Master's Thesis 2013

Candidate: Muhammad Mohsin

Title: Model Predictive control (MPC) with integral action; Reducing the control horizon and model free MPC.

**Telemark University College**

**Faculty of Technology**

M.Sc. Programme

**MASTER'S THESIS, COURSE CODE FMH606**

| | | |
|---|---|---|
| **Student:** | **Muhammad Mohsin** | |
| **Thesis title:** | **Model Predictive Control (MPC) with integral action: Reducing the control horizon and model free MPC.** | |
| **Signature:** | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | |
| **Number of pages:** | **146** | |
| **Keywords:** | **Model Predictive Control (MPC), Integral action, reducing control horizon, Model free MPC, DSR System identification algorithm, Four-tank process.** | |
| **Supervisor:** | **David Di Ruscio** | sign.: . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . |
| **Censor:** | | sign.:. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . |
| **External partner:** | **Prediktor, Olav Aaker** | sign.:. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . |
| **Availability:** | **Open** | |

**Archive approval** (supervisor signature): sign.: . . . . . . . . . . . . . . . . . . . . . . . . **Date** : . . . . . . . . . . . . .

**Abstract:**

Model Predictive Control (MPC) is the most widely used strategy in process industries due to remarkable features. It has the capability to control the non-minimum phase, unstable processes and handle the constraints in a systematic way. MPC with integral action is an effective method to achieve the offset free control which can remove the unknown slowly varying process and measurement noise respectively.

In this thesis, a multivariable four-tank process has been developed for simulation experiments and it is controlled at two operating conditions i.e. minimum and non-minimum phase setting. The mathematical models are constructed from the both physical and simulation data. Theoretical background of the state space model based MPC is described and the deviation variables are used to achieve the integral action in MPC. The proposed optimal controller has been implemented to control the level in lower tanks. The '*quadprog*' function and '*if-else'* technique are demonstrated to handle process constraints in MPC with integral action. The execution time for simulation is reduced using '*if-else'* method compared to '*quadprog*' function. The states are estimated by using the Kalman filter. A comparison in reducing control horizon in optimal control is also performed. The decentralized PI controller has been implemented to control the four-tank process and results are compared with MPC method.

Deterministic and Stochastic system identification and Realization 'DSR' algorithm has been proposed to formulate model free MPC. A linearized state space model is identified by the 'DSR' method and used in MPC algorithm. The proposed optimal control is more robust and faster than the traditional PI controller. Simulations are performed in MATLAB software.

# Table of Contents

## Preface

A master's thesis is a mandatory part for the completion of Master's degree in System and Control Engineering at Telemark University College (TUC) Porsgrunn, Norway. My research in this work presents Model Predictive Control (MPC) with integral action theory and implementation in four-tank simulation experiments. In order to complete this work, I used the knowledge learned in the Control Theory, Model Predictive Control, System identification and Optimal estimation during my master's studies.

The entire work was carried out at Telemark University College, Porsgrunn. The MATLAB software developed by Mathworks was used to perform simulations and m-files script written for this purpose are provided in the appendices. The schematic diagrams were drawn by using the Microsoft Visio 2010.

First of all, I would like to thank Almighty ALLAH for giving me the knowledge, skills and strength to achieve this goal.

I especially want to thank my supervisor, Associate Professor David Di Ruscio for his guidance and supervision during the last six months. He has given me unflinching encouragement and support that has been the greatest contribution of this work.

I would also like to thank Olav Aaker from Prediktor, Norway for his great interest in this project.

It is a great pleasure to express my special thanks to the faculty of technology at Telemark University College for providing an excellent learning environment and helping me in my ways during my stay in Norway.

My deepest gratitude goes to my wonderful parents, brothers and sisters for their unflagging love. They supported me financially and morally throughout my life. This dissertation was simply impossible without them. The words alone cannot express the thanks I owe to my family who supported me in all my pursuits.

Finally, I would like to sincerely thank my friends who love and care for me and especially my classmates for their valuable discussions and suggestions to achieve this goal.

Porsgrunn, 04 June 2013
Muhammad Mohsin

## Nomenclature

The symbols, subscripts and abbreviations used in the thesis are listed.

### Symbols

| | |
|---|---|
| $a_i$ | Cross-section area of the outlet hole in Tank $i$ |
| arg min | Minimizing argument |
| $A_i$ | Cross-section area of the tank $i$ |
| $(A, B, D, E)$ | State space model matrices |
| $(\tilde{A}, \tilde{B}, \tilde{D})$ | Augmented model matrices |
| $g$ | The acceleration of gravity |
| $G_-(s)$ | Transfer function of minimum phase process |
| $G_+(s)$ | Transfer function of non-minimum phase process |
| $h_i$ | The water level in the tank $i$ |
| $H$ | Hessian matrix |
| $H_L^d$ | Toeplitz matrix |
| I | Identity matrix |
| $J_k$ | Cost/ Objection function |
| $k_c$ | The pump gain |
| $K$ | Kalman Filter gain |
| $\tilde{K}$ | Kalman filter gain in innovation form |
| $K_p$ | Proportional gain in PI control |
| $L$ | Prediction horizon |
| $Lu$ | Control horizon |
| $n$ | System order |
| $O_L$ | Observability matrix |
| $P$ | Symmetric and positive semi-definite weighting matrix |

|  |  |
|---|---|
| $q_{in}$ | Volumetric flow rate into the tank |
| $q_{out}$ | Volumetric flow rate out of the tank |
| $Q$ | Symmetric and positive semi-definite weighting matrix |
| $r_{k+1\|L}$ | Specified reference signal vector |
| $R$ | Symmetric and positive semi-definite weighting matrix |
| $T$ | Time constant |
| $T_i$ | Integral time in PI control |
| $u_i$ | The voltage applied to the pump $i$ |
| $u_k$ | Actual control signal |
| $u^*_{k\|L}$ | Optimal future control signal vector |
| $\Delta u^*_{k\|L}$ | Optimal future deviation control signal vector |
| $U$ | Input data matrix |
| $v$ | Unknown slowly varying process disturbance |
| $w$ | Unknown slowly varying measurement disturbance |
| $Y$ | Output data matrix |
| $y_{k+1\|L}$ | Process output vector |
| $\gamma$ | Valve constant |
| $\Lambda$ | RGA matrix |

## Subscripts

|  |  |
|---|---|
| $i$ | Index $i = 1, 2, 3, 4, .......$ |
| $k$ | Discrete time |
| $k+1$ | Next sampling time |

## Abbreviations

| | |
|---|---|
| CARIMA | Controller Auto Regressive Integrated Moving  Average |
| CARMA | Controller Auto Regressive Moving Average |
| CV | Control Variables |
| DARMA | Deterministic Auto Regressive Moving Average |
| DMC | Dynamic Matrix Control |
| DSR | Deterministic and Stochastic system identification and Realization |
| EMPC | Extended Model Predictive Control |
| FCCU | Fractionator  Column of Fluid Catalytic Cracking Unit |
| GPC | Generalized Predictive Control |
| IMC | Internal Model Control |
| LQ | Linear Quadratic |
| LQR | Linear Quadratic Regulator |
| MAC | Model Algorithmic Control |
| MATLAB | Matrix Laboratory |
| MIMO | Multiple Input and Multiple Output |
| MPC | Model Predictive Control |
| MV | Manipulated Variables |
| PID | Proportional Integral Differential |
| PM | Prediction model |
| prbs | pseudo-random binary sequence |
| PVC | Poly-Vinyl Chloride |
| quadprog | Quadratic programming |
| RGA | Relative Gain Array |
| SCE | Systems and Control Engineering |
| SISO | Single Input and Single Output |
| SQP | Sequential Quadratic Programming |
| TUC | Telemark University College |
| UPC | Unified  Predictive  Control |

## Overview of Tables and Figures

A list of tables and figures are included in this section.

## List of Tables

## List of Figures

# 1. Introduction

In this chapter, the overview of the thesis topic is introduced with Model Predictive Control (MPC) background. The objective of the thesis and the main required tasks for successful completion of the work are listed. The outline of the thesis is also provided in this last part.

## 1.1 Overview

Classical proportional-integral-derivative (PID) Controllers have been used in the industries for decades and playing an important role to fulfill the operational demands of the industries. They ruled the process industries and the most widely used strategy due to their simple structure [1]. Due to the environmental regulations and fast changing economic market, industries were looking for an optimal controller that can increase the productivity of goods and reduce the operating cost. These requirements lead to the development of the Model Predictive Control which is an advanced control strategy that meet the requirement of the process industries [2]. MPC is the most widely used controller at present due to its ability to handle multivariable process and constraints in a simple way [3]. The ideas for developing predictive started since 1960's and first successful implementation of MPC reported by Richalet et al [4].

MPC belongs to a class of optimal control that uses a process model to compute future predicted outputs. These predicted outputs are then used to calculate a sequence of control inputs that are sent to the system for optimizing the plant future behavior [5]. An MPC algorithm consists of the cost function, constraints and a prediction model. The cost function measured the difference between the future output and specified reference and also find the control signals. The constraints are limitation of the process. The constraints for the MPC are an input amplitude constraint, input rate constraint and output constraint.

The prediction model is constructed from the process model that describe the relationship between the future outputs and control inputs. The difference between the predicted and classical controller is the use of the model. The process model can be finite impulse/step response model, state space models, or transfer function models [5]. The MPC can be classified into linear and nonlinear model predictive control based on the model used to construct the prediction model. Using a linear model lead to formulate a linear MPC and on the other hand, a nonlinear model resulted in nonlinear MPC.

There are several methods within MPC algorithms that differ from each other based on the process model used [6]. However, these algorithms have some problems with offset i.e. the process output is not equal to the specified reference at steady state. This offset problem can be solved by introducing the integral action. There are different methods to obtain integral action in MPC. In this thesis, it is achieved by using the deviation variables such that the output from the process is equal to the reference in steady state [7].

MPC with integral action method is formulated and implemented in benchmark process. Four-tank process is used as benchmark process, and simulation experiment is performed. System identification is a method to construct a mathematical model of the process based on the process input-output known data. System identification algorithm as Deterministic and Stochastic system identification and Realization (DSR) is used to identify the state space model of the four-tank process and then used it in the MPC with the integral action algorithm. The data for the four-tank process is generated by the simulation. The model formulation from the known input-output data and then using this model in MPC method is defined as model free MPC algorithm. This method is useful when the process model is not formulated by first principle.

## 1.2 Objective

The master thesis is a mandatory part of the master's degree in Systems and Control Engineering (SCE) at Telemark University College (TUC). The objective of this work is to give a theoretical description of model predictive control with integral action as well as perform simulation experiment on benchmark process. The well known nonlinear four-tank process is used for this purpose. The MATLAB software is used for the simulation experiments and the main tasks of the thesis are listed as,

- A short overview of the state space model based Model Predictive Control (MPC).
- Overview of different methods to achieve integral action in MPC algorithm.
- Theoretical description of MPC optimal controller with integral action method.
- Performance simulation experiments of MPC with integral action on the four-tank benchmark process.
- Performance comparison in reducing control horizon in MPC method.
- Identify the linearized state space model by using a system identification algorithm as DSR and use the identified model in the MPC method.

## 1.3 Thesis outline

The title of the thesis is “Model Predictive Control (MPC) with integral action: Reducing the control horizon and model free MPC”. The work is divided into different chapters and tasks are completed in a sequential way. In this section, a short overview of all the chapters is included.

In chapter 1, the overview of the thesis topic is introduced with Model Predictive Control (MPC) background. The objective of the thesis and the main required tasks for successful completion of the work are listed. The outline of the thesis is also provided in this last part.

In chapter 2, the basic ideas about the model predictive control is introduced. First of all, theory behind MPC is given and then the structure of the controller is summarized which consists of the cost function, constraints and prediction model. Finally, the working principle of the MPC is explained.

In chapter 3, state space model based predictive control is presented and different methods of achieving integral actions in the MPC optimal control algorithm are discussed. The formulation of MPC with integral action is described that will be used for simulation experiments of the four-tank process. Finally, the Kalman Filter algorithm steps are explained.

In chapter 4, the four-tank benchmark process is described and its physical model is formulated by writing down mathematic equations using the basic laws of physics as presented by many researchers. The nonlinear model is linearized for use in MPC with the integral action algorithm. The operating condition of the four-tank process as minimum and non-minimum phase are discussed at the end.

The chapter 5 is one of the main chapters, simulation experiments of MPC method on benchmark process is discussed. The parameter values for both operating conditions are taken from the literature. First of all, stability, observability and controllability of the linearized model of the four-tank process is analyzed. Constrained and unconstrained MPC with integral algorithm is implemented in the linearized model of four-tank minimum and non-minimum phase process. Constrained MPC is further explored with different constraints handling technique i.e. ‘*quadprog*’ function and ‘*if-else’* method. A decentralized PI controller is implemented to control the four-tank process for comparing the results with the MPC optimal controller. In the last part, simulation using different values of the control horizon in the MPC algorithm is performed. The experimental results for all the simulations are plotted and compared with each other.

In chapter 6, the system identification method is used to identify the model of the process and formulate a model free MPC algorithm. First of all, short overview of system identification algorithm as Deterministic and Stochastic system identification and Realization '*DSR*' is introduced. The input-output data of the four-tank process is generated by simulation for both minimum and non-minimum phase setting. The collected data is used in the '*DSR*' algorithm to construct a linearized state space model. The identified model is validated and then used in MPC with integral action to control the four-tank process.

The brief summary of all the simulation results obtained in this work is outlined in chapter 7, and some recommendations for the future work are given in the end.

The conclusions are pointed out in chapter 8.

# 2. Model Predictive Control

In this chapter, the basic ideas about the model predictive control are introduced. First of all, theory behind MPC is given and then the structure of the controller is summarized which consists of the cost function, constraints and prediction model. Finally, the working principle of the MPC is explained.

## 2.1 Introduction

Model predictive control belongs to a class of optimal control and the most commonly used technique in process industries. Model predictive control has a history of more than five decades and it is one of the challenging fields both in industrial and academic sectors. Several publications associated with MPC methodology provide a good introduction to practical issues. Design formulation, ability to handle constraints, online process optimization and simplicity of the design are the major aspects of model predictive control that make it attractive to practitioners and researchers [3].

## 2.2 Theory behind MPC

The conventional proportional-integral-differential (PID) controllers ruled the process industries for decades. Today's advance computing technology allows implementing more advanced control algorithms, but the most of the practitioner's preferred method is to design the robust and transparent process control structure which uses simple controller. This is the reason why the PID controllers are mostly used in the industry although many other sophisticated control algorithms have been developed, however, this strategy of control structure cases some limitation in process performance [3]. In the advance computing technology, the industry was looking for optimal control strategy. This demand leads to the development of model predictive control which is an effective optimal control strategy that fulfills the control requirement of process industries. MPC is an optimal model based control algorithm and it is regarded as the most advanced technique among all the control algorithms present today [2].

The development of modern control concepts has been started from the work of Kalman with the linear quadratic regulator (LQR) designed to minimize a quadratic cost function of states and inputs. The reason why the LQR was not the best choice for process industries because the nonlinearities of the real systems and the absence of constraint in its formulation and at the same time not much handy for instrument technicians and control engineers as mentioned by Nunes [5].

The ideas for developing model predictive control which is a special case of the optimal control theory have been started since 1960's according to *Garcia et al* [8]. The successful implementation of model predictive control in the industry reported by the researcher in late 1970's [5]. Particularly the one by Richalet et al. [4] presenting Model Predictive Heuristic Control (MPHC) which later known as Model Algorithmic Control (MAC [9]). It was implemented on a main fractionator column of Fluid Catalytic Cracking Unit (FCCU) in poly-Vinyl Chloride (PVC) plant, in the late seventies [2].

The theory of predictive control has been developed almost in all feature such as stability, nonlinearity and robustness [3]. Within the framework of predictive control, there are many different ways to design a predictive controller. There are different predictive controllers, each with different properties such as a Generalized Predictive Control (GPC, [10-12]), Dynamic Matrix Control (DMC, [13]), Unified Predictive Control (UPC, [14]), Internal Model Control (IMC, [8]) and Extended model based predictive control (EMPC, [15]) etc. After the successful implementation of model predictive control (MPC), it has become the most popular control strategy for process industries. The applications for MPC have now extended from petrochemicals and refining fields to food processing, automotive, metallurgy, aerospace and defense industries according to an industrial survey presented by Qin et al. [6].

MPC became the standard control strategy due to its constraints handling abilities and numerous advantages over traditional controllers. It is suitable for multivariable control designs where the interaction between manipulated variables (MV) and control variables (CV) is taken into consideration. It also has the ability to manage long time delay and non-minimum phase problem as discussed by Eng et al.[2].

## 2.3 Structure of Model Predictive Control

A Model predictive control algorithm consists of a cost function, constraints and a prediction model or model of the process [16]. It is a computer control algorithm that uses a model of the process which generally represents the complex behaviors of the dynamic systems. This model is used to predict the system's future response over a future time interval or normally known as the prediction horizon [2]. The future response of the system based on the current and past values of system output and on the future control actions. This information is used to calculate the optimal control signals for future actions [17]. Then the main concept of the optimization is to compute a vector of control inputs to be fed into the system in an optimal way and at the same time process constraints taken into concern [16]. The structure of the MPC is illustrated in the diagram shown in Figure 2.1 and below its main components are described.

