to be identified are the steady state pump gains K_{p1} and K_{p2} and the valve coefficients c_1, c_2, c_3 and c_4. This section will cover parameter calibration of the first principles model. Note that parameter calibration is not done for the subspace model, since DS-R toolbox returns a discrete time state space model which is already calibrated. A MATLAB script for calibrating the first principles model can be found in appendix B.

3.2.1 Data Reconciliation

Having flow sensors on the rig gives a great advantage when calibrating the parameters due to the fact that the flows into the tanks are always known. However, measurements from physical sensors come with some uncertainty due to noise and limited resolution. The uncertainty will affect the flow measurements which again affects the parameter calibration since the flow measurements are used to calibrate the parameters. Data reconciliation is performed on the flow measurements to reduce the uncertainty of the flow measurements before using them for calibration. The data reconciliation technique is found in a presentation made by North Carolina State University and University of Ottawa [12].

Due to conservation of mass, the flow into the three-way valve is equal to the sum of the flows out of the valve, hence satisfying the mass balance $q_1 = q_2 + q_3$ as illustrated in figure 3.6.

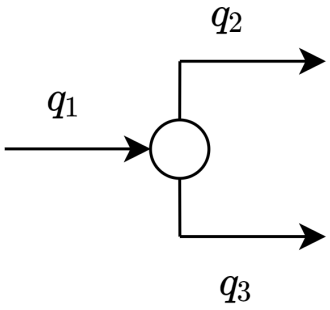


Figure 3.6: Flow split by tree way valve

Noise and uncertainty in one sensors is uncorrelated with the noise and uncertainty in the other sensors, and thus the mass balance is not satisfied when looking at the raw sensor data. By performing data reconciliation on the measured flows it is possible to obtain estimates of the flows that satisfy the mass balance. Optimal estimates are obtained

by solving the optimization problem shown in (3.16), constrained by the mass balance equation shown in (3.17).

$$\min_{(\hat{q}_1,\ \hat{q}_2,\ \hat{q}_3)} \quad J = \frac{(q_1-\hat{q}_1)^2}{{\sigma_1}^2} + \frac{(q_2-\hat{q}_2)^2}{{\sigma_2}^2} + \frac{(q_3-\hat{q}_3)^2}{{\sigma_3}^2} \tag{3.16}$$

subject to:

$$\hat{q}_1 - \hat{q}_2 - \hat{q}_3 = 0 \tag{3.17}$$

The problem becomes to minimize the objective J using the decision variables which are the flow estimates $\hat{q}_1, \hat{q}_2$ and $\hat{q}_3$. For each sensor number i, the deviation between the measured value and the estimate is weighted by the variance σ_i^2. In this way the measurements from sensors with a higher variance are considered less reliable than measurements from sensors with a lower variance. Flow estimates of the less accurate sensors will therefore differ more from the raw data than the estimate from the more accurate sensors.

To solve the optimization problem, the constraints are eliminated using Lagrangian multiplier so that the optimization problem becomes an objective function without constraints (3.18).

$$\min_{(\hat{q}_1,\ \hat{q}_2,\ \hat{q}_3,\ \lambda)} \quad J = \frac{(q_1-\hat{q}_1)^2}{{\sigma_1}^2} + \frac{(q_2-\hat{q}_2)^2}{{\sigma_2}^2} + \frac{(q_3-\hat{q}_3)^2}{{\sigma_3}^2} + \lambda(\hat{q}_1 - \hat{q}_2 - \hat{q}_3) \tag{3.18}$$

The minimum of objective J is found by taking the partial derivatives of J with respect to each of the decision variables and equating them to 0 as seen in (3.19).

$$\begin{aligned}
\frac{\partial J}{\partial \hat{q}_1} &= \frac{2(q_1-\hat{q}_1)}{\sigma_1^2} + \lambda = 0 \\
\frac{\partial J}{\partial \hat{q}_2} &= \frac{2(q_2-\hat{q}_1)}{\sigma_2^2} + \lambda = 0 \\
\frac{\partial J}{\partial \hat{q}_3} &= \frac{2(q_3-\hat{q}_1)}{\sigma_3^2} + \lambda = 0 \\
\frac{\partial J}{\partial \lambda} &= \hat{q}_1 - \hat{q}_2 - \hat{q}_3 = 0
\end{aligned} \tag{3.19}$$

The problem now consists of 4 equations and 4 unknowns, hence the unknowns can be found analytically. Arranging the problem into matrices results in the expression for the estimated flows seen in (3.20).

$$\hat{q} = q - VA_r^T \left(A_r V A_r^T\right)^{-1} A_r q \tag{3.20}$$

Where $\hat{q}$ is a vector containing the flow estimates, q is a vector containing the measured flows, V is a diagonal matrix containing the variances and A_r is a vector containing coefficients describing the direction of the flows into the three-way valve. The vectors and matrices are defined as seen in (3.21).

$$\hat{q} = \begin{bmatrix} \hat{q}_1 \\ \hat{q}_2 \\ \hat{q}_3 \end{bmatrix}, \quad q = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix}, \quad A_r = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}, \quad V = \begin{bmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \sigma_2^2 & 0 \\ 0 & 0 & \sigma_3^2 \end{bmatrix} \tag{3.21}$$

For each time step it is now possible to obtain estimates of the flows which satisfy the mass balance by using equation (3.20).

3.2.2 Calibration of Pump Gains

It has been established that the pumps should be modeled with constant gains. This gain should produce a linear relation between input and output that best fits the measured steady-state flows obtained from the physical process. This is done by taking the mean value of the flow measurements for each pump, and dividing it by the mean input signal to the respective pump. K_{p1} and K_{p2} is calculated as shown in (3.22).

$$K_{p1} = \frac{\bar{q}_5}{\bar{u_1}}, \quad K_{p2} = \frac{\bar{q}_6}{\bar{u_2}} \tag{3.22}$$

where $\bar{q}_5$ is the mean value of the flow coming from pump 1, $\bar{q}_6$ is the mean value of the flow coming from pump 2, $\bar{u}_1$ is the mean input signal to pump 1 and $\bar{u}_2$ is the mean input signal to pump 2.

3.2.3 Calibration of Valve Coefficients

The valve coefficients are dependent on the flow running through the valve and the level of water in the tank as seen in (3.23).

$$c = \frac{q}{\sqrt{h}} \tag{3.23}$$

Throughout an experiment the level and inflow of the upper tanks are always known. A change in level between two samples indicates how much water has been lost or gained during that time. This information is used to calculate the flow out of a tank at each time step using equation (3.24).

$$\dot{V}_{out(k)} = \dot{V}_{in(k)} - \frac{h_{(k+1)} - h_{(k)}}{\Delta t} A \tag{3.24}$$

The flows out of the upper tanks are now known, which means that the flow into the lower tanks are also known and can be calculated using equation (3.25).

$$\dot{V}_{out(k)} = \dot{V}_{in(k)} + \underbrace{\dot{V}_{out(k)}}_{\text{Upper Tank}} - \frac{h_{(k+1)} - h_{(k)}}{\Delta t} A \tag{3.25}$$

The flow through each discharge valve is now known for each time step and the discharge coefficients can be calculated using (3.23) by taking the mean value of the flow through the valve and dividing it by the mean value of the square root of the level.

3.3 Model Validation

After the model is calibrating it needs to be validated to see how well the model represents the physical system. Calibration and validation is performed using separate data sets which are both obtained with the same valve configurations on the physical process. The calibrated model is validated by simulating the model with the same pump inputs as used in the validation experiment. The simulated values are then compared to the measured data. The initial values of the validation data set are used as initial values for the simulation , however for the rest of the simulation the model is relying on the previous model output, meaning that the previous model outputs are used as initial values for computing the next model outputs at each time step. This can be seen in figure 3.7 which shows how the model and the real process are controlled using the same input, and how the outputs are compared. The reason for validating the model without feedback from the measured data is that when the model is used for MPC it does not have feedback during the prediction, and the accuracy of the prediction is therefor relying on the model performance alone. It therefor makes sense to validate how the model performs in open loop. The error between the model and experimental data indicates how the error propagates during the prediction. Results from model validation will be discussed in chapter 6.

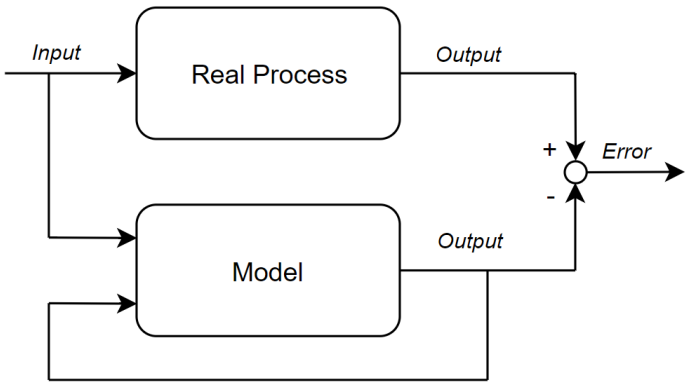


Figure 3.7: Diagram showing validation procedure

It was discovered that the actual values of γ_1 and γ_2 are not fixed, but in fact vary with the pump speed even though the three-way valve positions are fixed. This brings another level of uncertainty to the model. It is therefore of interest to find out both how well the parameters are calibrated, and how well the model performs. Since the flows are measured throughout the experiment, the exact values of γ_1 and γ_2 can be calculated for each sample and thus it is possible to eliminate the uncertainty of γ_1 and γ_2. Eliminating the uncertainty of the split factors ensures a more reliable validation of the valve coefficients and pump gains, however the model will have a fixed value of γ_1 and γ_2 when used in the Kalman filters and model based controllers. The model was therefore validated both with the model receiving the actual value of γ_1 and γ_2, and with γ_1 and γ_2 fixed.

3.4 Analysis

This section covers analysis of the linear quadruple tank process model which is important for the design of state estimators and controllers. This includes controllability, observability, stability. It will also cover analysis of the interaction that occurs due to the cross coupling of the 4 tanks.

3.4.1 Observability, Controllability and Stability

When designing state estimators and controllers it is important to know if the system is observable, controllable and stable. The states that are to be predicted need to be observable in order for the Kalman filter to work. The system is observable if the rank of the observability matrix (3.26) is equal to the number of states [13].

$$O_4 = \begin{bmatrix} C \\ CA \\ CA^2 \\ CA^3 \end{bmatrix} \tag{3.26}$$

It is known that the system order is 4 since there are 4 states, so the rank of the observability matrix needs to be 4 for the system to be observable. MATLAB is used to find the rank of O_4 using the function *rank.m*. MATLAB returns rank equal to 4 for the observability matrix (3.26), hence the system is observable.

To control a process it needs to be controllable. If the system is not controllable it means that the system has states that cannot be controlled by one of the control inputs [14]. A system is controllable if the rank of the controllability matrix (3.27) is equal to the system order.

$$C_4 = \begin{bmatrix} B & AB & A^2B & A^3B \end{bmatrix} \tag{3.27}$$

Again the MATLAB function *rank.m* is used to compute the rank of C_4, which returns the rank 4 for the controllability matrix, hence the system is controllable.

The poles of a system which is controllable and observable are the eigenvalues of the system matrix A. For a system to be stable the real part of the poles need to be negative. The system is therefore stable if the eigenvalues of A are negative. Due to the nature of the system matrix, the eigenvalues are the values of the diagonal of A. These values will always be negative since the model parameters are strictly positive. The system is therefore stable. [15]

3.4.2 Zero Location

If a system has fewer control inputs or outputs than the number of states, the system generally has zeros. This applies for the quadruple tank process since it has 4 states and 2 control inputs and outputs. If all the system zeros are located in the left half of the complex plane it is a minimum phase system, however if one or more zeros are located in the right half of the complex plane, the system is a non-minimum phase system.[16]

The system transfer matrix is analyzed in order to determine the location of the zeros. This matrix is obtained by taking the Laplace transform of the linearized model developed in 3.1.2 and arranging the individual transfer functions (3.28) into a matrix.

$$
\begin{aligned}
h_1(s) &= T_1 \frac{K_{p1}(1-\gamma_1)}{A(T_1 s+1)} u_1(s) \\
h_2(s) &= T_2 \frac{K_{p2}(1-\gamma_2)}{A(T_2 s+1)} u_2(s) \\
h_3(s) &= \frac{K_1 K_{p1}(1-\gamma_1)}{(T_1 s+1)(T_3 s+1)} u_1(s) + \frac{K_1 K_{p2}\gamma_2}{(T_3 s+1)} u_2(s) \\
h_4(s) &= \frac{K_2 K_{p2}(1-\gamma_2)}{(T_2 s+1)(T_4 s+1)} u_2(s) + \frac{K_2 K_{p1}\gamma_2}{(T_4 s+1)} u_1(s)
\end{aligned}
\tag{3.28}
$$

where $T_i = \frac{2A\sqrt{h_i^o}}{c_i}$ is the time constant of tank i, $K_1 = \frac{T_3}{A}$ and $K_2 = \frac{T_4}{A}$.

Arranging the transfer functions in (3.28) into a matrix gives the transfer matrix H(s) (3.29).

$$
\begin{bmatrix} h_3(s) \\ h_4(s) \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{K_1 K_{p1}(1-\gamma_1)}{(T_1 s+1)(T_3 s+1)} & \frac{K_1 K_{p2}\gamma_2}{(T_3 s+1)} \\ \frac{K_2 K_{p1}\gamma_1}{(T_4 s+1)} & \frac{K_2 K_{p2}(1-\gamma_2)}{(T_2 s+1)(T_4 s+1)} \end{bmatrix}}_{H(s)} \begin{bmatrix} u_1(s) \\ u_2(s) \end{bmatrix} \tag{3.29}
$$

Since the transfer matrix $H(s)$ is a 2 by 2 matrix, the zero polynomial is the numerator of the determinant of $H(s)$ when arranged so that the denominator of $H(s)$ is equal to the pole polynomial. The pole polynomial $\rho(s)$ is the smallest common denominator of all the under determinants of $H(s)$. Since the system is a 2 by 2 matrix, the pole polynomial is the smallest common denominator of the determinant of $H(s)$ as seen in (3.30) .[16]

$$
\rho(s) = (T_1 s+1)(T_2 s+1)(T_3 s+1)(T_4 s+1) \tag{3.30}
$$

The determinant of $H(s)$ when arranged so that the denominator is equal to $\rho(s)$ is shown in (3.31).

$$
|H(s)| = \frac{K(1-\gamma_1)(1-\gamma_2) - K\gamma_1\gamma_2(T_1 s+1)(T_2 s+1)}{(T_1 s+1)(T_2 s+1)(T_3 s+1)(T_4 s+1)} \tag{3.31}
$$

Where $K = K_1 K_2 K_{p1} K_{p2}$.

Since the denominator is equal to the pole polynomial, the zero polynomial $\pi(s)$ is equal to the numerator of (3.31) as seen in (3.32).

$$\pi(s) = K(1-\gamma_1)(1-\gamma_2) - K\gamma_1\gamma_2(T_1 s+1)(T_2 s+1) \tag{3.32}$$

The zeros are the roots of $\pi(s) = 0$. Equation (3.33) shows the zero polynomial equated to 0.

$$\underbrace{K\gamma_1\gamma_2 T_1 T_2}_{a}\ s^2 + \underbrace{K\gamma_1\gamma_2(T_1+T_2)}_{b}\ s + \underbrace{K\big(\gamma_1\gamma_2 - (1-\gamma_1)(1-\gamma_2)\big)}_{c} = 0 \tag{3.33}$$

The terms a and b will always be positive but the constant term c will change sign if $(1-\gamma_1)(1-\gamma_2)$ is greater than $\gamma_1\gamma_2$. If the constant term c in (3.33) is negative, one of the solutions of $\pi(s) = 0$ will be positive and thus result in one positive zero, and the system being non-minimum phase. For this to happen, $(1-\gamma_1)(1-\gamma_2)$ needs to be larger than $\gamma_1\gamma_2$. Therefore, the system will be minimum phase if $\gamma_1 + \gamma_2 > 1$ and non-minimum phase if $\gamma_1 + \gamma_2 < 1$.

3.4.3 Relative Gain Array

The relative gain array (RGA) is studied to get a better understanding of the process coupling. It provides information about steady-state process interaction using the steady state gains, given that the process is open loop stable [17]. The RGA is calculated from the transfer matrix by taking the Hadamard product (element by element multiplication) of the matrix and it's inverse transposed with $s = 0$ as shown in (3.34) [15].

$$RGA = H(s=0) \circ H(s=0)^{-T} \tag{3.34}$$

When putting $s = 0$ the transfer matrix consists of only the steady-state gains. The steady state gain matrix can be seen in (3.35).

$$H(s=0) = \begin{bmatrix} K_1 K_{p1}(1-\gamma_1) & K_1 K_{p2}\gamma_2 \\ K_2 K_{p1}\gamma_1 & K_2 K_{p2}(1-\gamma_2) \end{bmatrix} \tag{3.35}$$

Matrix (3.35) should be element wise multiplied with the inverse transposed. The matrix is therefor first inverted, and then transposed. The result can be seen in (3.36).

$$H(s=0)^{-T} = \begin{bmatrix} \frac{K_2K_{p2}(1-\gamma_2)}{K\left((1-\gamma_1)(1-\gamma_2)-\gamma_1\gamma_2\right)} & -\frac{K_2K_{p2}\gamma_1}{K\left((1-\gamma_1)(1-\gamma_2)-\gamma_1\gamma_2\right)} \\ -\frac{K_1K_{p2}\gamma_2}{K\left((1-\gamma_1)(1-\gamma_2)-\gamma_1\gamma_2\right)} & \frac{K_1K_{p1}(1-\gamma_1)}{K\left((1-\gamma_1)(1-\gamma_2)-\gamma_1\gamma_2\right)} \end{bmatrix} \tag{3.36}$$

Where $K = K_1K_2K_{p1}K_{p2}$.

By taking the Hadamard product of (3.35) and (3.36), one has obtained the relative gain array as seen in (3.37).

$$RGA = \begin{bmatrix} \frac{(1-\gamma_1)(1-\gamma_2)}{(1-\gamma_1)(1-\gamma_2)-\gamma_1\gamma_2} & -\frac{\gamma_1\gamma_2}{(1-\gamma_1)(1-\gamma_2)-\gamma_1\gamma_2} \\ -\frac{\gamma_1\gamma_2}{(1-\gamma_1)(1-\gamma_2)-\gamma_1\gamma_2} & \frac{(1-\gamma_1)(1-\gamma_2)}{(1-\gamma_1)(1-\gamma_2)-\gamma_1\gamma_2} \end{bmatrix} \tag{3.37}$$

By analyzing the RGA one can determine how inputs and outputs should be paired if using single loop controllers. An input should be paired with an output corresponding to a positive value in the RGA. It can be seen from (3.37) that the sign of any of the matrix indexes will be determined by the sign of the denominator. The RGA contains no other parameters than the γ values, hence these parameters determine the signs. If $(1-\gamma_1)(1-\gamma_2) > \gamma_1\gamma_2$, the denominator will be positive and the signs in the RGA remain unchanged. However if $(1-\gamma_1)(1-\gamma_2) < \gamma_1\gamma_2$, all the matrix positions change sign which means that the inputs and outputs should be paired the opposite way.

It was shown in section 3.4.2 that if the sum of γ_1 and γ_2 is less than 1, the system has a positive zero and is therefore non-minimum phase. By analyzing the RGA, it can be seen that if the sum of γ_1 and γ_2 is less than 1, the denominator is positive and the signs of the terms in the RGA remain unchanged. This means that output 1, which is tank 3, should be controlled by input 1 and output 2, which is tank 4 should be controlled by input 2. So in a non-minimum phase configuration the inputs and outputs should be paired in the opposite way of how the system is modeled, since pump 1 is directly linked to tank 4 and pump 2 is directly linked to tank 3. In this configuration the water from the upper tanks contribute more than the water coming directly from the pumps. This gives an intuitive interpretation of the interaction.

Based on the analysis above, there is a correlation between zero location and interaction. To illustrate the challenges of having positive zeros, the model was simulated and controlled by 2 PI controller which are single loop controllers. Figure 3.8 is a plot of controlling the simulator in a minimum phase configuration using 2 PI controllers.

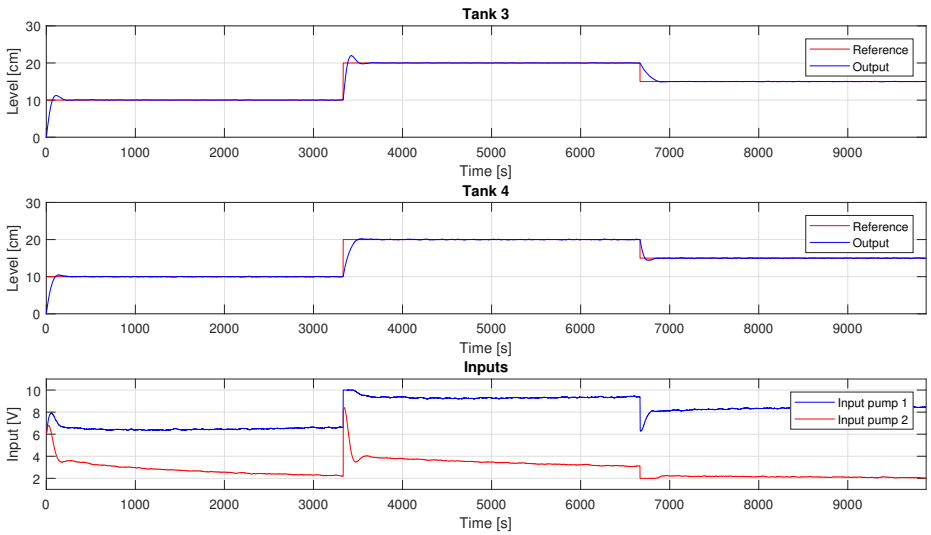


Figure 3.8: Controlling simulated process with PI controller in minimum phase

It can be seen that the controllers have no problem following the references and that the controller outputs (process inputs) are working within their range. Figure 3.9 shows the result of controlling the same simulator with the same 2 PI controllers, however now with a non-minimum phase configuration.

