Practical comparison of MPC Toolboxes

Emma Nilsson

LUND
UNIVERSITY

Department of Automatic Control

MSc Thesis
TFRT-6124
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Printed in Sweden by Tryckeriet i E-huset
Lund 2021

Abstract

Model predictive control can be used to control a range of processes, from self-driving cars to chemical plants. The education of the engineering students that in the future will design these controllers is an important matter. In this thesis, three Matlab toolboxes have been evaluated and compared from a student's perspective.

The toolboxes are Mathwork's own toolbox called Model Predictive Control Toolbox, Multi-Parametric Toolbox 3, and MATMPC. With a practical approach, three types of controllers have been created in each toolbox, a basic MPC controller, an MPC controller with integral action, and an MPC controller with gain scheduling. The documentation has been explored and the notation has been compared with the theory that is taught to the students. Different ways to implement integral action and gain scheduling have been evaluated and the controllers have been run with a real process to emulate a laboratory session and the results from the different toolboxes have been compared.

The notation in Multi-Parametric Toolbox 3 did best correspond to the students' knowledge about MPC and had in addition the best performance of the three toolboxes.

Acknowledgements

First of all, I would like to thank my supervisors Anton Cervin and Olle Kjellqvist for the guidance and support. A special thanks to Olle for always being available and finding the time to help.

I would also like to thank the department for inviting me to "fika", for all the inspiring Friday seminars and for making me feel at home.

Finally, I would like to thank my family, especially my parents for the support during my studies and most of all during this last year.

Contents

1

Introduction

1.1 Background and Motivation

In the 1970s, the engineers at Shell Oil developed a new control strategy that would later develop into Model Predictive Control (MPC) [Holkar and Waghmare, 2010]. Due to heavy computations in the algorithms, MPC was only suitable for relatively slow processes. Since then, the progress in algorithms and the development of computers have sped up computational times [Gros et al., 2020] and nowadays MPC is suitable also for fast systems [Fredlund and Sulejmanovic, 2017]. Some of the domains in which MPC is used are aerospace, automotive, power systems, pulp and paper [Johansson, 2015].

While the main concept of MPC is not hard to understand, there are many details to work with. For example, how the model should be formulated or how do the weights affect the controller. A good way to learn these things is through laboratory sessions. During the fall of 2020, the department of automatic control restructured some of its courses. In one of these courses a new laboratory session was planed, where the purpose is to strengthen the students' understanding of MPC.

1.2 Purpose and Aim

This thesis will do a practical comparison of three toolboxes and focus on user-friendliness and pedagogy. The three toolboxes that were examined are Mathwork's own toolbox called Model Predictive Control Toolbox, a toolbox called Multi-Parametric Toolbox 3, and one called MATMPC. In each toolbox, a basic MPC controller, an MPC controller with integral action, and an MPC controller with gain scheduling will lay the foundation for a discussion about the pros and cons of these toolboxes. The purpose of this thesis is to give a recommendation on what toolbox is suitable to use for a laboratory session for students learning about MPC.

1.3 Method

The toolboxes chosen for this thesis were to be toolboxes for Matlab, work for linear MPC, and run in the Simulink environment. Matlab was used because it is a familiar programming environment for the students at the Faculty of Engineering at Lund University, and specifically, Mathworks toolbox, Model Predictive Control Toolbox, and the Multi-Parametric Toolbox 3 were chosen because they are well-known toolboxes. MATMPC was chosen because it was interesting to look at a new toolbox. To investigate these toolboxes, the documentation has been critically reviewed and the notations have been studied. The same setup has been implemented in the different toolboxes and to emulate a laboratory session, the controllers have been run with a process called the quadruple tank. The result has been analysed and thoroughly discussed.

1.4 Limitations

This thesis only includes Matlab-based toolboxes with a focus on linear MPC. Only a basic setup, integral action and gain scheduling have been explored. Integral action has been limited to investigate two types of models besides the toolboxes' own functions.

1.5 Outline

In Chapter 2, the theories used in the thesis are described. Chapter 3 presents the quadruple tank, which is the process that was used to run experiments on. The toolboxes are introduced in Chapter 4, where it is also explained how they were used to create the controllers and implement integral action and gain scheduling. The results are then presented in Chapter 5 that is followed by a discussion and conclusion in Chapters 6 and 7 respectively.

2

Theoretical Background

This chapter will present the theories behind the control principles used to regulate the quadruple tank. As this thesis compares MPC toolboxes, the theory behind model predictive control will be explained, and to enhance the performance of the controller the theories of Kalman filtering, gain scheduling and different forms of integral action will be explored.

2.1 Model Predictive control

Prediction

In model predictive control, the control object's dynamics are described by a mathematical model. In the simplest case, a linear model is used:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k), \end{aligned} \tag{2.1}$$

which is a discrete-time model where u is the input signal and y is the output. The future states x can then be predicted by

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ x(k+2) &= Ax(k+1) + Bu(k+1) \\ &= A^2x(k) + ABu(k) + Bu(k+1) \\ x(k+3) &= Ax(k+2) + Bu(k+2) \\ &= A^3x(k) + A^2Bu(k) + ABu(k+1) + Bu(k+2) \\ &\vdots \\ x(k+H_p) &= A^{H_p}x(k) + A^{H_p-1}Bu(k) + \ldots + \sum_{i=0}^{H_p-H_u} A^iBu(k+H_u-1) \end{aligned}$$

where the control signal u is assumed to be constant for the last $H_p - H_u$ terms. These predictions can be represented in matrices.

$$\begin{bmatrix} x(k+1) \\ x(k+2) \\ \vdots \\ x(k+H_u) \\ x(k+H_u+1) \\ \vdots \\ x(k+H_p) \end{bmatrix} = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^{H_u} \\ A^{H_u+1} \\ \vdots \\ A^{H_p} \end{bmatrix} x(k) + \begin{bmatrix} B & 0 & \dots & 0 \\ AB & B & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{H_u-1}B & A^{H_u-2}B & \dots & B \\ A^{H_u}B & A^{H_u-1}B & \dots & AB+B \\ \vdots & \vdots & \vdots & \vdots \\ A^{H_p-1}B & A^{H_p-2}B & \dots & \sum_{i=0}^{H_p-H_u} A^i B \end{bmatrix} \begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+H_u-1) \end{bmatrix} \tag{2.2}$$

H_p is called the prediction horizon and is a design variable that determines how many samples into the future predictions for the output should be made, as shown in Figure 2.1. To be able to make these predictions, future values for the control signal are needed and H_u, called the control horizon, determines how many values of the future control sequence are used. The matrices in (2.2) can be rather large but can be precomputed.

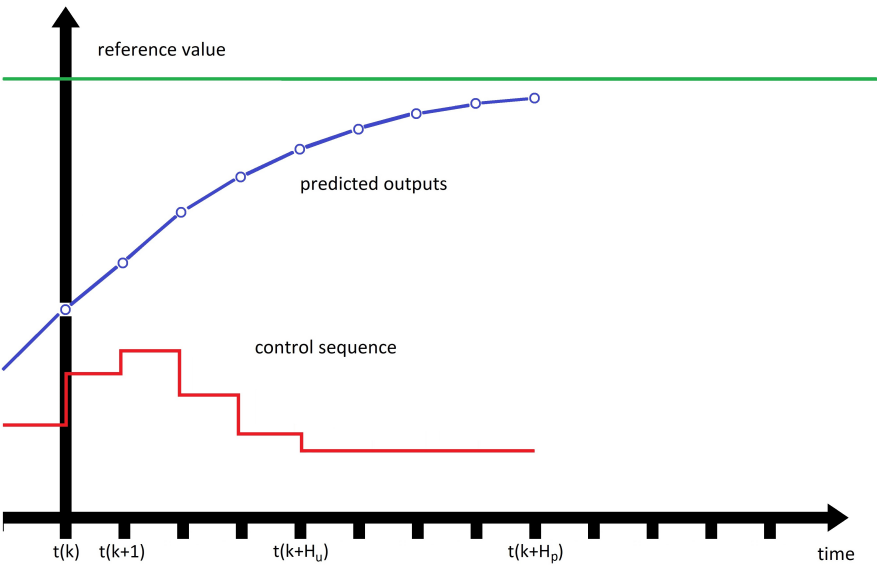

Figure 2.1: In every time sample, the controller uses a mathematical model and current data to predict future behavior. The prediction horizon, H_p is how many samples into the future that predictions are made, and the control horizon, H_u determines how far into the future that the control signal can be changed.

Optimization

The goal of control is often to reach a reference value as fast as possible. But there might be other factors than the speed that also needs consideration in optimization. Some systems have mechanical parts that wear out faster with erratic behavior. Others may have physical constraints, like how much water a tank can hold before it overflows or how far the wheels on a car can turn. In Model Predictive Control optimal control is achieved by minimizing the cost function

$$J(u) = \sum_{i=1}^{H_p} (x_i - r_i)^T Q(x_i - r_i) + u_i^T R u_i \tag{2.3}$$

where Q and R are called weights and are square matrices with appropriate dimensions. Q and R must also be positive semi-definite and positive definite respectively. These matrices are comprised of design values that determine if some states are more important than others and how important the states are compared to the input signal.

As a complement to the cost function, constraints are also defined. The ability to include constraints in the optimization in MPC is an advantage to many other controllers. When the MPC predicts the system's behavior it can early on take measures to prevent exceedances of the constraints. These constraints are most often put on the states, the input, and the slew rate:

$$\begin{aligned} -x(k) &\leq x_{\min} \\ x(k) &\leq x_{\max} \\ -u(k) &\leq u_{\min} \\ u(k) &\leq u_{\max} \\ -\Delta u(k) &\leq \Delta u_{\min} \\ \Delta u(k) &\leq \Delta u_{\max}. \end{aligned}$$

The goal of optimization is now redefined to find a control sequence that minimizes the cost function and satisfies the given constraints.

Quadratic programming

The optimization problem described above can easily be formulated as a quadratic programming (QP) problem. Quadratic programming is commonly used when developing software to solve linear MPC-optimization [Rossiter, 2003].

Receding horizon

Every time the system is sampled, an optimal control sequence is calculated. Only the first value from the control sequence is sent to the plant and the rest of the sequence is discarded. The result of this is that the control signal is always calculated

with the most recent data, which might be different from the predicted due to disturbances or modeling errors. In the next sample, when the controller solves the optimization problem, the time span has been moved one step into the future. This means that in every sample, the prediction horizon considers another time step into the future. For this reason model predictive control is sometimes called receding horizon control. Because the optimization is run every sample with new information, feedback occurs where the control signal is dependent upon the current state of the process.

Nonlinear MPC

If the model is nonlinear and can not be represented by equation (2.1) but instead of

$$\dot{x} = f(x,u) \tag{2.4}$$

then the problem can not be solved with the methods for linear MPC. There are instead other methods to solve this problem. One of these methods is called Sequential quadratic programming [Torrisi et al., 2016] but a closer investigation of the NMPC-problem is outside the scope of this thesis.

2.2 Kalman estimation

If one or several states can not be measured, a Kalman filter can be used to estimate the states. If the system is described by (2.1), the states $\tilde{x}$ can be estimated by

$$\tilde{x}(k+1) = A\tilde{x}(k) + Bu(k) + K(y(k) - C\tilde{x}(k)) \tag{2.5}$$

where K is the Kalman gain [Glad and Ljung, 2003].

2.3 Integral action

Integral action is used to remove steady-state errors and can for MPC be implemented in several ways. In this thesis, integral action has been implemented in two ways, by modeling an input disturbance and by explicitly integrating the error.

Input disturbance model

Assuming the continuous system

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{aligned} \tag{2.6}$$

a disturbance, d on the input-signal enters according to

$$\dot{x}(t) = Ax(t) + B(u(t) + d(t)). \tag{2.7}$$

A steady-state error can be assumed to be constant, $\dot{d} = 0$, which together with (2.6) gives

$$\begin{bmatrix} \dot{x}(t) \\ \dot{d}(t) \end{bmatrix} = \begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ d(t) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u(t) \tag{2.8}$$

$$y(t) = \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ d(t) \end{bmatrix} \tag{2.9}$$

This new model can be discretized with Matlabs function `c2d` and used in an MPC controller. The disturbance d can be estimated with a Kalman filter.

Integrating error

Another way to implement integral action is to keep track of the error $r - y$, where r is the reference and y is the output. By introducing a new state $x_i = \int (r(t) - y(t))dt$, the MPC can adjust the control signal to remove the error. For the discrete system in (2.1) the new model is given by

$$\begin{bmatrix} x(k+1) \\ x_i(k+1) \\ r(k+1) \end{bmatrix} = \begin{bmatrix} A & 0 & 0 \\ -C & I & I \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} x(k) \\ x_i(k) \\ r(k) \end{bmatrix} + \begin{bmatrix} B \\ 0 \\ 0 \end{bmatrix} u(k) \tag{2.10}$$

$$\begin{bmatrix} y(k) \\ y_i(k) \end{bmatrix} = \begin{bmatrix} C & 0 & 0 \\ 0 & I & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ x_i(k) \\ r(k) \end{bmatrix} \tag{2.11}$$

2.4 Gain Scheduling

For a nonlinear system, gain scheduling can be used to improve performance. The system is linearized around several stationary operating points and a controller is constructed for each linearized system. While the controlled plant is running, the controller with the stationary point closest to the current state is used. This means that gain scheduling uses different controllers for different regions.

Piecewise Affine System

An affine system can be defined[1] as

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) + f \\ y(k) &= Cx(k) + g. \end{aligned} \tag{2.12}$$

[1] In this definition, the notation used is the same as in the MPT3 documentation. f and g are constants and f is not the same as in (2.4).

Consider a linearized system

$$\Delta x(k) = A\Delta x(k) + B\Delta u(k) \tag{2.13}$$
$$\Delta y(k) = C\Delta x(k) \tag{2.14}$$

with the stationary operating point (x_0, u_0), and where $\Delta x = x - x_0$, $\Delta u = u - u_0$, and $\Delta y = y - y_0$. Then (2.13) can be also be presented as

$$x(k+1) - x_0 = A(x(k) - x_0) + B(u(k) - u_0). \tag{2.15}$$

Reorganized to

$$x(k+1) = Ax(k) + Bu(k) + (I - A)x_0 - Bu_0 \tag{2.16}$$

f in the affine system can be identified as

$$f = (I - A)x_0 - Bu_0. \tag{2.17}$$

By setting up several affine systems with different stationary operating point (Piecewise Affine System), gain scheduling can be achived.

3

The quadruple tank system

The process used for this thesis is a quadruple tank [Johansson, 2000]. As shown in Figure 3.1, there are four tanks distributed on two levels. The upper tanks empty their contents through a small hole down to the tank directly beneath. There are two pumps and each pump supplies water to two tanks, one tank on the upper level, and one tank on the lower level. Pump 1 supplies water to Tank 1 and Tank 4, Pump 2 supplies water to Tank 2 and Tank 3. The objective is to regulate the water level in the lower tanks.

3.1 Model

The control signal u applies voltage to the pumps, which can handle 0-10 V. The flow of water that the pumps produce is proportional to the voltage supplied to the pumps with a factor k_i. The flow of water is then divided between two tanks with a factor γ. If A_i denotes the cross-section of the tank, then the change of the water level h due to the pumps is

$$\frac{dh}{dt} = \frac{(1-\gamma)k_i}{A_i}u \tag{3.1}$$

for the upper tanks and

$$\frac{dh}{dt} = \frac{\gamma k_i}{A_i}u \tag{3.2}$$

for the lower. A special feature of the quadruple tank is that γ can be set with a switch on the process, but in this thesis γ will be constant.