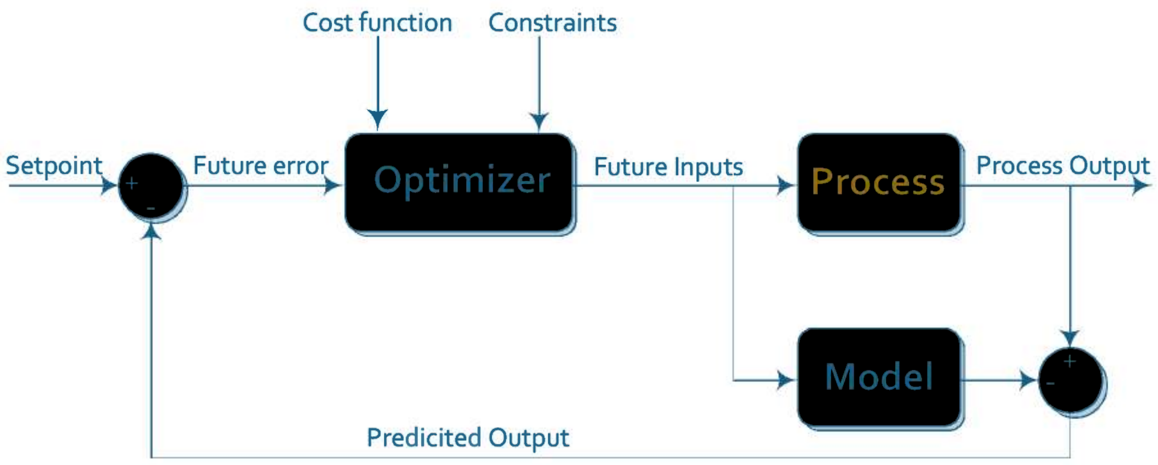


Figure 2.1: Structure of MPC [17].

### 2.3.1 Cost function

Cost function also called as objective or optimization function which is denoted by $J_k$ in this thesis. It is a scalar criterion that measures the difference between the future outputs $y_{k+1|L}$ and some specified future reference $r_{k+1|L}$ and at the same time finds the control signal $u_k$. This cost function is the measure of process behavior over the prediction horizon such that it is minimized with respect to the future control vector $u_{k+1|L}$ and only the first input is used. At the next time instant, $k := k+1$ the optimization process is repeated again which is known as a receding control horizon problem and mathematical derivations in this chapter are referenced from Ruscio [15]. The generally used cost function with MPC in scalar form is given in equation 2.1,

$$J_k = \sum_{i=1}^{L} ((y_{k+i} - r_{k+i})^T Q_i (y_{k+i} - r_{k+i}) + u_{k+i-1}^T P_i u_{k+i-1} + \Delta u_{k+i-1}^T R_i \Delta u_{k+i-1}) \tag{2.1}$$

Where

$Q_i \in R^{m \times m}$

$P_i \in R^{r \times r}$

$R_i \in R^{r \times r}$

are the user specified symmetric and positive semi-definite weighting matrices and $L$ is prediction horizon. For simplicity, these matrices could be chosen as, $Q_i = q\,\mathrm{I}_m$, $P_i = p\,\mathrm{I}_r$ and $R_i = r_0\,\mathrm{I}_r$ where $q, p$ and $r_0$ are taken as positive parameters. In the general case $Q_i, P_i$ and $R_i$ are diagonal weighting matrices, where $P_i$ is taken as zero in order to obtain MPC with integral action such that output is equal to the reference signal i.e. $y = r$ [15]. The cost function can be written in matrix form as,

$$J_k = (y_{k+1|L} - r_{k+1|L})^T Q(y_{k+1|L} - r_{k+1|L}) + u_{k|L}^T P u_{k|L} + \Delta u_{k|L}^T R \Delta u_{k|L} \tag{2.2}$$

Where

$Q \in R^{Lm \times Lm}$

$P \in R^{Lr \times Lr}$

$R \in R^{Lr \times Lr}$

are symmetric and positive semi-definite weighting matrices. MPC with offset free control, in other words MPC with integral action can be achieved by choosing $P = 0$. The control problem subjected to prediction model and process variable constraints is specified as equation 2.3.

$$u_{k|L}^* = \arg\min_{u_{k|L}} J_k(u_{k|L}) \tag{2.3}$$

The objective of the cost function is to minimize the difference between the process output $y_{k+1|L}$ and specified reference $r_{k+1|L}$ and at the same time minimize the control $u_{k|L}$ [15].

### 2.3.2 Constraints

The limitations to a process are known as constraints, and MPC became the standard control strategy due to its constraints handling abilities. Common types of constraints for model predictive control are the input amplitude constraint, input rate constraint and output constraint that can be written by following linear inequality form as in equation 2.4 [15].

$$A\Delta u_{k|L} \le b \tag{2.4}$$

Where $A$ is a matrix and $b$ is the vector. The more details about the common constraints is given below,

- **Input amplitude constraint**

It is amplitude constraints on the input signal which can be mathematically written as shown in equation 2.5,

$$u_{k|L}^{\min} \le u_{k|L} \le u_{k|L}^{\max} \tag{2.5}$$

The relationship between $u_{k|L}$ and $\Delta u_{k|L}$ can be defined as follows,

$$u_{k|L} = S\Delta u_{k|L} + cu_{k-1} \tag{2.6}$$

Rearranging the above equation as $S\Delta u_{k|L} = u_{k|L} - cu_{k-1}$ and the equation (2.5) is equivalent to equation (2.7),

$$\left.\begin{aligned} S\Delta u_{k|L} &\le \ \ u_{k|L}^{\max} - cu_{k-1} \\ -S\Delta u_{k|L} &\le -u_{k|L}^{\min} + cu_{k-1} \end{aligned}\right\} \tag{2.7}$$

Where $A = \begin{bmatrix} S \\ -S \end{bmatrix}$ and $b = \begin{bmatrix} u_{k|L}^{\max} - cu_{k-1} \\ -u_{k|L}^{\min} + cu_{k-1} \end{bmatrix}$, substituting $A$ and $b$ in equation (2.4) we have,

$$\begin{bmatrix} S \\ -S \end{bmatrix} \Delta u_{k|L} \le \begin{bmatrix} u_{k|L}^{\max} - cu_{k-1} \\ -u_{k|L}^{\min} + cu_{k-1} \end{bmatrix} \tag{2.8}$$

The input amplitude constraints in linear inequality form are given in the above equation.

- **Input rate constraint**

The limitations on the rate of change are stated as input rate constraints. Mathematically it can be written as,

$$\Delta u_{k|L}^{\min} \le u_{k|L} \le \Delta u_{k|L}^{\max} \tag{2.9}$$

The above equation is equivalent to

$$\begin{aligned} \Delta u_{k|L} &\le \ \ \Delta u_{k|L}^{\max} \\ -\Delta u_{k|L} &\le -\Delta u_{k|L}^{\min} \end{aligned} \tag{2.10}$$

The equation (2.10) can be written in linear inequality form, where $A = \begin{bmatrix} I \\ -I \end{bmatrix}$ and $b = \begin{bmatrix} \Delta u_{k|L}^{\max} \\ -\Delta u_{k|L}^{\min} \end{bmatrix}$

$$\begin{bmatrix} I \\ -I \end{bmatrix} \Delta u_{k|L} \le \begin{bmatrix} \Delta u_{k|L}^{\max} \\ -\Delta u_{k|L}^{\min} \end{bmatrix} \tag{2.11}$$

- **Output constraints**

The limitations on the output are defined as output constraints that can be written as

$$y_{\min} \le y_{k+1|L} \le y_{\max} \tag{2.12}$$

The prediction model in terms of control variable $u_{k|L}$ is given in equation (2.13),

$$y_{k+1|L} = p_L + F_L u_{k|L} \tag{2.13}$$

From equation (2.6) we have $u_{k|L}$ such that,

$$u_{k|L} = S\Delta u_{k|L} + cu_{k-1}$$

Substituting the $u_{k|L}$ in equation (2.13) we have,

$$y_{k+1|L} = p_L + F_L\left(S\Delta u_{k|L} + cu_{k-1}\right) \tag{2.14}$$

$$y_{k+1|L} = p_L + F_L S\Delta u_{k|L} + F_L cu_{k-1} \tag{2.15}$$

$$y_{k+1|L} = F_L S\Delta u_{k|L} + p_L + F_L cu_{k-1} \tag{2.16}$$

Further summarizing equation (2.16) gives,

$$y_{k+1|L} = F_L^{\Delta}\Delta u_{k|L} + p_{_L}^{\Delta} \tag{2.17}$$

Where

$$F_L^{\Delta} = F_L S$$
$$p_{_L}^{\Delta} = p_L + F_L cu_{k-1}$$

Combining the output constraint and prediction model in term of control change variable as given in equation (2.12) and (2.17) respectively, we have

$$y_{\min} \le F_L^{\Delta}\Delta u_{k|L} + p_{_L}^{\Delta} \le y_{\max} \tag{2.18}$$

The above equation is equivalent to

$$\left.\begin{array}{r} F_L^{\Delta}\Delta u_{k|L} \le \ y_{\max} - p_{_L}^{\Delta} \\ -F_L^{\Delta}\Delta u_{k|L} \le -y_{\min} + p_{_L}^{\Delta} \end{array}\right\} \tag{2.19}$$

Writing above equation in linear inequality form $A\Delta u_{k|L} \le b$, we have

$$\begin{bmatrix} F_L^{\Delta} \\ -F_L^{\Delta} \end{bmatrix}\Delta u_{k|L} \le \begin{bmatrix} y_{\max} - p_{_L}^{\Delta} \\ -y_{\min} + p_{_L}^{\Delta} \end{bmatrix} \tag{2.20}$$

Where,

$$A = \begin{bmatrix} F_L^{\Delta} \\ -F_L^{\Delta} \end{bmatrix}$$
$$b = \begin{bmatrix} y_{\max} - p_{_L}^{\Delta} \\ -y_{\min} + p_{_L}^{\Delta} \end{bmatrix}$$

Input amplitude, input change and output constraints from equations (2.8), (2.11) and (2.20) can be combined and written as linear inequality of the form $A\Delta u_{k|L} \le b$ respectively.

$$\begin{bmatrix} S \\ -S \\ I \\ -I \\ F_L^{\Delta} \\ -F_L^{\Delta} \end{bmatrix} \Delta u_{k|L} \le \begin{bmatrix} u_{k|L}^{\max} - cu_{k-1} \\ -u_{k|L}^{\min} + cu_{k-1} \\ \Delta u_{k|L}^{\max} \\ -\Delta u_{k|L}^{\min} \\ y_{\max} - p_{L}^{\Delta} \\ -y_{\min} + p_{L}^{\Delta} \end{bmatrix} \tag{2.21}$$

Where

$$A = \begin{bmatrix} S \\ -S \\ I \\ -I \\ F_L^{\Delta} \\ -F_L^{\Delta} \end{bmatrix} \text{ and } b = \begin{bmatrix} u_{k|L}^{\max} - cu_{k-1} \\ -u_{k|L}^{\min} + cu_{k-1} \\ \Delta u_{k|L}^{\max} \\ -\Delta u_{k|L}^{\min} \\ y_{\max} - p_{L}^{\Delta} \\ -y_{\min} + p_{L}^{\Delta} \end{bmatrix}$$

The solution to constraint problem can be solved by the quadratic programming. The control objective criterion and prediction model are given as,

$$J_k = (y_{k+1|L} - r_{k+1|L})^T Q (y_{k+1|L} - r_{k+1|L}) + \Delta u_{k|L}^T P \Delta u_{k|L} \tag{2.22}$$

Prediction model,

$$y_{k+1|L} = F_L \Delta u_{k|L} + p_L \tag{2.23}$$

By substituting equation (2.23) into (2.22) gives,

$$J_k = (F_L \Delta u_{k|L} + p_L - r_{k+1|L})^T Q (F_L \Delta u_{k|L} + p_L - r_{k+1|L}) + \Delta u_{k|L}^T P \Delta u_{k|L} \tag{2.24}$$

$$\begin{aligned} J_k = {} & \Delta u_{k|L}^T F_L^T Q F_L \Delta u_{k|L} + \Delta u_{k|L}^T F_L^T Q (p_L - r_{k+1|L}) + (p_L - r_{k+1|L})^T Q F_L \Delta u_{k|L} + \\ & (p_L - r_{k+1|L})^T Q (p_L - r_{k+1|L}) + \Delta u_{k|L}^T P \Delta u_{k|L} \end{aligned} \tag{2.25}$$

$$J_k = \Delta u_{k|L}^T (F_L^T Q F_L + P) \Delta u_{k|L} + \Delta u_{k|L}^T F_L^T Q (p_L - r_{k+1|L}) + (p_L - r_{k+1|L})^T Q F_L \Delta u_{k|L} + J_0 \tag{2.26}$$

Equation (2.26) can be written as

$$J_k = \Delta u_{k|L}^T H \Delta u_{k|L} + 2 f^T \Delta u_{k|L} + J_0 \tag{2.27}$$

Where,

$H = F_L^T Q F_L + P$

$f = F_L^T Q(p_L - r_{k+1|L})$

$J_0 = (p_L - r_{k+1|L})^T Q(p_L - r_{k+1|L})$

Subjected to $A\Delta u_{k|L} \le b$ the quadratic programming problem can be formulated as follows,

$$\min_{\Delta u_{k|L}}(\Delta u_{k|L}^T H \Delta u_{k|L} + 2f^T \Delta u_{k|L}) \tag{2.28}$$

Using the "*quadprog*" function in MATLAB, the problem can be solved such that,

$$\Delta u_{k|L} = quadprog(H, f, A, b) \tag{2.29}$$

And the control single $u_{k|L}$ can be computed as $u_{k|L} = \Delta u_{k|L} + u_{k-1|L}$ where $u_{k-1|L}$ must be known however it can be specified in the start [15].

## 2.3.3 Prediction model

Model predictive control requires a process model that describes the input to the output behavior of the process. Prediction model (PM) is usually constructed from the process model that describes the relationship between the future outputs and future control inputs. A linear dynamic process model can be written in the standard prediction model form as,

$$y_{k+1|L} = p_L + F_L u_{k|L} \tag{2.30}$$

Where $F_L \in \mathbb{R}^{Lm \times Lr}$ is a constant matrix derived from the process model, $L$ is a prediction horizon, and $p_L \in \mathbb{R}^{Lr}$ is a vector that depends on model parameters and a number of inputs and outputs that are older than time $k$. The prediction model in equation (2.30) is used in the MPC algorithm to compute the actual control vector $u_{k|L}^*$ [15].

In some MPC algorithm process deviation variables are computed, such as MPC with integral action computing the vector of future control deviation variables $\Delta u_{k|L}$. The prediction model in this case can be written as,

$$y_{k+1|L} = F_L^\Delta \Delta u_{k|L} + p_L^\Delta \tag{2.31}$$

In the MPC algorithm, this prediction model can be used to compute the control vector $\Delta u_{k|L}^*$ .The main purpose of using a prediction model given in equation (2.30) and (2.31) is to express the future predictions as a function of unknown future control vectors which MPC algorithm will compute [15].

## 2.4 Model Predictive Control principle

Current measurements and future outputs predicted by using the process model are the bases of the MPC calculations. At each sampling time, the MPC algorithm computes a sequence of control signals over the prediction horizon. The purpose of these control signals is to minimize the difference between the predicted controlled outputs and set point of the outputs or in other words predicted output reached set point in an optimal way [18].

The general principle of MPC control calculation is illustrated in Figure 2.2 for a SISO[1] control, where $y$, $\hat{y}$ and $u$ are actual output, predicted output and manipulated input respectively. The MPC algorithm computes a sequence of control signals $u(k+i-1)$ for $i=1,2,....,L$ at the current sampling time $k$. This sequence consists of the current control input $u(k)$ and $(L-1)$ future control inputs [19]. The first input of optimal sequence computed by the MPC is implemented, and the rest of control inputs are discarded. At the next sampling instance $k+1$, a new set of control signals $u(k+L+1)$ is calculated and again only the first is sent into the system. The entire process is repeated at subsequent sampling intervals.

In order to find the optimal control input vector, a cost function $J$ is to be minimized over a receding horizon consists of a finite number of steps $L$ in the future as mention by Byeongil Kim [17]. The number of samples the MPC controller predicts in the future called the prediction horizon $L$, and a number of control moves within the prediction horizon is called the control horizon [17].

[1] Single Input and Single Output

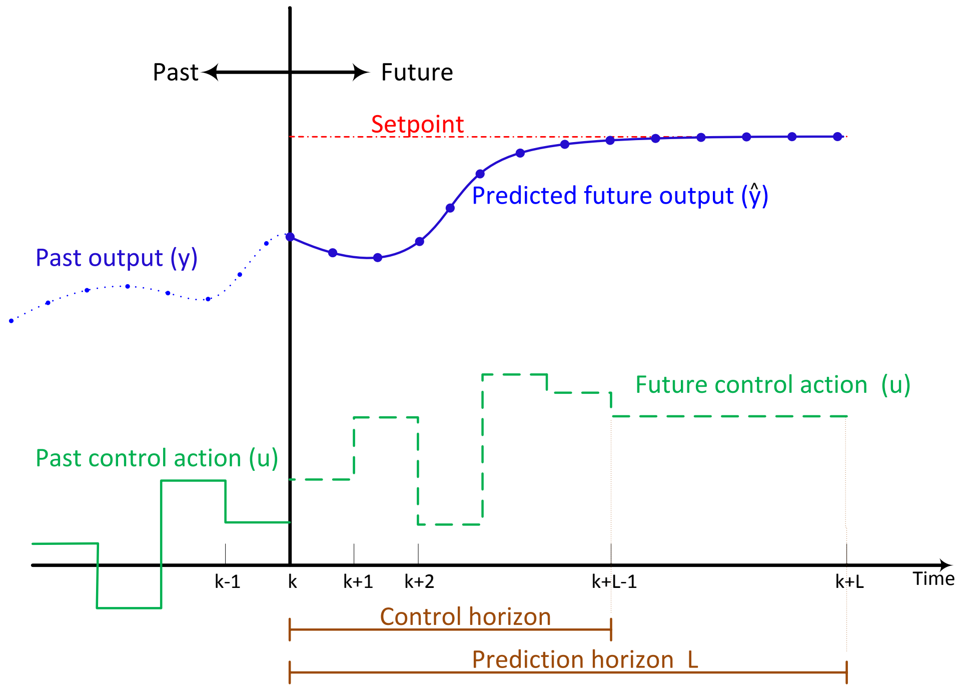


Figure 2.2: Model Predictive control concept [19]

In MPC algorithm prediction, the horizon is being shifted forward and for this reason MPC is also known as receding horizon control [20]. The basic idea of shifting forward prediction horizon from the present time instance $k$ to next time interval $k+1$ is illustrated in the diagram shown in Figure 2.3. At time instance $k$, an MPC controller predicts the $k+L$ outputs, and at the next sampling interval $k+1$, the prediction horizon moves forward as a result, the MPC controller predicts $k+L+1$ outputs [5].