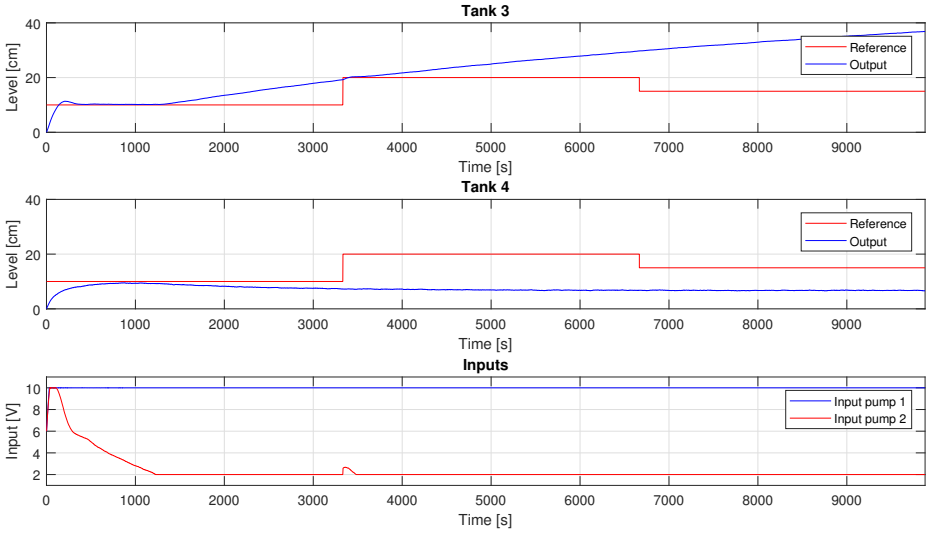


Figure 3.9: Controlling simulated process with PI controller in non-minimum phase

It is clearly seen that the outputs do not follow the references, but in fact deviate in opposite directions. The controller outputs are working at the bounds of their range,

with the input to pump 1 at maximum trying to raise the level in tank 4 and the input to pump 2 at minimum trying to lower the level in tank 3. Since the upper tank contribution is greater than the contribution directly from the pumps, pump 1 is now raising the level in tank 3 instead of lowering the level in tank 4. Pump 2 is trying to compensate for this by trying to lower the level in tank 3, but is in fact lowering the level in tank 4. This clearly shows that the controllers are working against each other.

The non-minimum phase configuration introduces inverse response to the process. Inverse response means that the output goes in the opposite direction of what is desired before it tends towards the desired value [2]. To illustrate inverse response, the simulation model was simulated from steady state levels with a decrease in input to pump 2 and an increase in input to pump 1. The resulting plot can be seen in figure 3.10.

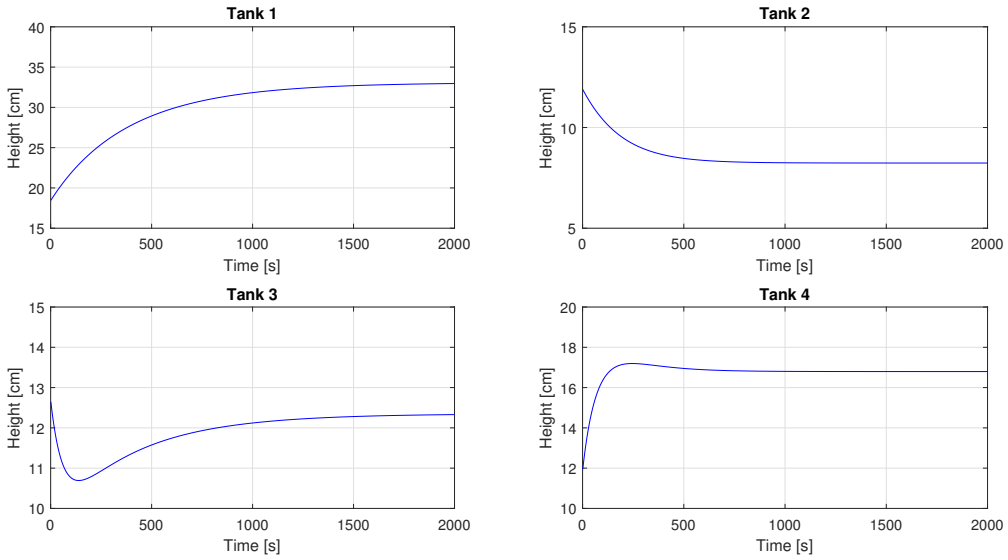


Figure 3.10: Simulation showing inverse response in non-minimum phase configuration

From the plot it can be seen that the level in tank 3 first decreases due to the decrease in input signal to pump 2. However the level stops decreasing when the level in tank 1 has raised and started to contribute significantly to the level in tank 3.

4 Kalman Filter

Model based controllers need information about all the states in the model, which in this case is all the tank levels, however only the levels of the lower tanks are typically measured. Although the process at USN has level sensors on all the tanks, the measurements of the upper tank levels are only used as a reference for comparisons, and thus the levels of the upper tanks need to be estimated. A commonly used state estimator is the Kalman filter. It estimates the states of a system with stochastic disturbances and measurement noise where at least one process variable is measured [13]. This chapter will cover the development of a Kalman filter for the linear models, an extended Kalman filter with a steady state Kalman gain for the non-linear model and an extended Kalman filter with time varying Kalman gain for the non-linear model.

4.1 Linear Kalman Filter

The Kalman filter is defined for linear systems, thus the linear Kalman filter is just called the Kalman filter. It predicts the process states using the discrete time linear state space model developed in 3.1.2. The algorithm consists of 3 steps which can be seen in (4.1).

$$\begin{aligned} \textbf{1.} \quad & \bar{y}_{(k)} = C_d \bar{x}_{(k)} \\ \textbf{2.} \quad & \hat{x}_{(k)} = \bar{x}_{(k)} + K_f \left[y_{(k)} - \bar{y}_{(k)} \right] \\ \textbf{3.} \quad & \bar{x}_{(k+1)} = A_d \hat{x}_{(k)} + B_d u_{(k)} \end{aligned} \tag{4.1}$$

The Kalman filter works in parallel with a physical process and estimates the states by exciting the process model with the same input as the physical process. The model outputs $\hat{y}_{(k)}$ are computed from the current input u_k and the previous state estimate $\bar{x}_{(k)}$, and are subtracted from the outputs $y_{(k)}$ of the physical process, resulting in an error $e_{(k)}$ called the innovation variable. The estimated states are corrected by multiplying the innovation variable with the Kalman gain matrix K_f and adding the result to the estimated states . The corrected states $\hat{x}_{(k)}$ are used to compute the state estimates for the next time step. When using a controller it is typically the corrected states $\hat{x}_{(k)}$ that are used . Figure 4.1 shows a block diagram of the Kalman filter in parallell with the physical process. [13]

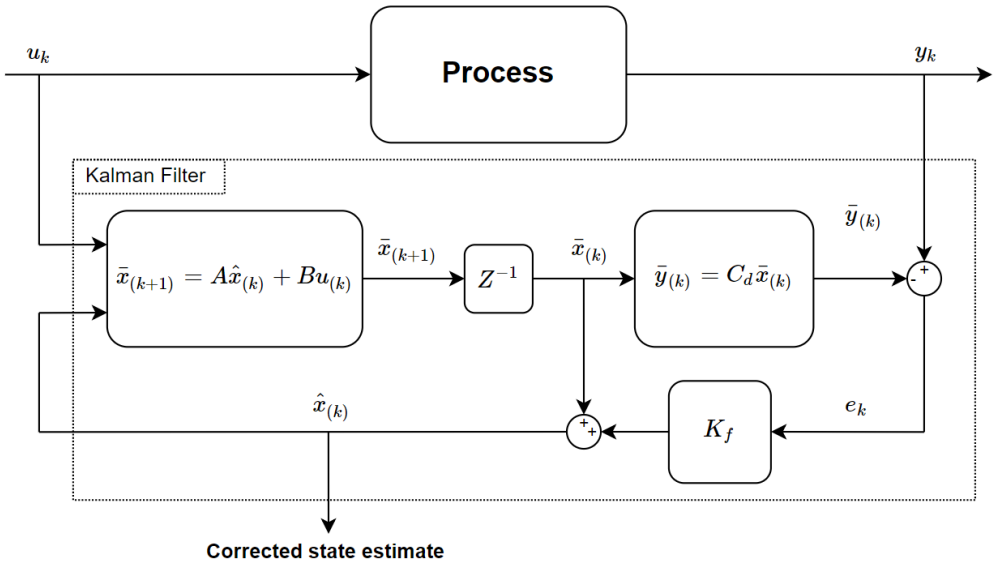


Figure 4.1: Block diagram of Kalman filter [13]

The Kalman filter gain is a time varying matrix, however for the linear Kalman filter with constant matrices the value of the Kalman filter gain matrix converges to a steady state value [13]. The steady state Kalman filter gain matrix K_f is found from the MATLAB function *kalman.m* which takes a state space model and the state and output covariance matrices as arguments. How the function is called can be seen in (4.2).

$$\begin{aligned} sys &= ss(A, [B\ G], C, 0) \\ [\sim, K] &= kalman(sys, Q_k, R_k) \end{aligned} \tag{4.2}$$

Where Q_k is the state covariance matrix, R_k is the output covariance matrix and G is the process noise gain matrix which is set equal to the identity matrix. The Kalman filter was implemented as a MATLAB function which takes the state space matrices , previous state estimate, current input, current process output and Kalman gain matrix as arguments. The function can be seen in figure 4.2.

```
function x = linearKalmanFilter(A,B,C,x,y,u,K)

    y_est = C*x;              % Computing estimated outputs
    x = x + K*(y - y_est);    % Correcting state estimate
    x = A*x + B*u;            % Updating state estimate
end
```

Figure 4.2: Matlab function for linear Kalman filter

4.2 Extended Kalman Filter

The Kalman filter is only defined for linear state space models, however it may be of interest to use a Kalman filter with a non-linear model. For this purpose the Kalman filter has been extended to use non-linear models and is referred to as the Extended Kalman Filter (EKF) [13]. The EKF can be used with a steady state gain as in the linear Kalman filter or with a varying gain. Both of these versions of the EKF will be developed in this section.

4.2.1 Extended Kalman Filter with Steady State Kalman Gain

The Extended Kalman filter with steady state gain uses the non-linear model to predict the state estimates and the Kalman gain obtained from the linear model to correct the states. The algorithm has the same structure at the linear version and can be seen in (4.3).

$$\begin{aligned} \textbf{1.} \quad & \bar{y}_{(k)} = g(\bar{x}_{(k)}) \\ \textbf{2.} \quad & \hat{x}_{(k)} = \bar{x}_{(k)} + K_f\left[(y_{(k)} - \bar{y}_{(k)}\right] \\ \textbf{3.} \quad & \bar{x}_{(k+1)} = f(\hat{x}_{(k)}, u_{(k)}) \end{aligned} \tag{4.3}$$

where $g\left(\bar{x}_{(k)}\right)$ is a non-linear output equation with respect to the predicted states $\bar{x}_{(k)}$ and $f\left(\hat{x}_{(k)}, u_{(k)}\right)$ is a non-linear state equation with respect to the corrected states $\hat{x}_{(k)}$ and the inputs u_k. The outputs of the quadruple tank process are level 3 and 4, thus the output equation is linear and the previously developed C_d matrix can be used. The only difference between the linear and extended Kalman filter is then the non-linear state estimation. This EKF with steady state Kalman gain was implemented as a MATLAB function which takes the model parameters ,the time step and the previous state estimate as arguments. The function for the Extended Kalman filter with steady state Kalman gain can be seen in appendix C.

4.2.2 Extended Kalman Filter with Time Varying Kalman Gain

Another approach to the Extended Kalman filter is to use a time varying Kalman filter gain. The states are then predicted using the non-linear model and corrected using a time varying Kalman gain. The gain is updated at each time step using the state and output covariance matrices, the system matrix A_d and output matrix C_d of the linear state space model which are linearized around the most recent state estimate. Computing the gain is a three step algorithm which can be seen in (4.4). [13]

$$
\begin{aligned}
\textbf{1.} \quad K_{f(k)} &= P_{p(k)} C_d^T \left[C_d P_{p(k)} C_d^T + R_k \right]^{-1} \\
\textbf{2.} \quad P_{c(k)} &= \left[I - K_{f(k)} C_d \right] P_{p(k)} \\
\textbf{3.} \quad P_{p(k+1)} &= A_d P_{c(k)} A_d^T + G_k Q_k G_k^T
\end{aligned}
\tag{4.4}
$$

where $P_{c(k)}$ is the auto-covariance of the corrected state error, $P_{p(k+1)}$ is the auto-covariance of the predicted state error and G_k is the process noise gain matrix. The initial value of $P_{p(k)}$ is set equal to the identity matrix. The complete algorithm for the Extended Kalman filter with time varying gain can be seen in (4.5).

$$
\begin{aligned}
\textbf{1.} \quad A_d &= I + \Delta t \frac{\partial f(x)}{\partial x}\Big|_{(\bar{x}_{(k)}, u_{(k)})} \\
\textbf{2.} \quad K_{f(k)} &= P_{p(k)} C_d^T \left[C_d P_{p(k)} C_d^T + R_k \right]^{-1} \\
\textbf{3.} \quad P_{c(k)} &= \left[I - K_{f(k)} C_d \right] P_{p(k)} \\
\textbf{4.} \quad P_{p(k+1)} &= A_d P_{c(k)} A_d^T + G_k Q_k G_k^T \\
\textbf{5.} \quad \bar{y}_{(k)} &= C_d \bar{x}_{(k)} \\
\textbf{6.} \quad \hat{x}_{(k)} &= \bar{x}_{(k)} + K_{f(k)} \left[y - \bar{y}_{(k)} \right] \\
\textbf{7.} \quad \bar{x}_{(k+1)} &= f(\hat{x}_{(k)}, u_{(k)})
\end{aligned}
\tag{4.5}
$$

where $f(x)$ is the continuous time state equation.

The algorithm in (4.5) was implemented in a MATLAB function which can be found in appendix D.

5 Model Based Controllers

Model based controllers use a mathematical model of the process to obtain information on how the process should be control [13]. This chapter covers the development of model based controllers in the form of a linear Model Predictive controller (MPC), a non-linear Model Predictive controller and a linear quadratic optimal controller (LQ).

5.1 Model Predictive Control

Model predictive control is the strategy of solving an optimal control problem with receding horizon. An optimal control problem is a dynamic optimization problem with a specified length of prediction called the prediction horizon. Solving an optimal control problem results in optimal control inputs for each time step of the whole prediction horizon, however the dynamic model used in an optimal control problem contains errors which result in model mismatch between the model and the physical process which should be controlled. The error will grow during the horizon length, thus the optimal control inputs computed for the whole prediction horizon may not result in the desired control behaviour. The solution is to introduce feedback in the form of a receding horizon. This is done by solving the optimal control problem and using only the first optimal control inputs on the process. At the next time step the horizon is slided one sample and a new optimal control problem is solved. The new optimal control inputs are used on the process and the horizon is again slided one time step. Advantages of model predictive control are that one can add bounds to the inputs, states and outputs, preventing the controller from controlling the process out of acceptable ranges. Another advantage is that future information about the reference allows the controller to start compensation before a reference change occurs, thus preventing aggressive controller behaviour. [10]

This section covers the development of a linear model predictive controller where the optimal control problem is solved subject to the linear process model and a non-linear predictive controller where the optimal control problem is solved subject to the non-linear process model.

5.1.1 Linear Model Predictive Control

Linear model predictive control uses a linear model to predict the future process behaviour. The linearized model developed in 3.1.2 is used as linear constraints for a linear optimal control problem. The objective of the optimal control problem is to minimize the error between the reference and the output for the whole prediction horizon with respect to the decision variable $u_{(k)}$ while constrained by the process model. It should also minimize the rate of change in the control input to prevent an aggressive controller. The objective function J for the optimal control problem can be seen in (5.1).

$$\min_{(u_{(k)})} J = \sum_{k=1}^{N} \left[e_{(k)}^T E e_{(k)} + \Delta u_{(k)}^T P \Delta u_{(k)} \right] \tag{5.1}$$

Where E is the weighting matrix for the errors, P is the weighting matrix for the rate of change of the inputs and $e_{(k)}$ is the error between the reference $r_{(k)}$ and the output $y_{(k)}$. The relation between the weighting matrices E and P determines how aggressive the controller is, hence they are used as tuning matrices. The constraints of the optimal control problem can be seen in (5.2).

$$\begin{aligned} x_{(k+1)} &= A_d x_{(k)} + B_d u_{(k)} \\ y_{(k)} &= C_d x_{(k)} \\ \Delta u_{(k)} &= u_{(k)} - u_{(k-1)} \\ e_{(k)} &= r_{(k)} - y_{(k)} \\ u_L &\leq u_k \leq u_U \end{aligned} \tag{5.2}$$

The optimal control input must be withing the accepted range, thus the inputs $u_{(k)}$ are bounded by an upper limit $u_U = 10V$ and lower limit $u_L = 2V$.

Although having introduced receding horizon, model mismatch may cause an offset between the reference and output due to the propagation of the error throughout the prediction horizon. Since the linear model is an approximation of the non-linear process, this model mismatch may be significant when operating the process far away from the operating points which the linear model was linearized around. An offset may also occur due to the process being subject to other unknown disturbances. To illustrate how model mismatch causes and offset, a simulation was performed where the non-linear model was controlled by a linear MPC. Figure 5.1 shows the result of the simulation.

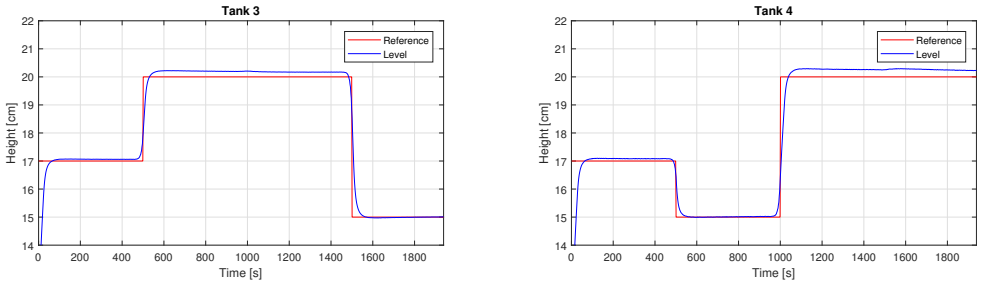


Figure 5.1: Linear MPC controlling non-linear simulation model

The offset that can be seen in figure 5.1 is due to model mismatch caused by linearization. It can be seen that the offset varies with the reference. This happens because the model is less accurate at levels which are far from the operating points than at levels that are close to the operating points. It is of interest to eliminate the offset caused by model mismatch and other unknown disturbances, and integral action was therefor added to the controller. Offset free MPC is achieved using a technique called the Δu formulation [10]. The process is assumed to be affected by constant or slowly varying disturbances. The discrete state space model is therefor expanded with disturbances v and w as seen in (5.3).

$$\begin{aligned} x_{(k+1)} &= A_d x_{(k)} + B_d u_{(k)} + v_{(k)} \\ y_{(k)} &= C_d x_{(k)} + w_{(k)} \end{aligned} \tag{5.3}$$

Since the disturbances are slowly varying, $v_{(k+1)} = v_{(k)} = v$ and $w_{(k+1)} = w_{(k)} = v$. This assumption is used to eliminate the disturbances by defining the deviation of states, inputs and outputs as seen in (5.4).

$$\begin{aligned} \Delta x_{(k+1)} &= x_{(k+1)} - x_{(k)} \\ \Delta x_{(k)} &= x_{(k)} - x_{(k-1)} \\ \Delta u_{(k)} &= u_{(k)} - u_{(k-1)} \\ \Delta y_{(k)} &= y_{(k)} - y_{(k-1)} \end{aligned} \tag{5.4}$$

The definitions in (5.4) are used to reformulate the state space model and eliminate v and w as seen in (5.5).

$$\begin{aligned} \Delta x_{(k+1)} &= A_d x_{(k)} + B_d u_{(k)} + \cancel{v} - \left(A x_{(k-1)} + B_d u_{(k-1)} + \cancel{v}\right) \\ y_{(k)} &= y_{(k-1)} + C_d x_{(k)} + \cancel{w} - \left(C_d x_{(k-1)} + \cancel{w}\right) \end{aligned} \tag{5.5}$$

The state equation and output equation are augmented to create an augmented state space model where the previous outputs $y_{(k-1)}$ are additional states. The new state space model can be seen in (5.6).

$$\begin{aligned} \underbrace{\begin{bmatrix} \Delta x_{(k+1)} \\ y_{(k)} \end{bmatrix}}_{\tilde{x}_{(k+1)}} &= \underbrace{\begin{bmatrix} A & 0 \\ C & I \end{bmatrix}}_{\tilde{A}} \underbrace{\begin{bmatrix} \Delta x_{(k)} \\ y_{(k-1)} \end{bmatrix}}_{\tilde{x}_k} + \underbrace{\begin{bmatrix} B \\ 0 \end{bmatrix}}_{\tilde{B}} \Delta u_{(k)} \\ y_{(k)} &= \underbrace{\begin{bmatrix} C & I \end{bmatrix}}_{\tilde{C}} \underbrace{\begin{bmatrix} \Delta x_{(k)} \\ y_{(k-1)} \end{bmatrix}}_{\tilde{x}_k} \end{aligned} \tag{5.6}$$

The new augmented state space model will be the linear constraints for the linear MPC with integral action. The new objective is now to minimize the errors and rate of change of inputs with respect to the decision variable $\Delta u_{(k)}$. This means that the optimal control inputs that are found from solving the optimal control problem are the deviation from the previous optimal control inputs, thus the new optimal value are added to the old optimal value so that $u_{opt(k)} = u_{opt(k-1)} + \Delta u_{opt(k)}$. The new objective function can be seen in (5.7).

$$\min_{(\Delta u_{(k)})} J = \sum_{k=1}^{N} \left[e_{(k)}^T E e_{(k)} + \Delta u_{(k)}^T P \Delta u_{(k)} \right] \tag{5.7}$$

The constraints are re-formulated since the formulation of the process model has changed. It is now no longer necessary to define $\Delta u_{(k)}$ since the process model is formulated using $\Delta u_{(k)}$, and bounds are now put on the rate of change of control input instead of the input itself. The new optimal control problem is subject to the linear constraints that can be seen in (5.8).

$$\begin{aligned} \tilde{x}_{(k+1)} &= \tilde{A}\tilde{x}_{(k)} + \tilde{B}\Delta u_{(k)} \\ y_{(k)} &= \tilde{C}\tilde{x}_{(k)} \\ e_{(k)} &= r_{(k)} - y_{(k)} \\ u_L &\leq \Delta u_{(k)} \leq u_U \end{aligned} \tag{5.8}$$

Solving an optimal control problem can be computational demanding and as a consequence it may take a lot of time. To reduce computational time the problem is reformulated to quadratic programming form to make use of MATLABs *quadprog.m* quadratic programming solver. When formulating the problem in this way the matrices become sparse, which reduces computational time [10]. The objective function on quadratic programming form is shown in (5.9).

$$\min_{(Z)} J = \frac{1}{2} z^T H z + C^T z \tag{5.9}$$

where z is a vector containing all the unknowns for the whole prediction horizon, H is a matrix containing the quadratic coefficients of the objective and C is a vector containing the linear coefficients of the objective. There are no linear terms in the objective function so the matrix C becomes a matrix of zeros equal to the length of z. The objective function is subject to the constraints shown in (5.10).

$$\begin{aligned} A_e z &= b_e \\ A_i z &\leq b_i \\ z_L \leq z &\leq z_U \end{aligned} \tag{5.10}$$

where A_e are the coefficients of the linear equality constraints, b_e are the constants of the linear equality constraints, A_i are the coefficients of the linear inequality constraints, b_i are the constants of the linear inequality constraints, Z_L are the lower bounds of the unknowns and Z_U are the upper bounds of the unknowns. The unknown vector z can be seen in (5.11).

$$z = \begin{bmatrix} \Delta u & \tilde{x} & e & y \end{bmatrix} \tag{5.11}$$

.