Something that also affects the water level is the holes at the bottom. The cross-section area of the holes is denoted a and gravity acceleration is denoted g. Using Bernoulli's law the change of the water level due to the hole can be expressed as

$$\frac{dh}{dt} = -\frac{a}{A_i}\sqrt{2gh}. \tag{3.3}$$

With the knowledge that the outflow from the upper tanks go into the lower tanks and with the denotation of the tanks and the pumps from Figure 3.1, the model for

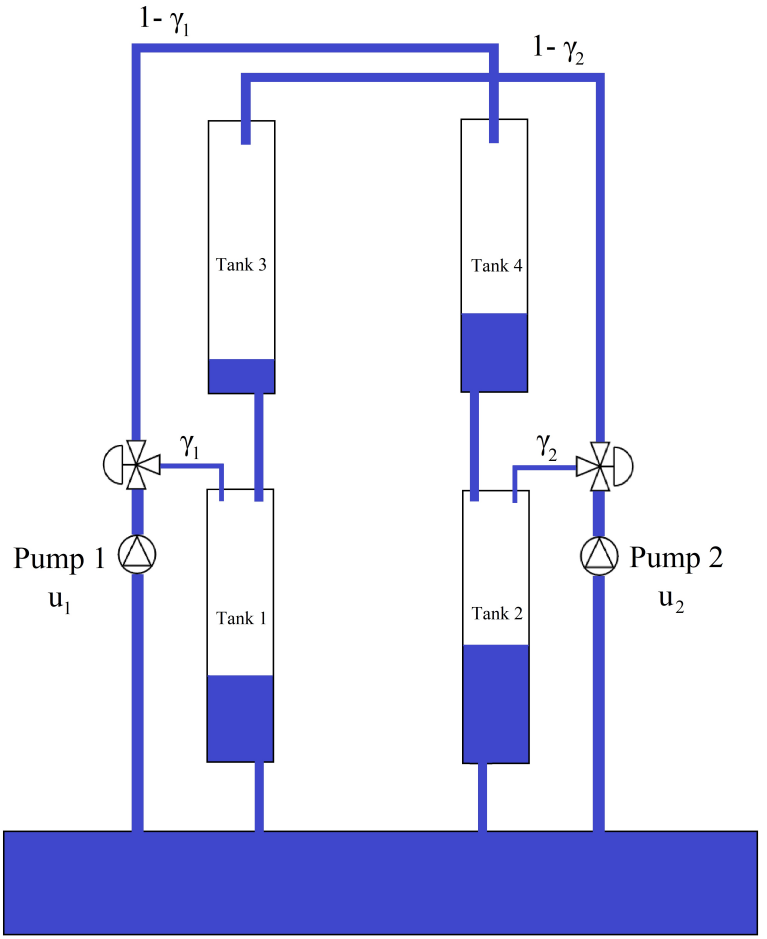

Figure 3.1: A schematic figure of the quadruple tank. The aim is to control the water level in Tank 1 and Tank 2.

the quadruple tank's behavior is modeled by

$$\frac{dh_1}{dt} = -\frac{a_1}{A_1}\sqrt{2gh_1} + \frac{a_3}{A_1}\sqrt{2gh_3} + \frac{\gamma_1 k_1}{A_1}u_1 \tag{3.4}$$

$$\frac{dh_2}{dt} = -\frac{a_2}{A_2}\sqrt{2gh_2} + \frac{a_4}{A_2}\sqrt{2gh_4} + \frac{\gamma_2 k_2}{A_2}u_2 \tag{3.5}$$

$$\frac{dh_3}{dt} = -\frac{a_3}{A_3}\sqrt{2gh_3} + \frac{(1-\gamma_2)k_2}{A_3}u_2 \tag{3.6}$$

$$\frac{dh_4}{dt} = -\frac{a_4}{A_4}\sqrt{2gh_4} + \frac{(1-\gamma_1)k_1}{A_4}u_1. \tag{3.7}$$

For the process, there are some physical constraints. The pumps can only be run with $0-10V$ and the water level can only be between $0-16cm$, as the tanks will overflow if it goes any higher. To have some margins, the constraints are set to $0-15cm$.

3.2 Linearized model

The model described above is nonlinear but this thesis is primarily analyzing toolboxes for linear MPC. Consequently, the model needs to be linearized around a stationary operating point $(h_1^0, h_2^0, h_3^0, h_4^0, u_1^0, u_2^0)$. With the water levels as states, $x = [h_1\ h_2\ h_3\ h_4]^T$ and $u = [u_1\ u_2]^T$, the linearized model is

$$\Delta\dot{x} = \begin{bmatrix} -\frac{1}{T_1} & 0 & \frac{A_3}{A_1 T_3} & 0 \\ 0 & -\frac{1}{T_2} & 0 & \frac{A_4}{A_2 T_4} \\ 0 & 0 & -\frac{1}{T_3} & 0 \\ 0 & 0 & 0 & -\frac{1}{T_4} \end{bmatrix} \Delta x + \begin{bmatrix} \frac{\gamma_1 k_1}{A_1} & 0 \\ 0 & \frac{\gamma_2 k_2}{A_2} \\ 0 & \frac{(1-\gamma_2)k_2}{A_3} \\ \frac{(1-\gamma_1)k_1}{A_4} & 0 \end{bmatrix} \Delta u \tag{3.8}$$

$$\Delta y = \begin{bmatrix} k_1^c & 0 & 0 & 0 \\ 0 & k_2^c & 0 & 0 \\ 0 & 0 & k_3^c & 0 \\ 0 & 0 & 0 & k_4^c \end{bmatrix} \Delta x \tag{3.9}$$

where $\Delta x = x - x_0$, $\Delta u = u - u_0$, and $\Delta y = y - y_0$. Furthermore, the sensor that registers the water level in the tanks provides a voltage and the water level relates to the sensor reading by $y_i = k_i^c x_i$. To simplify the expression and make the model more legible, T_i for $i \in [1,2,3,4]$ are introduced where

$$T_i = \frac{A_i}{a_i}\sqrt{\frac{2x_i^0}{g}}. \tag{3.10}$$

With the system linearized around a stationary operating point, the constraints also have to be redefined. For the linearized model, the constraints are

$$\begin{aligned} x_{min} &= 0 - x_0 \\ x_{max} &= 15 - x_0 \\ u_{min} &= 0 - u_0 \\ u_{max} &= 10 - u_0. \end{aligned} \tag{3.11}$$

For the quadruple tank, a suitable sampling interval is 0.5 s. This value was used throughout the thesis.

3.3 Parameters

To determine the parameters of the process, some experiments were run. To confirm the results, the experiments were run several times or in more than one way. All the values for the parameters are presented in Table 3.1.

The cross-sections, A_i were determined to 5.7 cm^2 by measuring the inner diameter of the tank to 2.7 cm but also by sealing the holes at the bottom of the tanks and filling them with 0.5 dl of water. By measuring the height of the water level to 8.7 cm and knowing the volume of the water the cross-section could be calculated.

The sensor reading relates to the water level by k^c. Filling the sealed tanks with a known volume of water, the corresponding water level was calculated. By comparing the water level with the sensor reading by $y = k^c h$, it was determined that the tanks had k^c values of approximately 4.5-5 V/cm.

Pump 1 supplies water to both Tank 1 and Tank 4. The same goes for Pump 2 with Tanks 2 and 3. γ describes how many percent of the water goes into the lower tanks and the switch is set so that the majority of the water goes to the upper tank. The empty tanks were sealed and then filled, using the pumps, with a nonspecific amount of water. The parameter γ could be calculated by

$$\gamma_1 = \frac{h_1}{h_1 + h_4} \tag{3.12}$$

where h_i is the water level in Tank i. The parameter γ for both pumps were calculated to 0.27.

To determine k_i, the propotional factor between the voltage supplied to the pumps and the water flow, $q = k_i u$, some calculations were needed. Consider (3.1) and (3.2). These describe how the water level changes when the tanks are sealed and the pumps are run. If u is constant and the water level is h^0 at a starting time t^0, then a simple integation of (3.1) leads to

$$h = h^0 + \frac{(1-\gamma)k_i}{A_i} u(t - t^0). \tag{3.13}$$

Rearranged it becomes

$$k_i = \frac{A_i(h - h^0)}{(t - t^0)(1-\gamma)u}. \tag{3.14}$$

Similar calculations starting from (3.2) leads to

$$k_i = \frac{A_i(h - h^0)}{(t - t^0)\gamma u} \tag{3.15}$$

for the lower tanks. By filling each tank while sealed and with constant u, and then inserting the values from two points into (3.14) or (3.15) a k_i could be calculated for each tank.

Table 3.1: Measured parameters of the quadruple tank.

Variable	Value	Description
A_i	5.7 cm^2	Cross-section of the tanks. i = 1,2,3,4.
a_1	0.058 cm^2	Cross-section of the outlet-holes of tanks 1-4.
a_2	0.072 cm^2	
a_3	0.065 cm^2	
a_4	0.066 cm^2	
g	981 cm/s^2	Gravity
γ_1	0.27	Percent of the water flow from pump i that goes into the lower tank. $i = 1,2$.
γ_2	0.27	
k_1	1.65 cm^3/V	Proportional factor between the voltage supplied to the pumps and the water flow.
k_2	1.65 cm^3/V	
k_1^c	0.492 V/cm	Proportional factor between the sensor reading of the water level and the actual water level.
k_2^c	0.495 V/cm	
k_3^c	0.553 V/cm	
k_4^c	0.544 V/cm	

The cross-section of the outlet-holes a_i can be calculated in a similar manner as k_i. Equation (3.3) can be rearanged

$$\frac{1}{\sqrt{h}}\frac{dh}{dt} = -\frac{a}{A_i}\sqrt{2g}. \tag{3.16}$$

Integration leads to

$$\sqrt{h} = \sqrt{h^0} - \frac{a}{A_i}\sqrt{\frac{g}{2}} \tag{3.17}$$

and a can then be calculated with

$$a = A_i\sqrt{\frac{2}{g}}\frac{\sqrt{h^0}-\sqrt{h}}{t-t^0}. \tag{3.18}$$

The experiment was conducted by filling up the tanks and then with $u = 0$ letting the water out. Looking at each tank seperatly, values from two points were inserted to the equation to calculate a.

4

Implementation

All the toolboxes that have been explored in this thesis can be used in Matlab. In this chapter, the setup for each toolbox will be described including Simulink models and code snippets. After a description of how to set up a basic MPC, the different ways to achieve integral action will be explored, and how to implement gain scheduling.

4.1 Simulink setup and basic MPC

Figure 4.1 shows the Simulink model used for running simulations. In this setup, the controller uses the states x in the MPC calculation. As it is possible in this process to measure all the tanks, the output y is simply converted to x by the gain $1/k^c$. This conversion is done here immediately after the block for the Simulated Quadtank. As the MPC is linearized and actually uses the states $\Delta x = x - x_0$, the x_0 is subtracted

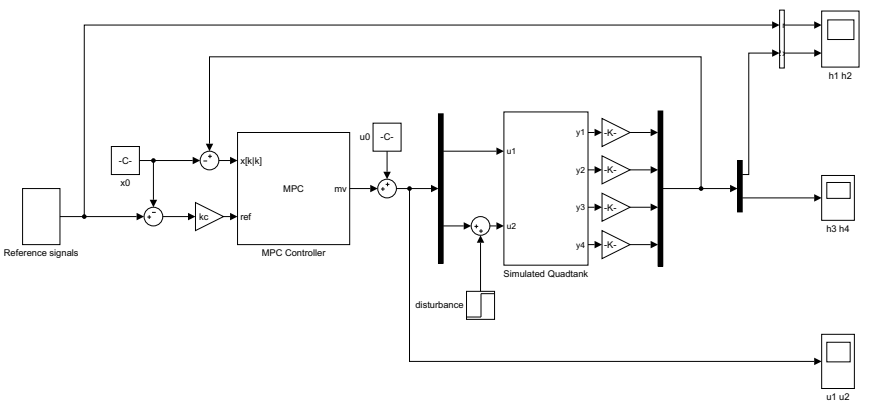

Figure 4.1: The setup for running the simulations in Simulink. The same setup is used for all toolboxes but here, the MPC block is from Mathworks toolbox.

from the signals with the states before it is supplied to the MPC block. For the same reason, the u_0 is added to the control signal before it is supplied to the simulation block.

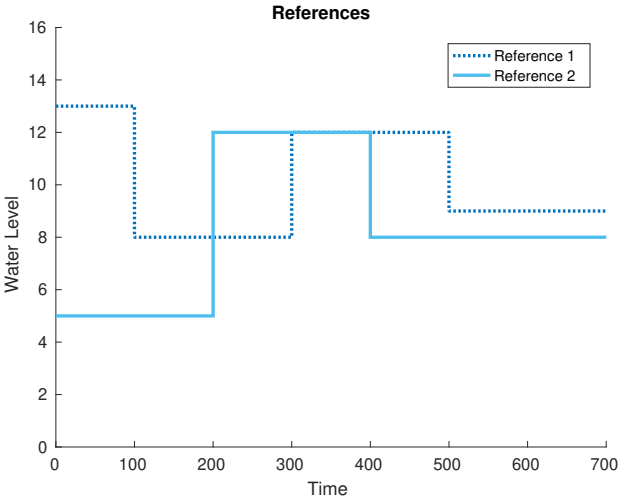

Figure 4.2: Reference signals for Tank 1 and Tank 2 for all controllers.

For this setup of the process, the objective is to control the lower tanks. The reference signals were chosen to examine the controller's behavior in three instances. It is interesting to observe the behavior of the reference change for all the controllers, both when the reference is set to a higher value, as well as a lower. As the model has been linearized it is also interesting to see the behavior at different water levels. To assess the controllers with gain scheduling, reference signals with different combinations of water levels were used. In addition to changing the reference signal, an input disturbance signal was added at 600 seconds, which were most relevant in the setups where the controllers used integral action. In Figure 4.2, the reference signal used when running all controllers can be seen. Reference 1 is used to control the water level in Tank 1, and Reference 2 for controlling the water level in Tank 2. The tanks can be studied in Figure 3.1. Because all the states can be measured in this process, there are four outputs. For each output, a reference signal is needed, but the objective is to control only the lower tanks and the upper tanks should not follow a reference. In this thesis, this is handled in two ways. Either, the output formula in (3.9) is changed to

$$\Delta y = \begin{bmatrix} k_1^c & 0 & 0 & 0 \\ 0 & k_2^c & 0 & 0 \end{bmatrix} \Delta x \tag{4.1}$$

in the MPC model, telling the controller that there are only two outputs. Or, references are sent to all outputs, but weights on the upper tanks are set to zero, telling

the controller that following these references have zero priority.

For all three toolboxes, the same basic Simulink setup was used in Matlab. The controllers were first run in simulations and then with the real process. The simulation block of the process is set up from the nonlinear model in (3.4)-(3.7). To simulate the limitations of the process, a saturation has been put on the water levels in the tanks to resemble that when the tank is empty, the water level can not go down and if the water level is higher than 16 cm, it will overflow and not go up higher. ε is a number slightly larger than zero and used as a lower saturation instead of 0 to avoid problems with numerical derivation. In reality, the pumps are limited to run at 0-10 V and so there is saturation on the input as well.

When the controllers were run with the real process, the simulation block was exchanged for a Simulink block that communicates with the process. Just like the simulation block, it takes two input signals for the pumps and supplies four readings of the water level in the tanks. To accompany the real process block, a low pass filter with 1.3 rad/s was added to the outputs to filter out measurement noise. The filter put on each output signal is

$$\frac{1}{0.25s^2+s+1}. \tag{4.2}$$

To linearize the model in (3.4)-(3.7), a stationary operating point was found for $h_1^0 = h_2^0 = 10$. With the parameters in Table 3.1, the stationary operating point was found at $x_0 = (10, 10, 3.1259, 7.8722)$ and $u_0 = (6.8098, 4.2261)$. If not stated otherwise, all the controllers use the model (3.8)-(3.9) with this stationary operating point in the MPC setup.

The Simulink integration routine used when running against both the simulated and real process was the ode45 default variable-step solver.

Mathworks Model Predictive Control Toolbox

Mathworks provides a large number of toolboxes for Matlab, and one of them is the Model Predictive Control Toolbox [Mathworks, 2020b]. The toolbox supplies a Simulink block in which the user can define an MPC-object. There is a substantial amount of documentation and examples for the toolbox, most of which describes how to use the *MPC Designer*. The *MPC Designer* is a graphical interface where the user can create an MPC-object.

Another way to create an MPC-object is at command line, or in a script. The controllers in this thesis have been created with a script.

Mathworks' toolbox has a particular definition of the input and output signals. Figure 4.3 shows how different signals are defined. The basic MPC-setup in this thesis only uses Measured Outputs as Output Variables (OV) and only Manipulated Variables (MV) as inputs.

The following code snippets show how to create an MPC-object with cost function weights and constraints.

```
mpcobj = mpc(model,Ts,Hp,Hu, Weights,constraints_MV,constraints_OV);
```

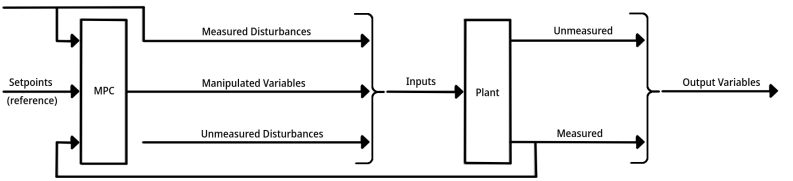

Figure 4.3: A diagram of how Mathworks defines the signals in an MPC design [Mathworks, 2020a].

where `model` is a discrete state-space representation of the model, `Ts` is the sample time, `Hp` and `Hu` are the prediction horizon and control horizon respectively. In this toolbox, constraints and weights are set for the outputs and can not be set for the states. Because of this, four references were given to the controller and the weights were set by

```
Weights.OV = [10 10 0 0];
Weights.MV = [1 1];
Weights.ManipulatedVariablesRate = [3 3];
```

As there are four outputs, constraints can be set on all water levels by

```
constraints_MV = struct('Min',u_min,'Max',u_max,...
                    ...'RateMin',du_min, 'RateMax', du_max);
constraints_OV = struct('Min',y_min,'Max',y_max);
```

where, for example `u_min = {-u1 -u2};`, and here `u1`$= u_1^0$ and `u2`$= u_2^0$ (3.11). Default settings for this toolbox are hard constraints for Manipulated Variables and soft constraints for Output Variables. All controllers in this toolbox use these default settings.