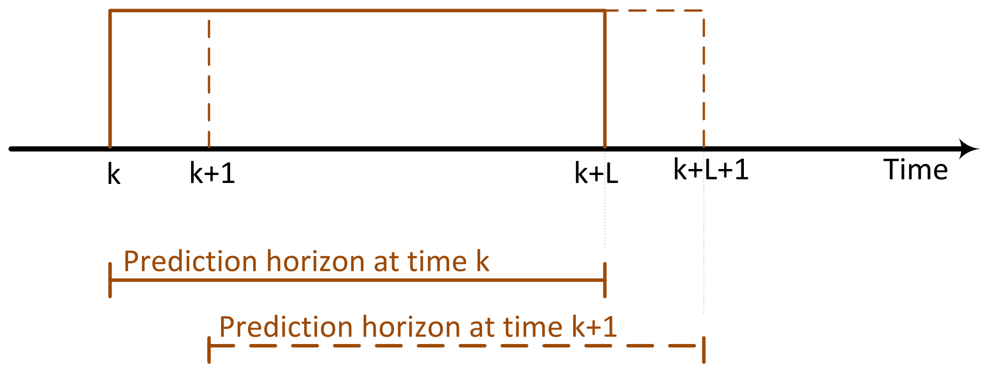


Figure 2.3: MPC receding prediction horizon [16]

# 3. State space model based MPC and integral action

In this chapter, state space model based predictive control is presented and different methods of achieving integral actions in the MPC optimal control algorithm are discussed. The formulation of MPC with integral action is described that will be used for simulation experiments of the four-tank process. Finally, the Kalman Filter algorithm steps are explained.

## 3.1 Introduction

Most of the MPC applications using prediction model based on a linear dynamic model of the process that will lead to the linear model predictive control. However, using the nonlinear process model for prediction will be resulted in the nonlinear model predictive control and it is a nonlinear optimization method that can be solved by Sequential Quadratic Programming (SQP) [21]. The problems with nonlinear MPC optimization are local minima, and there is no guarantee of nonlinear MPC to converge within the specified computation time. Hence, the model predictive control can be categorized into the linear model predictive control and nonlinear model predictive control by using the linear and nonlinear models in the prediction model respectively [15]. There are various methods to formulate the predict control algorithms that only different from each other based on the process model used for the cost function. MPC algorithms are using finite impulse response models, step response models, transfer function models or state space models for computing the future output predictions [5]. The general approach is to use a state space model as it is easy to convert any linear dynamic model into the state space model [15].

- Impulse/Step response model:

Impulse/ Step models are a special case of input and output models that can be formulated by simple experiments but required a large amount of parameters to be considered. These model parameters related to the impulse response matrices of the state space model. Matrix Algorithm Control (MAC [9]) and Dynamic Matrix Control (DMC [13]) algorithms use these models. However, these methods are not common because they rely on the model that describes only special case linear dynamic systems e.g. stable systems and systems without integrator, as mention by Ruscio [15].

- Transfer function model:

Transfer function models are preferred when it is easy to formulate a good model using physical laws or by system identification methods. It required fewer parameters to be estimated as compared to the Impulse/ Step response models. Generalized Predictive Control (GPC) is a class of predictive control that uses a transfer function model of the system [15, 22]. It uses different types of the transfer function models, examples are Controller Auto Regressive Integrated Moving Average (CARIMA), Deterministic Auto Regressive Moving Average (DARMA) or Controller Auto Regressive Moving Average (CARMA) model [5].

- State space model:

State space model similar to transfer function model can be formulated on the basis of physical laws or system identification methods. These models are mostly used for a time invariant system. In this thesis, the state space model based Model Predictive Control is in the main focus, hence its formulation is described in more details.

## 3.2 Extended Model Predictive Control (EMPC)

The prediction model can be formulated by using the state space model that leads to the Extended Model Predictive Control (EMPC) algorithm as presented by Rusico [23]. It can classify into EMPC$_1$ and EMPC$_2$ based on prediction model that uses the process actual variables and process deviation variables respectively. The formulation of the state space model based MPC is described below and the mathematical derivations in this chapter are referenced from Ruscio [15].

### 3.2.1 Extended Model Predictive Control (EMPC$_1$)

Prediction model is based on the actual variable in EMPC$_1$. A deterministic linear dynamic system can be written as a state space model given in Equation (3.1) and (3.2) as mention by Ruscio [15].

$$x_{k+1} = Ax_k + Bu_k \tag{3.1}$$

$$y_k = Dx_k \tag{3.2}$$

Using the above state space model , the prediction model can be formulated for $L = 4,$ where $L$ is prediction horizon.

For $k = k + 1$

$$y_{k+1} = Dx_{k+1} \tag{3.3}$$

Substituting the equation (3.1) into (3.3), gives

$$y_{k+1} = D(Ax_k + Bu_k) \tag{3.4}$$

$$y_{k+1} = DAx_k + DBu_k \tag{3.5}$$

For $k = k+2$

$$y_{k+2} = D(Ax_{k+1} + Bu_{k+1}) \tag{3.6}$$

$$y_{k+2} = DA^2x_k + DBu_k + DBu_{k+1} \tag{3.7}$$

For $k = k+3$

$$y_{k+3} = D(Ax_{k+2} + Bu_{k+2}) \tag{3.8}$$

$$y_{k+3} = DA^3x_k + DA^2Bu_k + DABu_{k+1} + DBu_{k+2} \tag{3.9}$$

Equations (3.2), (3.5), (3.7) and (3.9) can be written in matrix form as,

$$\underbrace{\begin{bmatrix} y_k \\ y_{k+1} \\ y_{k+2} \\ y_{k+3} \end{bmatrix}}_{y_{k|4}} = \underbrace{\begin{bmatrix} D \\ DA \\ DA^2 \\ DA^3 \end{bmatrix}}_{o_4} x_k + \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ DB & 0 & 0 \\ DAB & DB & 0 \\ DA^2B & DAB & DB \end{bmatrix}}_{H_4^d} \underbrace{\begin{bmatrix} u_k \\ u_{k+1} \\ u_{k+2} \end{bmatrix}}_{u_{k|3}} \tag{3.10}$$

Where $y_{k|4}$ is data matrix of output variables, $u_{k|3}$ is data matrix of input variables, $O_4$ extended observability matrix for the pair (D, A), and $H_4^d$ is lower block triangular Toeplitz matrix for (D,A,B) matrices [24]. The equation (3.10) can be written as,

$$y_{k|4} = O_4x_k + H_4^du_{k|3} \tag{3.11}$$

The above equation when the prediction horizon is equal to $L$ becomes,

$$y_{k|L} = O_Lx_k + H_L^du_{k|L-1} \tag{3.12}$$

Formulating the equation when $k = k+1$, the equation (3.12) becomes,

$$y_{k+1|L} = O_Lx_{k+1} + H_L^du_{k+1|L-1} \tag{3.13}$$

From equation (3.1) $x_{k+1} = Ax_k + Bu_k$, substituting in the above equation gives,

$$y_{k+1|L} = O_LAx_k + O_LBu_k + H_L^du_{k+1|L-1} \tag{3.14}$$

Writing the equation (3.14) in matrix form gives,

$$y_{k+1|L} = O_L A x_k + \begin{bmatrix} O_L B & H_L^d \end{bmatrix} \begin{bmatrix} u_k \\ u_{k+1|L-1} \end{bmatrix} \tag{3.15}$$

Where,

$p_L = O_L A x_k$ ,

$F_L = \begin{bmatrix} O_L B & H_L^d \end{bmatrix}$,

$u_{k|L} = \begin{bmatrix} u_k \\ u_{k+1|L-1} \end{bmatrix}$,

Substituting $p_L$, $F_L$ and $u_{k|L}$, the equation (3.15) becomes,

$$y_{k+1|L} = p_L + F_L u_{k|L} \tag{3.16}$$

The equation (3.16) is the prediction model that will be based for an MPC algorithm to compute the predicted outputs. The term $p_L$ in the above equation depends upon the $x_k$ which is the present state and resulting MPC will be a state feedback type algorithm [15].

$$J_k = (y_{k+1|L} - r_{k+1|L})^T Q (y_{k+1|L} - r_{k+1|L}) + u_{k|L}^T P u_{k|L} \tag{3.17}$$

Substituting prediction model from the equation (3.16) into the cost function given in equation (3.17), we have

$$J_k = (F_L u_{k|L} + p_L - r_{k+1|L})^T Q (F_L u_{k|L} + p_L - r_{k+1|L}) + u_{k|L}^T P u_{k|L} \tag{3.18}$$

$$J_k = u_{k|L}^T (F_L^T Q F_L + P) u_{k|L} + 2 F_L^T Q (p_L - r_{k+1|L}) u_{k|L} + (p_L - r_{k+1|L})^T Q (p_L - r_{k+1|L}) \tag{3.19}$$

$$J_k = u_{k|L}^T H u_{k|L} + 2 f^T u_{k|L} + J_0 \tag{3.20}$$

Where

$H = F_L^T Q F_L + P$

$f = F_L^T Q (p_L - r_{k+1|L})$

$J_0 = (p_L - r_{k+1|L})^T Q (p_L - r_{k+1|L})$

Minimizing the cost function given in equation (3.20) with respect to $u_{k|L}$ i.e, $\min\limits_{u_{k|L}} J_k$ ,

$$\frac{\partial J_k}{\partial u_{k|L}} = u_{k|L}^T H u_{k|L} + 2 f^T u_{k|L} + J_0 \tag{3.21}$$

The equation (3.21) becomes the future optimal control, where only the first element $u_{k|L}$, of control vector $u^*_{k|L}$ is used for control purpose [15].

$$u^*_{k|L} = -H^{-1} * f \tag{3.22}$$

### 3.2.2 Extended Model Predictive Control (EMPC$_2$)

The EMPC$_2$ is based on the process deviation variables and prediction model can be derived from the equations (3.1) and (3.2) by introducing the relationship $u_{k|L} = S\Delta u_{k|L} + cu_{k-1}$ [15]. The prediction model will be of the form,

$$y_{k+1|L} = p^{\Delta}_{L} + F^{\Delta}_{L}\Delta u_{k|L} \tag{3.23}$$

$p_L$ and $F_L$ are given by equation (3.15), where $F^{\Delta}_{L}$ and $p^{\Delta}_{L}$ are,

$$F^{\Delta}_{L} = F_L S$$
$$p^{\Delta}_{L} = p_L + F_L cu_{k-1}$$

The cost function,

$$J_k = (y_{k+1|L} - r_{k+1|L})^T Q (y_{k+1|L} - r_{k+1|L}) + u^T_{k|L} P u_{k|L} + \Delta u^T_{k|L} R \Delta u_{k|L} \tag{3.24}$$

Substituting prediction model from the equation (3.23) into the cost function given in equation (3.24), becomes

$$J_k = (F^{\Delta}_{L}\Delta u_{k|L} + p^{\Delta}_{L} - r_{k+1|L})^T Q (F^{\Delta}_{L}\Delta u_{k|L} + p^{\Delta}_{L} - r_{k+1|L}) + u^T_{k|L} P u_{k|L} + \Delta u^T_{k|L} R \Delta u_{k|L} \tag{3.25}$$

$$J_k = \Delta u^T_{k|L}(F^{\Delta^T}_{L} Q F^{\Delta}_{L} + R + S^T P S)\Delta u_{k|L} + 2(F^{\Delta^T}_{L} Q (p^{\Delta}_{L} - r_{k+1|L}) + S^T P c u_{k-1})^T \Delta u_{k|L} + J_0 \tag{3.26}$$

$$J_k = \Delta u^T_{k|L} H \Delta u_{k|L} + 2 f^T \Delta u_{k|L} + J_0 \tag{3.27}$$

Where

$$u_{k|L} = S\Delta u_{k|L} + cu_{k-1},$$
$$H = (F^{\Delta^T}_{L} Q F^{\Delta}_{L} + R + S^T P S),$$
$$f = (F^{\Delta^T}_{L} Q (p^{\Delta}_{L} - r_{k+1|L}) + S^T P c u_{k-1}),$$
$$J_0 = (p^{\Delta}_{L} - r_{k+1|L})^T Q (p^{\Delta}_{L} - r_{k+1|L}) + u^T_{k-1} c^T P c u_{k-1}$$

Minimizing the cost function given in equation (3.27) with respect to $\Delta u_{k|L}$ i.e, $\min\limits_{\Delta u_{k|L}} J_k$ ,

$$\frac{\partial J_k}{\partial u_{k|L}} = \Delta u_{k|L}^T H \Delta u_{k|L} + 2 f^T \Delta u_{k|L} + J_0 \tag{3.28}$$

The equation (3.28) becomes the future optimal control as,

$$\Delta u^*_{k\,|L} = -H^{-1} * f \tag{3.29}$$

Where only the first element $\Delta u_{k\,|L}$, of control vector $\Delta u^*_{k\,|L}$ is used for control purpose and actual control signal $u_k$ for MPC optimal control can be calculated as $u_k = \Delta u_k + u_{k-1}$ [15].

## 3.3 Integral action in MPC

Model predictive control has several algorithms that differ from each other based on the process model used for the cost function. These formulations have some problems with offset i.e. the process output $y$ is not equal to the set point $r$ in steady state. In order for the controllers to handle the offset problem, integral action is an effective method. Different methods have been presented to achieve offset free control or integral action in MPC algorithms. A short overview of these methods is discussed here, and formulation of MPC optimal controller with integral action used in the simulation experiments is present in the next section.

Åkesson [25] suggested a disturbance observer approach to obtain offset free control. There are always modeling error and disturbance that affect the proper working of the controller. These problems can be handled by introducing the integral action as a result, error free tracking of reference signal is obtained [25].

According to Muske and Bedgwell [26], most of the MPC algorithms use a constant output step disturbance model to achieve the integral action. The same approach was discussed by Rawling [27] to formulate the controller that effectively handles the steady state offset. Another method to achieve the offset free control is by the addition of integrating disturbance to the process model presented by Pannocchia and Rawlings [28]. A velocity form state-space method to get an offset free control pointed out by Pat and Garcia [29] which is a similar approach to augmenting the system model with a disturbance. In this method, state acts as a change in the original state, and outputs of the original system are the augmented states. Davison and Smith [30] pointed out that disturbance formulation as the standard approach to achieve integral action.

Morten [31] formulated the MPC with integral action by using the input changes as free variables in the optimization instead of using the input itself and presented the augmented model formulation as given below using the model in the equations (3.1) and (3.2),

$$\underbrace{\begin{bmatrix} \Delta x_{k+1} \\ u_k \end{bmatrix}}_{\tilde{x}_{k+1}} = \underbrace{\begin{bmatrix} A & B \\ 0 & I \end{bmatrix}}_{\tilde{A}} \tilde{x}_k + \underbrace{\begin{bmatrix} B \\ I \end{bmatrix}}_{\tilde{B}} \Delta u_k$$
$$y_k = \underbrace{\begin{bmatrix} D & I \end{bmatrix}}_{\tilde{D}} \tilde{x}_k \tag{3.30}$$

Where $\Delta u_k = u_k - u_{k-1}$.

In this work, the integral action in the MPC algorithm is achieved by using the deviation variables mention by Rusico [7] and described in the next part with details derivation.

## 3.4 Formulation of MPC optimal controller with integral action

The model of the process with disturbance can be written as

$$x_{k+1} = Ax_k + Bu_k + v \tag{3.31}$$

$$y_k = Dx_k + Eu_k + w \tag{3.32}$$

The equation (3.32) has a direct term from the input signal $u_k$ to the output signal $y_k$, however in MPC due to receding horizon control principle, the current information about the process is used for prediction and control. Therefore, in process model the term $E$ is set to zero because the input signal $u_k$ can not affect the output signal $y_k$ at the same time as mention by Wang [3]. Then the process model will be as

$$x_{k+1} = Ax_k + Bu_k + v \tag{3.33}$$

$$y_k = Dx_k + w \tag{3.34}$$

Where $A,B$ and $D$ are known system matrices, $x_k \in \mathbb{R}^n$ is a state variable vector, $u_k \in \mathbb{R}^r$ is control input vector, $y_k \in \mathbb{R}^m$ is output (measurement) vector. In equation (3.33) and (3.34), $v$ represent an unknown constant or slowly varying process disturbance and $w$ is an unknown constant or a slowly varying measurement noise vector [7, 32]. The disturbance $v$ and $w$ both are not known and the MPC algorithm required model free from unknown disturbances [15] which can be eliminated by introducing the terms $x_k$ and $y_{k-1}$.