Where Δu, $\tilde{x}$, e and y contain all the optimal values for the whole prediction horizon. The solver *quadprog.m* takes H, c, A_e, b_e, A_i, b_i, Z_L and Z_U as arguments, however the optimal control problem does not contain inequality constraints so the arguments for A_i and b_i are empty matrices. How the *quadprog.m* is called can be seen in (5.12).

$$u^* = quadprog(H, C, \underbrace{[\quad]}_{A_i}, \underbrace{[\quad]}_{b_i}, A_e, b_e, Z_L, Z_U) \tag{5.12}$$

A MATLAB function was made for computing the matrices H, c and A_e. This function is called on start up and is only called once since the matrices are constant throughout an experiment. The MATLAB script for this function can be found in appendix E. The *quadprog.m* solver function is implemented in a MATLAB function which is called every time the MPC runs. This function also builds the constant vector b_e since it changes every time the MPC runs due to new information about the reference and new initial states. A MATLAB script for the linear MPC function can be found in appendix F.

5.1.2 Non-Linear Model Predictive Control

Non-linear MPC uses a non-linear model to predict the future process behaviour. The objective of the non-linear optimal control problem is the same as for linear MPC as seen in (5.13).

$$\min_{(u_{(k)})} J = \sum_{k=1}^{N} \left[e_{(k)}^T E e_{(k)} + \Delta u_{(k)}^T P \Delta u_{(k)} \right] \tag{5.13}$$

The objective is now subject to the general non-linear constraints as seen in (5.14).

$$\begin{aligned} x_{(k+1)} &= f(x_{(k)}, u_{(k)}) \\ y_{(k)} &= g(x_{(k)}) \\ \Delta u_{(k)} &= u_{(k)} - u_{(k-1)} \\ e_{(k)} &= r_{(k)} - y_{(k)} \\ U_L &\leq u_{(k)} \leq U_u \end{aligned} \tag{5.14}$$

where $f(x_{(k)}, u_{(k)})$ is a non-linear state function and $g(x_{(k)}, u_{(k)})$ is a non-linear output function. Due to the outputs of the quadruple tank process being the states of the lower tanks, the output equation is equal to the linear case as seen in (5.15) .

$$g(x_{(k)}) = C_d x_{(k)} \tag{5.15}$$

The optimal control problem is non-linear and the *quadprog.m* solver can therefor not be used, thus the *fmincon.m* solver is used instead. Solving an optimization problem with *fmincon.m* allows for more flexibility than with *quadprog.m* since one is not forced to satisfy the strict matrix structure of the quadratic programming formulation, however solving the optimal control problem becomes more time consuming.

A MATLAB function called *computeObjective.m* was created for computing the objective for the whole prediction horizon. It simulates the non-linear model over the horizon length and sums up the objective J for each iteration. This function can be found in appendix G. The *computeObjective.m* function is called by *fmincon.m* which tries the function with different combination of inputs to find the optimal inputs that minimizes the function output J. The *fmincon.m* solver was called inside a MATLAB function which is called every time the MPC should run. At each time step the solver is given the last optimal control inputs as starting points for the optimization. From one time step to another it is likely that the new optimal control input is similar to the previous optimal control input. By using the previous found value as starting point the solver may find the optimal solution using fewer iterations and the computational time is reduced. The function which calls the *fmincon.m* solver can be seen in 5.2.

```
function [Uopt,it] = nonLinearMPC(u0,states,ref,ts,N,E,P,parameters)

% Making the objective function an anonymous function

    obj = @(u)computeObjective(u,states,ts,ref,N,parameters,E,P,u0(1,:)');

    lb = 2*ones(N,2);   % Creating lower bounds for inputs
    ub = 10*ones(N,2);  % Creating upper bounds for inputs

% Setting options for fmincon

    ops = optimset('Algorithm','sqp','Display','off','MaxIter',200);

% Calling fmincon

    [u,~,~,output]  = fmincon(obj,u0,[],[],[],[],lb,ub,[],ops);

    it = output.iterations;   % Number of iterations used to solve problem
    Uopt = [u(1,1);u(1,2)];   % Extracting the first optimal control inputs

end
```

Figure 5.2: MATLAB funtion for non-linear MPC

5.2 Linear Quadratic Optimal Control

A linear Quadratic Optimal Controller is an optimal controller in the sense that is computes an optimal control input based on a model of the process. For this case a controller with infinite horizon and integral action is considered [16]. The optimal control criteria is given by (5.16).

$$J_i = \frac{1}{2}\sum_{k=1}^{\infty}\left[(r-y_{(k)})^T E(r-y_{(k)}) + \Delta u_{(k)}^T P \Delta u_{(k)}\right] \tag{5.16}$$

where $\Delta u_{(k)} = u_{(k)} - u_{(k-1)}$, r is a constant reference and E and P are weighting matrices for the errors and inputs respectively. The model is a discrete time linear state space model with eliminated disturbances (5.17) as developed in 5.1.1.

$$\begin{aligned} \Delta x_{(k+1)} &= A_d \Delta x_{(k)} + B_d \Delta u_{(k)} \\ \Delta y_{(k)} &= C_d \Delta x_{(k)} \end{aligned} \tag{5.17}$$

where $\Delta x_{(k+1)} = x_{(k+1)} - x_{(k)}$, $\Delta x_{(k)} = x_{(k)} - x_{(k-1)}$, $\Delta u_{(k)} = u_{(k)} - u_{(k)}$ and $\Delta y_{(k)} = y_{(k)} - y_{(k-1)}$.

To achieve set point tracking with the constant reference the output equation is modified to include the reference as seen in (5.18).

$$r - y_{(k)} = r - y_{(k-1)} - C_d \Delta x_{(k)} \tag{5.18}$$

The expression obtained in (5.18) is augmented with the state equation to create an augmented state space model including the error as a state. The augmented state space model can be seen in (5.19).

$$\begin{aligned} \underbrace{\begin{bmatrix} \Delta x_{(k+1)} \\ r - y_{(k)} \end{bmatrix}}_{\tilde{x}_{(k+1)}} &= \underbrace{\begin{bmatrix} A_d & 0 \\ -C_d & I \end{bmatrix}}_{\tilde{A}_q} \underbrace{\begin{bmatrix} \Delta x_{(k)} \\ r - y_{(k-1)} \end{bmatrix}}_{\tilde{x}_k} + \underbrace{\begin{bmatrix} B_d \\ 0 \end{bmatrix}}_{\tilde{B}_q} \Delta u_{(k)} \\ \underbrace{r - y_{(k)}}_{\tilde{y}_{(k)}} &= \underbrace{\begin{bmatrix} -C_d & I \end{bmatrix}}_{\tilde{C}_q} \underbrace{\begin{bmatrix} \Delta x_{(k)} \\ r - y_{(k-1)} \end{bmatrix}}_{\tilde{x}_{q(k)}} \end{aligned} \tag{5.19}$$

The model to be used in the formulation of the LQ optimal controller is the augmented state space model (5.20).

$$\begin{aligned} \tilde{x}_{(k+1)} &= \tilde{A}_q \tilde{x}_{(k)} + \tilde{B}_q \tilde{u}_{(k)} \\ \tilde{y}_{(k)} &= \tilde{C}_q \tilde{x}_{(k)} \end{aligned} \tag{5.20}$$

The optimal control criteria is therefor also reformulated as seen in (5.21).

$$J_i = \frac{1}{2} \sum_{k=1}^{\infty} \left[\tilde{y}_{(k)}^T E \tilde{y}_{(k)} + \Delta u_{(k)}^T P \Delta u_{(k)} \right] \tag{5.21}$$

The maximum principle is used to solve the LQ optimal control problem, resulting in a controller formulation for the optimal control input u_k^* as shown in (5.22). [16]

$$\begin{aligned} G &= -\left(P + \tilde{B_q}^T R_c \tilde{B_q}\right)^{-1} \tilde{B_q}^T R_c \tilde{A_q} \\ \Delta u_{(k)}^* &= G \tilde{x}_{(k)} \end{aligned} \tag{5.22}$$

where R_c is a positive solution to the discrete algebraic Riccati equation (5.23).

$$R_c = \tilde{E} + \tilde{A}^T R_c \tilde{A} - \tilde{A}^T R_c \tilde{B} \left(P + \tilde{B}^T R_c \tilde{B}\right)^{-1} \tilde{B}^T R_c \tilde{A} \tag{5.23}$$

where $\tilde{E} = \tilde{C}^T E \tilde{C}$. Reformulating the optimal control input is shown in (5.24).

$$\begin{aligned} \Delta u_{(k)}^* &= \begin{bmatrix} G_1 & G_2 \end{bmatrix} \begin{bmatrix} \Delta x_{(k)} \\ r - y_{(k-1)} \end{bmatrix} \\ u_{(k)}^* &= u_{(k-1)}^* + G_1 \Delta x_{(k)} + G_2 \left(r - y_{(k-1)}\right) \end{aligned} \tag{5.24}$$

The controller formulation in (5.24) is similar to a PI controller. An advantage of the LQ controller is it's simplicity, however it does not support bounds. The control input is therefor manually restricted to $2 \leq u_{(k)}^* \leq 10$ and therefor the optimal control input may be located outside of the accepted input range. The LQ controller is implemented in a MATLAB function which computes the gain matrices G_1 and G_2. It takes the augmented state space model and input and error weight matrices as arguments. This function is called on start up and only runs once since the matrices are constant. The function can be seen in figure 5.3.

```
function G = OptimalControl(A,B,C,P,E)

Et = C'*E*C;                          % E tilde ( Error weights)

[R,~,~] = dare(A,B,Et,P);             % Solving discrete algebraic Riccati
    equation

G = -(P + B'*R*B)^(-1)*B'*R*A; % Computing gain matrices

end
```

Figure 5.3: MATLAB function for computing gain matrices for LQ controller

6 Results and Discussion

This chapter presents and discusses the results of the developed models, Kalman filters and model based controllers.

6.1 Model Development Results

This section covers the results of the model development and focuses on model validation. The model is re-calibrated (First principles model) or re-identified (Subspace identified model) whenever the opening of the discharge valves on the physical process are changed, so the validation presented here is only valid for a specific set of parameters, however it gives an indication of the performance of the model and calibration procedure. The parameters, which can be seen in table 6.1, were obtained from calibrating the model after the discharge valves were manually adjusted so that the process worked well in a minimum phase configuration.

Table 6.1: Parameters for validation

Parameter	**Value**	**Unit**
c_1	7.5844e-05	$m^{2.5}/\mathrm{s}$
c_2	8.9773e-05	$m^{2.5}/\mathrm{s}$
c_3	3.1148e-04	$m^{2.5}/\mathrm{s}$
c_4	2.9812e-04	$m^{2.5}/\mathrm{s}$
K_{p1}	1.8471e-05	m^3/s
K_{p2}	1.7805e-05	m^3/s
A	0.0289	m^2

6.1.1 Results for First Principles Non-linear Model

As discussed in section 3.3 the first principles model is validated in two steps. First it is validated using the actual measured values of γ_1 and γ_2 to get a more reliable validation of the model parameter. Then it is validated using fixed values for γ_1 and γ_2 to validate the model performance since this is how the model will be used in the Kalman filters and

model based controllers. Figure 6.1 shows the model validated when using varying values for γ_1 and γ_2. The actual values pf γ_1 and γ_2 can be seen in figure 6.2.

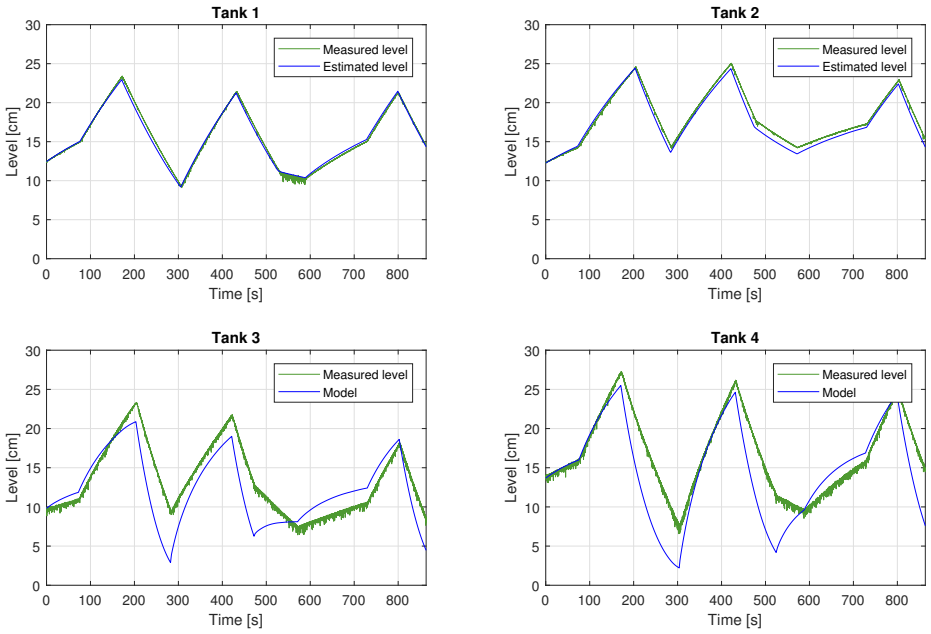


Figure 6.1: Validation of model with varying values for γ_1 and γ_2

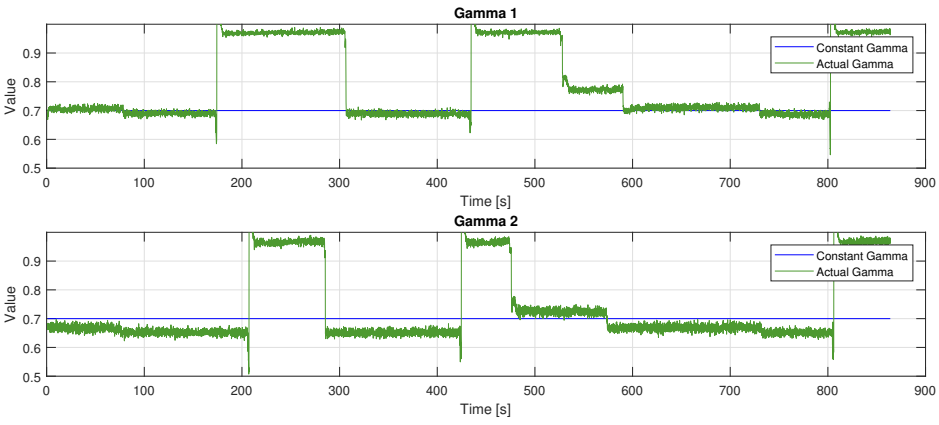


Figure 6.2: Varying γ_1 and γ_2

The plots show that the simulated levels on the upper tanks follow the measured levels very well, both when draining the tanks and when filling them up. This is an indication that both the valve coefficients and the pump gains are well calibrated. However, the plot shows that there is some deviation in the lower tanks. The measured levels are

generally higher than the simulated values. The reason for this is the idle pump speed which contributes with flow to the lower tanks even though the input signal is 0V. It can be seen from figure 6.2 that the values of γ_1 and γ_2 have some variation throughout the validation. Occasionally there are some major deviations. These occur when the pump speed is too low to provide flow to the upper tanks while they still provide flow to the lower tanks. To quantify the performance, the root mean square error (RMSE) was calculated for the validation. Table 6.2 shows the RMSE of validating the non-linear model.

Table 6.2: RMSE of first principles model with varying γ_1 and γ_2

	RMSE
Level 1	0.3441 cm
Level 2	0.6286 cm
Level 3	3.0797 cm
Level 4	3.8945 cm

To validate the performance of the model with a fixed split ratio, the simulation was carried out with values of γ_1 and γ_2 as used during the experiment, which was 0.7 for both γ_1 and γ_2. Figure 6.3 shows the result of validating the non-linear first principles model with fixed split factors.

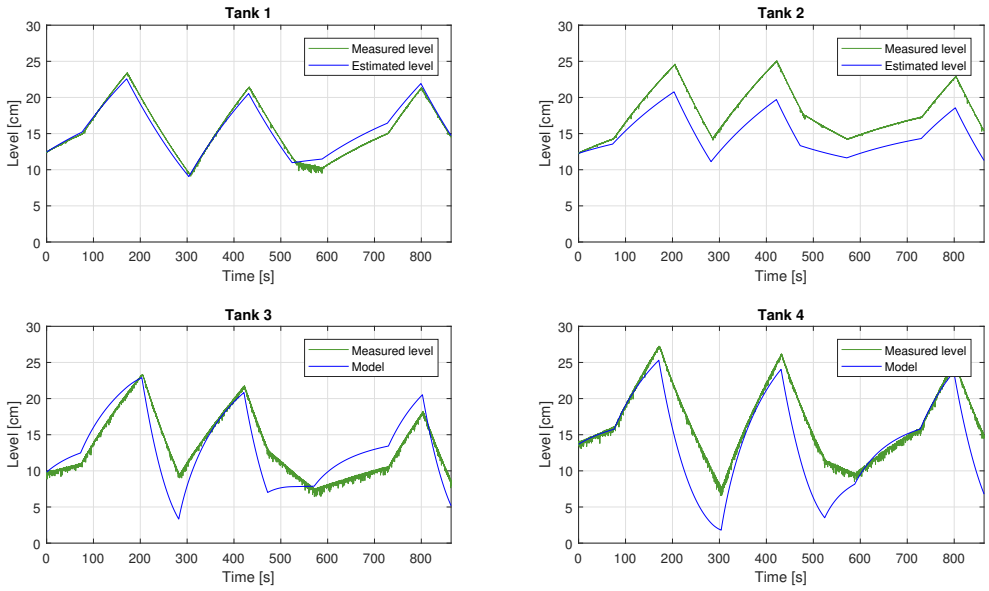


Figure 6.3: Validation of model with $\gamma_1 = 0.7$ and $\gamma_2 = 0.7$

The plot shows that there is a large error in the simulation of tank 2 , however the error is not so large in tank 1. The uncertainty of γ_2 is therefore greater than the uncertainty of γ_1. Fixing the split ratio does not affect the simulation of the lower tanks as much as the

upper tanks. This is due to the fact that the flow in and out of the lower tanks is greater than the flow in and out of the upper tanks since the flow out of the upper tanks goes into the lower tanks. An error in flow from the pumps therefore accounts for a smaller proportion of the total flow into the lower tanks than into the upper tanks.

6.1.2 Results for First Principles Linear Model

The linear model was validated since it is used for the linear Kalman filter, linear MPC and LQ controller. The validation was carried out with $\gamma_1 = 0.7$ and $\gamma_2 = 0.7$ and the resulting plot from the validation can be seen in 6.4. The model used in the validation was linearized around the operating points which can be seen in 6.3.