In the MPC-controller block, the user can choose between measured outputs or custom estimated states. When measured outputs are used, the controller block accepts the output y from the process, or, when a linearized model is used, Δy. The controller then has an internal Kalman estimator to get the states from the output. If instead, the custom estimated states are used, the user has to use an external estimator if one or several states can not be measured. In addition to choosing custom estimated states in the MPC-controller block, this property of the MPC object is set by

```
setEstimator(mpcobj,'custom');
```

One of this toolbox's features is the disturbance models. The user can design models for input-disturbance, output-disturbance, and measurement noise. If the user does not define disturbance and noise models, the toolbox will set it up automatically. To be able to test different models for integral action and gain scheduling, all disturbance and noise models were initially turned off. Output disturbance is set by a function

```
setoutdist(mpcobj,'model',tf(zeros(4,1)));
```

while noise models are set as a parameter

```
mpcobj.Model.Noise = tf(zeros(4,1));
```

Input disturbance can only be used if some of the inputs are defined as unmeasured inputs, which there are not for this setup.

Multi-Parametric Toolbox 3

Multi-Parametric Toolbox 3 (MPT3) is a toolbox for model predictive control, parametric optimization, and computational geometry [Herceg et al., 2013]. It is an open-source toolbox for Matlab. The documentation for the toolbox on the website is limited and not well organized, but there are many informative examples to read. In the installed toolbox, there is a document containing a description of available functions. As is the case with the website, this document is not well organized.

As the toolbox does not provide a Simulink block, the user needs to create a block of their own design if the use of Simulink is required. In this thesis, an S-function was used to create an MPC-object and to run the optimization with the toolbox. In the S-function, the MPC-calculations are run with the command

```
u = mpcobj.evaluate([x1;x2;x3;x4],'x.reference',[r1;r2;r3;r4]);
```

where `x1-x4` are the current states and `r1-r4` are the reference signals. In this toolbox, the user can choose if the reference-signals are for the states or the outputs. For this thesis, both ways were examined but most results presented for this toolbox uses references to states.

Unlike Mathworks toolbox, the MPT3 toolbox does not supply a state estimator, which means that if not all states can be measures, an external state-estimator is required to supply the controller with all the states.

To create an MPC-object, a model is created as an `LTISystem`-object from the toolbox

```
model = c2d(ss(A,B,C,D),Ts);
model = LTISystem(model);
```

where `A,B,C` are the matrices from equations (3.8) and (3.9), `D` is a zero-matrix and `Ts` is the sample time. Different properties are added to the LTI-system as parameters. One example of how to define constraints is

```
model.u.min = [-u1 -u2];
```

where u1= u_1^0 and u2= u_2^0 (3.11). The user can also choose to put constraints on the states or the output, simply by writing x or y instead of u. There are two ways to set constraints in the MPT3 toolbox, with the min/max described above, or with softMin/softMax. With min/max, the constraints that are set are hard constraints, while softMin/softMax are soft constraints. In all controllers set up with this toolbox, the min/max were used for all constraints, setting all constrains to hard constraints.

In the MPT3-toolbox, a QuadFunction-object can be created. This is used when weights are defined.

```
model.u.penalty = QuadFunction([1 0; 0 1]);
```

As with constraints, weights can be put on states or outputs. In this thesis, weights were put on states when x.reference was used and put on outputs when y.reference was used.

```
model.x.penalty = QuadFunction([[10 0 0 0;
                                  0 10 0 0;
                                  0 0 0 0;
                                  0 0 0 0]);
```

In one example on the website, weights on the slew rate were defined by model.u.deltaPenalty, but when an attempt was made to use this, a compilation error occurred.

To define the MPC-object, the command

```
mpcobj = MPCController(model,Hp);
```

is used, where Hp is the prediction horizon. The control horizon is not defined by the user in this toolbox and the documentation does not state how it is defined.

MATMPC

The MATMPC-toolbox is a new open-source toolbox from 2017 for Nonlinear MPC. It aims to be a fast and easy to use toolbox for nonlinear MPC, even for users with limited programming skills [Chen et al., 2019]. The documentation for the toolbox is sparse, and examples are few. There are many files of code, where comments also are rare.

When creating a model, the user has to define some dimensions.

```
%% Dimensions

nx=4;  % No. of differential states
nu=2;  % No. of controls
```

```
nz=0;  % No. of algebraic states
ny=6; % No. of outputs
nyN=4; % No. of outputs at the terminal point
np=0; % No. of model parameters
nc=0; % No. of general constraints
ncN=0; % No. of general constraints at the terminal point
nbx = 4; % No. of bounds on states
nbu = 2; % No. of bounds on controls

% state and control bounds
nbx_idx = 1:4; % indexs of states which are bounded
nbu_idx = 1:2; % indexs of controls which are bounded
```

No more information about the parameters is available, except finding where and how they are used in the code examples. The dimension set in the code snippet above is the actual dimensions for this process. Note that "`No. of outputs`" are set to 6, which is the number of states together with the number of inputs.

As this is a toolbox for nonlinear MPC, the model is not defined as matrices like in (2.1) but as a vector,

```
x_dot=[h1_dot;h2_dot;h3_dot;h4_dot];
```

where `h1_dot-h4_dot` is the right-hand side of (3.8). When the model is set up, the name of the script containing the model is inserted into a script called `Model_Generation.m`, which is then run. Before running the process or the simulation in Simulink, the script "`Initialization_Simulink.m`" must also be edited and run. In this script, there are some options that the user can choose from, for example

```
%% options
opt.hessian='Gauss_Newton';  % 'Gauss_Newton',
                             % 'Generalized_Gauss_Newton'
opt.integrator='ERK4'; % 'ERK4','IRK3, 'IRK3-DAE'
opt.condensing='default_full';%'default_full','no',
                              %'blasfeo_full(require blasfeo
                              %installed)','partial_condensing'
opt.qpsolver='qpoases';
opt.hotstart='no'; %'yes','no' (only for qpoases)
opt.shifting='no'; % 'yes','no'
opt.ref_type=1; % 0-time invariant, 1-time varying(no preview),
                % 2-time varying (preview)
opt.nonuniform_grid=0; % currently not supported
opt.RTI='no';
```

As with the parameters, there is no more information about the options to be found. In "Initialization_Simulink.m", prediction horizon, constraints, and weights are set, but as with the MPT3-toolbox, there is nowhere to set the control horizon. Constraints for the control signals are set with

```
% upper and lower bounds for controls (=nbu)
lb_u = [-u1;-u2];
ub_u = [10-u1;10-u2];
```

where u1= u_1^0 and u2= u_2^0 (3.11). No information were found on whether the constraints are hard or soft constraints.

Because the nonlinear MPC model is set up on the form of (2.4), the references are given for the states. Furthermore, the toolbox expects a reference for the control signal, and reference values need to be given for every point on the prediction horizon. As an example, for a model with 4 states, 2 control signals, and a prediction horizon of 60, a reference matrix of 6×60 is used. For the setups in this thesis, the reference is constant over the prediction horizon and set to zero for all signals except the reference to Tank 1 and Tank 2. The weights are put on the states and controls with

```
W=repmat([10 10 0 0 1 1]',1,N);
```

Like the reference, the weights are also given for all points on the prediction horizon N. In Mathworks toolbox, the weights on the slew rate were important when tuning the controllers, but in the MATMPC toolbox, there is no mention of how to set weights on the slew rate.

4.2 Integral Action

In Chapter 2, two ways of achieving integral action are described. These two ways were implemented for all three toolboxes. For the input disturbance model, the matrices were rewritten as (2.8)-(2.9) and the Simulink block in Figure 4.4 was added before the states were entered to the MPC-block. In (2.8) there are 6 states. From Port 1, called x_in, measurements from the tanks are entered. They are converted to the linearized model and then sent to the x_out as the four first states. The two last states are the modeled input disturbances which are estimated with a Kalman filter. The Kalman filter used is a Simulink block from the Control System Toolbox, where the discrete-time version of the filter was chosen.

For integrating error, the matrices were rewritten as (2.10)-(2.11). In (2.10), there are 8 states, first the 4 states that represent the water levels in the tanks, then 2 states for the integrated error and the last two states are the references for Tank 1 and Tank 2. Figure 4.5 shows the Simulink block added for this model. Both the measurements from the tanks and the references are converted to the linearized model and then sent to the x_out. In the middle, the operation $r - y$ is preformed then sent

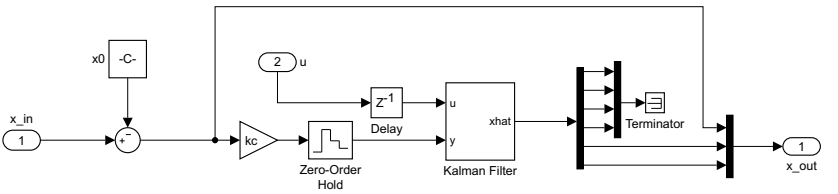

Figure 4.4: A Simulink sub-block added to the Simulink model in Figure 4.3 when integral action using the input disturbance model is implemented.

to the integrator block. Anti-windup for systems with muliple inputs and outputs was outside the scope for this thesis but as a simple anti-windup, the integrator is saturated.

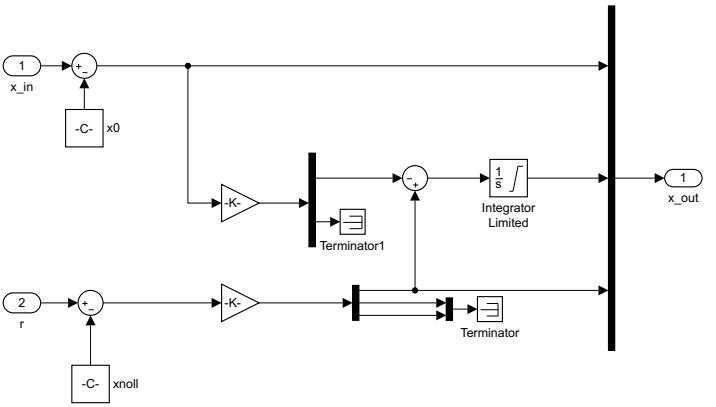

Figure 4.5: A Simulink sub-block added to the Simulink model in Figure 4.3 when integral action using the integrating error model is implemented.

These implementations of integral action were made in the same way for all toolboxes but the Mathworks toolbox and the MPT3 toolbox also have their own way of implementing integral action. As described in the section about implementing a basic MPC in Mathworks toolbox, there are disturbance models that will be active if not turned off by the user. Default for the toolbox is to use an output disturbance model that adds an integrator for all measured outputs. The integrators have dimensionless unity gain and are only added as long as it does not cause a violation of state observability. Another way is for the user to define the disturbance model. Matlab's disturbance models can be used to implement integral action in the form of input disturbance in a sampled version of (2.7). Adding the Bd to the model and defining d as an unmeasured input disturbance is done by

```
B = [B B];
```

then creating the model and defining what kind of signals it is with

```
model = setmpcsignals(model,'MV',[1 2],'UD',[3 4],'MO',[1 2]);
```

where Input Signal 1 and 2 are set as measured variables, inputs 3 and 4 are set as unmeasured disturbances and outputs 1 and 2 are set as measured outputs.

In a simplified version of Mathworks disturbance model

$$\begin{bmatrix} x_p(k+1) \\ x_{id}(k+1) \end{bmatrix} = \begin{bmatrix} A_p & B_{pd}C_{id} \\ 0 & A_{id} \end{bmatrix} \begin{bmatrix} x_p(k) \\ x_{id}(k) \end{bmatrix} + \begin{bmatrix} B_{pu} & B_{pd}D_{id} \\ 0 & B_{id} \end{bmatrix} \begin{bmatrix} u(k) \\ w_{id}(k) \end{bmatrix} \quad (4.3)$$

$$y(k) = \begin{bmatrix} C_p & D_{pd}C_{id} \end{bmatrix} \begin{bmatrix} x_p(k) \\ x_{id}(k) \end{bmatrix} + \begin{bmatrix} 0 & D_{pd}D_{id} \end{bmatrix} \begin{bmatrix} u(k) \\ w_{id}(k) \end{bmatrix} \quad (4.4)$$

A_p, B_{pu} and C_p can be identified as the matrices from (2.1). Considering a sampled version of (2.7), B_{pd} can be identified as the same as B_{pu}. Equation (2.9) shows that in the model, no disturbance affects the output directly, which means that D_{pd} is a zero-matrix.

To get the same input disturbance model form Mathworks disturbance model as the one in equations (2.8)-(2.9) the `setindist` were used as following

```
Aid = zeros(2);
Bid = 2*eye(2);
Cid = eye(2);
Did = zeros(2);
distMod = ss(Aid, Bid, Cid, Did);
setindist(mpcobj,'model',distMod);
```

where `distMod` is in continuous time and discretized by the toolbox. Although there were changes made to the system, the weights were still set on the same outputs and control signals, but when running the real process it was clear that the controller needed to be tuned differently than the other controller, and the best weights were found to be

```
Weights.OV = [10 10 0 0] ;
Weights.MV = [3 3];
Weights.ManipulatedVariablesRate = [40 40];
```

The MPT3 toolbox also has a function for adding integral action. The code

```
model.with('integrator');
```

is added when the MPC object is created. With the example, where this integration is used, there is a comment that this extends the model with an integrator state, but there is no more information found.

4.3 Gain Scheduling and Nonlinear MPC

In Mathworks toolbox, there is a controller block in Simulink for gain scheduling. The only real difference between this block and the regular MPC block is that in the gain scheduling block, the user can enter multiple MPC-objects and there is a port for a switch value. For this thesis, a user-defined block was used with if-statements checking what region the two lower tanks were in. The water levels were divided into three regions, $0-6$ cm, $6-10$ cm and $10-16$ cm, linearized around $h^0=4$, $h^0=8$ and $h^0=12$. Combinations of these water levels in Tank 1 and Tank 2 results in 9 controllers. The block producing the switch values also supplies the linearization point that corresponds to the different controllers.

In the MPT3 toolbox, an attempt was made to use the toolbox's function for Piecewise Affine System (PWA) for gain scheduling. Using the sampled matrices form (3.8) in (2.12) and with f identified as (2.17) an affine model can in MPT3 be created with

```
f = (eye(4)-A)*[h1; h2; h3; h4]-B*[u1; u2];
model = LTISystem('A',A,'B',B,'f',f,'C',C,'D',D,'Ts',Ts);
```

where `D` is a zero-matrix and `Ts` is the sample time of $0.5s$. To evaluate the function for PWA an MPC object was created with the affine model. With the linearized system as a model, the expected result would be $u=(0,0)$ if $x=(0,0,0,0)$ and the reference was $r=(0,0)$. With the affine model, this would be the same as getting $u=(u_1^0,u_2^0)$ if $x=(h_1^0,h_2^0,h_3^0,h_4^0)$ and the reference was $r=(k_1^c h_1^0, k_2^c h_2^0)$ (3.11). Calculations for a stationary operating point at $h_1^0=h_2^0=10$ leads to $u_0=(6.8098,4.2261)$ but

```
u = mpcobj.evaluate([h1;h2;h3;h4], 'y.reference', [kc1*h1;kc2*h1]);
```

results in $u=(5.2443,3.9821)$, where `h1`$=h_1^0$, `h2`$=h_2^0$, `h3`$=h_3^0$ and `h4`$=h_4^0$.

Another way to implement gain scheduling is to use the S-function in Simulink. Because the MPC object was already created in the S-function, only a minor adjustment was needed to create multiple controllers. In the execution part of the S-function, a setup with if-statements very similar to the one used with Mathworks toolbox were used.

Because the MATMPC toolbox is a toolbox for nonlinear MPC, it does not have any functions for gain scheduling. Instead, a controller based on the nonlinear system in (3.4)-(3.7) was created. The only changes needed were to enter the nonlinear equation instead of the linearized ones and to set the constraints as the actual values instead of the ones used for the linearized model.

5

Results

In this chapter, the results from the three different toolboxes, both simulations and running the quadruple tank will be presented as plots. There will be at least one plot from each toolbox showing the results from the basic setup, two types of integral action and either gain scheduling or nonlinear MPC. The plots will consist of two or three parts. First, all the water levels from the tanks will be presented in one plot. Because Tank 1 and Tank 2 are the water levels being controlled with references, these signals are more bold on the plots. There is also a thin line marking the constraints for the water levels. The control signals are presented below in a separate plot. Here, the constraints are also presented as thin lines. For some controllers, there are other interesting signals. For instance, in gain scheduling, the switch values are useful to study. These signals will be presented last in a separate plot.

While studying the plots for the real process, it is observed that the signal from Tank 2 is much noisier than the other signals. This is because the water coming into the tank was splashing the surface most of the time, while in the other tanks, the water was flowing gently along the side of the tank.

5.1 Mathworks Model Predictive Control Toolbox

In this section, the plots will show results from Mathworks toolbox.