Where

$$x_k = Ax_{k-1} + Bu_{k-1} + v \tag{3.35}$$

$$y_{k-1} = Dx_{k-1} + w \tag{3.36}$$

Subtracting the equation (3.35) from (3.33) and equation (3.36) from (3.34), gives

$$x_{k+1} - x_k = Ax_k + Bu_k + v - (Ax_{k-1} + Bu_{k-1} + v) \tag{3.37}$$

$$y_k - y_{k-1} = Dx_k + w - (Dx_{k-1} + w) \tag{3.38}$$

After simplifying above two equations,

$$x_{k+1} - x_k = A(x_k - x_{k-1}) + B(u_k - u_{k-1}) \tag{3.39}$$

$$y_k - y_{k-1} = D(x_k - x_{k-1}) \tag{3.40}$$

Where,

$\Delta x_{k+1} = x_{k+1} - x_k$

$\Delta x_k = x_k - x_{k-1}$

$\Delta u_k = u_k - u_{k-1}$

By substituting the above terms in equations (3.39) and (3.40), then write them in a matrix form.

$$\Delta x_{k+1} = A\Delta x_k + B\Delta u_k \tag{3.41}$$

$$y_k = y_{k-1} + D\Delta x_k \tag{3.42}$$

$$\underbrace{\begin{bmatrix} \Delta x_{k+1} \\ y_k \end{bmatrix}}_{\tilde{x}_{k+1}} = \underbrace{\begin{bmatrix} A & 0 \\ D & I \end{bmatrix}}_{\tilde{A}} \underbrace{\begin{bmatrix} \Delta x_k \\ y_{k-1} \end{bmatrix}}_{\tilde{x}_k} + \underbrace{\begin{bmatrix} B \\ 0 \end{bmatrix}}_{\tilde{B}} \Delta u_k \tag{3.43}$$

$$y_k = \underbrace{\begin{bmatrix} D & I \end{bmatrix}}_{\tilde{D}} \underbrace{\begin{bmatrix} \Delta x_k \\ y_{k-1} \end{bmatrix}}_{\tilde{x}_k} \tag{3.44}$$

Writing equations (3.43) and (3.44) in more compact form,

$$\Delta\tilde{x}_{k+1} = \tilde{A}\Delta\tilde{x}_k + \tilde{B}\Delta u_k \tag{3.45}$$

$$y_k = \tilde{D}\Delta\tilde{x}_k \tag{3.46}$$

The prediction model from the equations (3.45) and (3.46) can be formulated by same method presented in section (3.2.1). The prediction model will be in the form as,

$$y_{k+1|L} = p_L + F_L\Delta u_{k|L} \tag{3.47}$$

Where,

$p_L = O_L\tilde{A}\tilde{x}_k$

$F_L = \left[ O_L \tilde{B} \quad H_L^d \right]$

Where $O_L$ is extended observability matrix for the pair $(\tilde{D}, \tilde{A})$, and $H_4^d$ is a Toeplitz matrix for $(\tilde{D}, \tilde{A}, \tilde{B})$ matrices [15]. The cost function from the equation (3.24) is

$$J_k = (y_{k+1|L} - r_{k+1|L})^T Q(y_{k+1|L} - r_{k+1|L}) + u_{k|L}^T P u_{k|L} + \Delta u_{k|L}^T R \Delta u_{k|L} \tag{3.48}$$

The control weighting matrix $P = 0$, to obtain an MPC with integral action such that $y = r$ in steady state. Then substituting the prediction model equation (3.47) into the cost function equation (3.48) gives,

$$J_k = (F_L \Delta u_{k|L} + p_L - r_{k+1|L})^T Q(F_L \Delta u_{k|L} + p_L - r_{k+1|L}) + \Delta u_{k|L}^T R \Delta u_{k|L} \tag{3.49}$$

After simplifying the above equation, it is minimized with respect to $\Delta u_{k|L}$ i.e. $\min\limits_{\Delta u_{k\,|L}} J_k$ ,

$$\frac{\partial J_k}{\partial u_{k|L}} = \Delta u_{k|L}^T H \Delta u_{k|L} + 2 f^T \Delta u_{k|L} + J_0 \tag{3.50}$$

Where,

$$H = F_L^T Q F_L + R$$
$$f = F_L^T Q(p_L - r_{k+1|L})$$
$$J_0 = (p_L - r_{k+1|L})^T Q(p_L - r_{k+1|L})$$

The purpose of computing the control deviation variance $\Delta u_{k|L}$ is to achieve integral action in an optimal manner.

$$\Delta u_{k\,|L}^* = -H^{-1} + f \tag{3.51}$$

Where only the first element $\Delta u_{k\,|L}$, of the control deviation vector $\Delta u_{k\,|L}^*$ is used for control purpose and the actual control signal $u_k$ for MPC optimal control can be calculated as $u_k = \Delta u_k + u_{k-1}$ [15].

## 3.5 Kalman Filter

The theory of Kalman Filter was developed by Rudolf E. Kalman in 1960's. It is commonly used algorithm for estimating the unknown state variables of a dynamic system that are excited by the stochastic disturbances and measurement noise respectively [33]. This method produces an optimal estimation in such a way that the sum of estimation errors obtains a minimum mean value [15]. In this section, Kalman Filter algorithm is formulated which will be implemented in simulation experiments discussed in chapter 5. It will be used to estimate the states since all the states are not measured. A discrete time state space model can be written as

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + v_k \\ y_k &= Dx_k + w_k \end{aligned} \tag{3.52}$$

$v_k$ is the white process noise whereas $w_k$ is white measurement noise. The steps for calculating the state estimation by the Kalman Filter algorithm are as follows [15, 34],

- Finding the Kalman Filter Gain $K$.
- Define the initial Apriori or predicted state estimate

$$\overline{x}_k = x_k \tag{3.53}$$

- Find measurement model updating,

$$\overline{y}_k = D\overline{x}_k \tag{3.54}$$

- Finding the estimation error

$$e_k = y_k - \overline{y}_k \tag{3.55}$$

- Finding the aposteriori state estimate

$$\hat{x}_k = \overline{x}_k + Ke_k \tag{3.56}$$

- Finding the apriori state estimate update

$$\overline{x}_{k+1} = A\hat{x}_k + Bu_k \tag{3.57}$$

Where $\overline{x}_k$ is apriori or predicted state estimate and $\hat{x}_k$ is aposteriori state estimate [34]. Noted $h$ has been used in MATLAB m-file script to represent the state $x$. The Kalman filter in the innovation form can be written as,

$$\begin{aligned} \overline{x}_{k+1} &= A\overline{x}_k + Bu_k + \tilde{K}e_k \\ \overline{y}_k &= D\overline{x}_k + e_k \end{aligned} \tag{3.58}$$

The Kalman filter gain in innovation form $\tilde{K} = AK$, [15].

# 4. Simulation Experiments on benchmark process

In this chapter, the four-tank benchmark process is described and its physical model is formulated by writing down mathematic equations using the basic laws of physics as presented by many researchers. The nonlinear model is linearized for use in MPC with the integral action algorithm. The operating condition of the four-tank process as minimum and non-minimum phase are discussed at the end.

## 4.1 Four-tank Process

Four-tank or quadruple-tank process is a multivariable standard process used in many control laboratories for academic purpose first presented by *Johasson* [35]. It is a nonlinear system that consists of two pumps, two valves, two level sensors and four interconnected water tanks. The level sensors are connected to the lower tanks i.e. tank 1 and tank 2. The voltages to the pumps ($u_1$ and $u_2$) are the process input, and voltages from the level sensors ($y_1$ and $y_2$) are the process outputs. The schematic diagram of the four-tank process is illustrated in Figure 4.1.

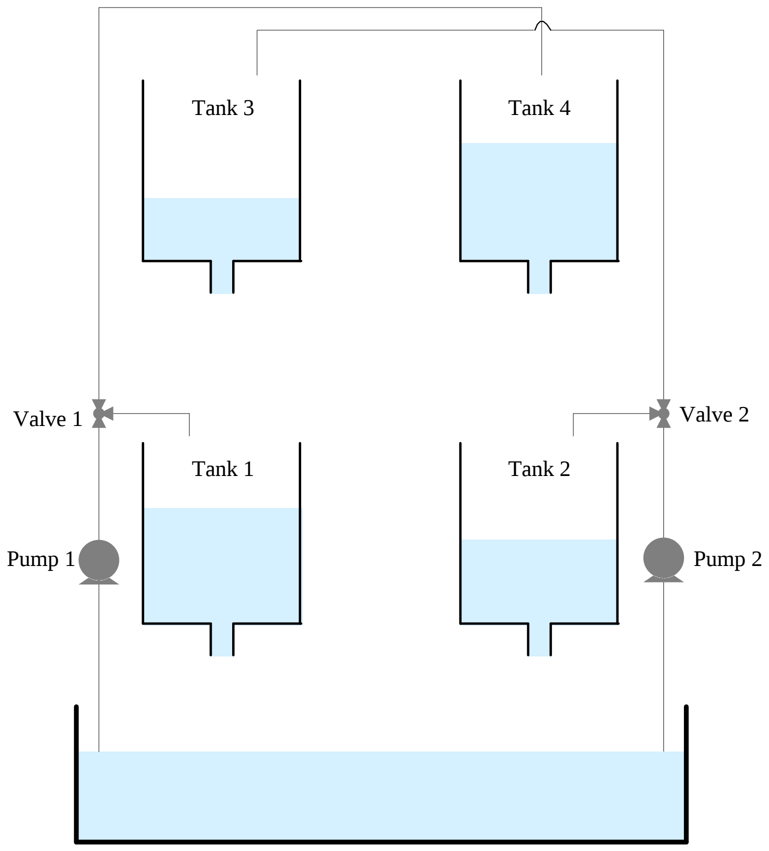


Figure 4.1: Four-tank process schematic diagram [35]

The voltage $u_1$ is applied to pump 1, for supplying water from the reservoir to tanks 1 and 4 while the voltage $u_2$ is applied to pump 2 for supplying water in tanks 2 and 3. The valve 1 is used to control the water flow in tanks 1 and 4 while valve 2 is used to control the water flow in tanks 2 and 3. The main goal is to control the liquid level in the tanks 1 and 2 therefore level sensors are used in lower tanks.

### 4.1.1 Physical model of four-tank process

The mathematical model of the four-tank process can be derived by using mass balances and Bernoulli's law as represented by Johansson [35]. The mass balance for $i^{th}$ tanks can be written as,

$$A_i \frac{dh_i}{dt} = -q_{i\,out} + q_{i\,in} \tag{4.1}$$

Where $q_{in}$ and $q_{out}$ are inflow and outflow of a tank respectively that can be modeled using Bernulli's law. $A_i$ is the cross-section area of $i^{\text{th}}$ tank and the potential energy in the tank will be equal to kinetic energy of the liquid in the tank such that [36],

$$mgh = \frac{1}{2} mv^2 \tag{4.2}$$

Solving above equation for $v$ gives,

$$v = \sqrt{2gh} \tag{4.3}$$

Multiplying the equation (4.3) with an area of the outlet hole $(a)$ of the tank gives the volumetric flow rate $q_{out}$ as,

$$q_{out} = av = a\sqrt{2gh} \tag{4.4}$$

Then the outflow of $i^{th}$ tanks can be written as,

$$q_{i\,out} = a_i v_i = a_i \sqrt{2gh}_i \tag{4.5}$$

The flow from pump 1 is $k_1 u_1$ that split into a flow $q_{1in} = \gamma_1 k_1 u_1$ to the tank 1 and a flow $q_{4in} = (1-\gamma_1) k_1 u_1$ to tank 4. Similarly, the flow from the pump 2 is $k_2 u_2$ that split into a flow $q_{2in} = \gamma_2 k_2 u_2$ to the tank 2 and a flow $q_{3in} = (1-\gamma_2) k_2 u_2$ to tank 3. The level measurement signals are $k_c * h_i$. By using equation (4.1) and (4.5) mass balance and Bernoulli's law can be extended for the four-tank process as [35],

$$A_1 \frac{dh_1}{dt} = -q_{1\,out} + q_{3\,out} + q_{1\,in} \tag{4.6}$$

$$A_2 \frac{dh_2}{dt} = -q_{2\,out} + q_{4\,out} + q_{2\,in} \tag{4.7}$$

$$A_3 \frac{dh_3}{dt} = -q_{3\,out} + q_{3\,in} \tag{4.8}$$

$$A_4 \frac{dh_4}{dt} = -q_{4\,out} + q_{4\,in} \tag{4.9}$$

Substituting the inflow $q_{in}$ and outflow $q_{out}$ of the tanks yield,

$$A_1 \frac{dh_1}{dt} = -a_1\sqrt{2gh_1} + a_3\sqrt{2gh_3} + \gamma_1 k_1 u_1 \tag{4.10}$$

$$A_2 \frac{dh_2}{dt} = -a_2\sqrt{2gh_2} + a_4\sqrt{2gh_4} + \gamma_2 k_2 u_2 \tag{4.11}$$

$$A_3 \frac{dh_3}{dt} = -a_3\sqrt{2gh_3} + (1-\gamma_2) k_2 u_2 \tag{4.12}$$

$$A_4 \frac{dh_4}{dt} = -a_4\sqrt{2gh_4} + (1-\gamma_1) k_1 u_1 \tag{4.13}$$

The nonlinear model of the four-tank process is described in above four differential equations and the parameter description is given in Table 4.1.

Table 4.1: The four-tank nonlinear model parameter description

| | |
|---|---|
| $A_i$ | Cross-section area of the tank $i$; |
| $a_i$ | Cross-section area of the outlet hole $i$; |
| $h_i$ | The water level in the tank $i$; |
| $u_i$ | The voltage applied to the pump $i$; |
| $\gamma_i$ | Valve constant for the valve $i$; |
| $k_i$ | Pump constant for the pump $i$ ; |
| $g$ | The acceleration of gravity; |
| $k_c$ | The pump gain |
| $i$ | $1, 2, ..., 4$ |

## 4.1.2 Linearization of four-tank process

The nonlinear differential equations (4.10)-(4.13) can be linearized at operating point given at the levels in the tanks $h_i^0$ and voltage $u_i^0$. Considering the variables such that $\dot{h}_i =: h_i - h_i^0$ and $u_i =: u_i - u_i^0$. The linearized state space model is given by [35]

$$\begin{aligned} \frac{dh}{dt} &= Ax + Bu \\ y &= Dx \end{aligned} \tag{4.14}$$

Where,

$$\frac{dh}{dt} = \begin{bmatrix} -\frac{1}{T_1} & 0 & \frac{A_3}{A_1 T_1} & 0 \\ 0 & -\frac{1}{T_2} & 0 & \frac{A_4}{A_2 T_2} \\ 0 & 0 & -\frac{1}{T_3} & 0 \\ 0 & 0 & 0 & -\frac{1}{T_4} \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix} + \begin{bmatrix} \frac{\gamma_1 k_1}{A_1} & 0 \\ 0 & \frac{\gamma_2 k_2}{A_2} \\ 0 & \frac{(1-\gamma_2)k_2}{A_3} \\ \frac{(1-\gamma_1)k_1}{A_4} & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{4.15}$$

$$y = \begin{bmatrix} k_c & 0 & 0 & 0 \\ 0 & k_c & 0 & 0 \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix} \tag{4.16}$$

The time constants for tank $i$ can be calculated as,

$$T_i = \frac{A_i}{a_i}\sqrt{2gh_i^0} \qquad for\ i = 1, 2, ...., 4.$$

## 4.1.3 Linear transfer function of the four-tank process

The linear transfer function of the four-tank process can be formulated by Laplace transform of equations (4.15) and (4.16) as mentioned by Numsomran [37].

$$\begin{bmatrix} y_1(s) \\ y_2(s) \end{bmatrix} = G(s).\begin{bmatrix} u_1(s) \\ u_2(s) \end{bmatrix} \tag{4.17}$$

$$G(s) = D(sI - A)^{-1} B \tag{4.18}$$

$$G(s) = \begin{bmatrix} G_{11}(s) & G_{12}(s) \\ G_{21}(s) & G_{22}(s) \end{bmatrix} \tag{4.19}$$

Where

$$G_{11}(s) = \frac{\gamma_1 c_1}{(1+sT_1)}$$

$$G_{12}(s) = \frac{(1-\gamma_2)c_1}{(1+sT_1)(1+sT_3)}$$

$$G_{21}(s) = \frac{(1-\gamma_1)c_2}{(1+sT_2)(1+sT_4)}$$

$$G_{22}(s) = \frac{\gamma_2 c_2}{(1+sT_2)}$$

Substituting above terms into the equation (4.19), we have

$$G(s) = \begin{bmatrix} \dfrac{\gamma_1 c_1}{(1+sT_1)} & \dfrac{(1-\gamma_2)c_1}{(1+sT_1)(1+sT_3)} \\ \dfrac{(1-\gamma_1)c_2}{(1+sT_2)(1+sT_4)} & \dfrac{\gamma_2 c_2}{(1+sT_2)} \end{bmatrix} \tag{4.20}$$

Where $c_1$ and $c_2$ in equation (4.20) are defined as, $c_1 = \frac{T_1 k_1 k_c}{A_1}$ and $c_2 = \frac{T_2 k_2 k_c}{A_2}$ described in [35]. According to the equations (4.17) and (4.19), the relationship between inputs and output will be as follows,

$$y_1(s) = G_{11}(s).u_1(s) + G_{12}(s).u_2(s) \tag{4.21}$$

$$y_2(s) = G_{21}(s).u_1(s) + G_{22}(s).u_2(s) \tag{4.22}$$

It will be used for input-output pairing to see how the change in inputs $u_1$ or $u_2$ affects the outputs $y_1$ or $y_2$.