Table 6.3: Operating points for linear model validation

	Value	**Unit**	**Comment**
h_1^o	19.0	cm	Operating point level 1
h_2^o	12.8	cm	Operating point level 2
h_3^o	11.9	cm	Operating point level 3
h_4^o	13.5	cm	Operating point level 4
u_1^o	6	V	Operating point input 1
u_2^o	6	V	Operating point input 2

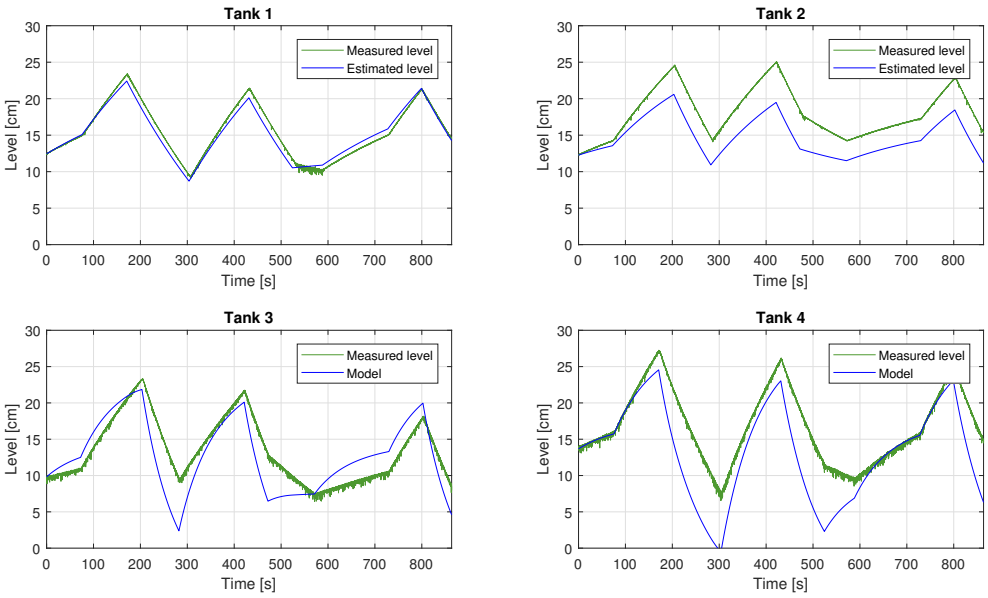


Figure 6.4: Validation of linear first principles model with $\gamma_1 = 0.7$ and $\gamma_2 = 0.7$

From looking at the plot, the performance of the linear model looks to be similar to the performance of the non-linear model. The root mean square error from validation of the non-linear and linear model can be seen in table 6.4.

Table 6.4: RMSE of first principle models with $\gamma_1 = 0.7$ and $\gamma_2 = 0.7$

	Non-linear Model	**Linear Model**
Level 1	0.7381 cm	0.6943 cm
Level 2	3.1717 cm	3.2919 cm
Level 3	2.3749 cm	2.5808 cm
Level 4	2.9538 cm	3.5887 cm

From table 6.4 it can be seen that the performance of the non-linear and the linear model is similar, although the RMSE is slightly greater for the linear model than for the non-linear model. This is as expected since the physical process is non-linear and the linear model is only an approximation around the operating points. Due to the deviation between the measured and simulated values it is expected that there will be some propagation of error in the prediction in the model predictive controllers. Advantages of using a linear model may compensate for the reduced model performance since reduced computational time is gained.

6.1.3 Results for Subspace Model

Due to the internal states of the subspace models being in a different coordinate system than the measured levels, only the levels of the lower tanks were validated. Validation data sets were gathered from the physical process with time steps of $\Delta t = 0.4s$ and $\Delta t = 1s$. The subspace models were simulated against the measured data using the same inputs as used in the validation experiment. The outputs of the simulation model were compared to the measured data. The resulting RMSE values from the validations can be seen in table 6.5 and a plot of the validation of the model with $\Delta t = 1s$ can be seen in figure 6.5.

Table 6.5: RMSE of subspace identified model

Model time step	**RMSE level 3**	**RMSE level 4**
$\Delta t = 0.4s$	3.32 cm	1.12 cm
$\Delta t = 1s$	3.75 cm	0.79 cm

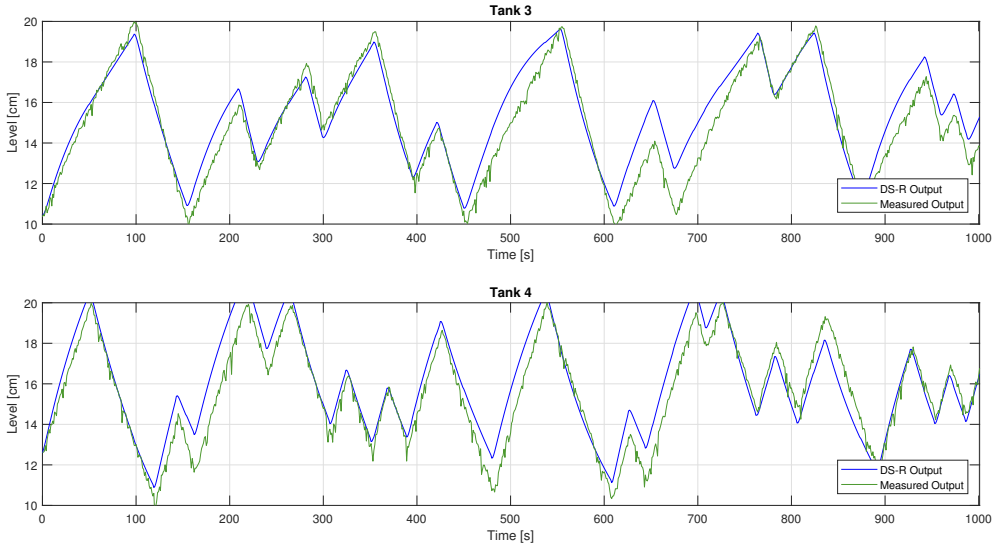


Figure 6.5: Validation of subspace identified model with time step $\Delta t = 1s$

It can be seen from the table in 6.5 that increasing the time step from 0.4s to 1s does not pose any significant difference in performance. The performance of level 4 is actually decreased by increasing the time step. The quadruple tank process is a relatively slow process, thus it is expected that the time step can be increased a reasonable amount without significantly affecting the model performance. The RMSE values of level 3 are slightly larger for the subspace model than the first principles models, however the subspace model has a much lower RMSE in level 4. This combined with the fact that the performance of the upper tanks is not documented makes it hard to determine what is the preferred model.

6.2 Kalman Filter Results

This section covers the results of the Kalman filters developed in chapter 4. The filters are used to estimate the tank levels, however they could also be used to estimate the flows through the discharge valves since the flow is only dependent on the level in the tank. The estimation performance is evaluated by comparing the estimates to the measured levels. Flow estimation is not considered since there is no flow sensor on the discharge vales, thus there is no ground for comparison. The filters were first tested with the simulation model to verify that they were correctly implemented before they were tested against experimental data gathered from the physical process. Noise was added to the simulation model for the Kalman filter simulation. The noise was designed to replicate the noise in the physical sensors by analyzing the standard deviation of the signals from the level sensors on the physical process and adding random noise to the simulated values equivalent to the standard deviation.

It was of interest to see how good a well calibrated Kalman filter performs and how the Kalman filters perform if considerable model mismatch is introduced. Model mismatch of 10% and 20% was introduced in the parameters c_1 and c_2 since they in combination affect all 4 tanks. The calibrated model initially has some model mismatch which may be canceled out by the additional mismatch, resulting in increased performance instead of decreased performance. It was therefor ensured that the parameter errors were introduced in the direction which decreased performance since it is of interest to look at the worst case scenario. All filters were tested against the same data set so that the performance could be compared. Both the simulation and experimental testing was performed in MATLAB. The experimental data was gathered with the physical process configured in minimum phase. The covariance matrices Q_k and R_k were used as tuning matrices, where the diagonal values of Q_k were set equal to 0.01 and the diagonal values of R_k were set equal to 1 for all experiments. These values were found to give the best results. Table 6.6 shows the model parameters used for the Kalman filters.

Table 6.6: Parameters for validation

Parameter	**Value**	**Unit**
c_1	7.5844e-05	$m^{2.5}/\mathrm{s}$
c_2	8.9773e-05	$m^{2.5}/\mathrm{s}$
c_3	3.1148e-04	$m^{2.5}/\mathrm{s}$
c_4	2.9812e-04	$m^{2.5}/\mathrm{s}$
K_{p1}	1.8471e-05	m^3/s
K_{p2}	1.7805e-05	m^3/s
A	0.0289	m^2
γ_1	0.7	
γ_2	0.7	

6.2.1 Results for Linear Kalman Filter with First Principles Linear Model

When obtaining results for the linear Kalman filter, the model was linearized around the operating points which can be seen in table 6.7.

Table 6.7: Operating points for linear Kalman filter

	Value	Unit	Comment
h_1^o	19.0	cm	Operating point level 1
h_2^o	12.8	cm	Operating point level 2
h_3^o	11.9	cm	Operating point level 3
h_4^o	13.5	cm	Operating point level 4
u_1^o	6	V	Operating point input 1
u_2^o	6	V	Operating point input 2

The Kalman filter was simulated against the non-linear model to verify that the filter had been correctly implemented. Figure 6.6 shows a plot obtained from simulating the linear Kalman filter with the non-linear simulation model using identical model parameters for the Kalman filter and the simulation model.

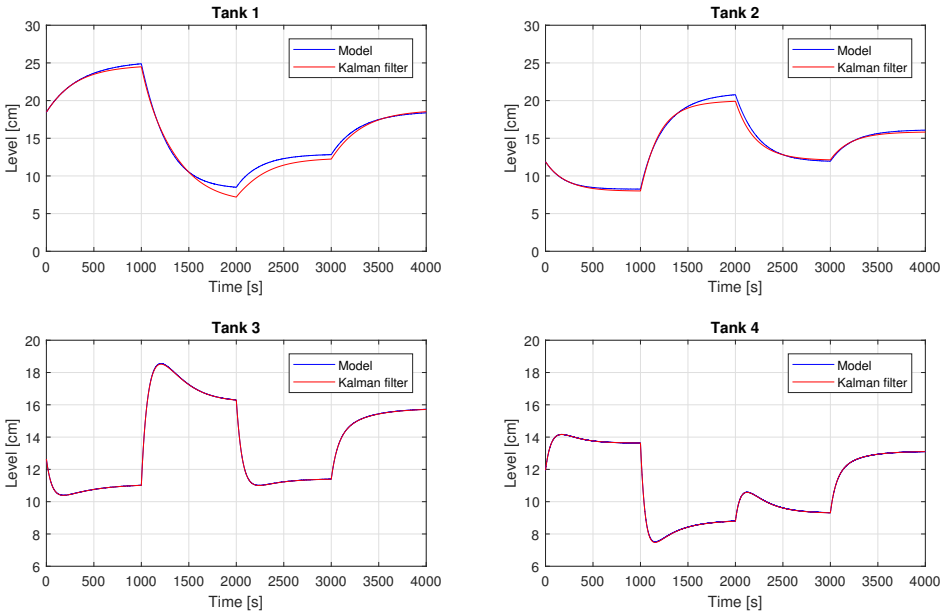


Figure 6.6: Linear Kalman filter on simulation model with the same parameters

It can be seen that the estimated outputs follow the simulator outputs without noticeable deviation, however the levels of the upper tanks deviate slightly. This is due to the model mismatch caused by linearization. The levels follow well when they are close to

the operating points and deviate when moving away from the operating points, thus this is expected behaviour.

It has been established that the actual value of γ_1 and γ_2 on the physical process varies with the pump speed. It was therefore of interest to see how an error in the γ values affects the estimation of the unknown states. A simulation was therefor performed with a 5% error in the γ values. The γ values and root mean square errors of the simulations are shown in table 6.8 and a plot from the simulation with errors in the γ values can be seen in figure 6.7.

Table 6.8: RMSE for Linear Kalman Filter Simulation

γ values	**RMSE level 1**	**RMSE level 2**	**Comment**
$\gamma_1 = 0.7$, $\gamma_2 = 0.7$	0.61 cm	0.33 cm	Correct γ_1 and γ_2
$\gamma_1 = 0.735$,$\gamma_2 = 0.735$	3.26 cm	2.78 cm	5% error in γ_1 and γ_2

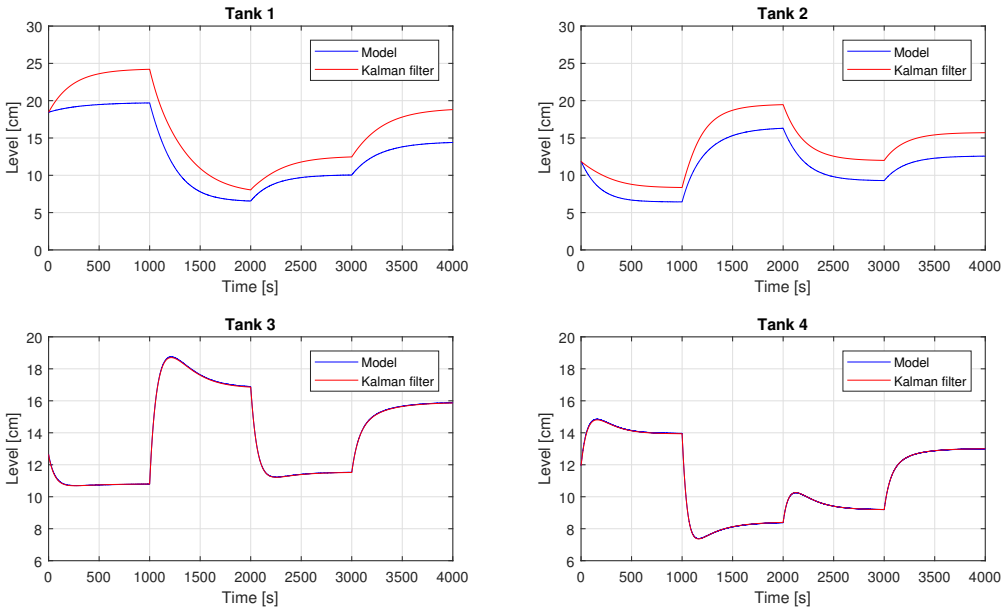


Figure 6.7: Linear Kalman filter on simulation model with 5% errors in γ_1 and γ_2

From the plot in 6.7 it can be seen that even in simulations the uncertainty of the γ values produce noticeable errors in the state estimation. It is therefore expected that the state estimates on the physical process will deviate due to the γ uncertainty.

The Kalman filter was tested on experimental data after verifying the performance in simulations. Table 6.9 shows the root mean square errors of testing the Kalman filter with 0%, 10% and 20% error in c_1 and c_2. A plot of the linear Kalman filter performance with 0% model mismatch can be seen in figure 6.8.

Table 6.9: RMSE from experiment with linear Kalman Filter

Parameters	RMSE level 1	RMSE level 2	Mismatch
$c_1 = 7.584e-5\ cm^{2.5}$ $c_2 = 8.977e-5\ cm^{2.5}$	2.95 cm	1.69 cm	0%
$c_1 = 6.826e-5\ cm^{2.5}$ $c_2 = 8.079e-5\ cm^{2.5}$	2.92 cm	2.43 cm	10%
$c_1 = 6.068e-5\ cm^{2.5}$ $c_2 = 7.182e-5\ cm^{2.5}$	2.88 cm	3.32 cm	20%

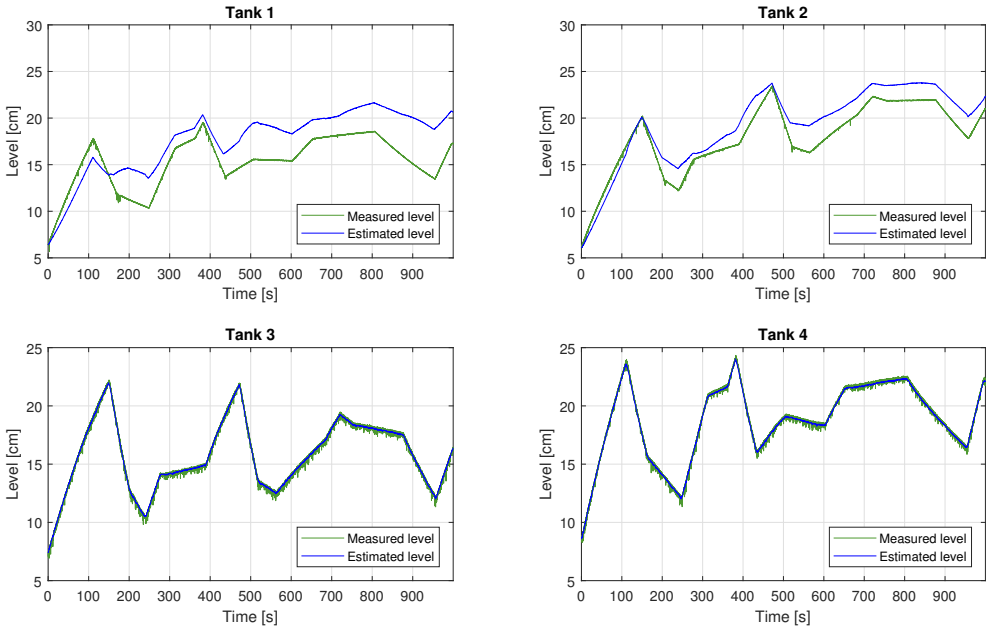


Figure 6.8: Linear Kalman filter on physical process with 0% model mismatch

The resulting plot from the experiment shows that the Kalman filter has no problem estimating the output states, however it does not estimate the upper levels accurately. It can be seen that there is an estimation error of approximately $5cm$ in level 1 towards the end of the experiment. This is equivalent to an error of approximately 25% which would not be acceptable if the unknown states for example were subject to constraints. For plots from experiments with model mismatch see appendix H.

6.2.2 Results for Linear Kalman Filter with Subspace Identified Models

The DS-R returns a realization which is not in the same coordinate system as the measured levels. The Kalman filter was therefor only tested with the lower tank levels. Experiments were performed with time steps $\Delta t = 0.4s$ and $\Delta t = 1s$. The results from using a Kalman filter obtained from a data set with $\Delta t = 0.4s$ can be seen in figure 6.9.

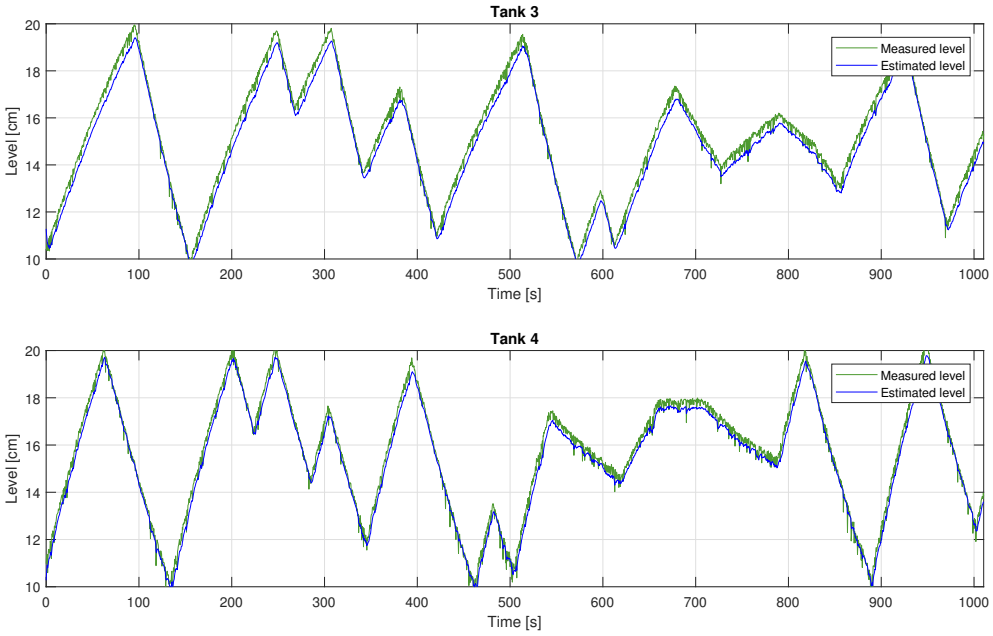


Figure 6.9: Linear Kalman filter from DS-R with $\Delta t = 0.4s$

The plot shows that the estimated outputs follow the measured data. Figure 6.10 shows a plot of using a Kalman filter obtained from a data set with $\Delta t = 1s$. The root mean square error was computed for both cases and is shown in table 6.10

Table 6.10: RMSE for Kalman filter from DS-R

Δt	**RMSE level 3**	**RMSE level 4**
0.4 s	0.39 cm	0.30 cm
1 s	0.60 cm	0.67 cm

From looking at the plot it can be seen that the estimated levels do not follow the measured data as well as in the case where $\Delta t = 0.4s$. The RMSE values confirms the observation made from the plot, showing that the RMSE is approximately twice in the case where $\Delta t = 1s$ compared to $\Delta t = 0.4$. The reason why there is a difference between the two Kalman filters may be because both models are calibrated with approximately $1000s$ of

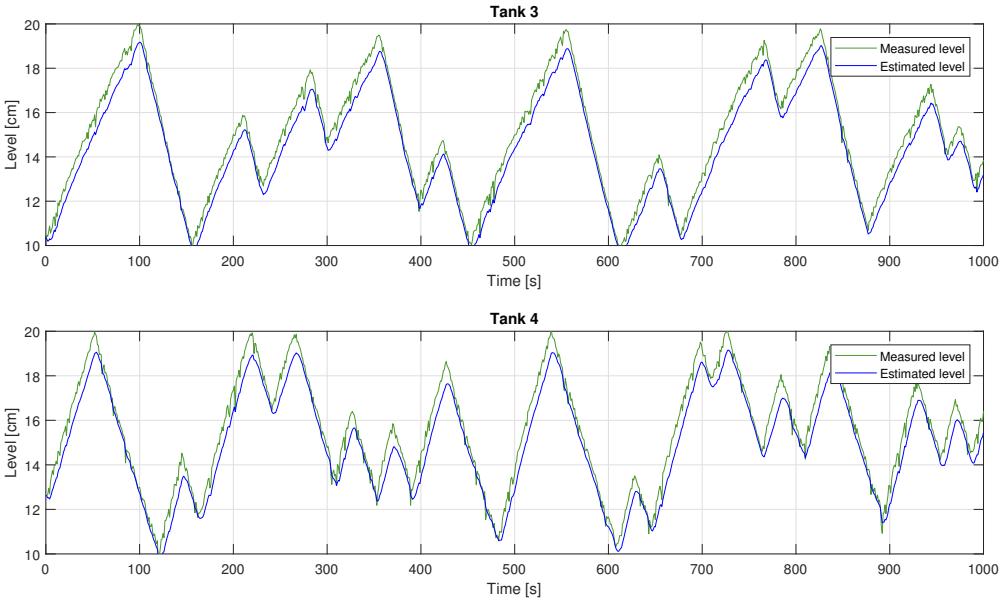


Figure 6.10: Linear Kalman filter from DS-R with $\Delta t = 1s$

data. The model with $\Delta t = 1s$ therefor has less than half the number of samples for calibration. This may result in a less accurate model. Another possible reason is that the model simply becomes less accurate with a larger time step.