- Figure 5.1. Basic MPC, simulation.
- Figure 5.2. Basic MPC, real process.
- Figure 5.3. Integral action with Input disturbance model, external Kalman filter, simulation.
- Figure 5.4. Integral action with Input disturbance model, external Kalman filter, real process.
- Figure 5.5. Integral action with Input disturbance model, toolbox's disturbance model, simulation.

- Figure 5.6. Integral action with Input disturbance model, toolbox's disturbance model, real process.
- Figure 5.7. Default disturbance model, simulation.
- Figure 5.8. Default disturbance model, real process.
- Figure 5.9. Integral action with Integrating error model, simulation.
- Figure 5.10. Gain scheduling, simulation.
- Figure 5.11. Gain scheduling, real process.

Figure 5.1 shows the plot from a simulation where the controller is created with the Mathworks toolbox. For this controller, no special features, like integral action or gain scheduling are implemented and all default disturbance models are turned off. For this simulation, it can be observed that the controller works very well, with only a slight stationary error. All the water levels and the control signals are within the constraints, and because there is no integral action, the input disturbance on Control Signal 1 is clearly visible at 600 s, which is expected. When running the same controller with the real process, the control signals are still within the constraints but the water level in Tank 2 violates the constraints at 300 s. This does not result in any special action or warnings from the controller. Another interesting behavior to notice is that there are some oscillations after the reference changes before the system settles.

Figure 5.3 shows the simulation where the controller uses the model in (2.8)-(2.9) to model an input disturbance and implement integral action. An external Kalman filter is used to estimate the states representing the disturbance. The plot from running this controller with the real process is presented in Figure 5.4. In simulation all the signals are within the constraints and the input disturbance at 600 s is warded off with the integral action. The real process in Figure 5.4 shows the same amount of oscillations as in Figure 5.2 and the constraints are violated at 200 s and 300 s. The most interesting with this controller is that although there is visible integral action the stationary error does not disappear completely, not even in simulation. Most obvious is in Figure 5.3 for the water level in Tank 1 between 400 s and 500 s.

The next controller, presented with simulation in Figure 5.5 and with the real process in Figure 5.6, also uses model (2.8)-(2.9). Instead of an external Kalman filter, this controller uses the toolboxes disturbance model to implement the integral action. The plot for the real process shows much oscillation and while the input disturbance at 600 s is counteracted, the water levels in Tank 1 and Tank 2 do not follow the reference very closely. Also worth noting is that the difference between the states that the toolbox has estimated and the actual states is considerable at times, even in the simulation. Constraint violation happens one time in the real process for the water level in Tank 1, right after startup.

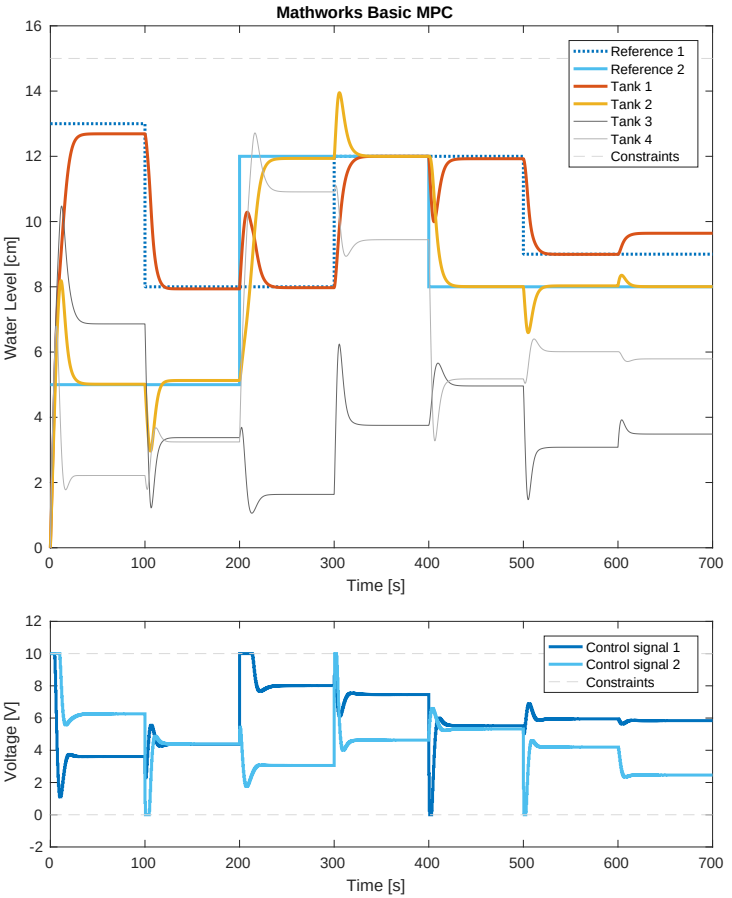

Figure 5.1: Simulation of a controller created with Mathworks toolbox. Upper plot showing water levels in Tanks 1-4 together with reference signals for Tank 1 and Tank 2. The lower plot shows control signals. This controller uses the model in (3.8)-(3.9) with no disturbance models or gain scheduling.

In Figures 5.7 and 5.8 the plots from simulations and running the process with a controller using Mathworks toolbox's default disturbance models are shown. While the behavior in the simulation is satisfactory, the real process shows constraint violations of the water level in both Tank 3 and Tank 4 on more than one occasion. And while the water levels in Tank 1 and Tank 2 follow the references sufficiently, the system is highly oscillatory and hardly settles before the next reference change.

The simulation, with the controller using model (2.10)-(2.11) is shown in Figure 5.9. In the simulation, it can be seen that stationary error is removed as well as the

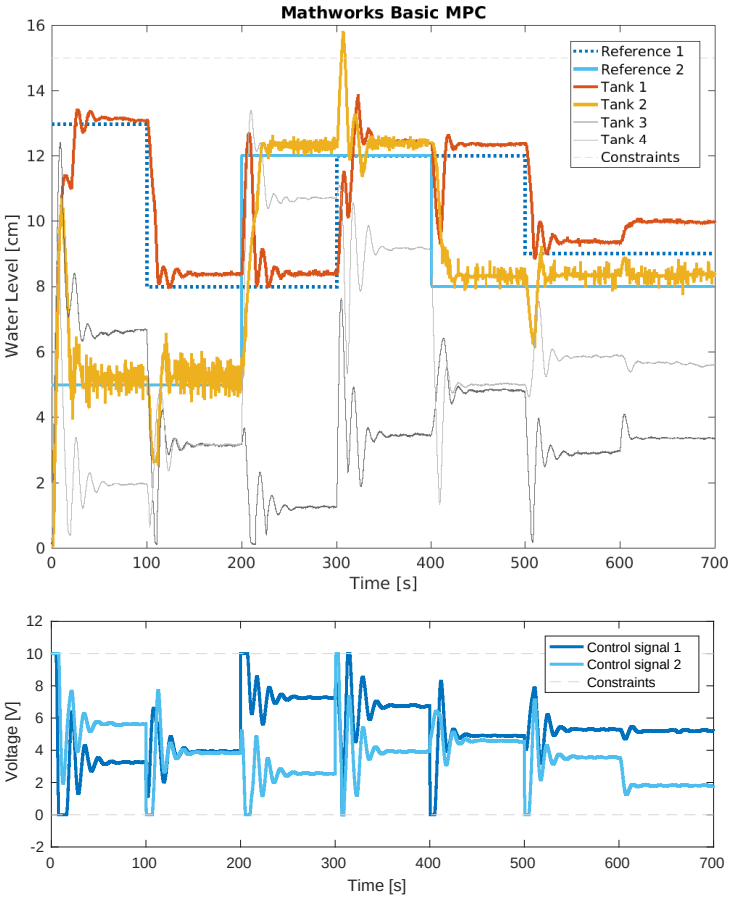

Figure 5.2: Running of the real process with a controller created with Mathworks toolbox. Upper plot showing water levels in Tanks 1-4 together with reference signals for Tank 1 and Tank 2. The lower plot shows control signals. This controller uses the model in (3.8)-(3.9) with no disturbance models or gain scheduling.

disturbance at 600 s. In the plot of the integrated error, the saturation implemented to achieve anti-windup can be seen.

Gain scheduling was implemented in the controller and Figure 5.10 shows a plot of the simulation while Figure 5.11 shows a plot of the real process. In the plots of the switch values, it can be seen that the controller uses different MPC objects both during the simulation and while running the real process. While the water levels in the simulation follow the reference very closely, it seems to have a constant offset in the real process. In the real process, there are also some constraints violations. The

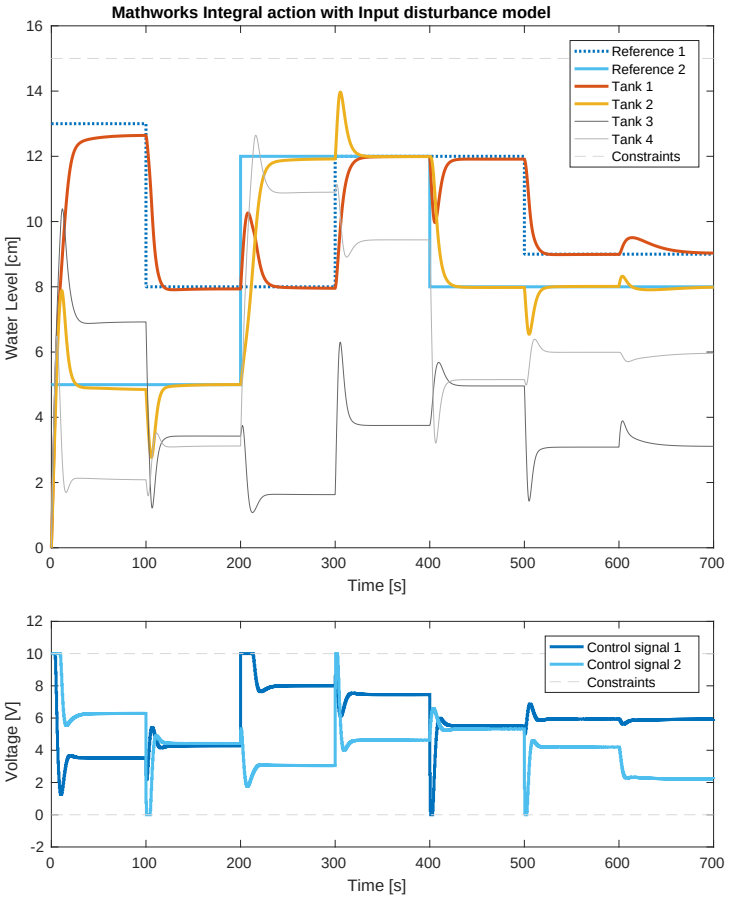

Figure 5.3: Simulation of a controller created with Mathworks toolbox. Upper plot showing water levels in Tanks 1-4 together with reference signals for Tank 1 and Tank 2. The lower plot shows control signals. This controller uses the model in (2.8)-(2.9) and an external Kalman filter.

water level in Tank 2 is slightly above 15 at 300 s, but more interesting, the control signals violate the constraints 3 times.

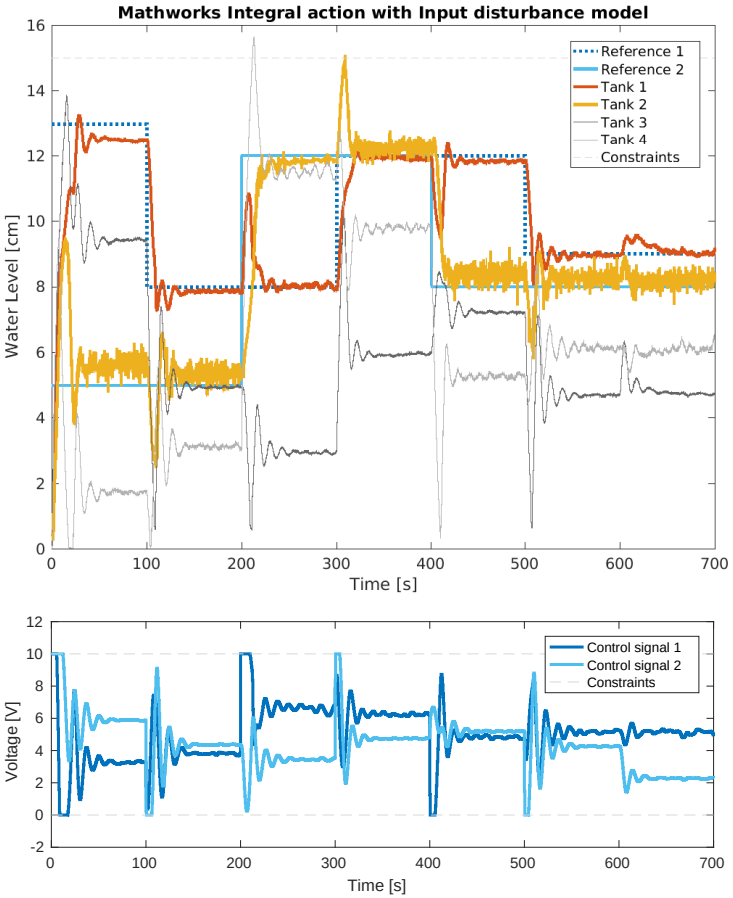

Figure 5.4: Running of the real process with a controller created with Mathworks toolbox. Upper plot showing water levels in Tanks 1-4 together with reference signals for Tank 1 and Tank 2. The lower plot shows control signals. This controller uses the model in (2.8)-(2.9) and an external Kalman filter.

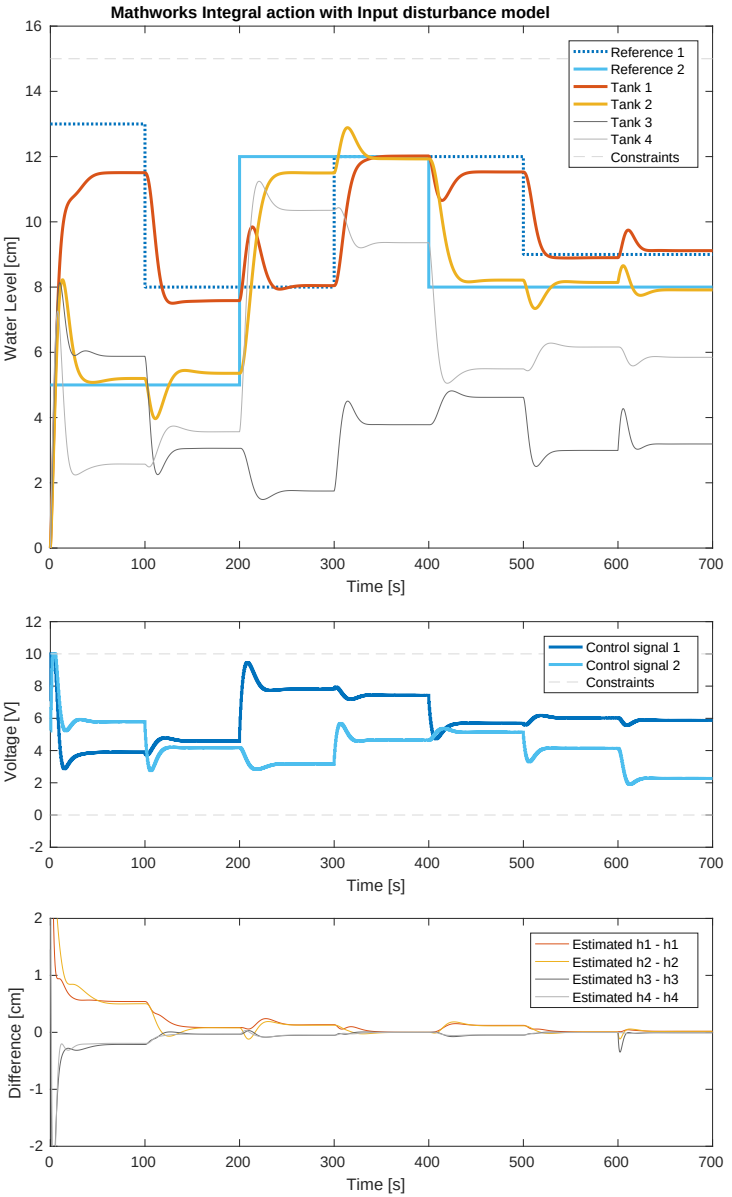

Figure 5.5: Simulation of a controller created with Mathworks toolbox. Upper plot showing water levels in Tanks 1-4 together with reference signals for Tank 1 and Tank 2. The middle plot shows control signals. The lower plot shows the difference between the toolbox's estimate of the water levels and the actual water levels. This controller uses the model in (2.8)-(2.9) and implements this with the toolbox's disturbance model.