### 4.1.4 Operating conditions

In this thesis, the four-tank process model and control are compared at two operating conditions defined as a minimum phase system and non-minimum phase system. They are characterized by plotting the system poles and zeros locations on the complex *s*-plane, whose axes correspond to real and imaginary parts of the complex variable *s* [38].

- **Minimum phase system**

A minimum phase system does not have zeros or poles in the right half of the complex s plane. According to *Ogata* [39], the range of phase angle in such system is minimum. In the four-tank minimum phase process, the sum of the valve constants $\gamma_1$ and $\gamma_2$ is greater than one but less than two *i.e.*, $1 < \gamma_1 + \gamma_2 < 2$. According to Johansson, in this case the flow of liquid to lower tanks is greater than the flow in the upper tanks [35].

- **Non-minimum phase system**

A non-minimum phase system has zeros or poles in the right half of the complex plane and according to *Ogata* [39], the range of phase angle in such system is larger than the minimum value. In the four-tank process, if the sum of the valve constants $\gamma_1$ and $\gamma_2$ is greater than zero but less than one *i.e.* $0 < \gamma_1 + \gamma_2 < 1$, then the system will have non-minimum phase characteristics. According to Johansson, the flow of liquid to lower tanks is smaller compared to the upper tanks. Therefore, it is hard to control the level in this phase [35].

# 5. Simulation Experiments

The simulation experiments of MPC method on benchmark process are discussed in this chapter. The parameter values for both operating conditions are taken from the literature. First of all, stability, observability and controllability of the linearized model of the four-tank process is analyzed. Constrained and unconstrained MPC with integral algorithm is implemented in the linearized model of the four-tank minimum and non-minimum phase process. Constrained MPC is further explored with different constraint handling techniques i.e. '*quadprog*' function and '*if-else'* method. A decentralized PI controller is implemented to control the four-tank process for comparing the results with the MPC optimal controller. In the last part, simulation using different values of the control horizon in the MPC algorithm is performed. The experimental results for all the simulations are plotted and compared the performance with each other.

## 5.1 Four-tank Process

Four-tank process is taken as a main benchmark process. The Kalman Filter is implemented for state estimation, and the performance comparison of reducing control horizon in MPC method is also pointed out. The results of simulation are discussed, and relevant MATLAB scripts are given in appendices. The parameter values for the four-tank process are given in the following table,

Table 5.1: Parameter values of the four-tank process [35]

| Parameter | Values |
|---|---|
| $A_1$ *and* $A_3$ [cm$^2$] | 28 |
| $A_2$ *and* $A_4$ [cm$^2$] | 32 |
| $a_1$ *and* $a_3$ [cm$^2$] | 0.071 |
| $a_2$ *and* $a_4$ [cm$^2$] | 0.057 |
| $k_c$ [V/cm] | 0.5/1.0 |
| $g$ [cm/s$^2$] | 981 |

## 5.2 Simulation of nonlinear model

The physical model of the four-tank process derived in section 4.1.1 from equations (4.10) -(4.13) is simulated to find the steady state values of the system in both minimum and non-minimum phase characteristics. The parameter values in both cases are taken from the literature and tabulated in relevant sections. The tanks are assumed to be empty at the start, and a constant voltage is applied such that the liquid level reached the stable point. A combine MATLAB program is written for both operating points, and a built-in function " *input* " is used to select either minimum phase or non-minimum phase process by typing " 1 " or " 2 " respectively. A data cursor display the values of points on the plotted lines and script files with the supporting function are given in Appendix 2.

### 5.2.1 Minimum Phase system

The system will be minimum phase if the sum of the valve constants $\gamma_1$ and $\gamma_2$ is greater than one but less than two *i.e.* $1 < \gamma_1 + \gamma_2 < 2$, as mentioned in the previous chapter [35]. In this case, the sum of valves constants is 1.3 which states that the system is minimum phase. A constant voltage of 3 [V] is applied to pumps 1 and 2, and run the simulations for a time span of 1000 [s]. The four tanks are assumed empty at the start, therefore the initial values are set equal to zero. The parameter values of the four-tank process are taken from the Table 5.1, and minimum phase parameter values are given in Table 5.2 [35].

Table 5.2: Parameter values of minimum phase [35]

| Parameter | Values |
|---|---|
| Input voltage $u_1$ [V] | 3.00 |
| Input voltage $u_2$ [V] | 3.00 |
| Pump 1 constant $k_1$ [$cm^3$/V] | 3.33 |
| Pump 2 constant $k_2$ [$cm^3$/V] | 3.35 |
| Valve 1 constant $\gamma_1$ | 0.70 |
| Valve 2 constant $\gamma_2$ | 0.60 |

The Figure 5.1 shows the results after simulating the minimum phase system. It can be seen from the plot, levels in tanks 1 and 2 reached at steady state in 450 and 530 second respectively. Similarly, the levels in tanks 3 and 4 are stable after 130 and 190 seconds respectively and reached steady state faster than tanks 1 and 3. The steady state levels in four tanks are pointed by using the data cursor tool in MATLAB where x-axis represented the simulation time [s] and y-axis are the level [cm]. The steady state levels in four-tank minimum phase process are given in Table 5.3

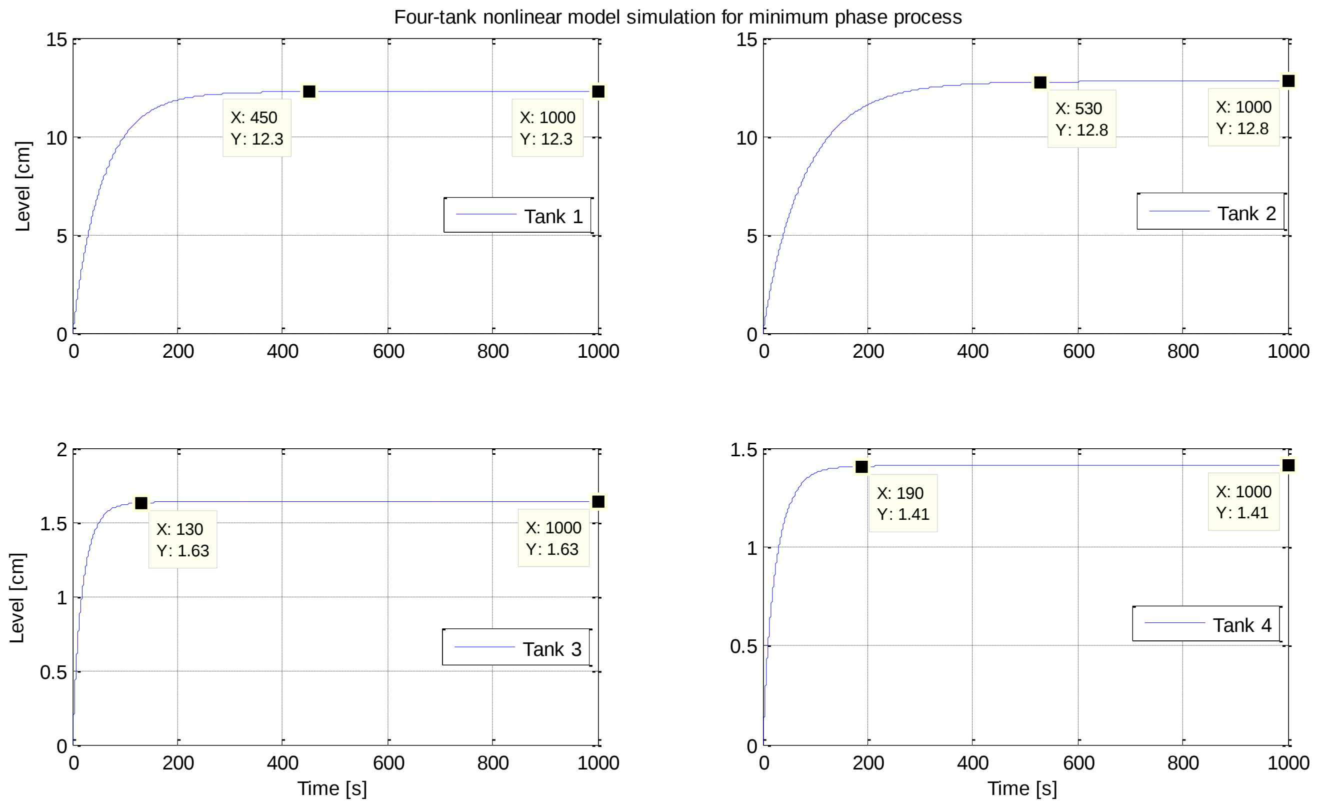


Figure 5.1: Four-tank nonlinear model simulation for minimum phase process

Table 5.3: Steady state levels in four-tank minimum phase process

| Parameter | Values |
|---|---|
| Level in tank 1 $h_1^0$ [cm] | 12.30 |
| Level in tank 2 $h_2^0$ [cm] | 12.80 |
| Level in tank 3 $h_3^0$ [cm] | 1.63 |
| Level in tank 4 $h_4^0$ [cm] | 1.41 |

### 5.2.2 Non-minimum Phase system

The system will be non-minimum phase if the sum of the valve constants $\gamma_1$ and $\gamma_2$ is greater than zero but less than one *i.e.* $0 < \gamma_1 + \gamma_2 < 1$, as mentioned in the previous chapter [35]. In this case, the sum of valves constants is 0.77 which states that the system is non-minimum phase. A constant voltage of 3.15 [V] is applied to pumps 1 and 2 and simulation performed for a time span of 1000 [s]. The four tanks are supposed to be empty at the start, therefore the initial values are set equal to zero. The parameter values of the four-tank process are taken from the Table 5.1, and non-minimum phase parameter values are given in Table 5.4 [35].

Table 5.4: Parameter values of non-minimum phase [35]

| Parameter | Values |
|---|---|
| Input voltage $u_1$ [V] | 3.15 |
| Input voltage $u_2$ [V] | 3.15 |
| Pump 1 constant $k_1$ [cm$^3$/V] | 3.14 |
| Pump 2 constant $k_2$ [cm$^3$/V] | 3.29 |
| Valve 1 constant $\gamma_1$ | 0.43 |
| Valve 2 constant $\gamma_2$ | 0.34 |

The Figure 5.2 shows the results after simulating the non-minimum phase system. The levels in tanks 1 and 2 reached a steady state in 500 and 644 second respectively and then remained stable rest of simulation time. Similarly, the levels in tanks 3 and 4 are stable after 260 and 393 seconds respectively. The level in these tanks reached to steady state faster than tanks 1 and 2. A similar trend was seen in Figure 5.1, however in non-minimum phase, the process is slower than in minimum phase. The steady state levels in four tanks are pointed by using the data cursor tool in MATLAB where x-axis represented the simulation time [s] and y-axis are the level [cm]. The steady state levels in the four-tank non-minimum phase are given in the Table 5.5.

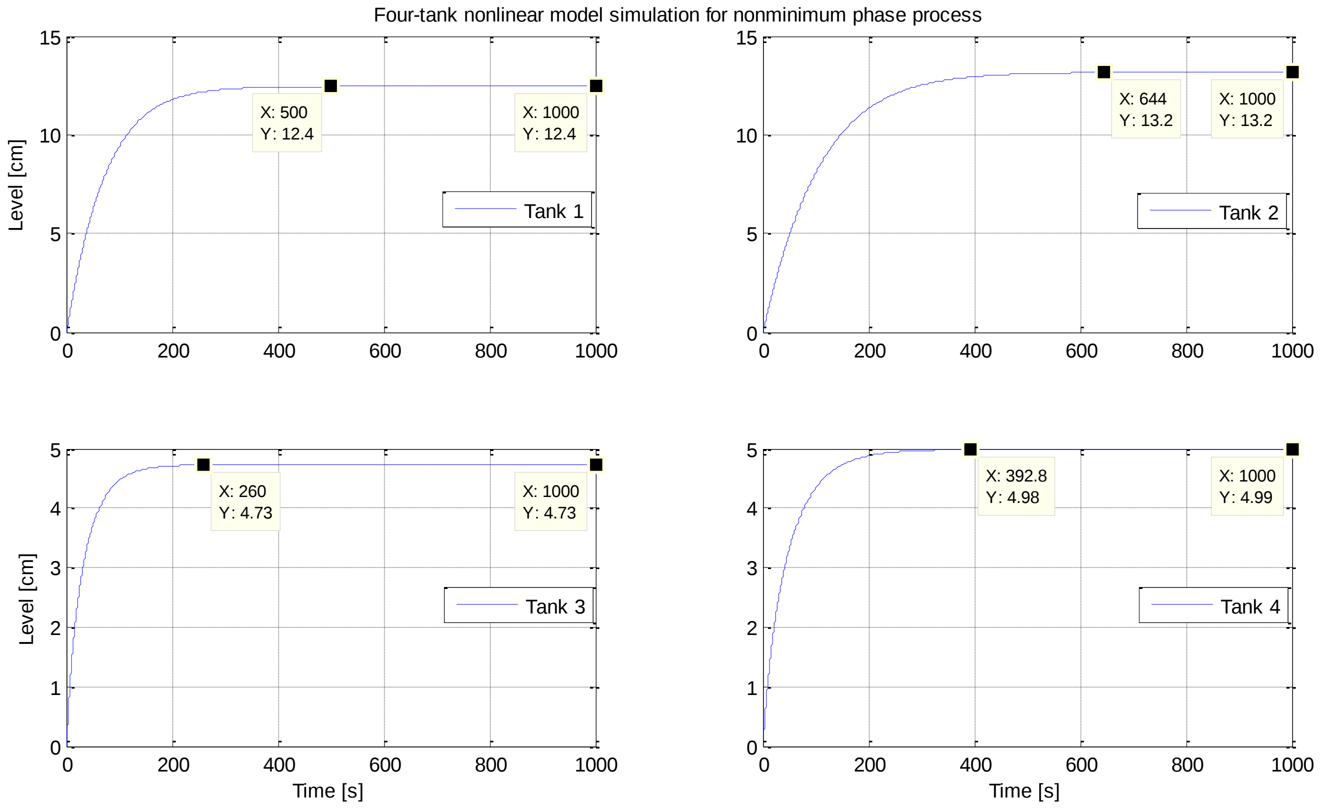


Figure 5.2: Four-tank nonlinear model simulation for non-minimum phase process

Table 5.5: Steady state levels in the four-tank non-minimum phase process

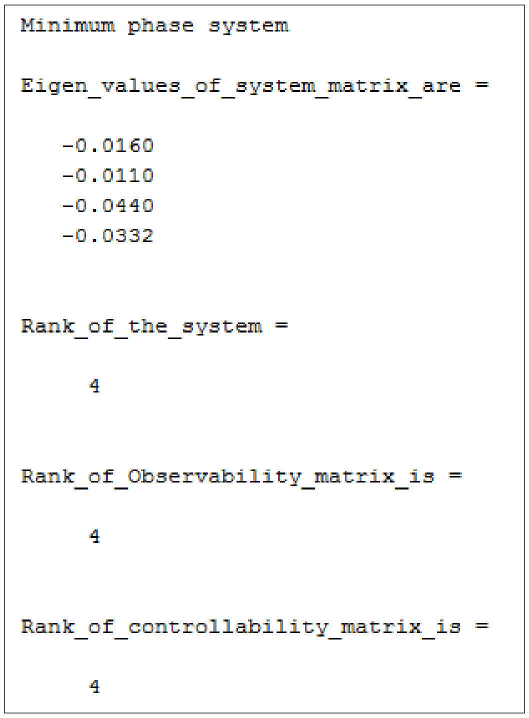
| Parameter | Values |
|---|---|
| Level in tank 1 $h_1^0$ [cm] | 12.4 |
| Level in tank 2 $h_2^0$ [cm] | 13.2 |
| Level in tank 3 $h_3^0$ [cm] | 4.73 |
| Level in tank 4 $h_4^0$ [cm] | 4.99 |

## 5.3 Observability and controllability analysis of linearized model

Linearized model of the four-tank process presented in section 4.1.2 is analyzed before implementing the MPC optimal controller. Eigenvalues are computed to check the stability. According to Rusico [32], the real part of the eigenvalues are negative in stable systems and they lie in the left part of the complex plane. A system will be observable if the rank of the observability is equal to the rank of the system, similarly a system will be controllable if the rank of the controllability is equal to the rank of the system. Therefore, observability and controllability and then rank of these matrices are computed to analyze the system observability and controllability. A combine MATLAB program is written for both operating conditions. The MATLAB built-in function " *input* " is used to select either minimum phase or non-minimum phase process by typing " 1 " or " 2 " respectively. The program script is attached in Appendix 3, and results are discussed in the next sections.

### 5.3.1 Minimum phase

The four-tank minimum phase process is stable as the real part of the eigenvalues are negative. The system is also observable and controllable as the rank of observability and controllability matrices are equal to the rank of the system. A screen dump from the MATLAB code is shown in the Figure 5.3.