6.2.3 Results for Extended Kalman Filter with Steady State Gain

It was not possible to see any deviation between the simulator and the Extended Kalman filter with steady state gain when there was no model mismatch. This makes sense since the filter and the simulation model use identical models. To evaluate the filter performance on the simulator it was simulated with an error in the γ values as done with the linear Kalman filter. The root mean square errors were computed for the simulation with no errors in the γ values and for the simulation with 5% error in the γ values. The RMSE values can be seen in table 6.11 and a plot from the simulation with errors in the γ values can be seen in 6.11

Table 6.11: RMSE from simulation with Extended Kalman Filter with steady state Kalman Gain

γ values	RMSE level 1	RMSE level 2	Comment
$\gamma_1 = 0.7$, $\gamma_2 = 0.7$	0.0011 cm	0.00082 cm	Correct γ_1 and γ_2
$\gamma_1 = 0.735$,$\gamma_2 = 0.735$	3.52 cm	2.90 cm	5% error in γ_1 and γ_2

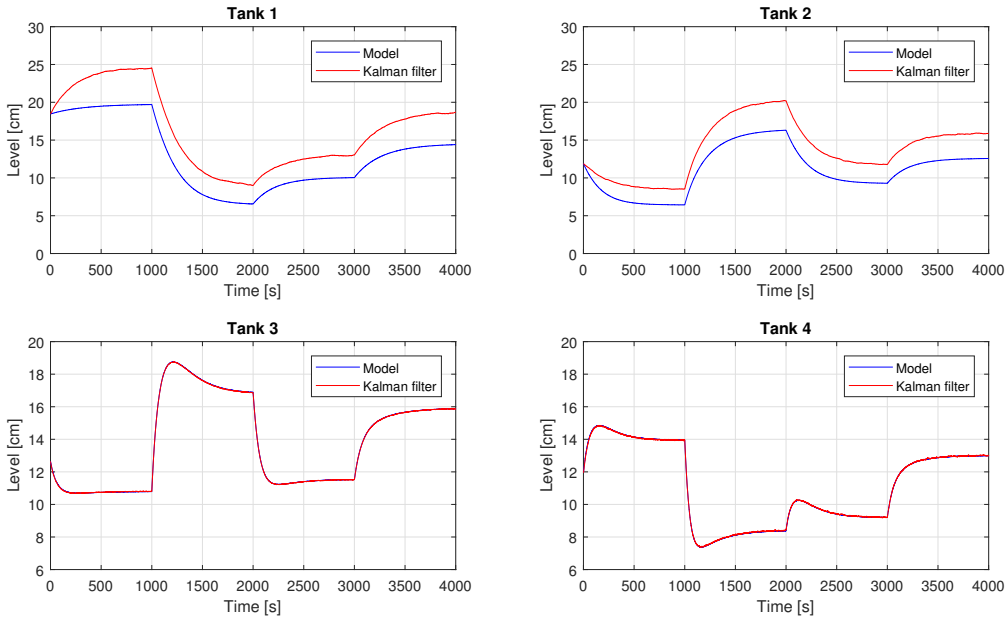


Figure 6.11: Simulation of Extended Kalman filter with steady state gain and 5% errors in γ_1 and γ_2

By comparing the results shown in table 6.11 to the results of linear Kalman filter simulation it is apparent that when there is no model mismatch the Extended Kalman filter performs a lot better than the linear Kalman filter, just as expected. However when errors are introduced in the γ values, the RMSE of the Extended Kalman filter with steady state gain exceeds the RMSE of the linear Kalman filter. This shows that the linear Kalman filter is more robust to model mismatch than the extended Kalman filter. The reason for this is that modeling errors linearly affect the linear Kalman filter , while they affect the Extended Kalman filter in a non-linear manner.

Table 6.12 shows the RMSE values of testing the Extended Kalman filter on experimental data with 0%, 10% and 20% errors in the valve coefficients c_1 and c_2. Figure 6.12 shows a plot the experiment when no additional model mismatch is introduced.

Table 6.12: RMSE from experiment with Extended Kalman Filter with steady state Kalman gain

Parameters	**RMSE level 1**	**RMSE level 2**	**Mismatch**
$c_1 = 7.584e-5\ cm^{2.5}$ $c_2 = 8.977e-5\ cm^{2.5}$	3.99 cm	1.25 cm	0%
$c_1 = 6.826e-5\ cm^{2.5}$ $c_2 = 8.079e-5\ cm^{2.5}$	7.36 cm	4.65 cm	10%
$c_1 = 6.068e-5\ cm^{2.5}$ $c_2 = 7.182e-5\ cm^{2.5}$	11.56 cm	9.15 cm	20%

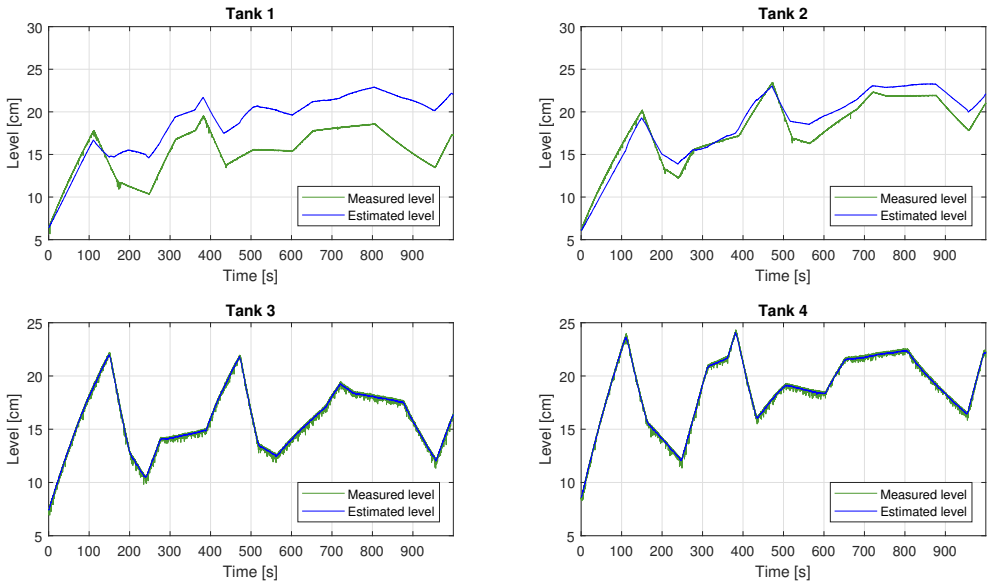


Figure 6.12: EKF with steady state Kalman gain on physical process with no model mismatch

From looking at figure 6.12 it is not clear if the performance of the EKF with steady state Kalman gain is better or worse than the linear Kalman filter, however the RMSE values shows that there is a slightly larger error in the EKF case when no additional model mismatch is introduced. The difference is not big enough to conclude weather of not the filter is better or worse, since a different experiment may result in an RMSE that favours the EKF. The drawbacks of using the EKF do become apparent when looking at the performance when additional model mismatch is introduced. It can be seen from table 6.12 that the RMSE values grows a lot when adding mismatch. The error increases much more for the EKF than for the the linear Kalman filter, which supports the observations that the linear Kalman filter is more robust to modeling errors. For plots from experiments with model mismatch see appendix I.

6.2.4 Results for Extended Kalman Filter with Time Varying Kalman Gain

The EKF with time varying Kalman gain performed similarly to the steady Kalman gain case in simulations. It was not possible to detect any deviation between the estimates and the simulation model from looking at the plot of the simulation when no model mismatch was introduced. As discussed in 6.2.3 this is expected behaviour since the filter and simulator uses identical models. A simulation was performed to see the robustness of the EKF with time varying Kalman gain due to errors in the γ values. Table 6.13 shows the RMSE values of the simulations. A plot of the simulation where the γ values were subject to 5% errors can be seen in figure 6.13.

Table 6.13: RMSE from simulation with Extended Kalman Filter with time varying Kalman Gain

γ values	RMSE level 1	RMSE level 2	Comment
$\gamma_1 = 0.7$, $\gamma_2 = 0.7$	0.0010 cm	0.00075 cm	Correct γ_1 and γ_2
$\gamma_1 = 0.735$,$\gamma_2 = 0.735$	3.50 cm	2.91 cm	5% error in γ_1 and γ_2

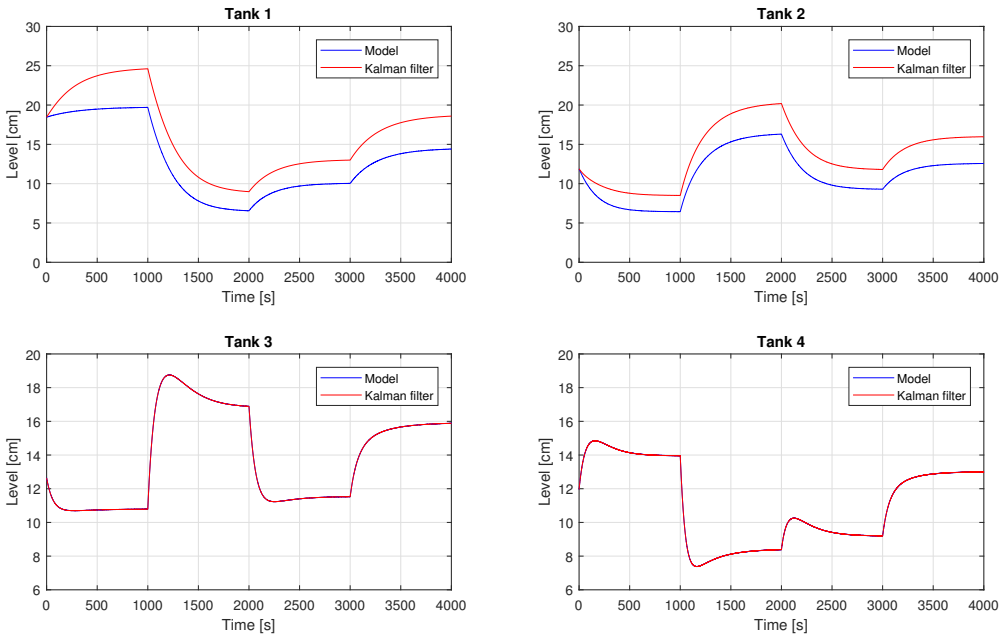


Figure 6.13: Simulation of Extended Kalman filter with time varying gain and 5% errors in γ_1 and γ_2

It can be seen that the performance of the EFK with time varying Kalman gain is very similar to the performance of the EKF with steady state gain when performing simulations. A possible reason for this is that the Kalman gain converges to a steady state gain matrix if the system matrix A_d is constant. If the system matrix which is linearized at each time step does not change much, the time varying Kalman gain matrix will resemble the steady state Kalman gain matrix, thus resulting in 2 filters which are similar.

Table 6.14 shows the root mean square error values of testing the EKF with time varying Kalman gain on experimental data while introducing 0%, 10% and 20% model mismatch. A plot from testing the EKF with time varying Kalman gain on experimental data when no model mismatch is introduced can be seen in figure 6.14.

Table 6.14: RMSE from experiment with Extended Kalman Filter with time varying Kalman gain

Parameters	RMSE level 1	RMSE level 2	Mismatch
$c_1 = 7.584e-5\ cm^{2.5}$ $c_2 = 8.977e-5\ cm^{2.5}$	3.91 cm	1.01 cm	0%
$c_1 = 6.826e-5\ cm^{2.5}$ $c_2 = 8.079e-5\ cm^{2.5}$	7.17 cm	4.14 cm	10%
$c_1 = 6.068e-5\ cm^{2.5}$ $c_2 = 7.182e-5\ cm^{2.5}$	11.16 cm	8.29 cm	20%

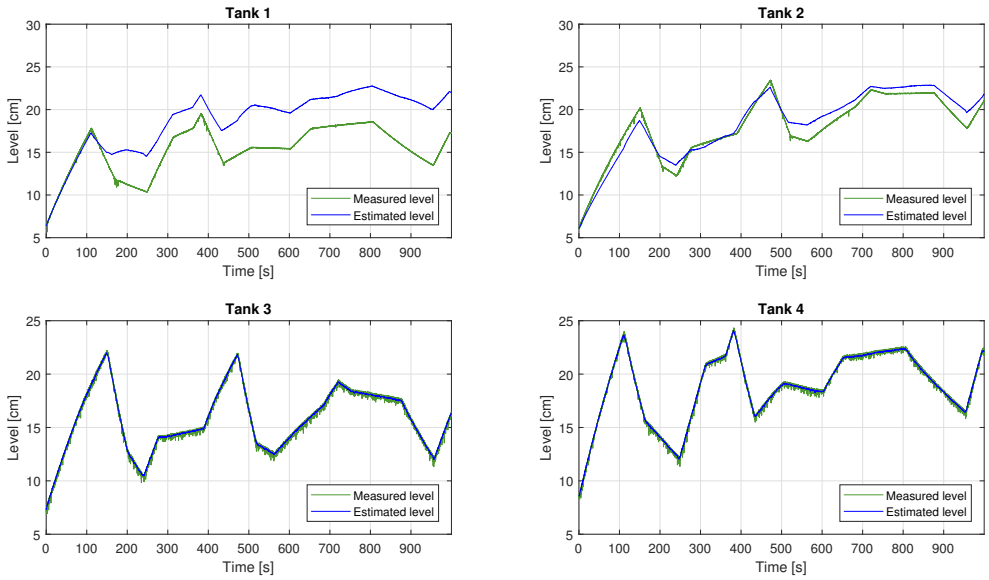


Figure 6.14: EKF with time varying Kalman gain on physical process with no model mismatch

The results are similar to the steady state Kalman gain case. The RMSE due to model mismatch increases in a similar manner, however the RMSE is consistently lower in the case where time varying Kalman gain is used. The reason for this is that when the Kalman gain is frequently updated, it obtains a more correct value according to the model that is predicting the states. This causes the state estimates to be corrected based on a linear model approximation which should be quite accurate at the current levels, instead of correcting the states based on a linear model which is only accurate at a fixes level. When moving away from the points where the model was linearized the steady state Kalman gain becomes incorrect. For plots from experiments with model mismatch see appendix J.

6.3 Model Based Control Results

This section covers the results of the model based controllers developed in chapter 5. All the controllers were first tested with the simulation model to verify that they were correctly implemented before they were tested on the physical process. The controllers were run in combination with the linear Kalman filter both in simulations and on the physical process. The initial states of the MPC are the corrected state estimates of the Kalman filter. This gives the Kalman filter a second purpose in addition to state estimation, where the Kalman filter is used for filtration of the noisy output signals.

An experiment was designed where the references vary within the process window. The same experiment was performed for all controllers to make it possible to compare the results. The experiments were all done by starting with empty tanks and maximum pump inputs for both pumps. When both level 3 and level 4 had reached 10cm the controller was activated. By starting with empty tanks, all experiments are as similar as possible. Experiments with the model based controllers were carried out on the physical process with 0%, 10% and 20% model mismatch. Model mismatch was introduced in the parameters c_1 and c_2 since they in combination affect all 4 tanks. The mismatch was introduced in both the MPC model and the Kalman filter.

The fact that the quadruple tank process at USN is a slow process was exploited by increasing the time step of the controllers, meaning that the sample time of the LabVIEW program was increased for allowing a higher computational time of the controllers. This makes it possible to have a longer horizon for the MPCs since an increased horizon results in more variables to optimize, and thus a longer computational time. The model parameters used for the experiments can be seen in table 6.15.

Table 6.15: Parameters used for model based control experiments

Parameter	**Value**	**Unit**
c_1	7.5844e-05	$m^{2.5}/s$
c_2	8.9773e-05	$m^{2.5}/s$
c_3	3.1148e-04	$m^{2.5}/s$
c_4	2.9812e-04	$m^{2.5}/s$
K_{p1}	1.8471e-05	m^3/s
K_{p2}	1.7805e-05	m^3/s
A	0.0289	m^2
γ_1	0.7	
γ_2	0.7	

6.3.1 Results for Linear MPC with First Principles Linear Model

The linear first principles model used in the linear MPC was linearized around the operating points shown in table 6.16.

Table 6.16: Operating points for linear MPC

	Value	Unit	Comment
h_1^o	19.0	cm	Operating point level 1
h_2^o	12.8	cm	Operating point level 2
h_3^o	11.9	cm	Operating point level 3
h_4^o	13.5	cm	Operating point level 4
u_1^o	6	V	Operating point input 1
u_2^o	6	V	Operating point input 2

The experiments with the linear MPC were carried out with the same horizon, prediction time step and program time step for both the simulation and the physical process. The horizon was chosen so that the MPC could predict a sufficient time ahead to account for a reasonable step change within the process window. The simulation model was simulated with different time steps Δt to find a time step for the prediction horizon which still resulted in good model performance. The experiments for the linear MPC with the first principles model was carried out with the parameters shown in table 6.17.

Table 6.17: MPC parameters for linear MPC with first principles linear model

Parameter	Value	Unit
Experiment length	440	s
Program time step	0.4	s
MPC prediction time step	5	s
MPC prediction horizon	60	s

A plot of simulating the linear MPC with integral action in combination with the linear Kalman filter can be seen in figure 6.15 and the resulting RMSE between the reference and outputs can be seen in table 6.18.

Table 6.18: RMSE from simulation with linear MPC with first principles linear model

RMSE level 3	RMSE level 4
0.898 cm	0.922 cm

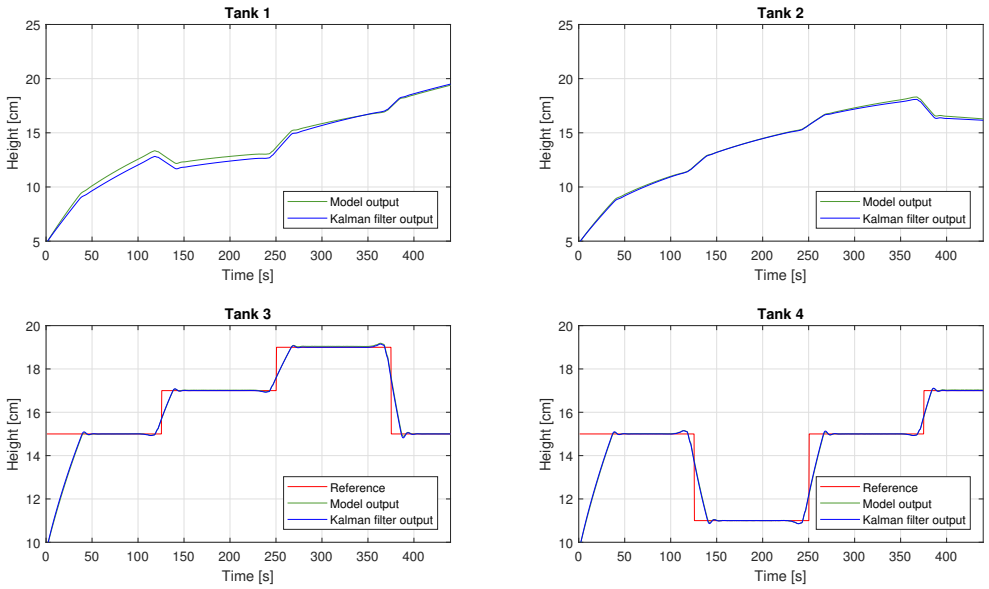


Figure 6.15: Controlling simulator with linear MPC in minimum phase

It can be seen that the MPC controls the simulation model without problems. The response clearly indicates integral action since there is no steady state offset and there is a tendency of oscillation in the levels before they settle. It can also be seen that the Kalman filter estimates both the output and the upper levels well. The deviation seen in the upper levels is due to model mismatch as discussed in 6.2.

Table 6.19 shows the resulting RMSE values between the references and outputs for experiments done on the physical process where 0%, 10% and 20% mode mismatch was introduced. A plot from the experiment with 0% additional model mismatch can be seen in figure 6.16.

Table 6.19: RMSE from experiment with linear MPC with integral action on physical process

Parameters	**RMSE level 3**	**RMSE level 4**	**Mismatch**
$c_1 = 7.584e-5\ cm^{2.5}$ $c_2 = 8.977e-5\ cm^{2.5}$	0.967 cm	0.70 cm	0%
$c_1 = 8.605e-5\ cm^{2.5}$ $c_2 = 1.045e-4\ cm^{2.5}$	0.764 cm	0.756 cm	10%
$c_1 = 9.387e-5\ cm^{2.5}$ $c_2 = 1.140e-4\ cm^{2.5}$	0.897 cm	0.706 cm	20%

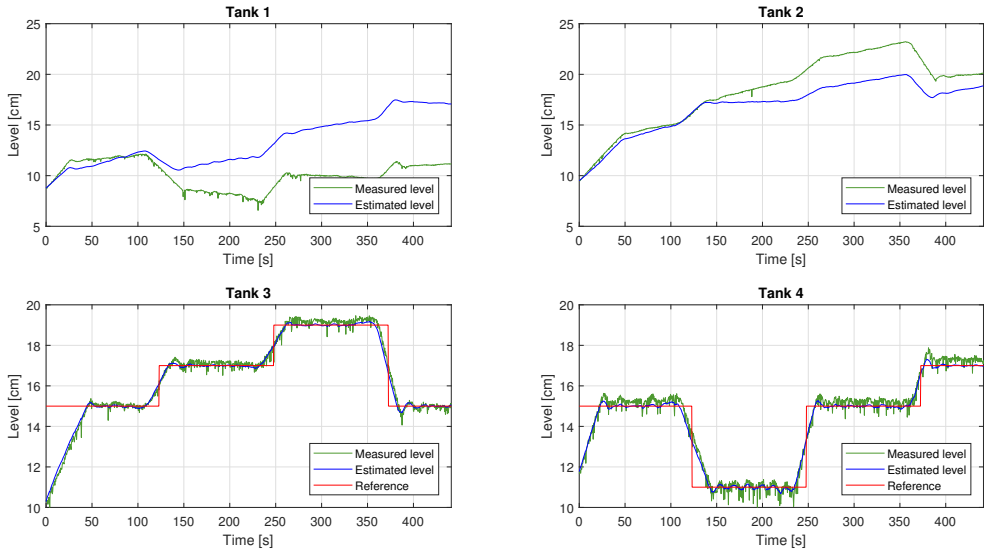


Figure 6.16: Controlling real process with linear MPC in minimum phase

From table 6.19 it can be seen that model mismatch randomly affects the MPC performance. When introducing 10% model mismatch the RMSE value in level 3 decreases slightly, and when introducing 20% model mismatch the RMSE in level 3 is less than when there is 0% model mismatch, although greater than when there is 10% mismatch. The variations in RMSE are likely caused by noise, thus being random. This shows that model mismatch, within a reasonable amount, does not affect the performance of the MPC. The MPC is therefor robust against modeling errors. This is expected since the MPC has feedback and therefor starts the prediction with correct levels at each time step.