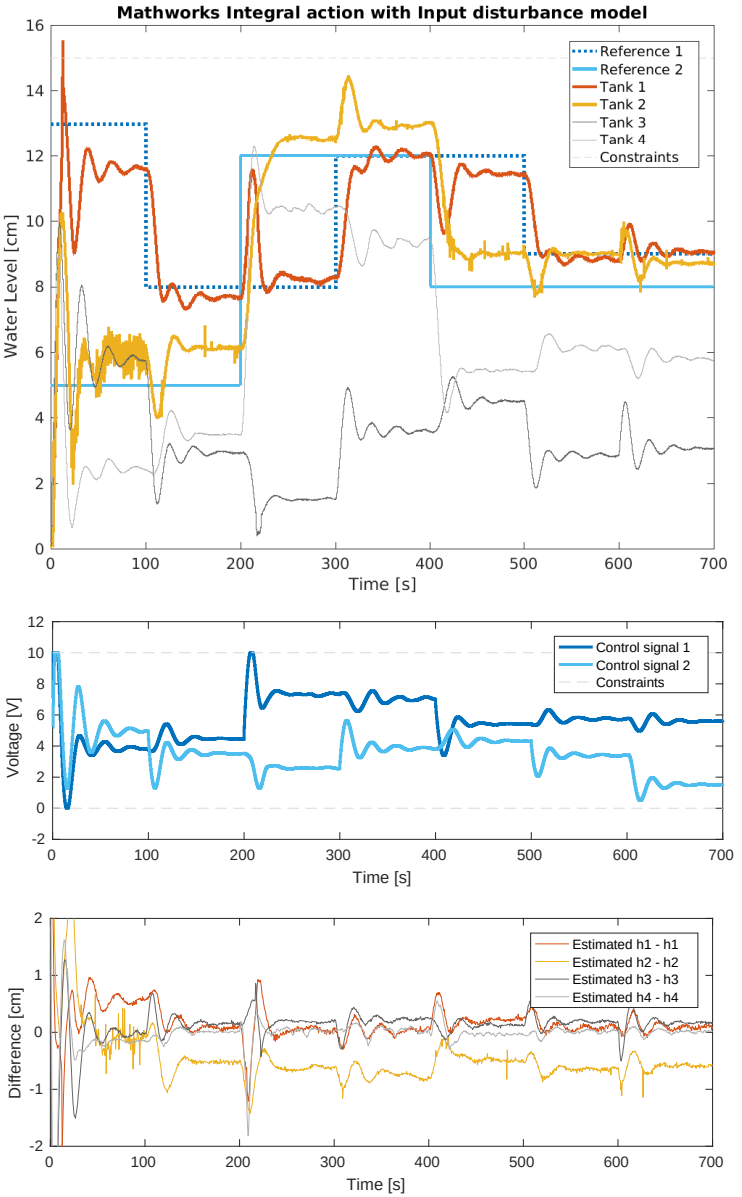

Figure 5.6: Running of the real process with a controller created with Mathworks toolbox. Upper plot showing water levels in Tanks 1-4 together with reference signals for Tank 1 and Tank 2. The middle plot shows control signals. The lower plot shows the difference between the toolbox's estimate of the water levels and the actual water levels. This controller uses the model in (2.8)-(2.9) and implements this with the toolbox's disturbance model.

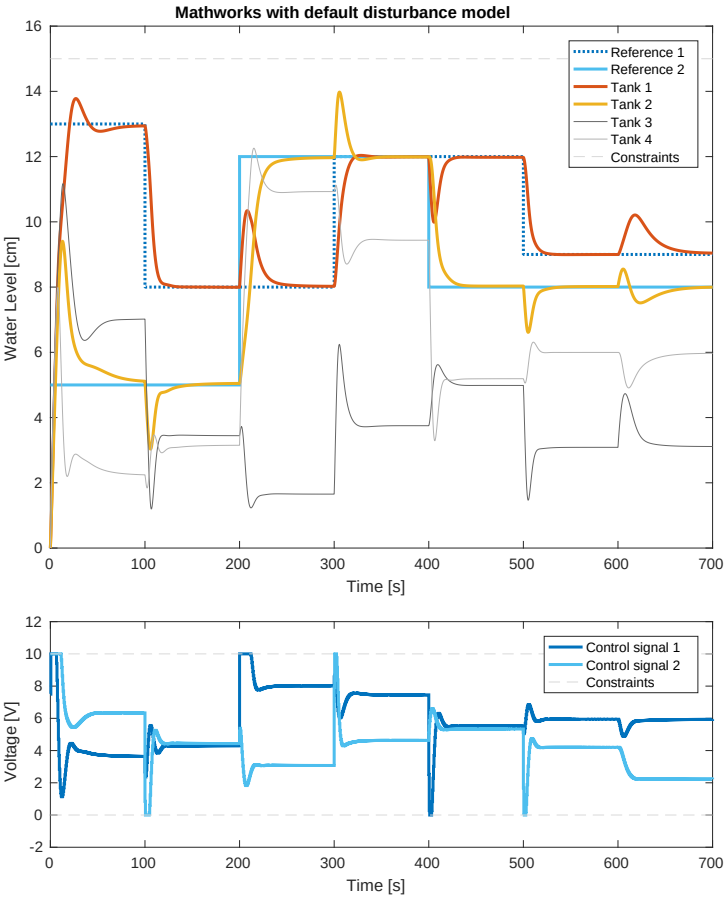

Figure 5.7: Simulation of a controller created with Mathworks toolbox. Upper plot showing water levels in Tanks 1-4 together with reference signals for Tank 1 and Tank 2. The lower plot shows control signals. This controller uses the toolbox's default settings for disturbance model.

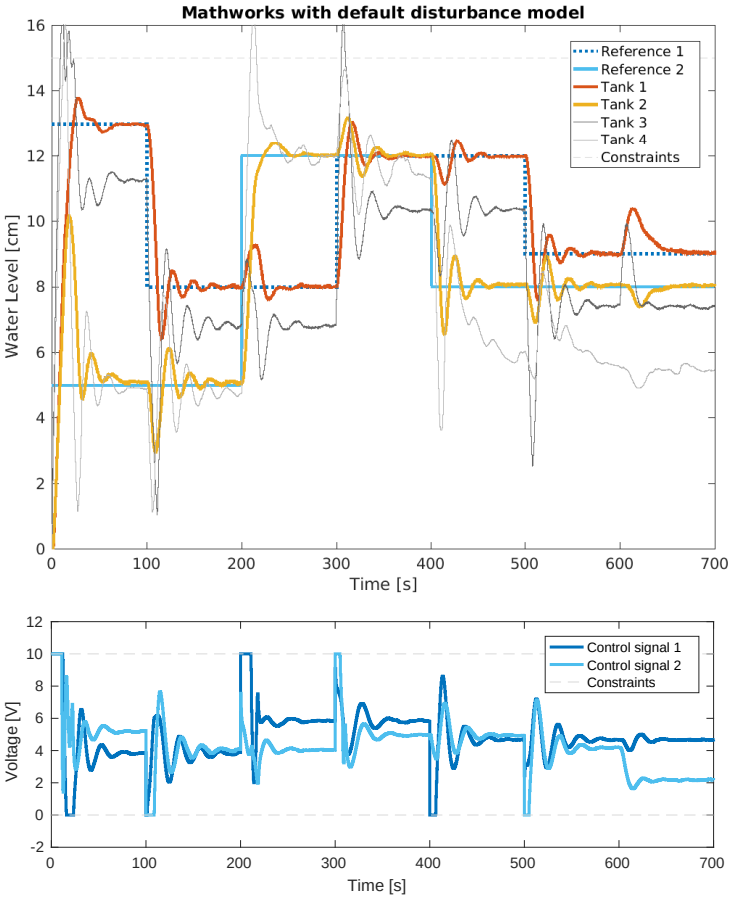

Figure 5.8: Running of the real process with a controller created with Mathworks toolbox. Upper plot showing water levels in Tanks 1-4 together with reference signals for Tank 1 and Tank 2. The lower plot shows control signals. This controller uses the toolbox's default settings for disturbance model.

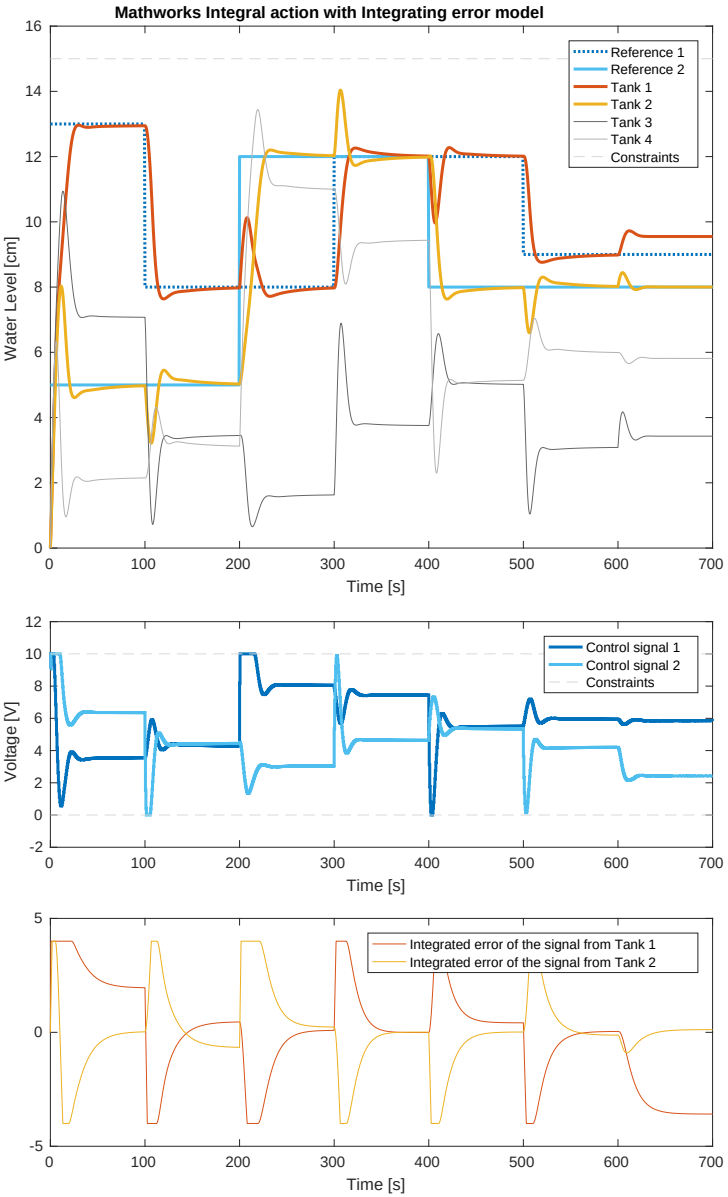

Figure 5.9: Simulation of a controller created with Mathworks toolbox. Upper plot showing water levels in Tanks 1-4 together with reference signals for Tank 1 and Tank 2. The middle plot shows control signals. The lower plot shows the integrated error. This controller uses the model in equations (2.10)-(2.11).

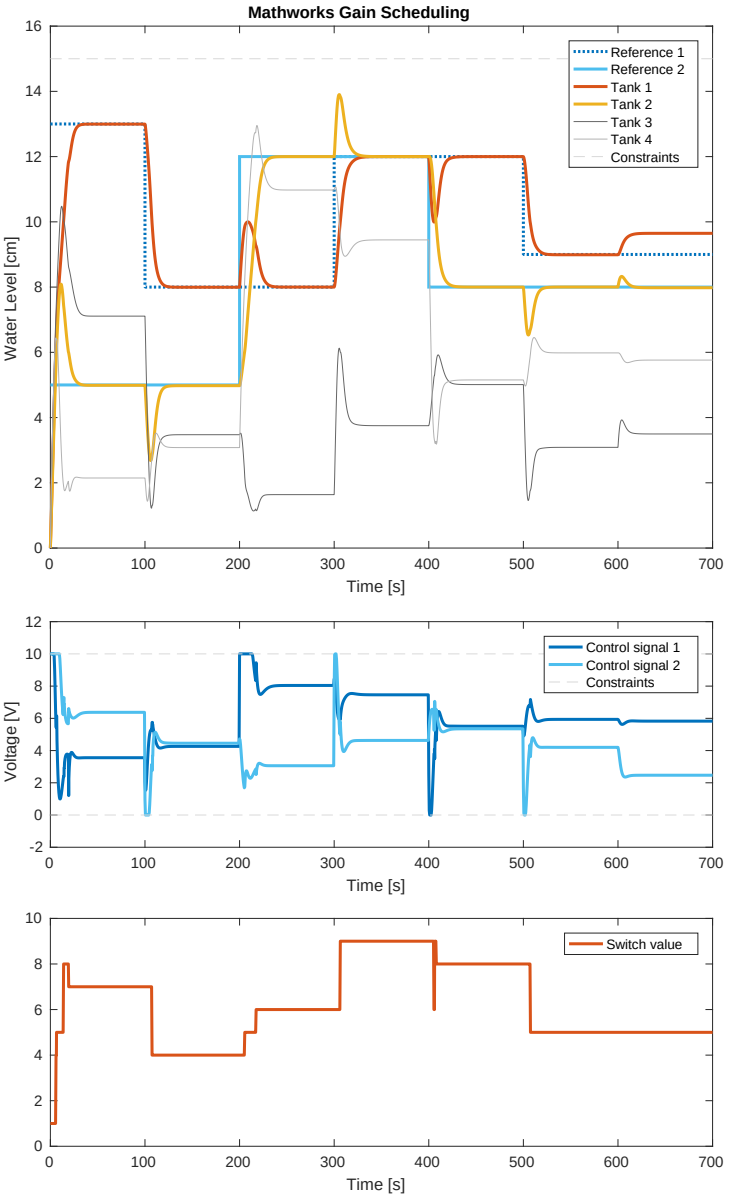

Figure 5.10: Simulation of a controller created with Mathworks toolbox. Upper plot showing water levels in Tanks 1-4 together with reference signals for Tank 1 and Tank 2. The middle plot shows control signals. The lower plot shows the switch value during the simulation. This controller uses gain scheduling and the MPC objects used in the controller are numbered 1-9.

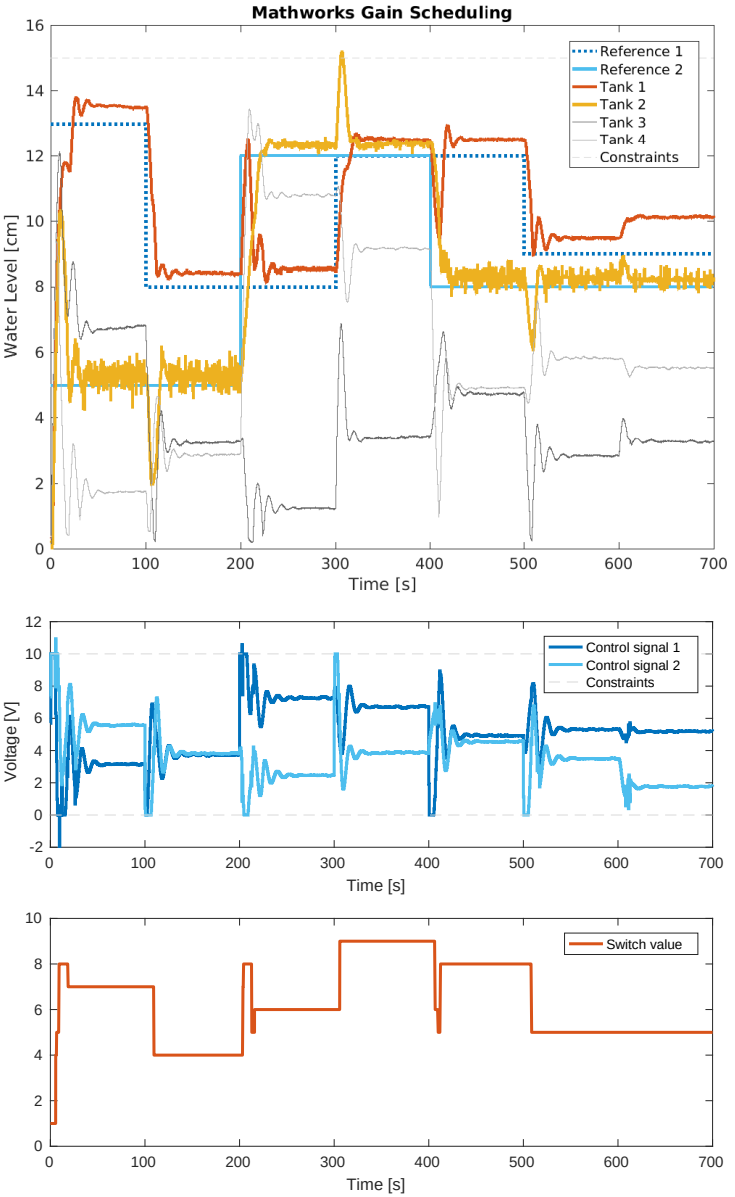

Figure 5.11: Running of the real process with a controller created with Mathworks toolbox. Upper plot showing water levels in Tanks 1-4 together with reference signals for Tank 1 and Tank 2. The middle plot shows control signals. The lower plot shows the switch value during the simulation. This controller uses gain scheduling and the MPC objects used in the controller are numbered 1-9.