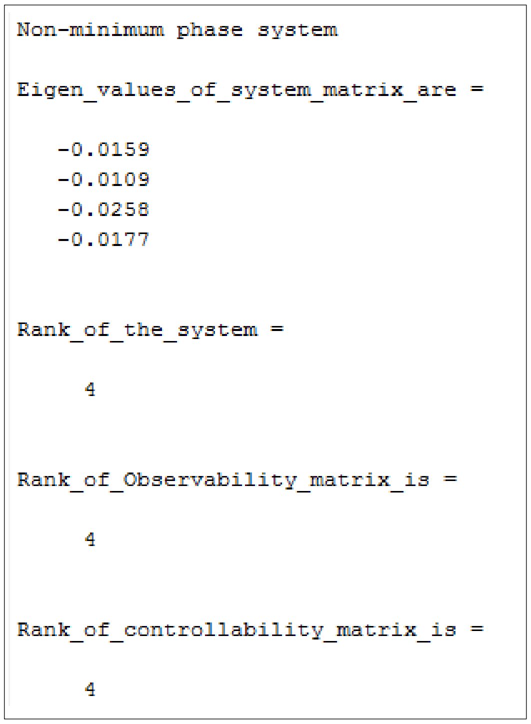


```
Minimum phase system

Eigen_values_of_system_matrix_are =

   -0.0160
   -0.0110
   -0.0440
   -0.0332


Rank_of_the_system =

     4


Rank_of_Observability_matrix_is =

     4


Rank_of_controllability_matrix_is =

     4
```

Figure 5.3: Observability and controllability analysis of minimum phase model

### 5.3.2 Non-minimum phase

The four-tank non-minimum phase process is stable as the real part of the eigenvalues are negative. The rank of observability and controllability matrices are equal to the rank of the system, therefore the four-tank non-minimum phase process is observable and controllable. A screen dump from the MATLAB code is shown in the Figure 5.4.



```
Non-minimum phase system

Eigen_values_of_system_matrix_are =

   -0.0159
   -0.0109
   -0.0258
   -0.0177


Rank_of_the_system =

     4


Rank_of_Observability_matrix_is =

     4


Rank_of_controllability_matrix_is =

     4
```

Figure 5.4: Observability and controllability analysis of non-minimum phase model

## 5.4 Implementation of MPC optimal control with integral action

MPC optimal control with integral action is implemented after analyzing the linearized model of the four-tank process. The mathematical formulation of MPC with integral action is given in section 3.4. The steps are also explained to achieve integral action from the discrete model of an augmented state space model. The sampling time of 0.1 second is used in all the experiments. The constrained and unconstrained MPC with integral are implemented in minimum and non-minimum cases. The constraints are handled using "*if-else*" method and MATLAB " *quadprog* " function, Kalman filter is used to for state estimation. Appropriate values of the weighting matrices $Q$ and $R$ are assigned to weighted the output and input variables respectively.

Where

$$Q = \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix}, \text{ and } R = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$$

### 5.4.1 Constrained MPC with integral action

Implementation of constrained MPC with integral action is divided into simulation of minimum and non-minimum phase process and the results are plotted. First of all, the state space continuous model is converted into discrete time using the "*c2dm*" function. The integral action in MPC is achieved by augmenting the state space model. The input amplitude constraints are implemented and voltage supply to the pumps restrained from minimum 0 [V] to maximum 5 [V]. The MPC with constraints becomes a quadratic programming problem that is solved by " *quadprog* " function. It is a built-in MATLAB function used to solve equality constraint. The "*if-else*" method also implemented to handle constraints in the MPC optimal controller. A MATLAB program is written for both operating points. The "*input"* function is used to select constrained MPC with integral action for either a minimum or non-minimum phase process by typing "1" or "2" respectively. The main script file along with six other supporting files is attached with necessary comments. The MATLAB file using '*quadprog*' function and '*if-else'* are provided in Appendices 4 and 5 respectively.

#### 5.4.1.1 Minimum phase system

In this case, the parameter values are taken from the Table 5.2 and constrained MPC with integral action is implemented. The initial level in tanks 1 and 2 are 11.5 and 12.5 [cm] respectively and small reference of ±0.1 [cm] is given by using the MATLAB m-file script "*prbs1.m"*. The sampling time of 0.1 second and prediction horizon $L$ of 15 seconds is used both strategies of constraints handling for comparison. The constrained MPC with integral action in the four-tank minimum phase system using "*quadprog*" function and "if-else" method is simulated, and results are discussed below with plots.

- **Using "*quadprog*" function**

The input amplitude constraints are implemented by using '*quadprog*' function and the control signals for pumps 1 and 2 are restricted to 0-5 [V] as shown in lower two plots of the Figure 5.5. The simulation is performed by changing the set point every 150 seconds and levels in Tanks 1 and 2 are controlled desirably. In the upper two plots, the red line and blue represents the set point and output level respectively.  The output level changes before the change in the set point which is the basic principle of the MPC optimal controller. The simulation result of four-tank minimum phase process by implementing the constrained MPC with integral action is illustrated in Figure 5.5, and the elapsed time for the simulation is 14.90 seconds.

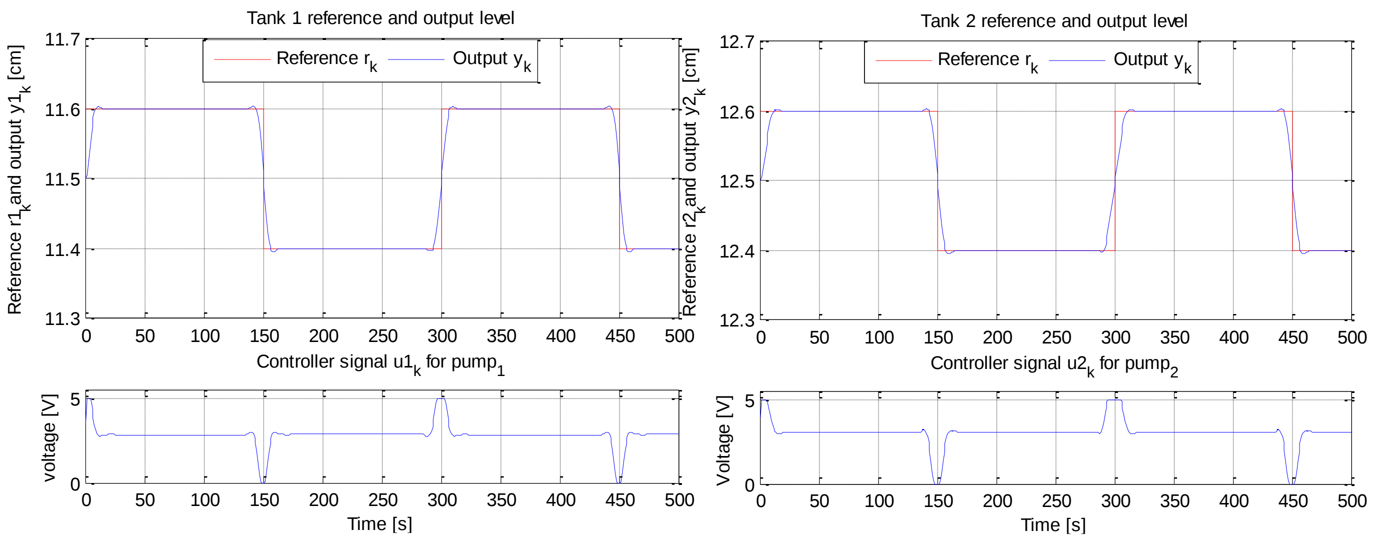


Figure 5.5: The simulation result of four-tank minimum phase process with constrained MPC with integral action using "*quadprog*" function. The upper two plots illustrated the reference signal and output levels for tanks 1 and 2. The lower two plots are the controller signals for pumps 1 and 2.

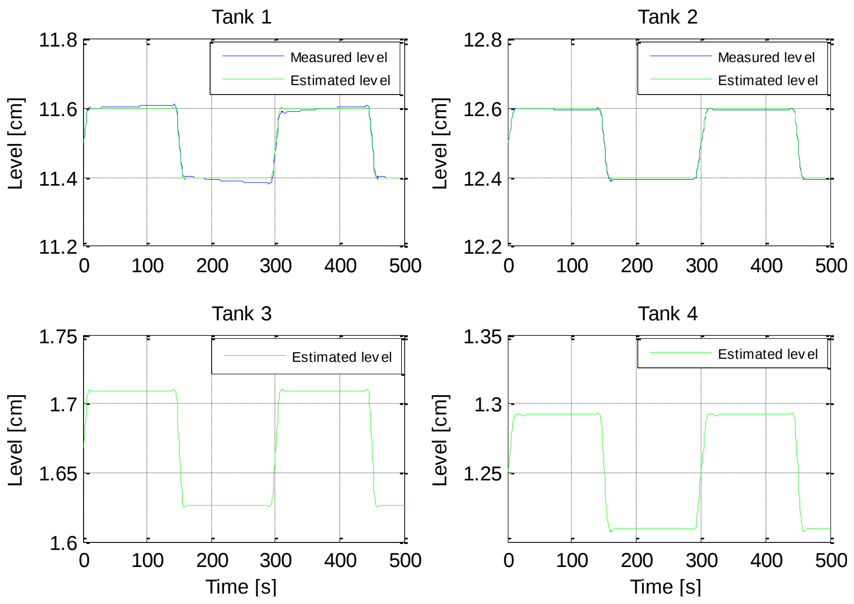


Figure 5.6: Results of estimated level during the implementation of constrained MPC with integral action using "*quadprog*" function  in four-tank minimum phase process. Upper two plots are a comparison of estimated vs measured level in tanks 1 and 2. The lower two plots are estimated level in Tanks 3 and 4.

The Kalman filter is implemented to estimate the states of the four tanks. The levels are measured only in tanks 1 and 2, therefore estimated and measured levels are compared in upper two plots. The lower two plots show the estimated levels in tanks 3 and 4. It can be seen from the Figure 5.6, the estimated and measured levels in tank 2 are similar, however there is a small difference in levels for tank 1. The blue line and green lines represent the measured and estimated levels respectively.

- **Using *"If-else"* method**

This strategy is implemented using if-else loop such that,

```
umin=0
umax=5

 if       u<umin
          u=umin;
 elseif   u>umax
          u=umax
 end
```

The parameter values used in the above method kept the same and simulation is performed. The simulation results of constrained MPC with integral action for the minimum phase system using "*if-else*" method is shown in Figure 5.7, and the elapsed time for the simulation is 3.40 seconds. A small undershoots and overshoots have been seen in both the tanks 1 and 2, especially in tank 2 when comparing it with the Figure 5.5. The MATLAB m-script file is attached in appendix 5.

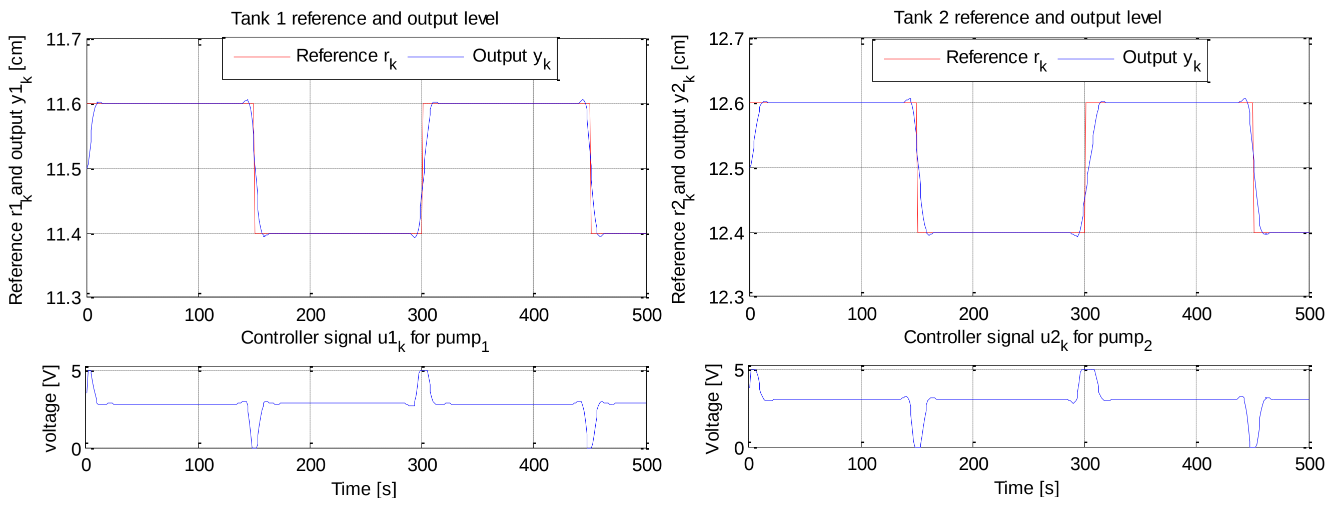


Figure 5.7: The simulation result of four-tank minimum phase process with constrained MPC with integral action using "*if-else*" method for constraints handling. The upper two plots illustrated the reference signal and output levels for tanks 1 and 2. The lower two plots are the controller input signal for pumps 1 and 2.

It can be seen from the Figure 5.8 there is a difference between the estimated and measured levels in tanks 1 and 2. However, these differences are smaller when "*quadprog*" function is used to handle constraints as shown in Figure 5.6. The blue line and green lines represent the measured and estimated levels respectively.

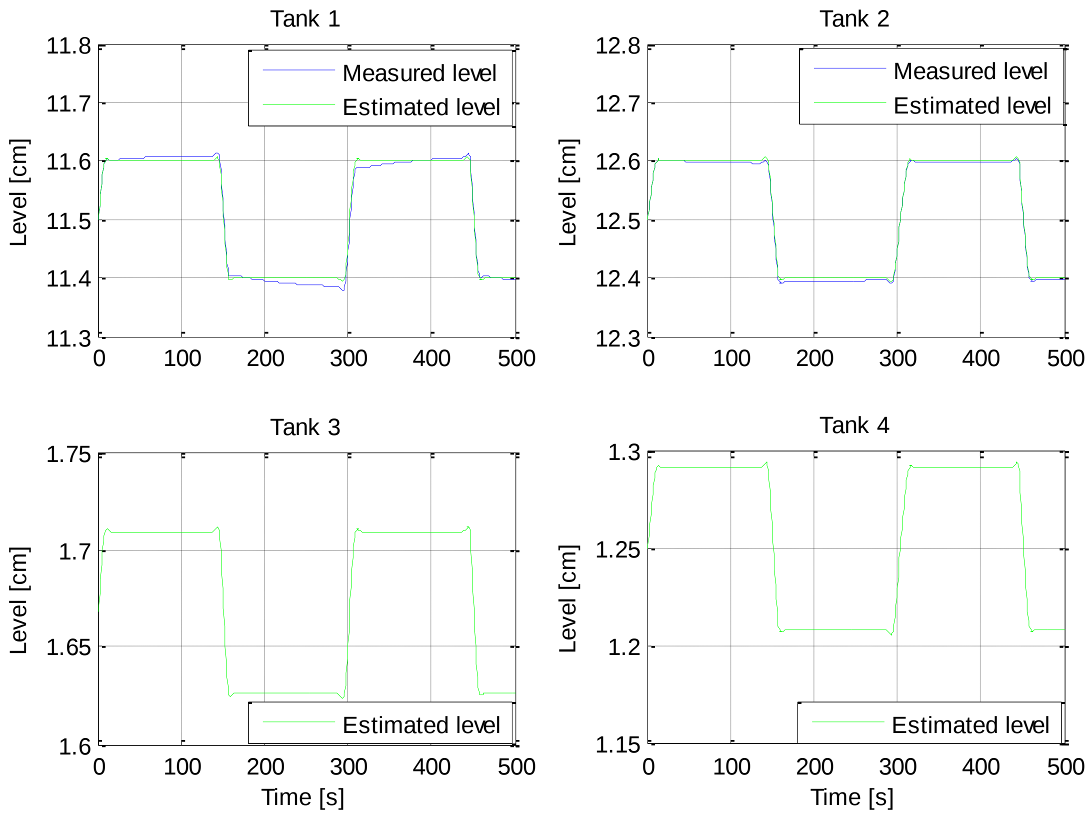


Figure 5.8: Results of estimated level during the implementation of constrained MPC with integral action using "*if-else*" method in four-tank minimum phase process. Upper two plots are a comparison of estimated *vs* measured level in tanks 1 and 2. The lower two plots are estimated level in Tanks 3 and 4.

- **Comparison remarks**

It is interesting that the elapse time for simulation in '*if-else'* method is smaller compared with the "*quadprog*" strategy. The reason for a higher execution time in latter strategy is that it has to execute a lot of complex calculations.

### 5.4.1.2 Non-minimum phase system

The parameter values for the non-minimum phase are taken from the Table 5.4 and constrained MPC with integral action is implemented similar to the minimum phase system. The initial level in tanks 1 and 2 is set to 11.5 and 12.5 [cm] respectively and small reference of ±0.1 [cm] is given using the MATLAB m-file script "*prbs1.m"*. The sampling time of 0.1 second and prediction horizon of 15 seconds are used in both strategies of constraints handling for comparison. The constrained MPC with integral action in four-tank non-minimum phase system is simulated, and results are discussed below with plots.

- **Using "*quadprog*" function**

In this method, the input amplitude constraints are implemented and the control signal for pumps 1 and 2 are restricted to 0-5 [V] as shown in lower two plots of the Figure 5.9. The simulation is performed using the same method as in minimum phase system, and the levels in Tanks 1 and 2 are controlled according to specified reference. It can be seen from the plots that the change in output levels is slower than minimum phase process. The simulation result of four-tank non-minimum phase process by implementing the constrained MPC with integral action is illustrated in Figure 5.9, and the elapsed time of 16.14 seconds for the simulation is observed.