From figure 6.16 it can be seen that the output estimates closely follow the measured values, although with less noise. This shows that the Kalman filter works well as a filter in addition to estimating the states. The estimated outputs looks to have a tendency of following below the measured levels. This is due to the major noise spikes pointing downwards and thus lowering the mean value of the signal. The spikes are caused by troubled water due to the water flowing into the lower tanks from the upper tanks, hence the measured values of the upper tanks are less noisy. The estimated levels of the upper levels deviate significantly from the measured levels. This again shows that the MPC is robust against errors since the initial states of the MPC are the estimated states.

6.3.2 Results for Linear MPC with Subspace Identified Model

Experiments with linear MPC with the subspace identified model were performed with two models, 1 with a time step of 0.4 seconds and 1 with a time step of 1 second. The MPC prediction time step is limited to the time step of the realized model. Therefor the horizon was adjusted so that the solver could solve the optimization problem within the time step of the model. For a prediction time step of 0.4s the maximum possible horizon was 20 seconds and for a prediction time step of 1 second the maximum possible horizon was 60 seconds. The solver generally solves the problem in less than 1 second, however the maximum observed computational time was close to 1 second.

The MPCs were first tested on the simulation model before they were tested on the physical process. They were tested in combination with the Kalman filter obtained from the model identification. The reason for this is that the Kalman filter provides state estimates which are in the same coordinate system as the states of the MPC model. Since the estimated values of the upper tanks are in a different coordinate system than the measured levels, only the lower tanks are presented.

A plot from controlling the simulation model with the linear MPC using the model with a time step of $\Delta t = 0.4s$ can be seen in figure 6.17 and a plot from controlling the simulation model with the linear MPC using the model with a time step of $\Delta t = 1s$ can be seen in figure 6.18. The resulting RMSE values of the simulations can be seen in table 6.20.

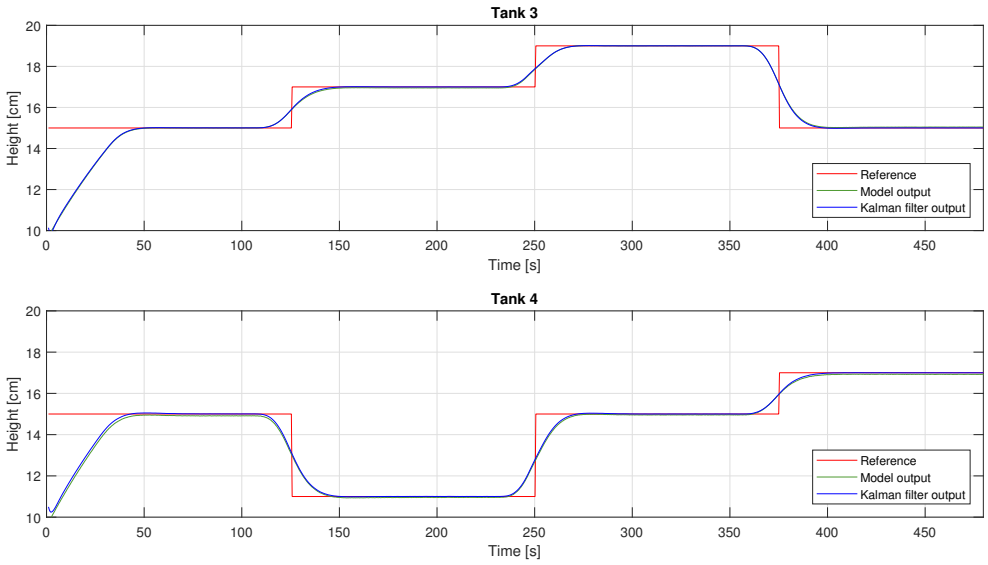


Figure 6.17: Results from controlling simulation model in minimum phase with linear MPC with subspace identified model, $\Delta t = 0.4s$

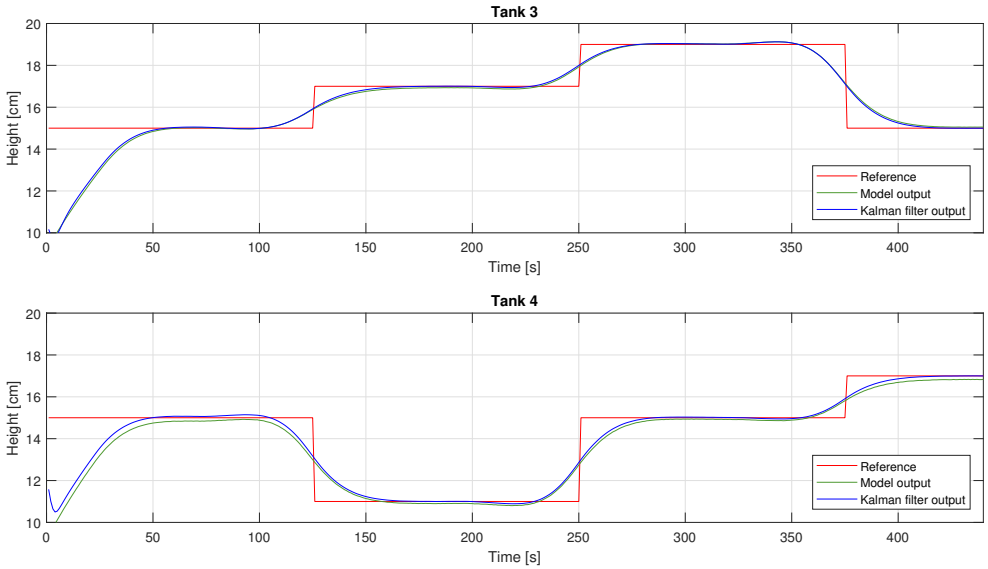


Figure 6.18: Results from controlling simulation model in minimum phase with linear MPC with subspace identified model, $\Delta t = 1s$

Table 6.20: RMSE from simulation with linear MPC with subspace identified models

Model time step	**RMSE level 3**	**RMSE level 4**
$\Delta t = 0.4s$	0.882 cm	0.887 cm
$\Delta t = 1s$	0.985 cm	0.993 cm

The RMSE values show that the model with $\Delta t = 1s$ performs worse than the model with $\Delta t = 0.4s$, similar to the observations made of the Kalman filters. It can be seen from the plots that the Kalman filter estimation of the outputs deviate more in the case where the model has a time step of $\Delta t = 1s$ than in the case where $\Delta t = 0.4s$. Since the MPC takes feedback from the estimated states and not the outputs, this explains the higher RMSE in the $\Delta t = 1s$ case.

Since it was not possible to adjust the model parameters to introduce model mismatch, experiments were performed with both models where discharge valve 1 and 2 were manually adjusted on the physical setup. A plot from controlling the process with no model mismatch and a model time step of 0.4 seconds can be seen in figure 6.19 and a plot from controlling the process with no model mismatch and a model time step of 1 second can be seen in figure 6.20. The RMSE values for all the physical experiments done with the linear MPC with subspace identified model can be seen in table 6.21.

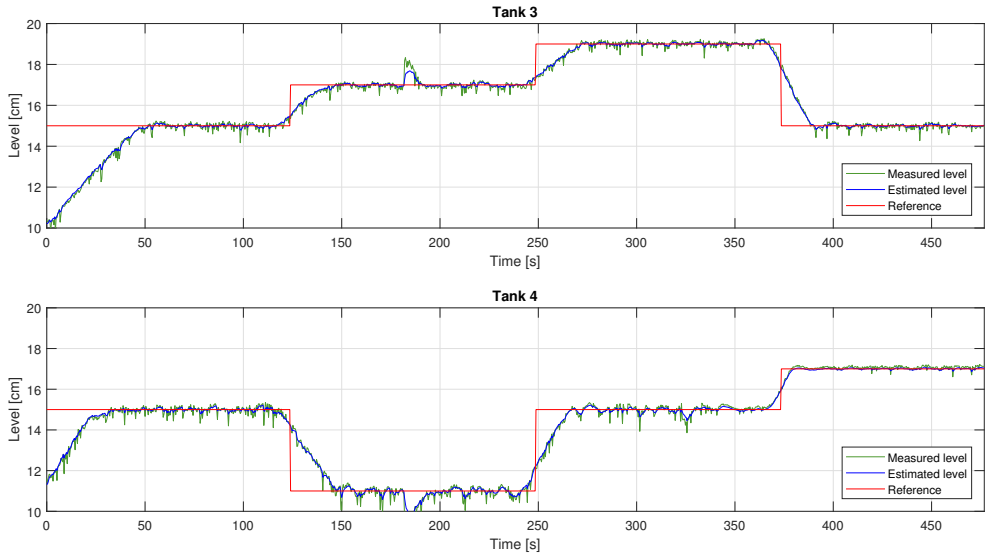


Figure 6.19: Results from controlling real process in minimum phase with linear MPC with subspace identified model $\Delta t = 0.4s$

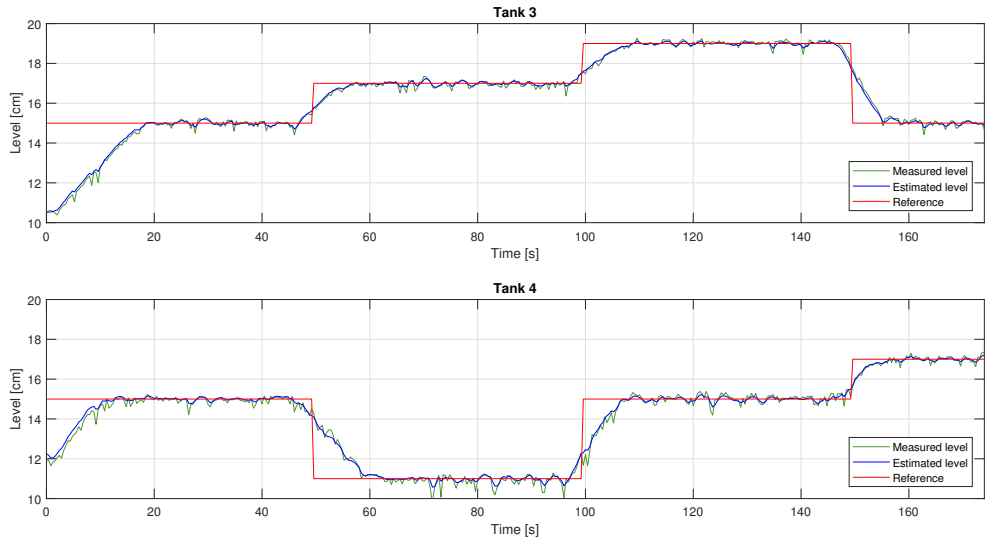


Figure 6.20: Results from controlling real process in minimum phase with linear MPC with subspace identified model where $\Delta t = 1s$

Table 6.21: RMSE from experiment with linear MPC with subspace identified model on physical process

Model time step	RMSE level 3	RMSE level 4	Comment
$\Delta t = 0.4s$	0.997 cm	0.767 cm	No model mismatch
$\Delta t = 0.4s$	0.978 cm	0.654 cm	Model mismatch
$\Delta t = 1s$	1.018 cm	0.813 cm	No model mismatch
$\Delta t = 1s$	1.148 cm	0.740 cm	Model mismatch

The RMSE of level 4 decreases when model mismatch is introduced for both the case when $\Delta t = 0.4s$ and for the case when $\Delta t = 1s$. This is an indication that the model mismatch cancels the initial model mismatch in tank 4. In level 3 the model mismatch increases the RMSE, however not much. This increase in RMSE is so small that it likely is random and may not be the same for a different experiment. This shows that the MPC is robust to model mismatch. By comparing the RMSE values of the $\Delta t = 0.4s$ and $\Delta t = 1s$ one can argue that an increased sample time does not affect the MPC performance significantly. The RMSE values are generally higher, however the difference is small. It is expected that the model with $\Delta t = 1s$ performs worse than the model with $\Delta t = 0.4s$, however the increased time step has also given the benefit of an increased horizon. It is likely that this is the reason that MPC with the less accurate model performs similar to the MPC with the more accurate model.

6.3.3 Results for Non-linear MPC

The non-linear MPC was simulated with both a minimum-phase and non-minimum phase configuration. The challenge of controlling the process in non-minimum phase is that the controller needs to capture the zero dynamics caused by the positive zero. The stability of the controller depends on the controller being able to see the whole inverse respons. Through trial and error it was found that in order to achieve a reasonable set point tracking on the simulator in non-minimum phase, a horizon of 400s was needed. The prediction time step was set to be 20 seconds to reduce computational time. Figure 6.21 shows the levels of the non-minimum phase experiment and the inputs of the experiment can be seen in figure 6.22.

The experiment was carried out for a long time. This was necessary in order to get the level to settle. It can be seen from the plot that the controller uses the levels of the upper tanks to control the level in the lower tanks. When there is a step in the reference of tank 3 it is the input to pump 1 which is increased, while the input to pump 2 is decreased. This is the opposite behaviour of how the controller reacts in a minimum phase configuration. It shows that the MPC manages to see the inverse response and uses the opposite input output pairing than in the minimum phase configuration. From the level plot it can be seen that the level in the tank which is not subject to a reference change is slightly excited in order to compensate for the reference change in the other tank. It is also possible to

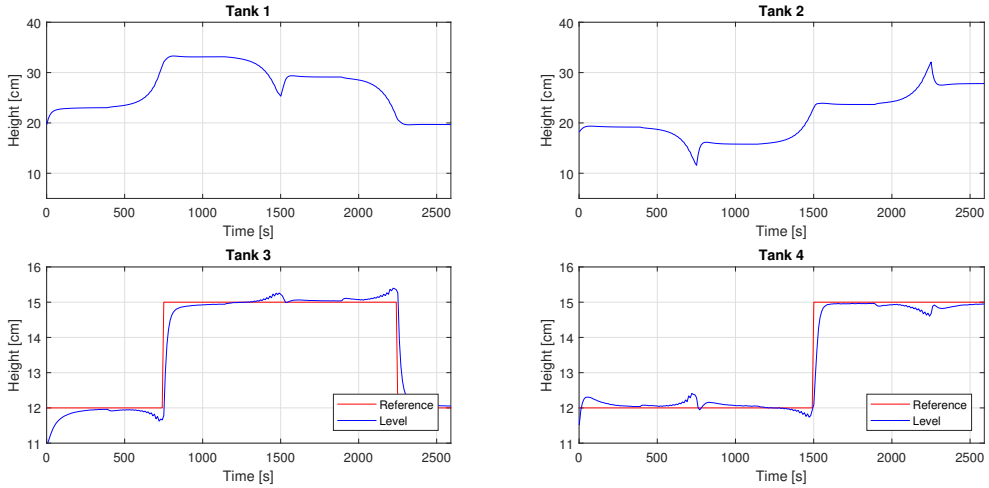


Figure 6.21: Results from controlling simulator in non-minimum phase with non-linear MPC

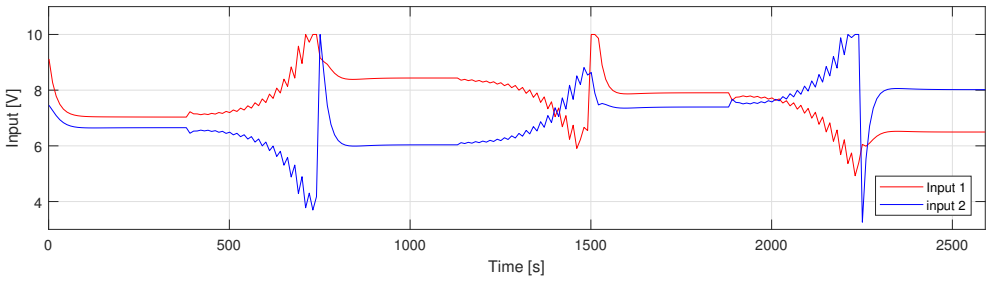


Figure 6.22: Inputs of experiment when controlling simulation model in non-minimum phase with non-linear MPC

see the inverse response that occurs before a reference change. This observation confirms that there is high interaction and shows that the MPC theoretically is able to handle non-minimum phase control. Controlling the physical process in non-minimum phase was not successful and is therefor not presented.

Experiments performed with the non-linear MPC in minimum phase were carried out similar to the linear MPC case, using the same horizon and MPC prediction time step. The RMSE values between the references and outputs when controlling the simulation model with the non-linear MPC is presented in table 6.22 and a level plot of the experiment can be seen in figure 6.23.

Table 6.22: RMSE from simulation with non-linear MPC

RMSE level 3	**RMSE level 4**
0.004 cm	0.039 cm

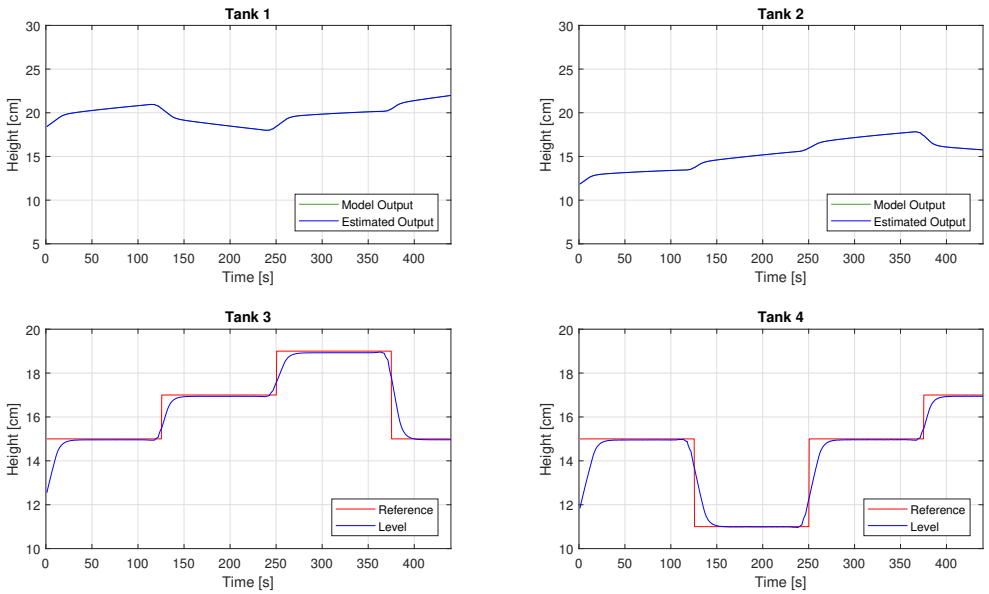


Figure 6.23: Controlling the simulation model with non-linear MPC in minimum phase

It can be seen from the plot that the MPC has no problem controlling the simulation model in minimum phase, and the RMSE values in table 6.22 are lower than for the linear MPC case. This is expected since the the model of the non-linear MPC and the simulation model are identical.

Table 6.23 presents the RMSE values of running the non-linear MPC on the physical process when 0%, 10% and 20% model mismatch is introduced. A plot from the experiment when no additional model mismatch was introduced is shown in figure 6.24.

Table 6.23: RMSE from experiment with non-linear MPC on physical process

Parameters	**RMSE level 3**	**RMSE level 4**	**Mismatch**
$c_1 = 7.584e-5\ cm^{2.5}$ $c_2 = 8.977e-5\ cm^{2.5}$	0.950 cm	0.878 cm	0%
$c_1 = 8.605e-5\ cm^{2.5}$ $c_2 = 1.045e-4\ cm^{2.5}$	0.961 cm	0.846 cm	10%
$c_1 = 9.387e-5\ cm^{2.5}$ $c_2 = 1.140e-4\ cm^{2.5}$	0.973 cm	0.864 cm	20%

It can be seen from the RMSE values that model mismatch does not decrease the performance of the MPC significantly. What can be seen from the plot of the levels is that there is a steady state offset which increases with the value of the reference. The offset looks similar to the offset occurring when using the linear MPC on the non-linear simulation model. The non-linear MPC does not have integral action and is therefor not

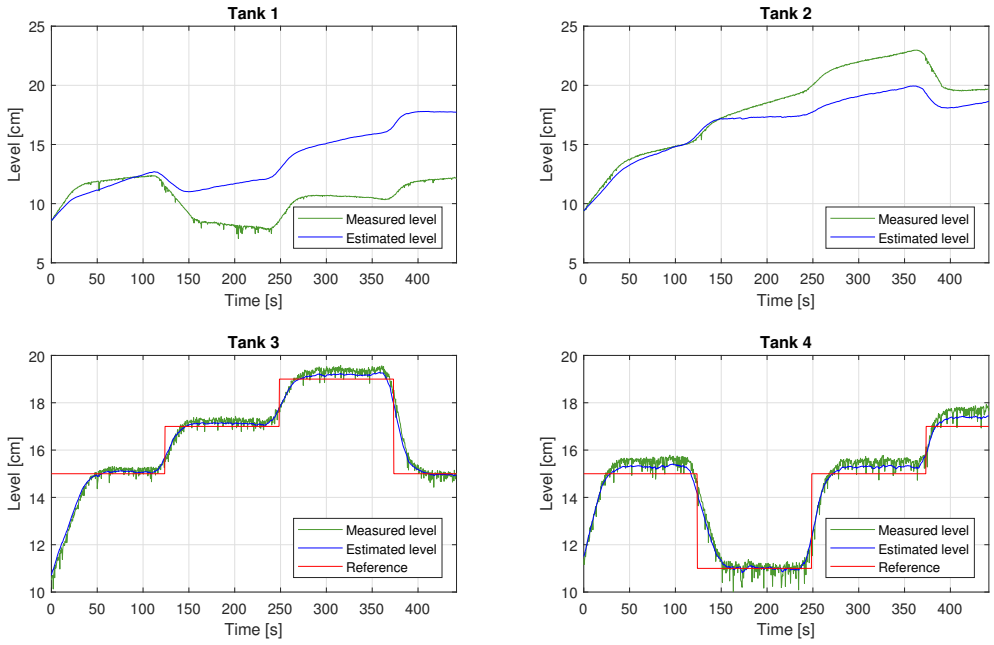


Figure 6.24: Controlling real process with non-linear MPC in minimum phase

compensating for the offset. Since the offset increases when the reference increases it is likely that the non-linearity of the physical process and the process model is slightly different, causing a model mismatch which increases with the level.