5.2 Multi-Parametric Toolbox 3

In this section, the plots will show results from the MPT3 toolbox.

- Figure 5.12. Basic MPC, simulation.
- Figure 5.13. Basic MPC, real process.
- Figure 5.14. Integral action with Input disturbance model, external Kalman filter, real process.
- Figure 5.15. Integral action with toolbox's function, simulation.
- Figure 5.16. Integral action with Integrating error model, real process.
- Figure 5.17. Gain scheduling, simulation.
- Figure 5.18. Gain scheduling, real process.

The plots from running the simulation and the real process with a basic MPC, made with the MPT3 toolbox is shown in Figures 5.12 and 5.13. For this controller, no integral action or gain scheduling was implemented. The simulation follows the reference closely and has no constraints violations. The real process has no constraint violations either, settles rather quickly after reference changes, and has only a small stationary error. As expected, the input disturbance at 600 s is not counteracted, because the controller has no integral action.

In Figure 5.15, a plot of a simulation is shown where the controller implements integral action with the toolbox's function. Contrary to what might be expected of integral action, the water levels in Tank 1 and Tank 2 still have a stationary error a large amount of the time, but more importantly, the controller does not seem to counteract the input disturbance at all.

In the next two figures, integral action is used, first in Figure 5.14 with the input disturbance model, and then with integrated error model in Figure 5.16. Both controllers are running the real process and are clearly handling the input disturbance at 600 s, but the one with integrator error model in Figure 5.16 compensates for stationary errors more than the input disturbance model in Figure 5.14. While there is no constraint violation in Figure 5.16, there is one occasion where the water level in Tank 2 violates the constraint and goes above 15 cm at 300 s. When there were a constraint violation, the toolbox returned "NaN" (Not a Number) instead of a control signal.

Plots from running the controller with gain scheduling can be studied in Figures 5.17 and 5.18. Like the controller made in Mathworks toolbox, the simulation with gain scheduling in the MPT3 toolbox follows the reference well and have no constraints violations, but when running the real process there are some stationary errors and the water level in Tank 2 violates the constrains slightly at 300 s.

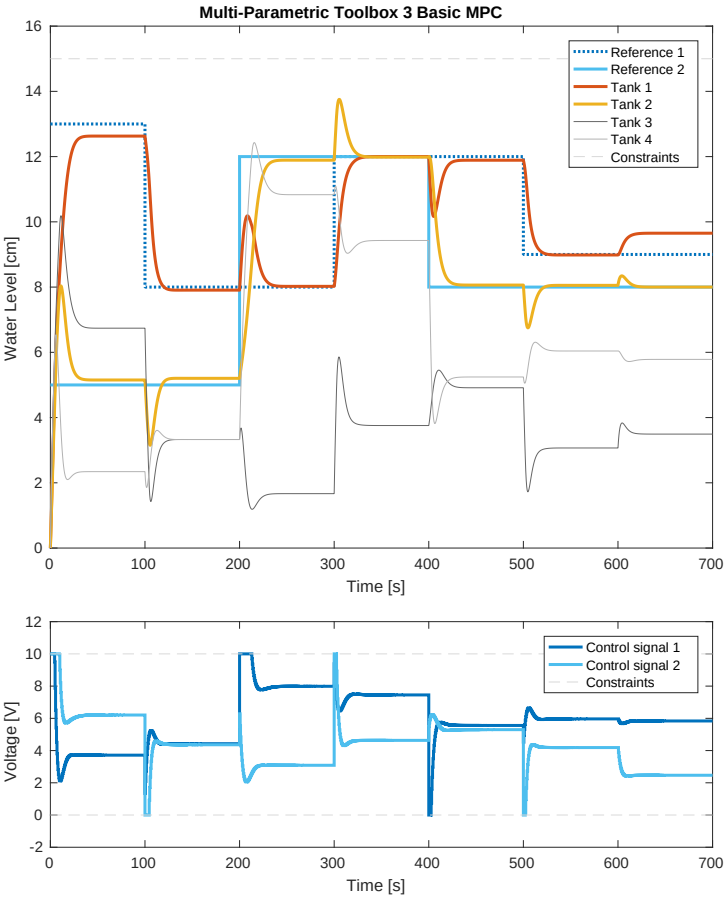

Figure 5.12: Simulation of a controller created with Multi-Parametric Toolbox 3. Upper plot showing water levels in Tanks 1-4 together with reference signals for Tank 1 and Tank 2. The lower plot shows control signals. This controller uses the model in (3.8)-(3.9) with no disturbance models or gain scheduling.

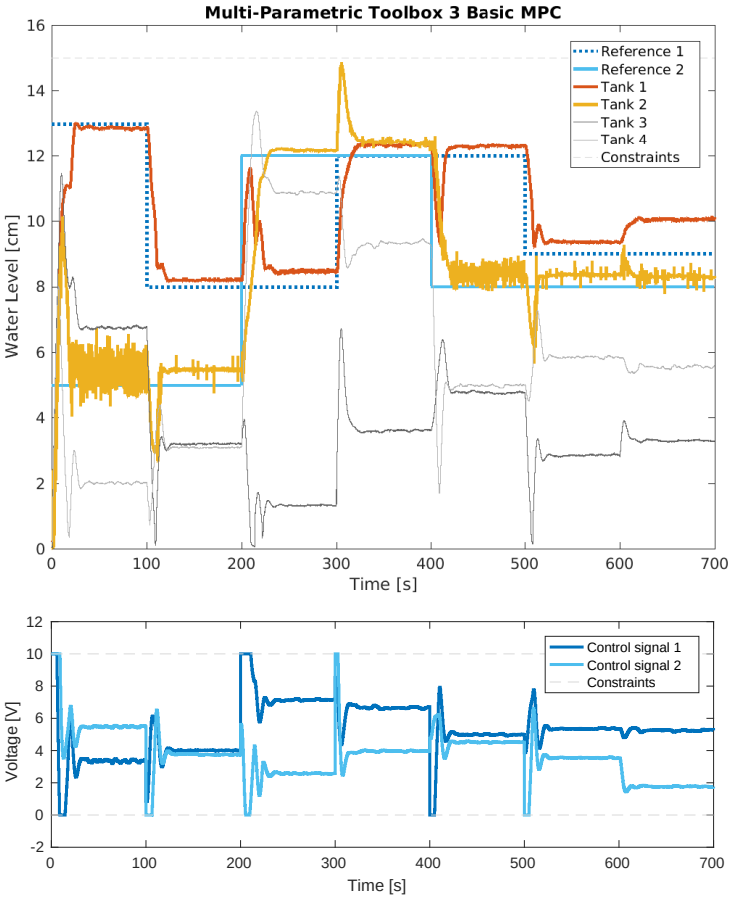

Figure 5.13: Running of the real process with a controller created with Multi-Parametric Toolbox 3. Upper plot showing water levels in Tanks 1-4 together with reference signals for Tank 1 and Tank 2. The lower plot shows control signals. This controller uses the model in (3.8)-(3.9) with no disturbance models or gain scheduling.

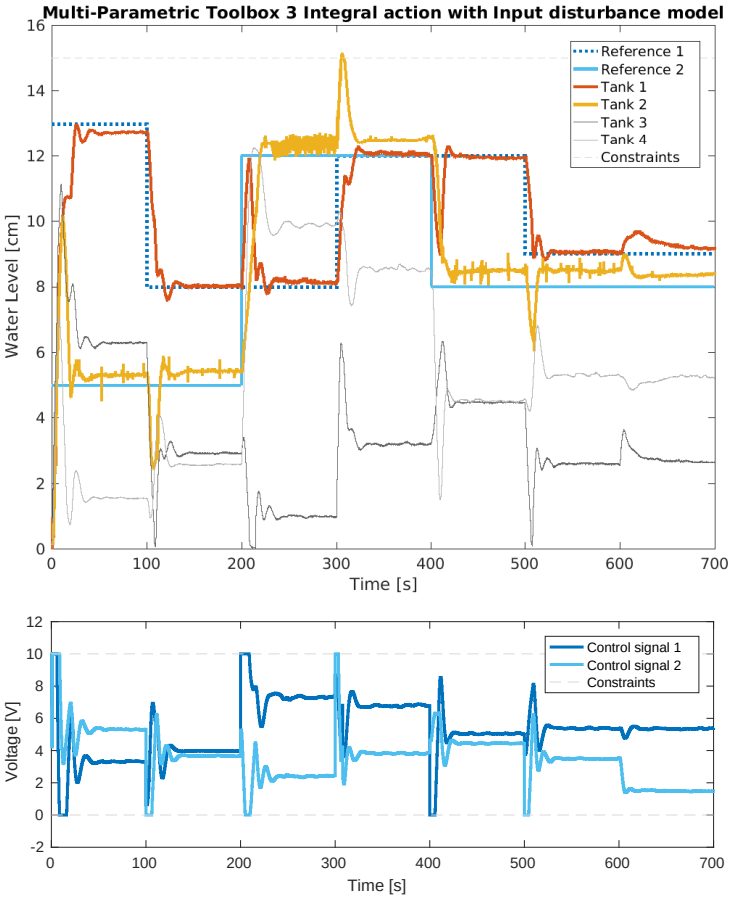

Figure 5.14: Running of the real process with a controller created with Multi-Parametric Toolbox 3. Upper plot showing water levels in Tanks 1-4 together with reference signals for Tank 1 and Tank 2. The lower plot shows control signals. This controller uses the model in (2.8)-(2.9) and an external Kalman filter.

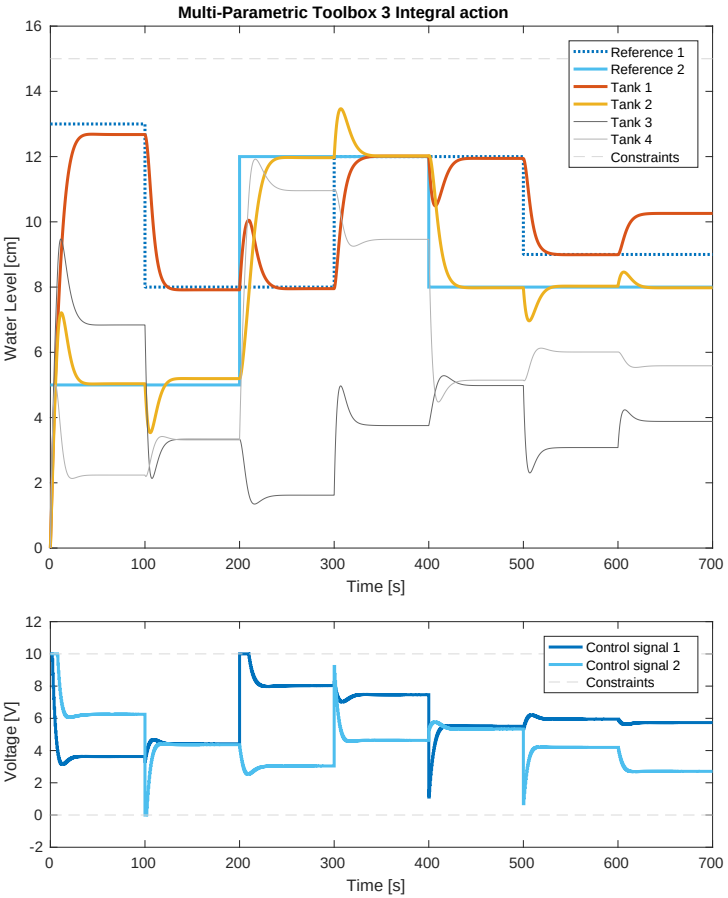

Figure 5.15: Simulation of a controller created with Multi-Parametric Toolbox 3. Upper plot showing water levels in Tanks 1-4 together with reference signals for Tank 1 and Tank 2. The lower plot shows control signals. This controller uses the toolbox's function for integral action.

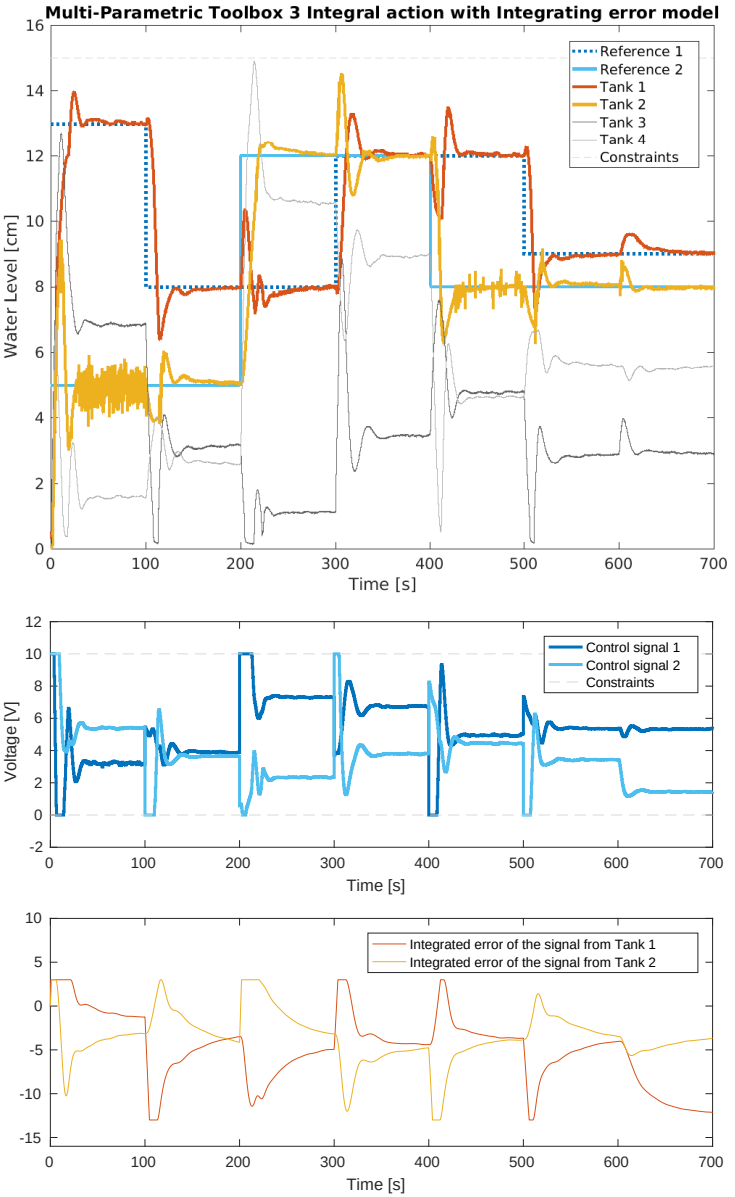

Figure 5.16: Running of the real process with a controller created with Multi-Parametric Toolbox 3. Upper plot showing water levels in Tanks 1-4 together with reference signals for Tank 1 and Tank 2. The middle plot shows control signals. The lower plot shows the integrated error. This controller uses the model in (2.10)-(2.11).

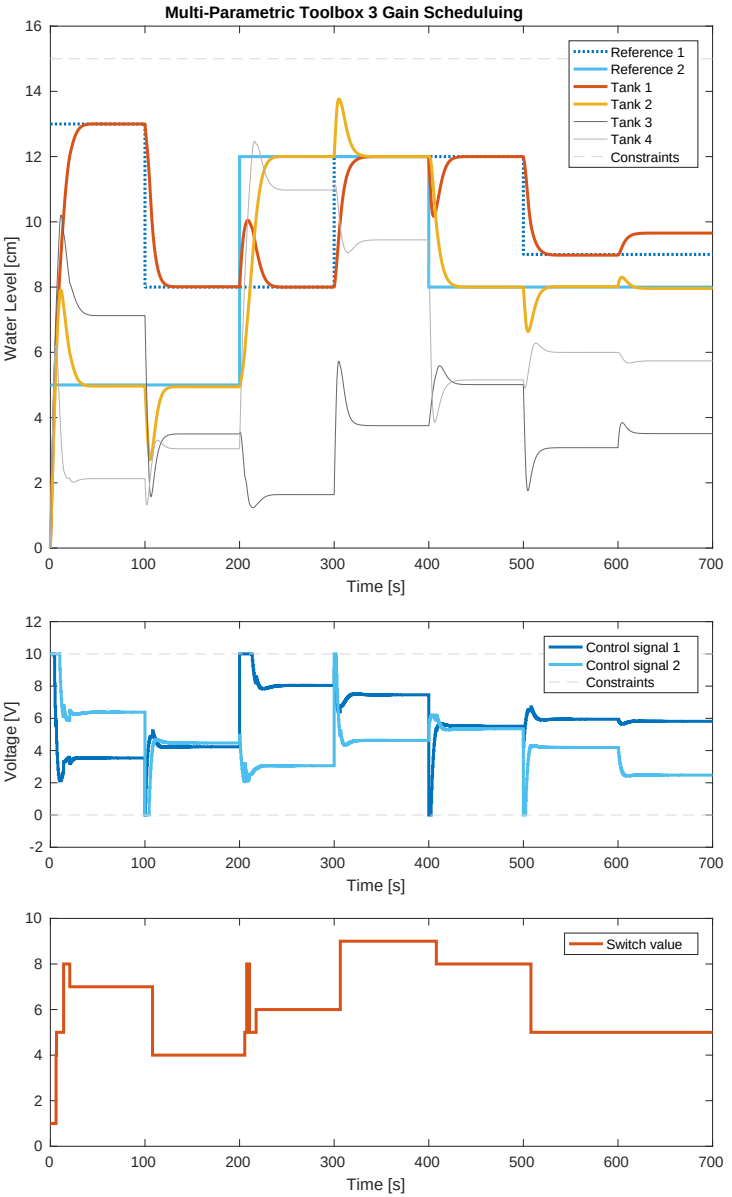

Figure 5.17: Simulation of a controller created with Multi-Parametric Toolbox 3. Upper plot showing water levels in Tanks 1-4 together with reference signals for Tank 1 and Tank 2. The middle plot shows control signals. The lower plot shows the switch value during the simulation. This controller uses gain scheduling and the MPC objects used in the controller are numbered 1-9.