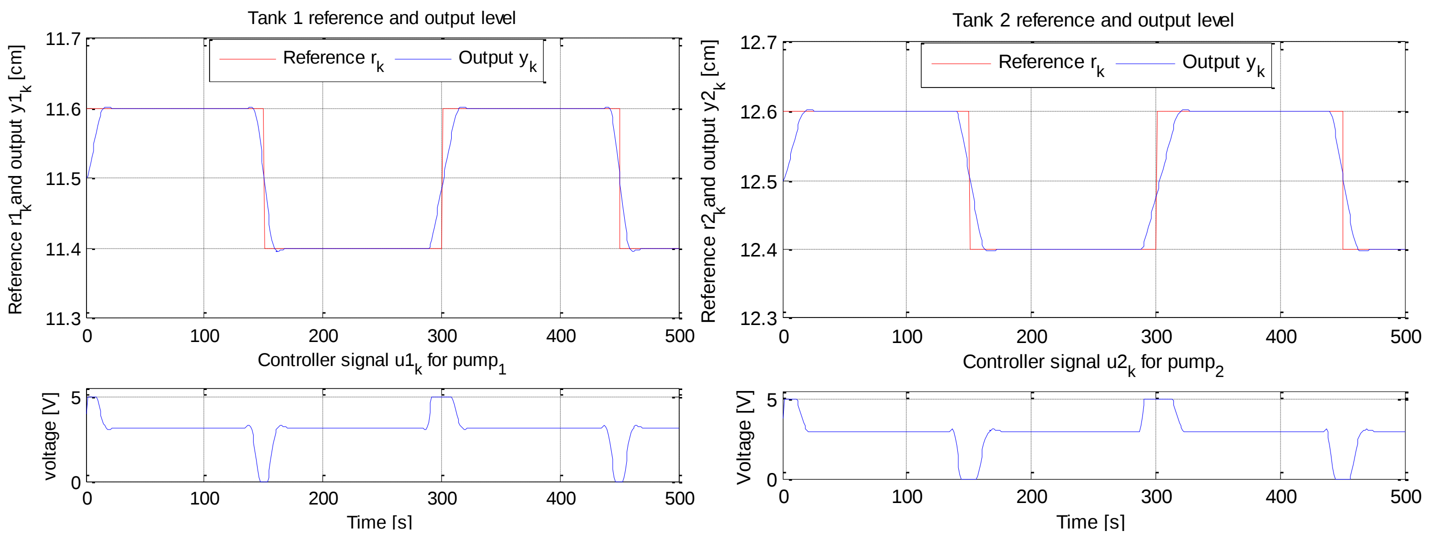


Figure 5.9: The simulation result of four-tank non-minimum phase process with constrained MPC with integral action using "*quadprog*" function for constraint handling. The upper two plots illustrated the reference signal and output levels for tanks 1 and 2. The lower two plots are the controller input signal for pumps 1 and 2.

The figure 5.10 illustrates the comparison of estimated and measured levels for tanks 1 and 2 in upper two plots and the estimated levels for tanks 3 and 4 in lower two plots. In this case, the estimated and measured levels are not matched with each other for tank 1, but the difference is small in tank 2. The blue line and green lines represent the measured and estimated levels respectively.

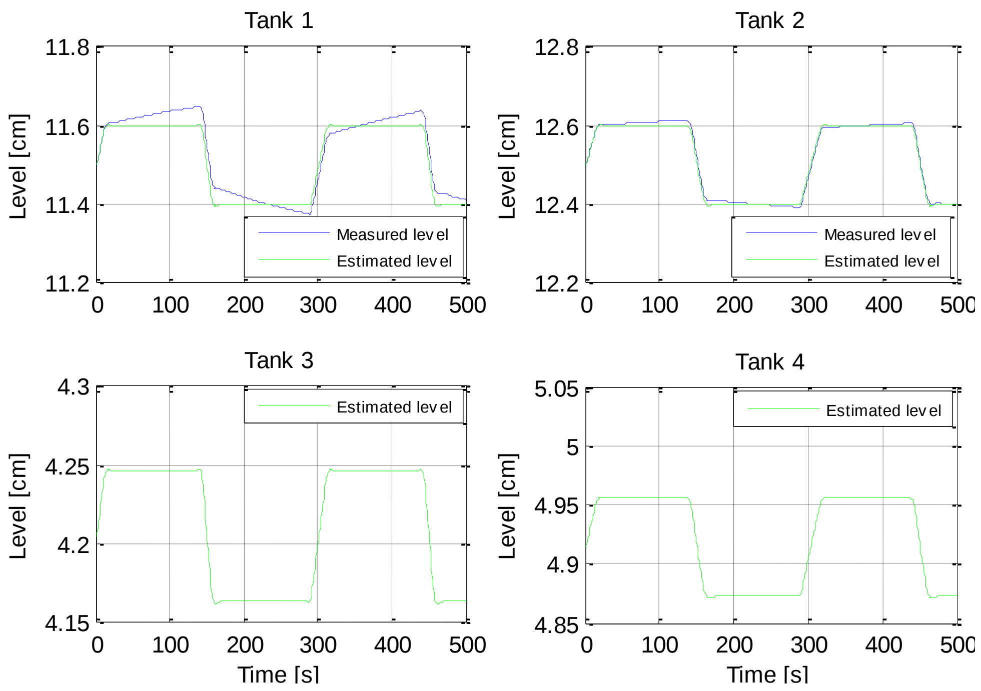


Figure 5.10: Results of estimated level during the implementation of constrained MPC with integral action using "*quadprog*" function in four-tank non-minimum phase process. Upper two plots are a comparison of estimated *vs* measured level in tanks 1 and 2. The lower two plots are estimated level in Tanks 3 and 4.

- **Using "*If-else*" method**

The constraints in MPC with integral action are handled using "*if-else*"method similar to the one implemented in the minimum phase process. The parameter values used in the above method kept the same and simulation is performed. The elapsed time for the simulation is 5.67 that is again smaller compared with the "*quadprog*" method. The simulation results for non-minimum phase system using "*if-else*" method are shown in Figure 5.11 and controller response in this case is slow. A small undershoots and overshoots have been seen in both the tanks 1 and 2. The MATLAB m-script file is attached in appendix 5.

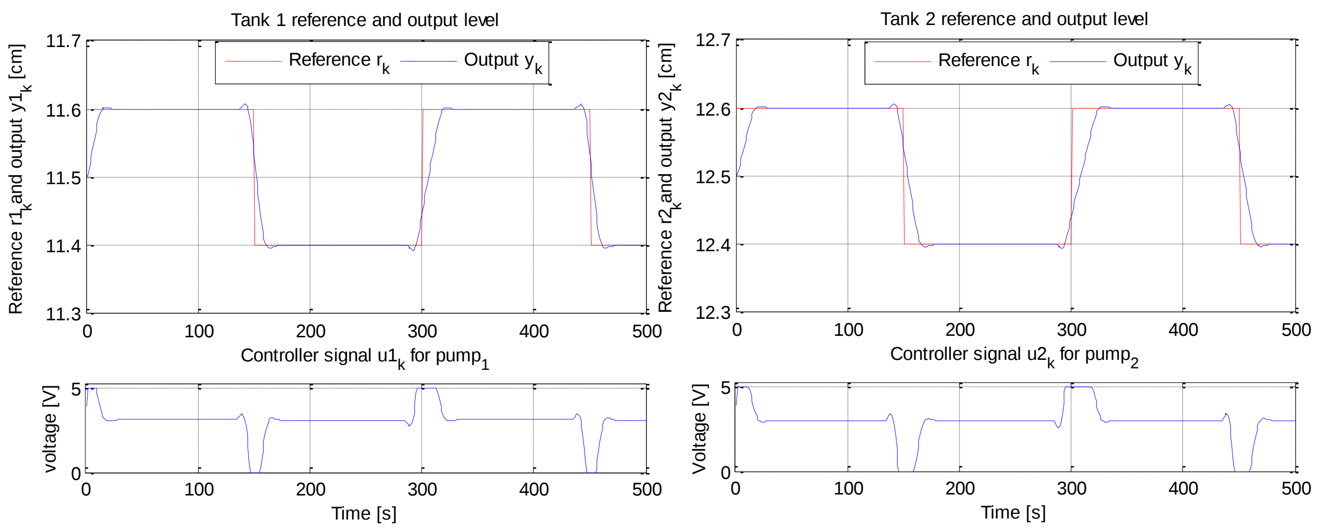


Figure 5.11: The simulation result of four-tank non-minimum phase process with constrained MPC with integral action using *"if-else"* for constraints handling. The upper two plots illustrated the reference signal and output levels for tanks 1 and 2. The lower two plots are the controller input signal for pumps 1 and 2.

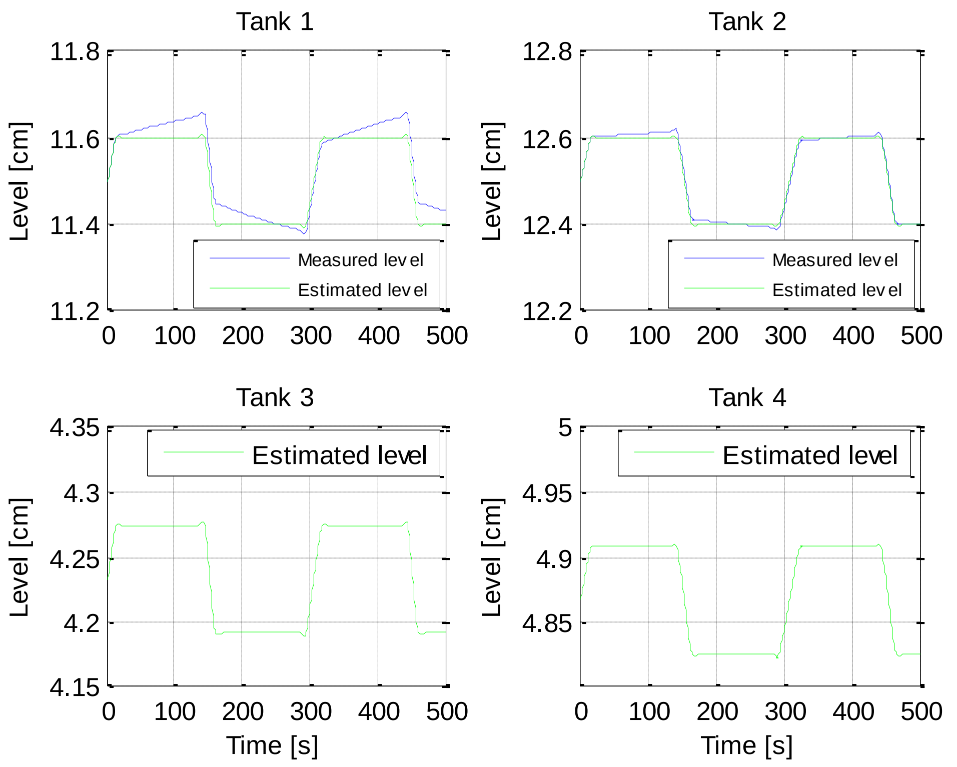


Figure 5.12 Results of estimated levels during the implementation of constrained MPC with integral action using "*if-else*" method in four-tank non-minimum phase process. Upper two plots are a comparison of estimated *vs* measured level in tanks 1 and 2. The lower two plots are estimated level in Tanks 3 and 4.

The figure 5.12 illustrates the comparison of estimated and measured levels for tanks 1 and 2 in upper two plots. The estimated levels for tanks 3 and 4 are shown in lower two plots. It can be seen that the difference between the estimated and measured levels in upper two plots increased as compared to the Figure 5.10, when "*quadprog*" was used to handle constraints.

- **Comparison remarks**

The output level in non-minimum phase is changing slowly compared with minimum phase system. The control signals to the pumps 1 and 2 are confined within the limit of 0-5 [V] using two different approaches. The execution time is smaller by using "*if-else*" method, on the other hand, the performance of the controller is better when constraints are solved by "*quadprog*" function in the MPC optimal controller.

## 5.4.2 Unconstrained MPC with integral action

In this section, the simulations of MPC with integral action in minimum and non-minimum phase systems are performed without implementing the constraints on input voltage. A MATLAB program is written for both operating points, and the " *input* " function is used to select unconstrained MPC with integral action for either a minimum or non-minimum phase process by typing " 1 " or " 2 " respectively. The main script file is attached in Appendix 6 with necessary comments.

### 5.4.2.1 Minimum phase system

For minimum phase system, the parameter values are taken from the Table 5.2, and MPC with integral action is implemented. The initial levels in tanks 1 and 2 are 11.5 and 12.5 [cm] respectively and small reference of ±0.2 [cm] is given using the MATLAB m-file script "*prbs1.m*". The sampling time of 0.1 second and prediction horizon of 15 seconds is used. The simulation is performed by changing the set point every 150 seconds. However, the controller has higher undershoots and overshoots compared with the constrained MPC algorithm as presented in prevision section. The red line and blue represent the set point and output level respectively in upper two plots of the Figure 5.13. The lower two plots shows the input signals for pumps 1 and 2 varying from -8 to 12 [V]. It is not good practice to use voltage higher than what is needed. Moreover, it is not only the wastage of energy but also affect the proper working of the device. The reasons for bigger undershoots and overshoots is that control signals are not restricted to minimum or maximum limits. The simulation result of four-tank minimum phase process by implementing the unconstrained MPC with integral action is illustrated in Figure 5.13.

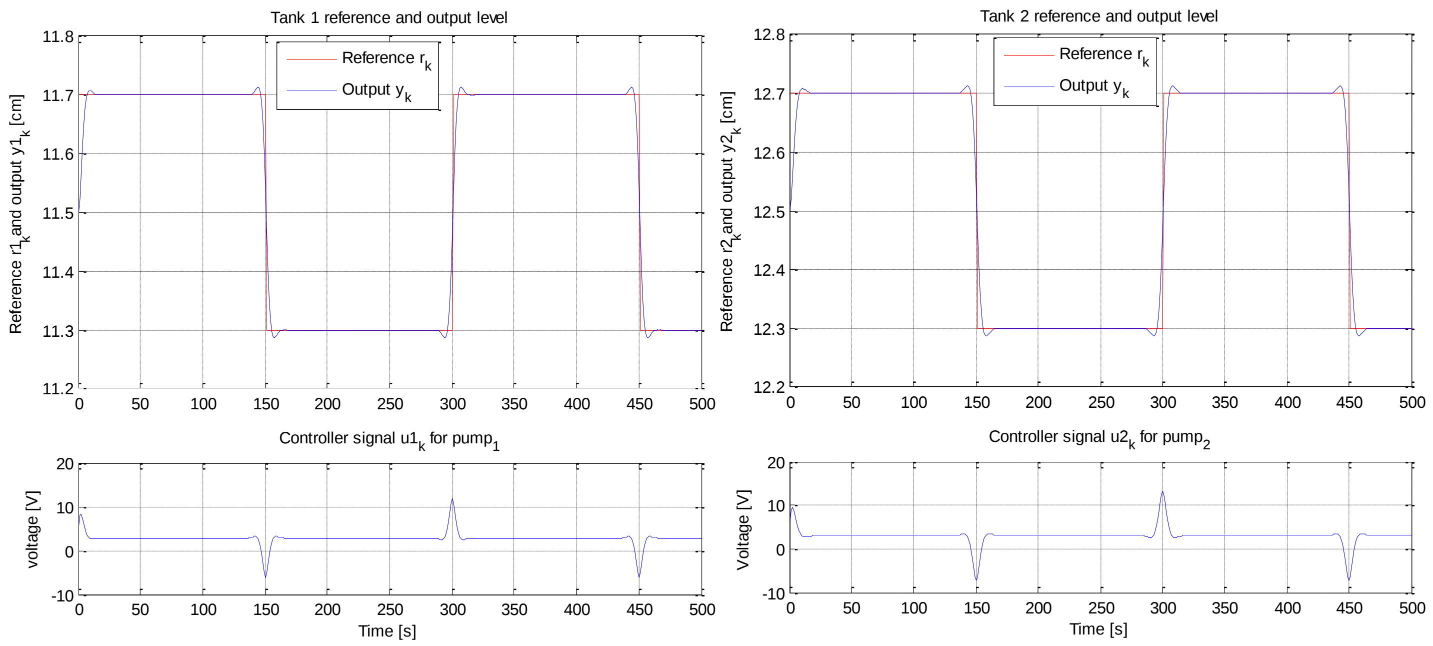


Figure 5.13:The simulation result of four-tank minimum phase process with unconstrained MPC with integral action. The upper two plots illustrated the reference signal and output levels for tanks 1 and 2. The lower two plots are the controller input signal for pumps 1 and 2.

It can be seen from the Figure 5.14 the estimated and measured levels in tank 2 have a small difference however, in tank 1 the measured level is higher than the estimated level. The lower two plots show the estimated levels in tanks 3 and 4. The blue line and green lines represent the measured and estimated levels respectively.