The estimated levels of the upper tanks resemble the estimated level of the upper tanks in the linear MPC case, where the Kalman filter fails to estimate the levels correctly and produces a significant estimation error. Even with a large estimation error the MPC obtains a satisfying control.

6.3.4 Results for Linear Quadratic Optimal Control

Results from controlling the simulation model and the physical process with the developed LQ controller are presented below. The experiments were carried out with a program time step of 0.2 seconds, ensuring that both the Kalman filter estimation and controller would be able to run within the program time step. Figure 6.25 shows the result of controlling the simulation model with the LQ controller and the resulting RMSE values between the references and outputs can be seen in table 6.24.

The plot shows that the outputs follow the references without any problem, however without predictive control. The controller does not compensate for a reference change

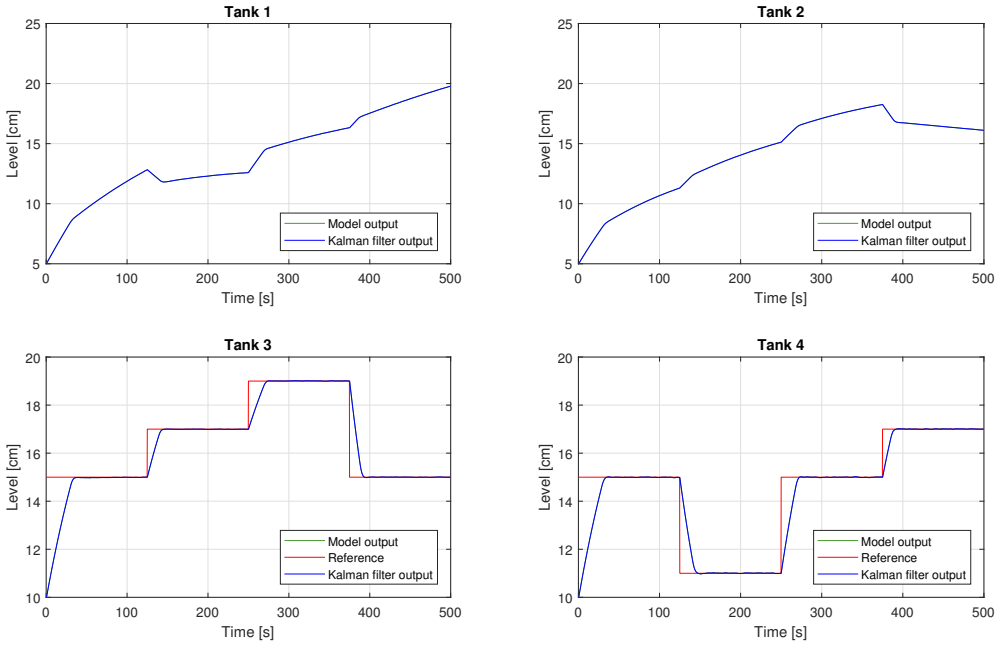


Figure 6.25: Controlling simulator with LQ controller in minimum phase

Table 6.24: RMSE from simulation with LQ controller

RMSE level 3	**RMSE level 4**
2.30 cm	3.45 cm

before it has happened. This explains why the resulting RMSE values are much greater than in the linear MPC case.

Experiments were performed where the LQ controller was used to control the physical process while introducing 0%, 10% and 20% model mismatch. The resulting levels for the experiment where no additional model mismatch was introduced are shown in figure 6.26 and the RMSE values of the 3 experiments can be seen in table 6.25.

Table 6.25: RMSE from experiment with LQ controller on physical process

Parameters	**RMSE level 3**	**RMSE level 4**	**Mismatch**
$c_1 = 7.584e-5\ cm^{2.5}$ $c_2 = 8.977e-5\ cm^{2.5}$	1.09 cm	1.12 cm	0%
$c_1 = 8.605e-5\ cm^{2.5}$ $c_2 = 1.045e-4\ cm^{2.5}$	1.065 cm	1.087 cm	10%
$c_1 = 9.387e-5\ cm^{2.5}$ $c_2 = 1.140e-4\ cm^{2.5}$	1.083 cm	1.024 cm	20%

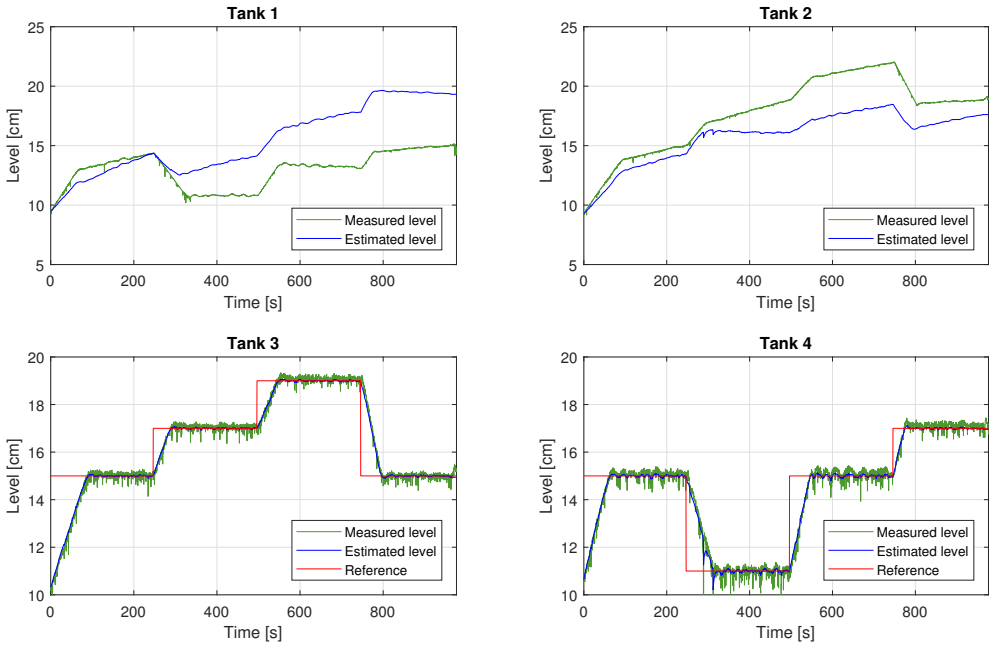


Figure 6.26: Controlling real process with LQ controller in minimum phase

It can be seen from table 6.25 that model mismatch does not have a significant impact on the RMSE. The RMSE in level 4 decreases with model mismatch as also observed in previous results. The controller clearly produces offset free control with good set point tracking, indicating that integral action is working. There is minimal overshoot when compensating for a reference change, however there is some oscillatory behaviour. This is due to the noise signal varying randomly so that the Kalman filter produces a slightly oscillating output estimate. The controller is constantly trying to compensate for the varying output, thus resulting in small oscillations.

7 Conclusion and Future Work

This chapter covers conclusions drawn from the results of the project and recommendations for future work. Conclusions about model development, Kalman filter development and development of model based controllers is considered.

7.1 Conclusion

7.1.1 Conclusion of Model Development

The models have thoroughly been developed and calibrated to establish confidence in the model so that the results from the Kalman filters and model based controllers can be considered to be reasonable. Validation of the models clearly show that there is model mismatch between the obtained models and measured data, however the model mismatch is considered to be reasonable and expected.

Validation has shown that the performance of the non-linear model is better than the linear model and that it is challenging to determine if the subspace identified model is better or worse than the first principles models. The performance of the linear model is not significantly worse than the former and is the preferred model for implementation in Kalman filters and MPCs. Both the first principles modeling and subspace identification approaches have advantages and weaknesses. The weakness of subspace identification being the lack of flexibility to change the model once it has bean identified. This can especially be inconvenient when using model based controllers and Kalman filters since the time step is restricted to the time step of the calibration data. The benefit of using a subspace identified model is that with little effort one can obtain a model which works well. The weakness of first principles modeling is that it is time consuming and may be difficult if the process is physically complex. The advantage is the flexibility one has since the parameters easily can be updated without having to obtain a calibration data set from the physical process. For this particular project the first principles models have been the preferred models due to flexibility, allowing frequent changes in the γ values.

7.1.2 Conclusion of Kalman Filters

Simulations of the Kalman filters has established confidence that the filters have been correctly implemented before applying them to the physical process. It has been shown from both simulations and experiments that the linear Kalman filter is the preferred filter. This is because it is the most convenient filter to implement and is most robust to model mismatch.

Experiments show that even when the model is well calibrated the estimates of the upper tanks deviate significantly. Depending on the application, the estimation error may be considered unacceptable. When using the Kalman filter to obtain state estimates for the model based controllers the estimation error looks to be insignificant due to the robustness of the controllers, however if the estimates should be used for other purposes which require an accurate estimate, the performance of the state estimation may be insufficient. A typical example of this may be if the Kalman filter is used in combination with an MPC where there has been put bounds on the states. If the state estimate is incorrect the actual value of the state may exceed the bounds while the estimate is still within the accepted range. Another example is if the Kalman filter should be used to keep a process running while performing maintenance. If a sensor is to be changed on a process and the Kalman filter can sufficiently estimate the value which the sensor measured, the process can keep running by using the Kalman filter estimate while the sensor is being replaces. If it is not possible to rely on the estimate, the Kalman filter cannot be used for this.

7.1.3 Conclusion of Model Based Controllers

A linear MPC using the first principles linear model, a linear MPC using the subspace identified model, a non-linear MPC using the non-linear first principles model and a linear quadratic controller using the linear first principles model has been developed an evaluated through simulations and experiments. The linear MPCs and the LQ controller was developed with integral action and the non-linear MPC was developed without integral action. The results have shown successful control of both the simulation model and the physical process and it has been shown that all controllers are robust to reasonable model mismatch because of using feedback.

The model predictive controllers perform better than the linear quadratic controller due to allowing predictive control, The LQ controller provides good control and has the advantage of being simpler to both develop and implement. The non-linear MPC has a higher computational time than the linear controllers. Since all controllers have shown robustness to modeling error, the non-linear MPC is not a preferred controller for this specific process, however the preferred choice of controller will be dependent on the context in which the controller should be used.

7.2 Future Work

7.2.1 Modeling of Pumps

From validation of the models it was seen that the idle speed of the pumps caused a flow to the lower tanks which affected the model performance. It could be interesting to try to model the idle pump speed and the hydro static pressure difference between the upper and lower tanks so that the model describes the flow into the tanks more accurately. One could also include the pump dynamics in the model, however as it was shown in the model development, the pump dynamics are a lot faster than the tank dynamics so including the pump dynamics will likely not make the model significantly more accurate.

7.2.2 Controlling the Process in Non-minimum Phase

It was shown through simulations that the MPC theoretically is capable of controlling the process in a non-minimum phase configuration. However it was not shown experimentally as no sufficient results were obtained. The challenge is to configure the process so that it is possible to control it within the process window. Controlling the process in non-minimum phase requires the control to be done over a long time, thus the MPC needs to have a long horizon and computational time increases.

7.2.3 State Estimation Techniques

One could explore other state estimation techniques like observers or the unscented Kalman filter to see if they achieve state estimates with reasonable accuracy.

7.2.4 Non-linear MPC with Integral Action

It may be of interest to evaluate the performance of a non-linear MPC when introducing integral action. Through this project is has been shown that the linear MPC is preferred due to a lower computational time, however data documenting the performance of a non-linear MPC with integral could be obtained.

Bibliography

[1] J. L. R. N. Karl Henrik Johansson, 'A multivariable laboratory process with an adjustable zero', 1998.

[2] S. Dormido and F. Esquembre, 'The quadruple-tank process: An interactive tool for control education', 2003.

[3] D. D. Ruscio, 'Model predictive control with integral action: A simple mpc algorithm', 2013.

[4] S. N. Mohd. Azam, 'Linear discrete-time state space realization of a modified quadruple tank system with state estimation using kalman filter', 2017.

[5] D. D. Gamage, *Experimental subspace identification and model predictive control of a four tanks system*, 2012.

[6] B. B. Kharel, *Comparing methods for system identification on the quadruple tank process*, 2014.

[7] *Whatis*, Accessed 14.05.2018. [Online]. Available: `https://whatis.techtarget.com/definition/state-machine`.

[8] M. R. Hansen and T. O. Andersen, *Hydraulic components and systems*, 2012.

[9] W. KLONOWSKI, 'Simplifying principles for chemical and enzyme reaction kinetics', 1983.

[10] R. Sharma, 'Lecture notes for the course iia 4117: Model predictive control', 2017.

[11] D. D. Ruscio, *Subspace system identification theory and applications*, 2014.

[12] University of Ottawa and North Carolina State University, *Introduction to data reconciliation*, Presentation, 2003.

[13] F. Haugen, *Advanced dynamics and control*, 2010.

[14] *Nptel*, Accessed 30.04.2018. [Online]. Available: `http://nptel.ac.in/courses/108103008/29`.

[15] D. D. Ruscio, *System theory state space analysis and control theory*, 2016.

[16] ——, *Optimal model based control: System analysis and design*, 2016.

[17] *Umass*, Accessed 01.05.2018. [Online]. Available: `http://www.ecs.umass.edu/che/che446/2014/multiloop_control.pdf`.

Appendix A

Task Description

University College
of Southeast Norway

Faculty of Technology, Natural Sciences and Maritime Sciences, Campus Porsgrunn

FMH606 Master's Thesis

Title: Experimental Evaluation of MIMO Observers and Model Based Controllers

USN supervisor: Carlos F. Pfeiffer

Co-supervisor: Roshan Sharma

Task background:

Mutlivariable control is one of the most important applications of advenced model based control. Most advanced techniques require the use of state space models to etimate internal state variables for control laws, or to predict the outputs. Popular methods to estimate state variables are Kalman filters, Luenberguer observers, high gain observers, etc.

There are many scientific publications on these topics, but most of them only show results with simulations, and very few present experimental data. However, an important step for the adoption of such methods by industria require experimental tests, at least at pilot level.

At USN we have built a multivariable, pilot size rig consisting of four interacting tanks. All the leveles and incoming flows are meassured, and the information can be feed on line to matlab and simulink applications, for use with MIMO observers and advaced, model based controllers.

Task description:

The tasks that should be completed in this master thesis are:

- Test and calibrate availabe nonlinear models of the quadruple tank system. Modify the model in case it is necessary.
- Design and run subspace identification test to get experimental state-space models.
- Design and test different observers to estimate levles and flows. Use both first principle derived models, and models obtained using subspace identification. Report the theoretic and the real beahvior of the estimated variables.
- Desing and test different model based controllers, inlcuding MPC (linear and non-linear), using first principle models and sub-space identification models.
- Test the performance of the observers and controllers when the characteristic of the process change, but the model is not updated (robustness to model mismatch).
- Write a detalied report with the conclusions.

Student category: Master in Industrial IT and Automation.

Practical arrangements: Practical work will be carried out in USN, using the system depicted in figure 1.

Address: Kjølnes ring 56, NO-3918 Porsgrunn, Norway. **Phone:** 35 57 50 00. **Fax:** 35 55 75 47.

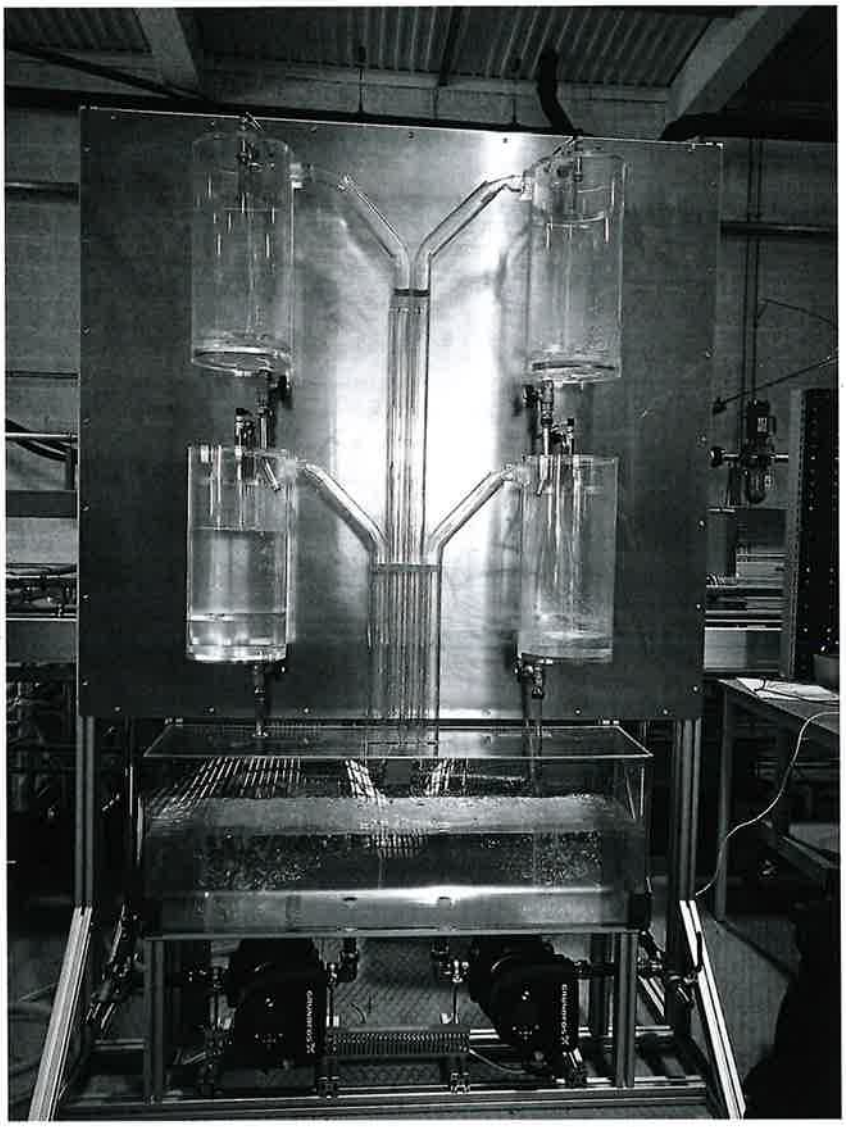


Figure 1: Four tanks experimental rig.