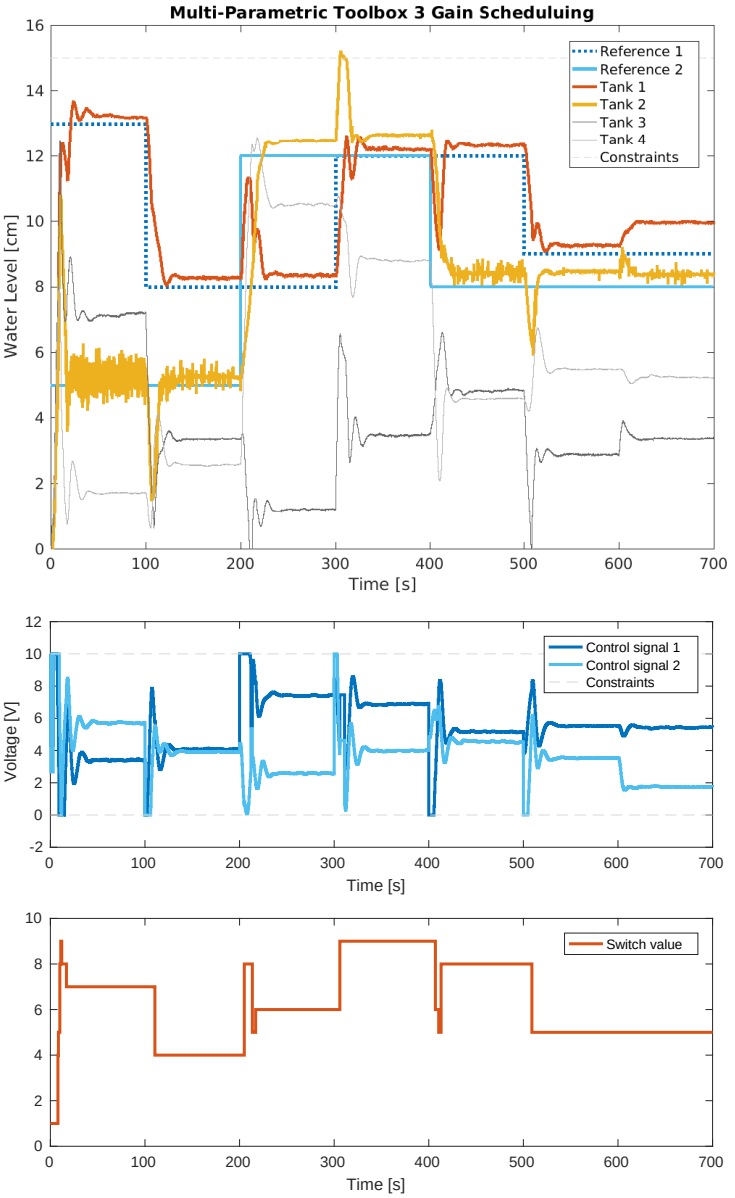

Figure 5.18: Running of the real process with a controller created with Multi-Parametric Toolbox 3. Upper plot showing water levels in Tanks 1-4 together with reference signals for Tank 1 and Tank 2. The middle plot shows control signals. The lower plot shows the switch value during the simulation. This controller uses gain scheduling and the MPC objects used in the controller are numbered 1-9.

5.3 MATMPC

In this section, the plots will show results from the MATMPC toolbox.

- Figure 5.19. Basic MPC, real process.
- Figure 5.20. Integral action with Input disturbance model, external Kalman filter, real process.
- Figure 5.21. Integral action with Integrating error model, real process.
- Figure 5.22. Gain scheduling, real process.

Figure 5.19 shows a plot of a controller made with the MATMPC toolbox, which is controlling the real process. This controller does not implement integral action or gain scheduling and uses the linearized model in (3.8)-(3.9). There are stationary errors with this controller and the constraints are violated several times. This constraint violation did not result in any warnings or actions from the toolbox, but while tuning the weigths for this controller, there were constraint violations where the water level almost reached 16 cm, which caused an error that shut down the process.

Figure 5.20 shows the results from running a controller with input disturbance model and an external Kalman filter. The controller removes the input disturbance at 600 s but there are still many times where there are stationary errors and the controller does not counteract them. Furthermore, the water level in Tank 2 violates the constraint at 300 s.

In Figure 5.21 the controller is run with the real process and the interating error model. The system is oscillating some after the reference changes but no constraints are violated. The saturation on the integrated error was set higher than in the other toolboxes because in this controller the integrated error otherwise reached the limit, which halts the integral action. The integral action in the controller is very slow.

Because the MATMPC toolbox is a toolbox for Nonlinear MPC, it does not have a function for gain scheduling. Instead, a controller using nonlinear MPC was created and the results from running the real process can be studied in Figure 5.22. There is a slight stationary error but not much oscillations. The input disturbance at 600 s is not counteracted but that was expected. One constraint violation appears at 300 s for the water level in Tank 2.

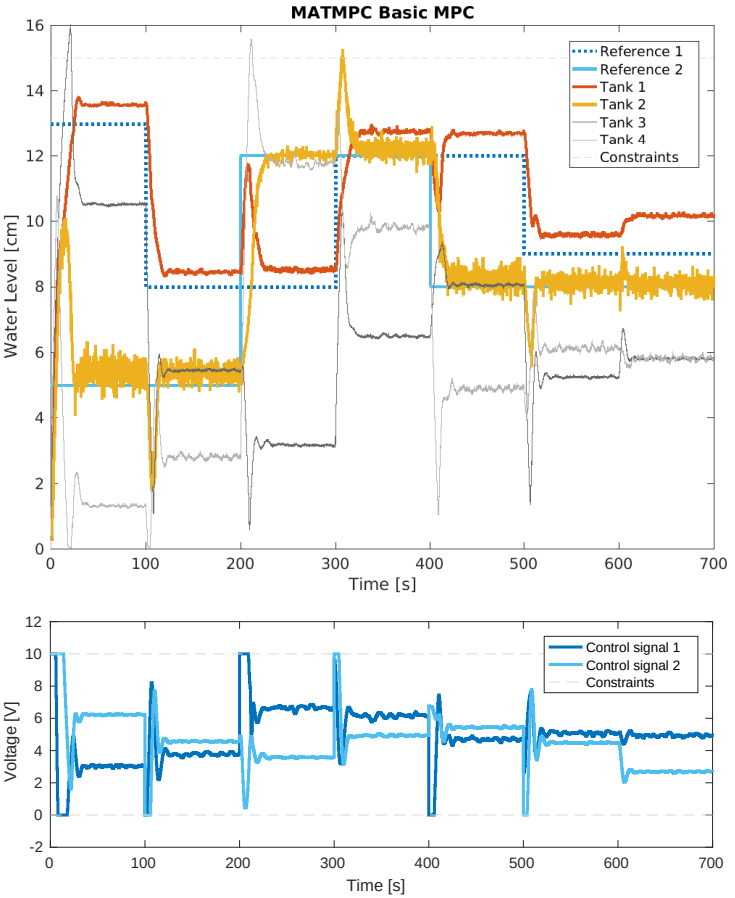

Figure 5.19: Running of the real process with a controller created with the MATMPC toolbox. Upper plot showing water levels in Tanks 1-4 together with reference signals for Tank 1 and Tank 2. The lower plot shows control signals. This controller uses the model in (3.8)-(3.9) with no disturbance models or gain scheduling.

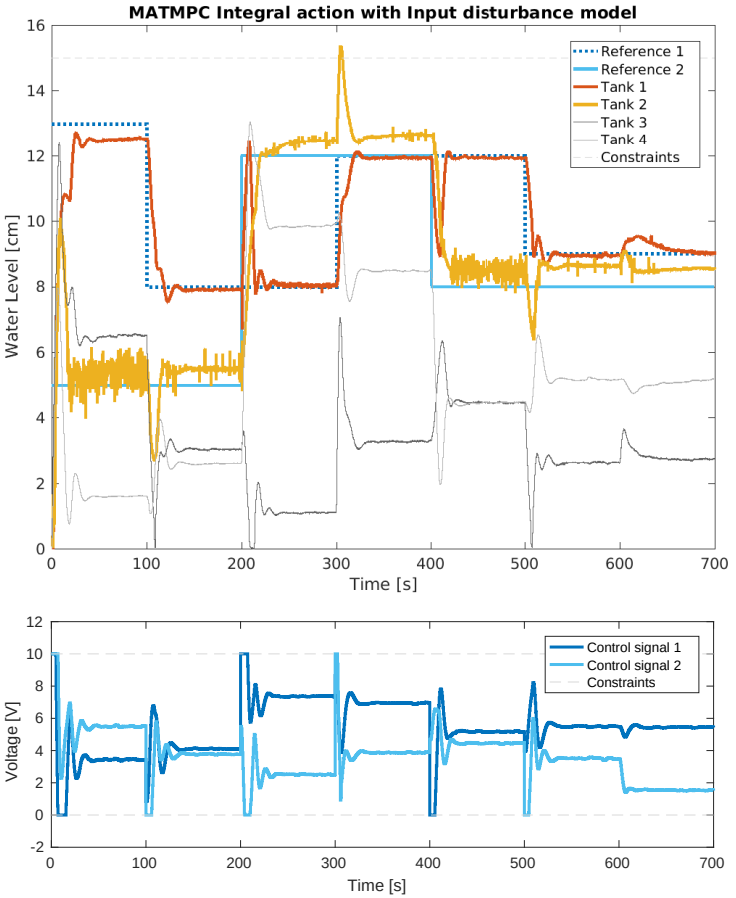

Figure 5.20: Running of the real process with a controller created with the MATMPC toolbox. Upper plot showing water levels in Tanks 1-4 together with reference signals for Tank 1 and Tank 2. The lower plot shows control signals. This controller uses the model in (2.8)-(2.9) and an external Kalman filter.

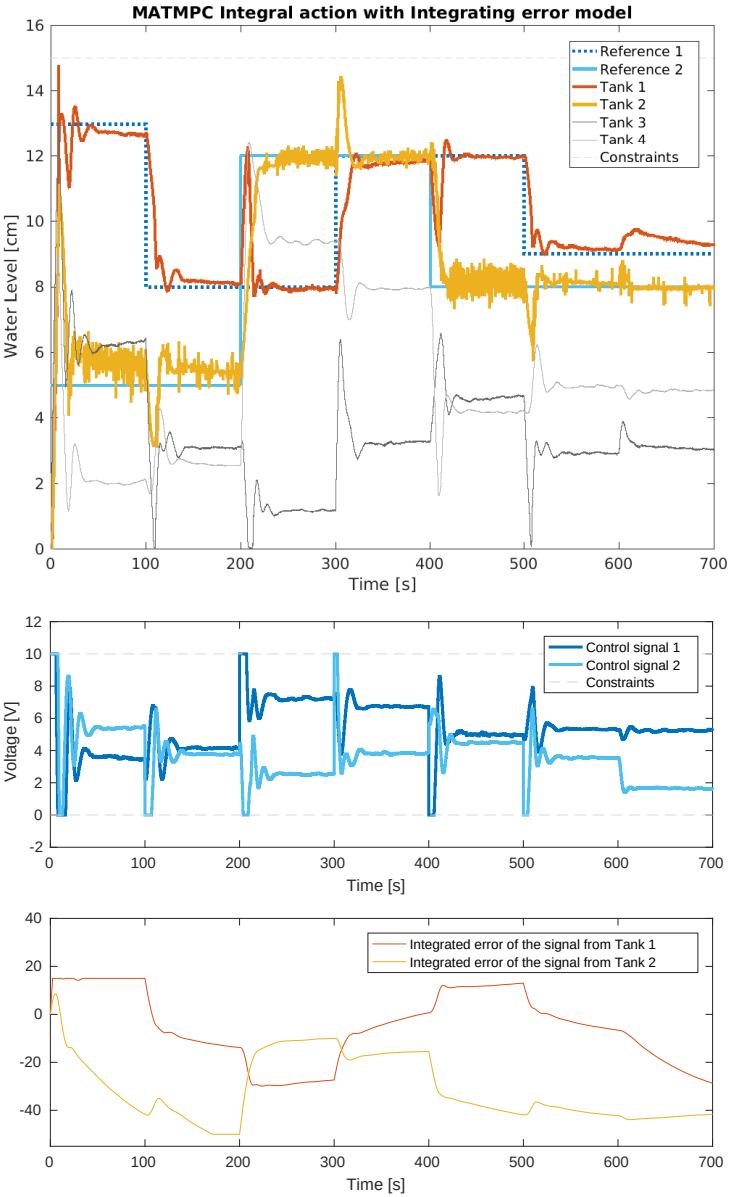

Figure 5.21: Running of the real process with a controller created with the MATMPC toolbox. Upper plot showing water levels in Tanks 1-4 together with reference signals for Tank 1 and Tank 2. The middle plot shows control signals. The lower plot shows the integrated error. This controller uses the model in (2.10)-(2.11).

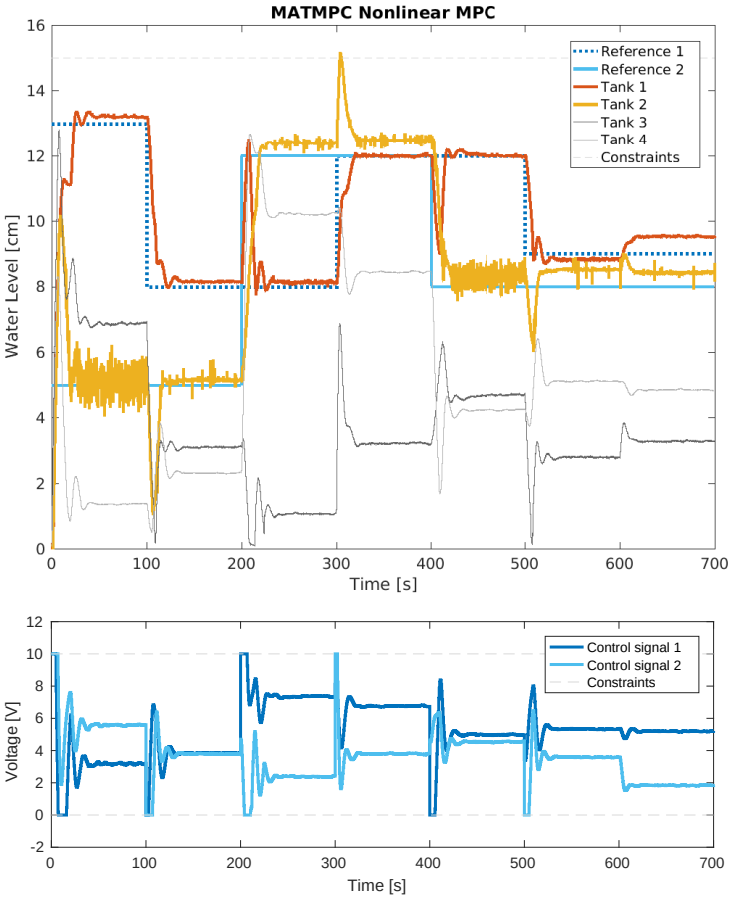

Figure 5.22: Running of the real process with a controller created with the MATMPC toolbox. Upper plot showing water levels in Tanks 1-4 together with reference signals for Tank 1 and Tank 2. The lower plot shows control signals. This controller uses Nonlinear MPC.

6

Discussion

This chapter will compare the three toolboxes and discuss their differences but also their similarities. To start the chapter is one section about each toolbox, mostly looking at the documentation but also addressing how well the toolboxes' notation correlates to the theory that is taught to the students at the department of automatic control at Lund University. These sections also include some discussion of the aspects that did not have an equivalent in the other toolboxes.

In theory, if set up the same way, all toolboxes should produce the exact same results, at least for a basic setup. The differences in the results in Chapter 5 show that MPC is a complex controller, both to design and implement and it is difficult to know whether the differences are because of wrongful usage, shortcomings in the documentation, or internal details of the implementation. In Section 6.4, the differences in the results will be discussed, as well as the different ways to implement integral action, gain scheduling how to configure weights and constraints in the different toolboxes.