Signatures:

Student (date and signature): 31.01.2018 Sondre Fjereide

Supervisor (date and signature): 31.01.2018 Carlos L. Pfeffer

Co-supervisor (date and signature): 31.01.2018

Appendix B

MATLAB Script for Calibrating First Principles Model

```
% Parameter Calibration

close all; clear all; clc;
format short
data = importdata('processCalibration.txt'); % Logged data set

A  = 0.0289;                          % Area of tanks
ts = 0.1;                             % Sample time of logged data
N  = floor((length(data))*ts)/ts; % Length of data set

q1 = data(:,5);         % Flow from pump into tank 1
q2 = data(:,6);         % Flow from pump into tank 2
q3 = data(:,7);         % Flow from pump into tank 3
q4 = data(:,8);         % Flow from pump into tank 4
q5 = data(1:N-1,9);     % Flow from pump 1
q6 = data(1:N-1,10);    % Flow from pump 2

L1 = data(:,1);         % Level tank 1
L2 = data(:,2);         % Level tank 2
L3 = data(:,3);         % Level tank 3
L4 = data(:,4);         % Level tank 4

u1 = data(1:N-1,11);    % Signal to pump 1
u2 = data(1:N-1,12);    % Signal to pump 2

% Computing flow through each valve at each timestep

for i = 1:N-1

 qout1(i,1) = q1(i) - A*(L1(i+1)-L1(i))/ts;              % Qout tank 1
 qout2(i,1) = q2(i) - A*(L2(i+1)-L2(i))/ts;              % Qout tank 2
 qout3(i,1) = q3(i) + qout1(i) - A*(L3(i+1)-L3(i))/ts;   % Qout tank 3
 qout4(i,1) = q4(i) + qout2(i) - A*(L4(i+1)-L4(i))/ts;   % Qout tank 4
```

```
end

sL1 = sqrt(L1(1:N-1,:));          % Square root of level in tank 1
sL2 = sqrt(L2(1:N-1,:));          % Square root of level in tank 2
sL3 = sqrt(L3(1:N-1,:));          % Square root of level in tank 3
sL4 = sqrt(L4(1:N-1,:));          % Square root of level in tank 4

% Soling equation " K = Q/sqrt(h) " for each valve

c1 = mean(qout1)/mean(sL1);     % Valve coefficient for tank valve 1
c2 = mean(qout2)/mean(sL2);     % Valve coefficient for tank valve 2
c3 = mean(qout3)/mean(sL3);     % Valve coefficient for tank valve 3
c4 = mean(qout4)/mean(sL4);     % Valve coefficient for tank valve 4

% Displaying valve coefficients in table

Coefficients = [c1 c2 c3 c4 ]';
rownames = {'c1','c2','c3','c4'};
ValveCoefficients = table(Coefficients,'rowNames',rownames)

% ---------- Calibrating pump gains from logged data ----------

% Sorting data for calibrating pump gains

% Initializing indexes

pq5 = 1; pq6 = 1;

for i = 1:N-1

    if u1(i) >= 2

        q5logged(pq5) = q5(i);
        u1logged(pq5) = u1(i);

        pq5 = pq5 + 1;
    end
    if u2(i) >= 2

        q6logged(pq6) = q6(i);
        u2logged(pq6) = u2(i);

        pq6 = pq6 + 1;
    end
end

Qp1 = mean(q5logged); % Mean flowrate of pump 1
Up1 = mean(u1logged); % Mean input signal to pump 1

Qp2 = mean(q6logged); % Mean flowrate of pump 2
Up2 = mean(u2logged); % Mean input signal to pump 2
```

```
Kp1 = Qp1/Up1;          % Slope of flowrate for pump 1 (Upper tank)
Kp2 = Qp2/Up2;          % Slope of flowrate for pump 2 (Upper tank)

% Displaying pump gains in table

Gain = [Kp1 Kp2]';
rownames = {'Kp1','Kp2'}';
PumpGains = table(Gain,'rowNames',rownames)
```

Appendix C

MATLAB Function for Extended Kalman Filter with Steady State Gain

```
function x = extKalmanFilterSsG(x,y,u,K,parameters,ts)

    % Parameters

    %Valve coefficients

    c1  = parameters(1);  % Valve coefficient for discharge valve 1
    c2  = parameters(2);  % Valve coefficient for discharge valve 2
    c3  = parameters(3);  % Valve coefficient for discharge valve 3
    c4  = parameters(4);  % Valve coefficient for discharge valve 4

    % Pump gains

    Kp1 = parameters(5);  % Pump gain for pump 1
    Kp2 = parameters(6);  % Pump gain for pump 2

    % Other parameters

    A   = parameters(7);   % Tank area
    g1  = parameters(8);   % Gamma value for three-way valve 1
    g2  = parameters(9);   % Gamma value for three-way valve 2


    %---------------- Kalman filter algorithm ------------------------

    y_est = [x(3);x(4)];   % Getting predicted output

    x = x + K*(y - y_est); % Correcting estimated states

    % Estimating states for next timestep

    x1 = x(1) + ts/A*(Kp1*u(1)*(1-g1) - c1*sqrt(x(1)));

    x2 = x(2) + ts/A*(Kp2*u(2)*(1-g2) - c2*sqrt(x(2)));
```

```
    x3 = x(3) + ts/A*(Kp2*u(2)*g2 + c1*sqrt(x(1)) - c3*sqrt(x(3)));
    x4 = x(4) + ts/A*(Kp1*u(1)*g1 + c2*sqrt(x(2)) - c4*sqrt(x(4)));
    % Creating state vector
    x = [x1;x2;x3;x4];
end
```

Appendix D

MATLAB Function for Extended Kalman Filter with Time Varying Kalman Gain

```
function [x,Pp] = myExtendedKalmanFilter(x,y,u,parameters,ts,Pp,Q,R)

    c1  = parameters(1);    % Discharge coefficient valve 1
    c2  = parameters(2);    % Discharge coefficient valve 2
    c3  = parameters(3);    % Discharge coefficient valve 3
    c4  = parameters(4);    % Discharge coefficient valve 4
    Kp1 = parameters(5);    % Pump gain pump 1
    Kp2 = parameters(6);    % Pump gain pump 2
    A   = parameters(7);    % Tank area
    g1  = parameters(8);    % Split factor 1
    g2  = parameters(9);    % Splot factor 2

  % Linearizing system matrix

    Ac = [-c1/(2*A*sqrt(x(1)))  0  0  0;
           0 -c2/(2*A*sqrt(x(2)))  0  0;
           c1/(2*A*sqrt(x(1)))    0 -c3/(2*A*sqrt(x(3)))  0;
           0  c2/(2*A*sqrt(x(2)))  0 -c4/(2*A*sqrt(x(4)))];

  I = eye(4);

  Ad = I + ts*Ac;      % Discretizing matrix A

  Cd = [0  0  1  0;
        0  0  0  1];

  % Updating Kalman Gain

  G = eye(4);

  % Computing current Kalman gain
  K  = Pp*Cd'*(Cd*Pp*Cd' + R)^-1;

  % Computing current auto-covariance of corrected state
```

```
    Pc = (I - K*Cd)*Pp;

    % Computing next time step auto-covariance of predicted state
    Pp = Ad*Pc*Ad' + G*Q*G';

    % Updating state estimate

    y_est = Cd*x;

    x = x + K*(y - y_est);

    h1 = x(1) + ts/A*(Kp1*u(1)*(1-g1) - c1*sqrt(x(1)));    % Tank 1

    h2 = x(2) + ts/A*(Kp2*u(2)*(1-g2) - c2*sqrt(x(2)));    % Tank 2

    h3 = x(3) + ts/A*(Kp2*u(2)*g2 + c1*sqrt(x(1)) - c3*sqrt(x(3))); % Tank 3

    h4 = x(4) + ts/A*(Kp1*u(1)*g1 + c2*sqrt(x(2)) - c4*sqrt(x(4))); % Tank 4


    x = [h1;h2;h3;h4];

end
```

Appendix E

MATLAB Script for Formulation of QP Problem

```
function [H,Ae,c] = buildIntegralMPCdu(Q,P,A,B,C,N)

% Defining number of states, inputs and outputs
    nx = 6; ny = 2; nu  = 2;
    nz  = N*(nx + nu + 2*ny); % Total number of unknowns

% Quadratic coefficients

    H11 = kron(eye(N),P);     % Coefficients for du'P du
    H22 = zeros(N*nx,N*nx);   % Coefficients for x (non in objective)
    H33 = kron(eye(N),Q);     % Coefficients for e' E e
    H44 = zeros(N*ny,N*ny);   % Coefficients for y (non in objective)

    H   = blkdiag(H11,H22,H33,H44);

% Linear coefficients

    c   = zeros(nz,1);

% State equation dxk+1 = Adx + Bdu

    Ae1du = kron(-eye(N),B);
    Ae1x  = eye(N*nx)-kron(diag(ones(N-abs(-1),1),-1),A);
    Ae1e  = zeros(N*nx,N*ny);
    Ae1y  = zeros(N*nx,N*ny);

% e equation ek = rk - yk

    Ae2du = zeros(N*ny, N*nu);
    Ae2x  = zeros(N*ny, N*nx);
    Ae2e  = eye(N*ny);
    Ae2y  = eye(N*ny);

% y equation yk = Cdxk
```

```
    Ae3du = zeros(N*ny,N*nu);
    Ae3x = -kron(eye(N),C);
    Ae3e = zeros(N*ny,N*ny);
    Ae3y = eye(N*ny);


Ae = [Ae1du Ae1x Ae1e Ae1y;
      Ae2du Ae2x Ae2e Ae2y;
      Ae3du Ae3x Ae3e Ae3y];

end
```

Appendix F

MATLAB Function for Linear MPC with Integral Action

```
function du = integralMPCdu(H ,Ae, c ,Ubl,Ubu, A, x0, R, N)

% Defining number of states, inputs and outputs
    nx = 6; ny = 2; nu  = 2;
    nz  = N*(nx + nu + 2*ny); % Total number of unknowns

% Creating bounds vectors

    lowerLims = [Ubl(1);          % Lower bound of du1
                 Ubl(2);          % Lower bound of du2
          -inf*ones(nz/N-nu,1)]; % Lower bound of remaining unknowns

    ZL = kron(ones(N,1),lowerLims);  % Expanding to the whole horizon

    upperLims = [Ubu(1);          % Upper bound of du1
                 Ubu(2);          % Upper bound of du2
           Inf*ones(nz/N-nu,1)]; % Upper bound of remaining unknowns

    ZU = kron(ones(N,1),upperLims);  % Expanding to the whole horizon

% Creating constants vector

    be1  = [A*x0;zeros((N-1)*nx,1)]; % Constants in x = Ax + Bu
    be2  = reshape(R,N*ny,1);        % Constants in e = r - y
    be3  = zeros(N*ny,1);            % Constants in y = Cx

    be   = [be1;be2;be3];            % Constant vector

    ops = optimset('Display','off'); % Setting options for quadprog

% Solving problem

    [u_opt]= quadprog(H,c,[],[],Ae,be,ZL,ZU,[],ops);
```

```
% Extracting the first optimal delta u's

    du1 = u_opt(1);
    du2 = u_opt(2);

% Sorting into vector
    du = [du1;du2];
end
```

Appendix G

Matlab Function for Computing Objective of Non-linear MPC

```
function J = computeObjective(u,states,ts,ref,N,parameters,Q,P,u0)

    % Extracting model paramteres

    c1  = parameters(1);  % Valve coefficient tank 1
    c2  = parameters(2);  % Valve coefficient tank 2
    c3  = parameters(3);  % Valve coefficient tank 3
    c4  = parameters(4);  % Valve coefficient tank 4
    Kp1 = parameters(5);  % Pump gain (pump 1 to tank 1)
    Kp2 = parameters(6);  % Pump gain (pump 2 to tank 2)
    A   = parameters(7);  % Tank area
    g1  = parameters(8);  % Gamma value for split valve 1
    g2  = parameters(9);  % Gamma value for split valve 2

    % Extracting initial states

    h1 = states(1);       % Inital level 1
    h2 = states(2);       % Inital level 2
    h3 = states(3);       % Inital level 3
    h4 = states(4);       % Inital level 4

    % Initializing variables and computing constants

    J = 0;            % Initializing objective
    uk_last = u0;     % Getting previous optimal control input
    a = ts*1/A;       % Initializing constant delta t / area

    %/////////////////////////////////////////////////////////////////
    %     Simulating the process for the whole prediction horizon
    %/////////////////////////////////////////////////////////////////

for i = 1:N-1

    uk1 = u(i,1);           % Extracting input 1 for current time step
```

```
    uk2 = u(i,2);              % Extracting input 2 for current time step

    %------------------ Simulating the process ------------------------------

    h1 = h1 + a*(Kp1*uk1*(1-g1) - c1*sqrt(h1) );                % Tank 1

    h2 = h2 + a*(Kp2*uk2*(1-g2) - c2*sqrt(h2));                 % Tank 2

    h3 = h3 + a*(Kp2*uk2*g2     + c1*sqrt(h1) - c3*sqrt(h3)); % Tank 3

    h4 = h4 + a*(Kp1*uk1*g1     + c2*sqrt(h2) - c4*sqrt(h4)); % Tank 4

    %--------------------- Computing objective ------------------------------

    uk      = [uk1;uk2];     % Input vector for current time step

    e = [(ref(i,1) - h3);    % Errors of current time step
         (ref(i,2) - h4)];

    % Updating objective

    J = J + e'*Q*e + ((uk-uk_last)'/ts*P*(uk-uk_last)/ts);

    uk_last = uk;  % Saving current input for next iteration

end
```

Appendix H

Result Plots from Experiment with Linear Kalman Filter

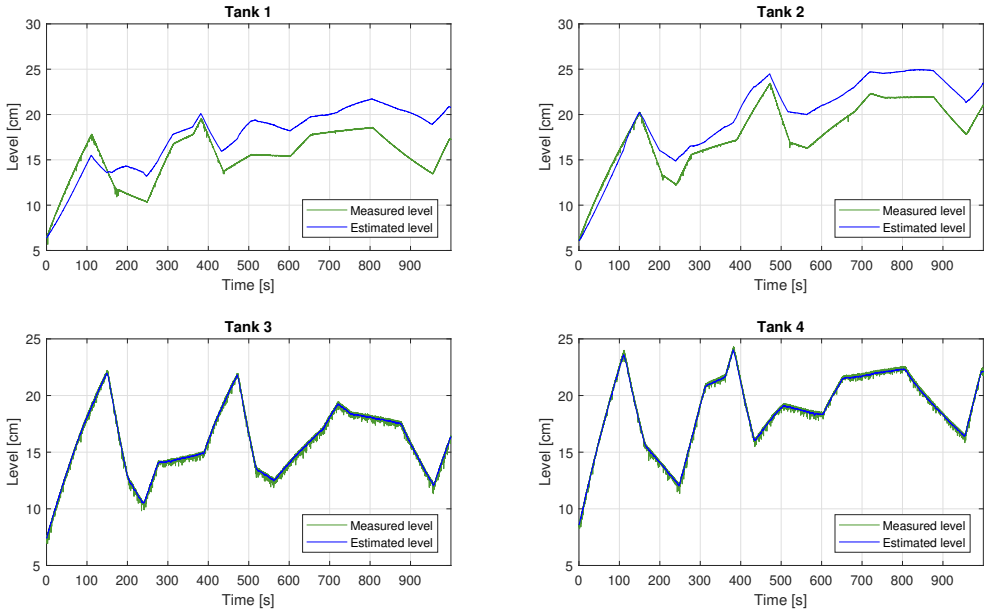


Figure H.1: Linear Kalman filter on physical process with 10% model mismatch

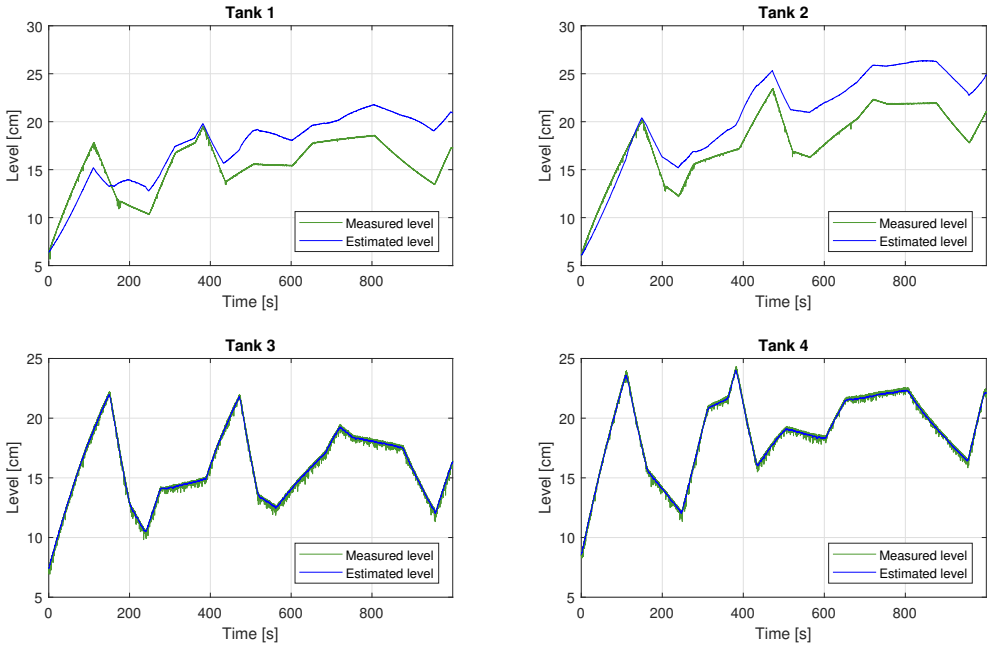


Figure H.2: Linear Kalman filter on physical process with 20% model mismatch

Appendix I

Result Plots from Experiment with Extended Kalman Filter with Steady State Kalman Gain

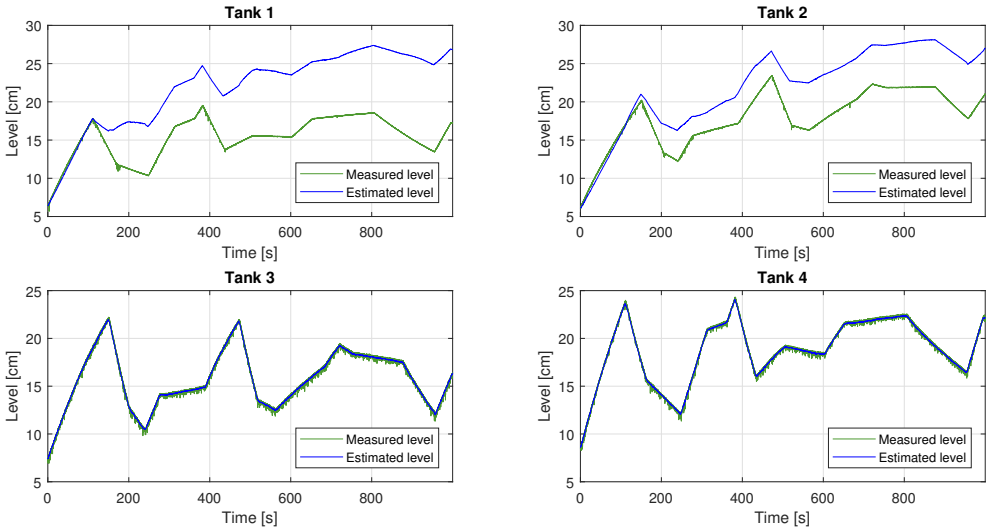


Figure I.1: Extended Kalman filter with steady state Kalman gain on physical process with 10% model mismatch

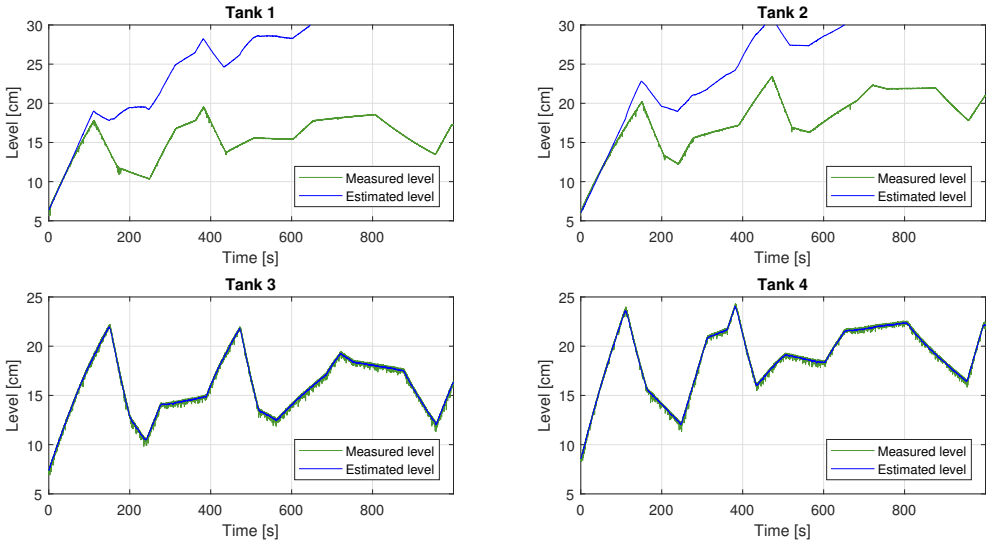


Figure I.2: Extended Kalman filter with steady state Kalman gain on physical process with 20% model mismatch

Appendix J

Result Plots from Experiment with Extended Kalman Filter with Time Varying Kalman Gain

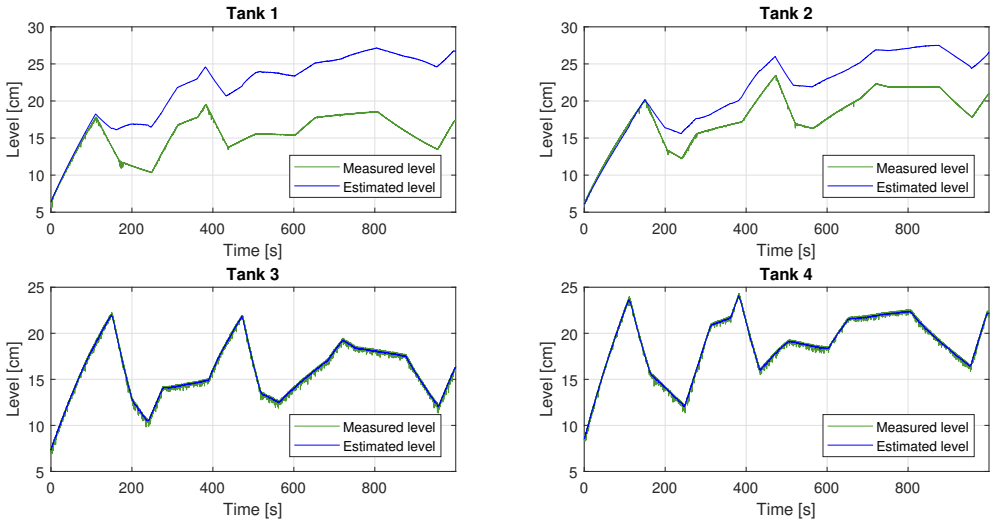


Figure J.1: Extended Kalman filter with time varying Kalman gain on physical process with 10% model mismatch

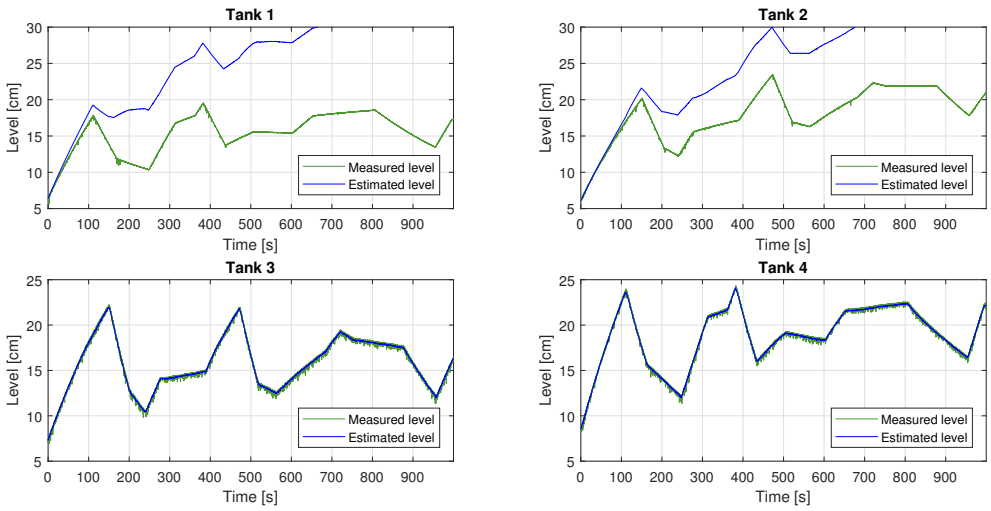


Figure J.2: Extended Kalman filter with time varying Kalman gain on physical process with 20% model mismatch