6.1 Mathworks Model Predictive Control Toolbox

A first impression of Mathworks toolbox could be that it is a well-documented toolbox. On the website, there is a significant amount of text and examples. Much of it describes the graphical interface *MPC Designer*. A brief attempt was made to make a controller with this graphical interface but despite all documentation and example, when the author of this thesis had worked with it for a day, still not understanding how to enter a model, the conclusion was that it might not be very user friendly and perhaps not suitable for educational purposes.

Figure 4.3 shows how Mathworks define signals for an MPC design. While learning about MPC, these definitions never came up. It is not obvious how MV, OV, MO and other signals relate to the predictions in (2.2) or the cost function in (2.3). This could make it harder for students to see the connection between theory and practice.

From the code snippets for Mathworks toolbox in Section 4.2, it is easily observed that setting the properties for the controller is done in various ways. For example, setting weights have no similarities with how the constraints are set. As most of the documentation handles the graphical interface, it can be hard to find examples of how to set the properties with code. Finding an example of how it is done for a single-input, single-output system is also not very useful because of the variety in the code and no conclusions can be drawn on how to expand it to a multi-input, multi-output case.

The disturbance and noise models in the toolbox are rather complicated. Equations (4.3)-(4.4) show a simplified version, only including the input-disturbance model. Information on how to use these models are spread out to at least three different locations, not referencing to the others. Together with the fact that the input and output-disturbance models are set with a function, while noise models are set as a parameter makes it difficult to use. While having as default to add integrators to the output disturbance might be good for an inexperienced user wanting to set up a controller, it is not desirable for educational purposes. It can also be argued that it might be harder for an inexperienced user to troubleshoot if the need arises.

With all the effort the creators of this toolbox put into developing the disturbance models, it is surprising to find that when implementing gain scheduling, there is no function for defining in what areas the different MPC objects should operate, neither is there any suggestion in the documentation.

6.2 Multi-Parametric Toolbox 3

At first, when examining the website for the toolbox, there does not seem to be much documentation. After working with it for a while, the conclusion is that there is much information and mostly in the form of examples. Unfortunately, it is not well organised however, because there is not too much information, the readers can go through it all to find what they are looking for. While there are good examples of how to create the MPC object and set the properties, sometimes the documentation does not describe how things work. For example, this thesis describes in Section 4.2 that the user can add integral action with this toolbox, but the only information is that it extends the model with an integrator state. If something does not work, it is hard to troubleshoot with this lack of information.

While setting up the controllers, no information was found on how to set the control horizon. It could be assumed to be the same as the prediction horizon, which is set by the user, but there is nothing in the documentation that supports it.

In Section 4.1 about this toolbox, it is described how a model is created and then how all the properties are set as parameters. This makes it easy to read and easy to draw conclusions on how to expand from a single-input, single-output system to a multi-input, multi-output case. Another advantage of this toolbox is that all properties can be set on states or outputs. For the process used in this thesis, it

appeared natural to set constraints on the states. Even though it was possible to put them on the outputs, it seemed more like a workaround.

For the MPT3 toolbox, no Simulink block was available. This was solved by using an S-function. While setting up the S-function required some experience with Simulink, using it was simple. In the S-function, an MPC object was created and in every time step, the function `evaluate` was used to calculate the control signal.

In this toolbox, there were three different functions that did not work. When trying to use weights on the slew rate a compilation error occurred. When using the toolbox's function for integral action the result in Figure 5.15 shows that the input disturbance was not counteracted, and while trying to use an affine model, the MPC object did not return the expected control signal.

6.3 MATMPC

For the MATMPC toolbox, the documentation is practically nonexistent. There is a document stating the dimensions for some of the variables and parameters used, and some simple notation in the code. The only other information is with the installation instructions, where the user is prompted to look at the examples, which are only code, and change it to suit the system. Here it is also stated what files to run, and in what order. This was necessary because there were many files and all of them have plenty of code. So even with these short instructions, it was a challenge to understand where the user should do the changes and how.

As with the MPT3 toolbox, MATMPC does not state any information about the control horizon but it is probably the same as the prediction horizon.

6.4 Comparing the toolboxes

This section will start with a comparison of the results by looking at the plots. The best plot from the different toolboxes is compared in the categories Basic MPC, MPC with integral action, and MPC with gain scheduling or nonlinear MPC. Because the simulations in the different categories were very similar in the different toolboxes, the focus will be on the real process. Thereafter, the different ways of implementing integral action will be discussed, looking at the toolboxes' functions for integral action as well as the result of changing the models together with an external Kalman filter or with the integrator. Next follows a section about the implementation and results of gain scheduling and nonlinear MPC and then this section ends with a thorough discussion about how to configure weights and constraints in the different toolboxes.

Comparing the results

In simulations, the different implementations gave very similar results in all toolboxes. Comparing a simulation of basic MPC in Mathworks toolbox and MPT3

toolbox in Figures 5.1 and 5.12, it is hard to tell the difference. The same goes for gain scheduling in Figures 5.10 and 5.17. More interesting is to compare the control of the real process. Looking at the basic setup for MPC in Figures 5.2, 5.13 and 5.19, the MPT3 toolbox is less oscillative than Mathworks toolbox and have a smaller stationary error than the MATMPC toolbox. MPT3 is also the only one not violating the constraints. Integral action was implemented in a couple of different ways and not all toolboxes had the best result with the same type of implementation. Comparing the controllers with best results with integral action, which can be seen in Figures 5.4, 5.16 and 5.20, it can be observed that Mathworks toolbox still has a problem with oscillations and constraint violations while MATMPC can not seem to counteract the stationary errors. While the MPT3 toolbox has a bit of aggressive behavior, it does not violate the constraints and does counteract the stationary errors. Comparing gain scheduling in Figures 5.11 and 5.18 with the nonlinear MPC in MATMPC in Figure 5.22 it is hard to tell if one toolbox is better then the others. All three violate the constraints and all three have a larger stationary error than expected.

Integral action

Looking more closely at the different implementations of integral action, non of the toolboxes' functions gave a good result. While the MATMPC did not have any functions for integral action, the MPT3 toolbox had one and in Mathworks toolbox, the toolbox's disturbance models were used in two ways, the default setting that resulted in integral action and an input disturbance model. Worst was the result of the MPT3 model in Figure 5.15, where the input disturbance at 600 s was not counteracted at all, which suggests that the integral action is not working. In Mathworks toolbox, both implementations show that the input disturbance is counteracted, and simulations of the default controller in Figure 5.7 show a good result but looking at the real process in Figure 5.8, this controller is not well behaved and exhibits oscillations and the worst case of constraint violation in all the toolboxes. Studying the results from integral action with Mathworks input-disturbance model in Figure 5.6, it is clear that it is very oscillative and has a big stationary error. Compared with the other controllers this one needed a very high weight on the slew rate to counteract the worst of the oscillations, but obviously, it was not enough. Looking at the difference between the, by the controller, estimated states and the measured states there is one state where the difference is not centered around zero. Not even in the simulation in Figure 5.5, the estimation seems to work properly. One possibility for this is that the toolbox's Kalman filter does not work properly if there is no active output-disturbance model.

Two ways of implementing integral action, which was done for all three toolboxes in the same way were to model an input disturbance and to model an integrated error. For the input disturbance model, the results doesn't differ much between the toolboxes, which can be seen in Figures 5.4, 5.14 and 5.20. What is inter-

esting with this model is that it does not remove stationary errors caused by model errors. This is easily observed in the simulation in Figure 5.3 for Tank 1 between 100-200 s or between 400-500 s. Integrated error model did on the other hand remove all errors, which is clear in Figure 5.9, but this model had other drawbacks. Because the controllers with this model needed anti-windup, a saturation was put on the integrator. If the error was big enough, the integrator stopped working due to the saturation. This might be preventable with another setup for the anti-windup. Another interesting behavior with this model is that the results varied between the toolboxes. The MPT3 toolbox show quite good results in Figure 5.16 while the MATMPC controller performs quite poorly in Figure 5.21. Overall, it was not hard to implement these models with any of the toolboxes and they gave considerably better results than the toolboxes' own functions.

Gain Scheduling

Implementing gain scheduling gave a very good result in simulation. Comparing for example basic MPC controller and gain scheduling in Mathworks toolbox in Figures 5.1 and 5.10, the improvement is clearly visible. Looking at the results from running the real process on the other hand did not show the same improvements. The same comparison for the real process in Mathworks can be done with Figures 5.2 and 5.11 or with MPT3 in Figures 5.13 and 5.18. Here it is hard to tell if there has been an improvement. For the controller for nonlinear MPC in MATMPC in Figures 5.19 and 5.22, the same results can be observed. This is probably due to model errors. Model errors can actually cancel out some of the stationary errors in the basic controllers. An improvement in the controller might not then be visible in the results.

Only Mathworks toolbox offered a solution for gain scheduling, but with the S-function used for MPT3, it was just as easy. Unfortunately the PWA did not work in MPT3, because it had much potential. In MATMPC, implementing nonlinear MPC was not harder than implementing the linearized MPC. Considering that with nonlinear MPC, there was no need to linearize the model or convert the constraints, it was actually easier.

Weights and Constraints

The three toolboxes all have different ways to define weights. The way MPT3 defines weights, match the theory taught to the students at the department of automatic control at Lund University, with separate quadratic matrices for control signal and states or outputs. In Mathworks toolbox, the weights are defined as vectors, which in theory isn't wrong since the toolbox states that the weight matrices are diagonal with the squares of the weight vectors as the diagonal elements. So while it is not wrong, it might not help to strengthen the students' understanding of MPC. MATMPC also defines weights as a vector and contains the weights for both the states and for the control signal. This vector is then replicated so that these weights are set for every

point on the prediction horizon. This is also a correct but confusing way to set the weights. While tuning the controllers in Mathworks toolbox, weights on the slew rate turned out to be important. However, in the two other toolboxes, weights on the slew rate could not be set. Interestingly, MPT3 and MATMPC worked just fine without weights on the slew rate.

In Mathworks toolbox, constraints can only be put on the output signals. For this process, it would be more natural to put them on the states. It can be confusing, as a student, if the original problem needs to be reformulated. In Mathworks toolbox, the constraints were set to default with hard constraints on the control signals and soft constraints on the output signals, while in the MPT3 toolbox, all constraints were set to hard. In MATMPC on the other hand was no information about whether the constraints were hard or soft but some assumptions can be drawn. As the control signal never violates the constraints, these can be assumed to be hard. The constraints for the states on the other hand are violated in more than one result, without any actions from the toolbox, but only for small violations. While setting up the controllers in this toolbox the author noticed that the toolbox shut down the process if it was a major violation. In Mathworks toolbox, there is also no action when constraints are violated, probably because they are soft. In MPT3, the controller sends NaN (Not a Number) when a constraint is violated. This sends out a clear message and gives the user the possibility to set up a handler, to deal with constraint violations if necessary.

7

Conclusion

In this thesis, three Matlab toolboxes for MPC have been compared for the purpose of helping students understand the concept of MPC. To examine the toolboxes, three types of controllers were implemented, a basic MPC controller, an MPC controller with integral action, and an MPC controller with gain scheduling. The different elements that were evaluated were documentation and notation, implementation of integral action and gain scheduling, and performance.

Of the three toolboxes compared, MPT3 (Multi-Parametric Toolbox 3) was by far the toolbox in which the notation best correlates to the theory that the students at the department of automatic control at Lund University learn. Setting up the MPC object, the code was simple and consistent. Although most of the results were good, the plots show that the controllers from the MPT3 toolbox were a little bit better than the others. Another upside of this toolbox is that the user can choose to set the properties on the states or the outputs, this was for example useful when setting the constraints for this process. Even if the MPT3 toolbox does not have working functions for integral action and gain scheduling, it is not a major problem as the user can easily make their own solution. The really big drawback with this toolbox is that it has no Simulink block or even a description of how to use it in Simulink. An S-function is a good solution but hard for the inexperienced Simulink user.

For the Mathworks Model Predictive Control Toolbox there is a great amount of documentation, but despite that, or perhaps because of that, it can be hard finding the information needed. Setting up the MPC object, the code was not consistent and it corresponds poorly with the theory about MPC that the students know. The disturbance and noise models are not good for educational purposes. It makes it hard for the students to observe different behaviors of the MPC and besides being a complicated model, it is also the default when creating a controller and not obvious how to turn it off. An upside to this toolbox is the Simulink block for gain scheduling even though the users have to make their own switch, deciding when the different MPC object should be used.

The big drawback for the MATMPC toolbox was the nonexistent documentation. The big amount of code for the user to sort through did not help either. The notation does not correspond to the students' knowledge of MPC but because this

is actually a toolbox for nonlinear MPC, some part of it being hard to understand can be because of that. The positive side to this toolbox was that after figuring out how to create a controller for the linearized model, it was really easy to do it for the nonlinear model.

In conclution, the MPT3 toolbox overall gives good performance and also seems to be the simplest to explain and use from a pedagogical perspective.

Future work

While the students are familiar with Matlab, there are other programming languages that they also have worked with and future work could explore some of these. If a toolbox is intuitive enough, even a language that they have not worked with before could be used. Other things to consider for future work could be to investigate the best way to teach integral action in MPC. Perhaps compare the methods from this thesis to others and maybe the best way to implement them, which could include a more sophisticated way of anti-windup. Yet another thing to look into is a better method for the switch value in gain scheduling.

Bibliography

Chen, Y., M. Bruschetta, E. Picotti, and A. Beghi (2019). "Matmpc - a matlab based toolbox for real-time nonlinear model predictive control". In: *2019 18th European Control Conference (ECC)*, pp. 3365–3370. DOI: 10.23919/ECC.2019.8795788.

Fredlund, J. K. and K. S. Sulejmanovic (2017). *Autonomous driving using Model Predictive Control methods*. MA thesis. Department of Automatic Control, Lund University.

Glad, T. and L. Ljung (2003). *Reglerteori Flervariabla och Olinjära metoder*. Studentlitteratur.

Gros, S., M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl (2020). "From linear to nonlinear mpc: bridging the gap viathe real-time iteration". *International Journal of Control* **93**:1, pp. 62–80. DOI: https://doi.org/10.1080/00207179.2016.1222553.

Herceg, M., M. Kvasnica, C. Jones, and M. Morari (2013). "Multi-Parametric Toolbox 3.0". In: *Proc. of the European Control Conference*. http://control.ee.ethz.ch/~mpt. Zürich, Switzerland, pp. 502–510.

Holkar, K. S. and L. M. Waghmare (2010). "An overview of model predictive control". *International Journal of Control and Automation* **3**:4, pp. 47–64.

Johansson, K. (2000). "The quadruple-tank process: a multivariable laboratory process with an adjustable zero". *Control Systems Technology, IEEE Transactions on* **8**, pp. 456–465. DOI: 10.1109/87.845876.

Johansson, R. (2015). *Predictive and Adaptive Control*. Lund University.

Mathworks (2020a). *Linearize simulink models using mpc designer*. Last accessed 11 December 2020. URL: https://www.mathworks.com/help/mpc/gs/linearize-simulink-models-using-mpc-designer.html.

Mathworks (2020b). *Model predictive control toolbox*. Last accessed 11 December 2020. URL: https://www.mathworks.com/products/model-predictive-control.html.

Rossiter, J. A. (2003). *Model-based predicitve control: a practical approach*. CRC Press LLC.

Torrisi, G., S. Grammatico, R. S. Smith, and M. Morari (2016). "A variant to sequential quadratic programming for nonlinear model predictive control". In: *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 2814–2819. DOI: 10.1109/CDC.2016.7798688.

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER'S THESIS

Date of issue
March 2021

Document Number
TFRT-6124

Author(s)
Emma Nilsson

Supervisor
Olle Kjellqvist, Dept. of Automatic Control, Lund University, Sweden
Anton Cervin, Dept. of Automatic Control, Lund University, Sweden
Pontus Giselsson, Dept. of Automatic Control, Lund University, Sweden (examiner)

Title and subtitle
Practical comparison of MPC Toolboxes

Abstract

Model predictive control can be used to control a range of processes, from selfdriving cars to chemical plants. The education of the engineering students that in the future will design these controllers is an important matter. In this thesis, three Matlab toolboxes have been evaluated and compared from a student's perspective.

The toolboxes are Mathwork's own toolbox called Model Predictive Control Toolbox, Multi-Parametric Toolbox 3, and MATMPC. With a practical approach, three types of controllers have been created in each toolbox, a basic MPC controller, an MPC controller with integral action, and an MPC controller with gain scheduling. The documentation has been explored and the notation has been compared with the theory that is taught to the students. Different ways to implement integral action and gain scheduling have been evaluated and the controllers have been run with a real process to emulate a laboratory session and the results from the different toolboxes have been compared.

The notation in Multi-Parametric Toolbox 3 did best correspond to the students' knowledge about MPC and had in addition the best performance of the three toolboxes.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
1-68

Recipient's notes

Security classification

http://www.control.lth.se/publications/