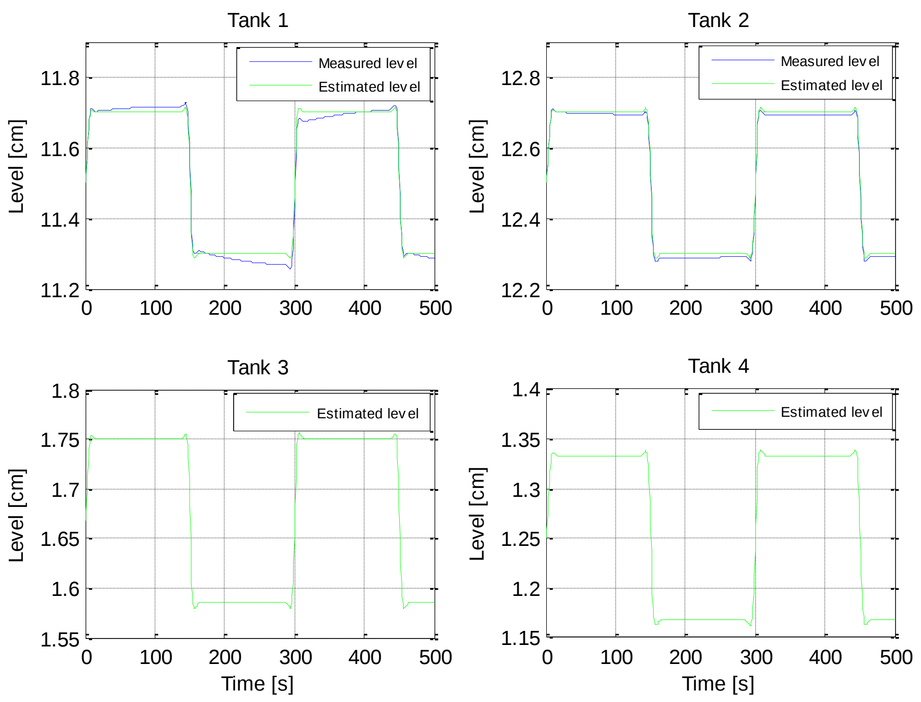


Figure 5.14: Results of estimated levels during the implementation of unconstrained MPC with integral action in four-tank minimum phase process. Upper two plots are a comparison of estimated *vs* measured level in tanks 1 and 2. The lower two plots are estimated level in tanks 3 and 4.

### 5.4.2.2 Non-minimum phase system

For non-minimum phase system, the parameter values are taken from the Table 5.4, and similar method as in minimum phase process is implemented. The initial level in tanks 1 and 2 are 11.5 and 12.5 [cm] respectively and small reference of ±0.2 [cm] is given using the MATLAB m-file script "*prbs1.m*". The sampling time of 0.1 second and prediction horizon of 15 seconds is used. The simulation is performed by changing the set point every 150 seconds. The controller achieves the set point. However, it has more higher undershoots and overshoots than minimum phase process. The red line and blue represent the set point and output level respectively in upper two plots of the Figure 5.15. It can be seen from the lower plots, the input signals for pumps 1 varying from -8 to 15 [V] and for pump 2 varying from -10 to 18 [V]. The reasons for bigger undershoots and overshoots is that control signals are not restricted to minimum or maximum limits. The simulation result of four-tank minimum phase process by implementing the unconstrained MPC with integral action is illustrated in Figure 5.15.

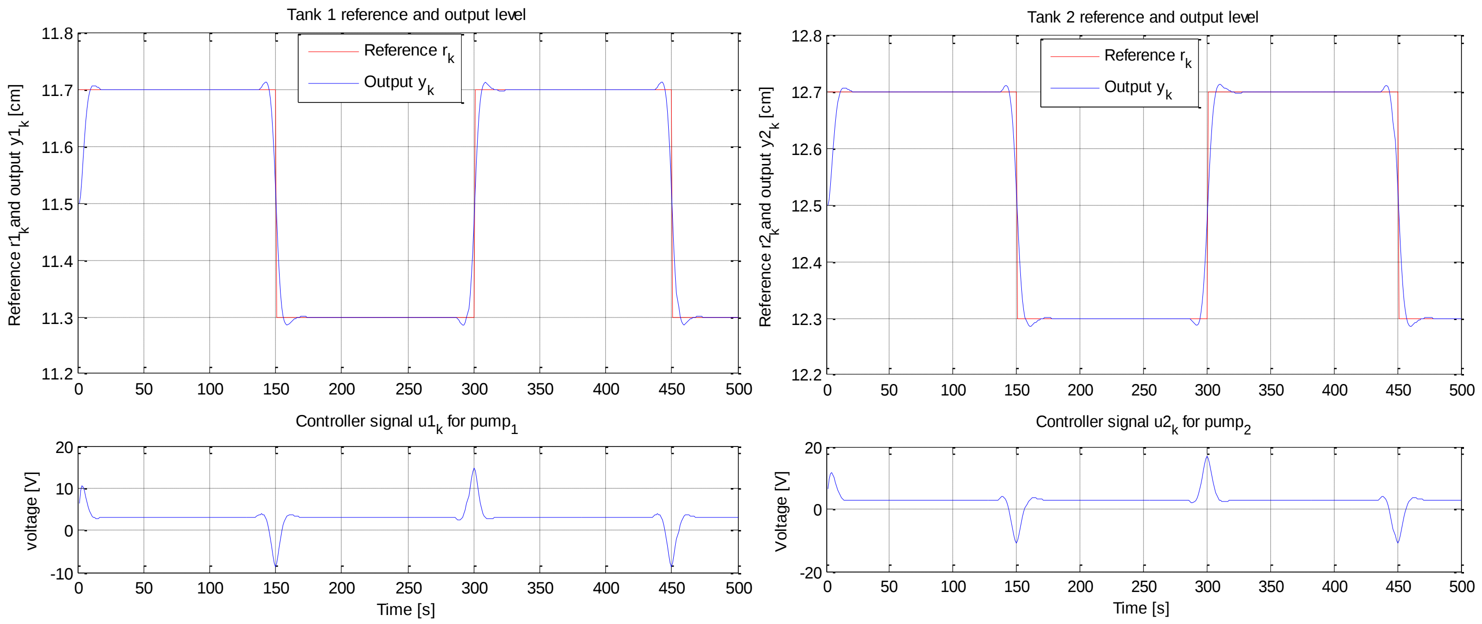


Figure 5.15: The simulation result of four-tank non-minimum phase process with unconstrained MPC with integral action. The upper two plots illustrated the reference signal and output levels for tanks 1 and 2. The lower two plots are the controller input signal for pumps 1 and 2.

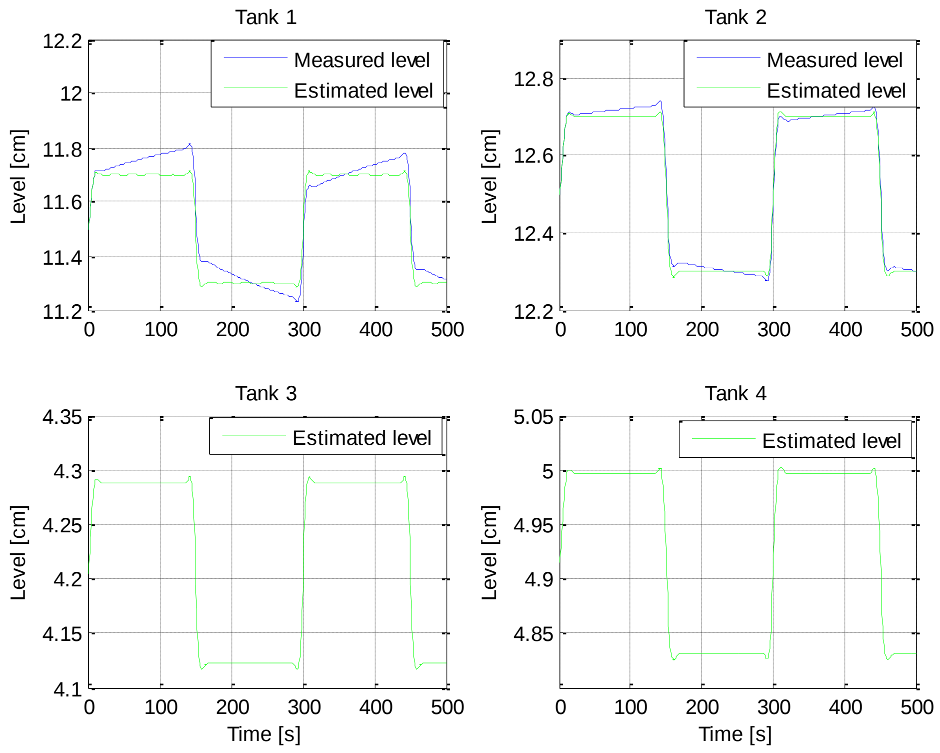


Figure 5.16: Results of estimated levels during the implementation of unconstrained MPC with integral action. Upper two plots are a comparison of estimated *vs* measured level in tanks 1 and 2. The lower two plots are estimated level in Tanks 3 and 4.

The difference between the estimated and measured levels in the tank 1 is quite large compared with the minimum phase process. It can be seen from the Figure 5.16 that the difference in tank 2 is not small, however it is following the trend. The lower two plots show the estimated levels in tanks 3 and 4. The blue line and green lines represent the measured and estimated levels respectively.

### 5.4.3 Performance comparison in reducing control horizon

The control horizon defines as the number of samples within the prediction horizon $L$ of which the MPC optimal controller could affect the control action. The MPC optimal controller tries to become aggressive if the number of control horizon increase. The reason for aggressive behavior is due to increase in computational requirement [16]. Reducing the number of unknown future controls, the cost function will be [15],

$$J_k = \sum_{i=1}^{L} ((y_{k+i} - r_{k+i})^T Q_i (y_{k+i} - r_{k+i}) + u_{k+i-1}^T P_i u_{k+i-1}) + \sum_{i=1}^{Lu} \Delta u_{k+i-1}^T R_i \Delta u_{k+i-1} \qquad (5.1)$$

In the cost function given in the equation (5.1), $L$ is prediction horizon and $Lu$ is control horizon. In the previous section, MPC with integral action for minimum and non-minimum phase process, control horizon was equal to prediction horizon i.e. $Lu = L.$ However, in reducing the number of unknown future controls, the control horizon is chosen such that $1 \le Lu < L$ and performance of MPC with integral action is compared. The combined MATLAB program is written for both operating points, and the "*input"* function is used to select MPC with integral action for either a minimum or non-minimum phase process by typing "1" or "2" respectively. The program m-file script is provided in Appendix 7, and different values of control horizon are tested.

#### 5.4.3.1 Minimum phase

Reducing the control horizon in MPC optimal control, three cases are formulated in minimum phase process to check the performance of the controller. The control horizon with different values of 10, 4 and 2 seconds in MPC method are used, and results are discussed. The sampling time of 0.1 second and prediction horizon of 15 seconds kept constant for all cases. In section 5.4.1, MPC with integral action using control horizon equal to prediction horizon already implemented, therefore it is not repeated here. The parameter values are taken from the Table 5.2 and constrained MPC with integral action is implemented. The initial level in tanks 1 and 2 are 11.5 and 12.5 [cm] respectively and small reference of ±0.1 [cm] is given using the MATLAB m-file script *"prbs1.m"*. The simulation is performed by changing the set point every 100 seconds.

- **Case 1: Lu=10 and L=15**

  In this case, a control horizon of 10 and prediction horizon of 15 seconds are used and the output levels are controlled according to the given set points as illustrated in Figure 5.17. There are small undershoots in both tanks 1 and 2. The MPC with integral using control horizon 10 give better performance as compared to the case when $Lu = L$ shown in the Figure 5.5. In the upper two plots, the red line and blue represent the set point and output level respectively whereas in lower two plots, control signal for pumps 1 and 2 are shown.

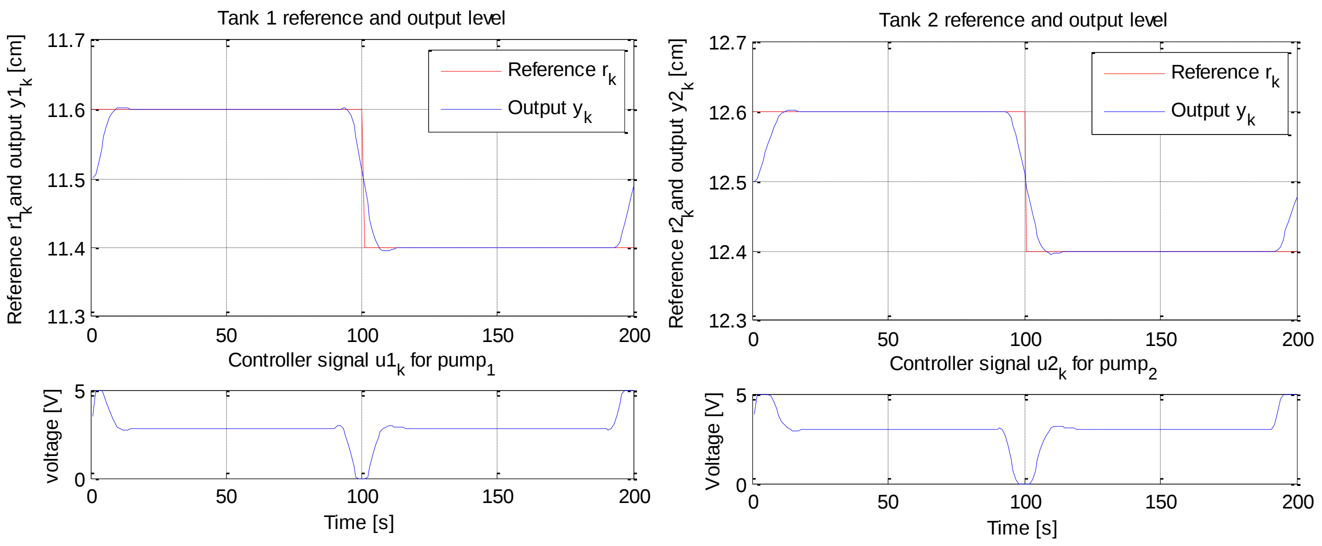


Figure 5.17: Simulation of four-tank minimum phase process with MPC with integral action in reducing control horizon. Control horizon Lu=10, Prediction horizon L=15.

- **Case 2: Lu=4 and L=15**

  In this case, a control horizon of 4 and prediction horizon of 15 seconds is used and then simulation is performed. There are slightly bigger undershoots and overshoots for both the tanks 1 and 2 as compared to the case when control horizon was 10 seconds. The settling time is almost 20 seconds. The simulation results from implementing MPC with integral action using control horizon of 4 seconds are illustrated in the Figure 5.18, the red line and blue represent the set point and output level in upper two plots respectively. The control input signals for pumps 1 and 2 are shown in lower two plots of the Figure 5.18.

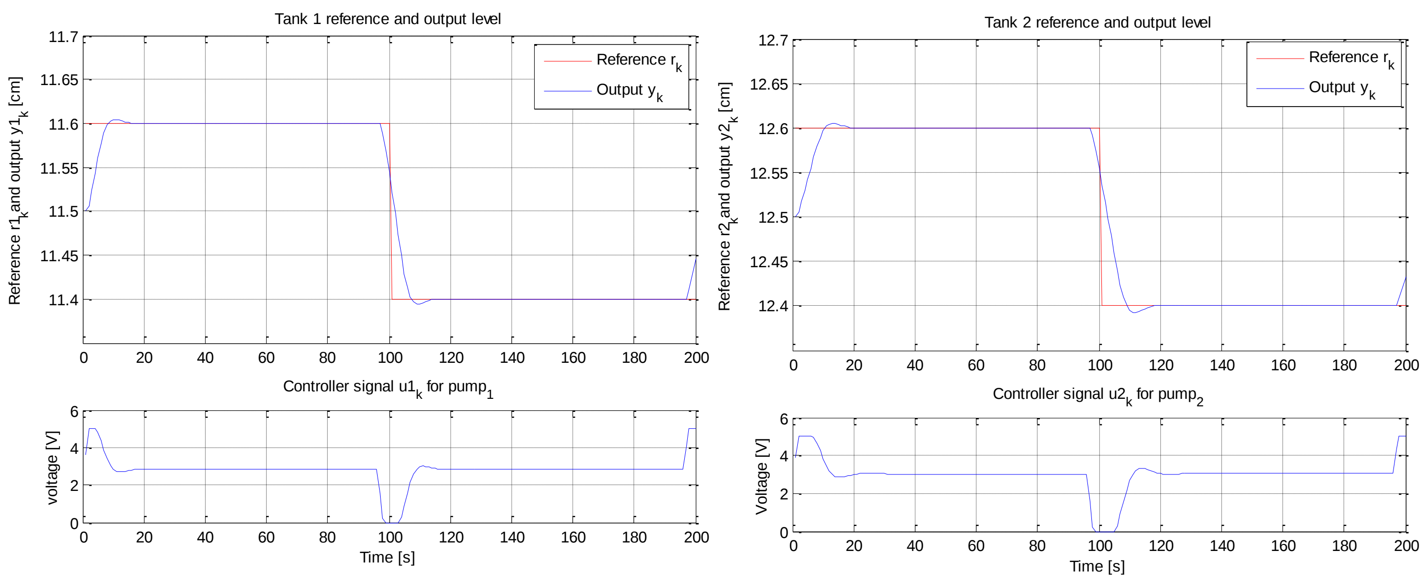


Figure 5.18: Simulation of four-tank minimum phase process with MPC with integral action in reducing control horizon. Control horizon Lu=4, Prediction horizon L=15.

- **Case 3: Lu=2 and L=15**

  An extreme case when the control horizon of 2 seconds is used in MPC with the integral action algorithm. The output level for both tanks showed fluctuation and behavior is more like a PI controller. The settling time is double as compared to above cases and it reached 50 and 40 seconds for tanks 1 and 2 respectively. The simulation results from implementing MPC with integral action using control horizon of 2 seconds are illustrated in the Figure 5.19, where the red line and blue represent the set point and output level in upper two plots respectively. The control input signals also have a lot of variation between 0-5 [V] for pumps 1 and 2 as shown in lower two plots of the Figure 5.19.

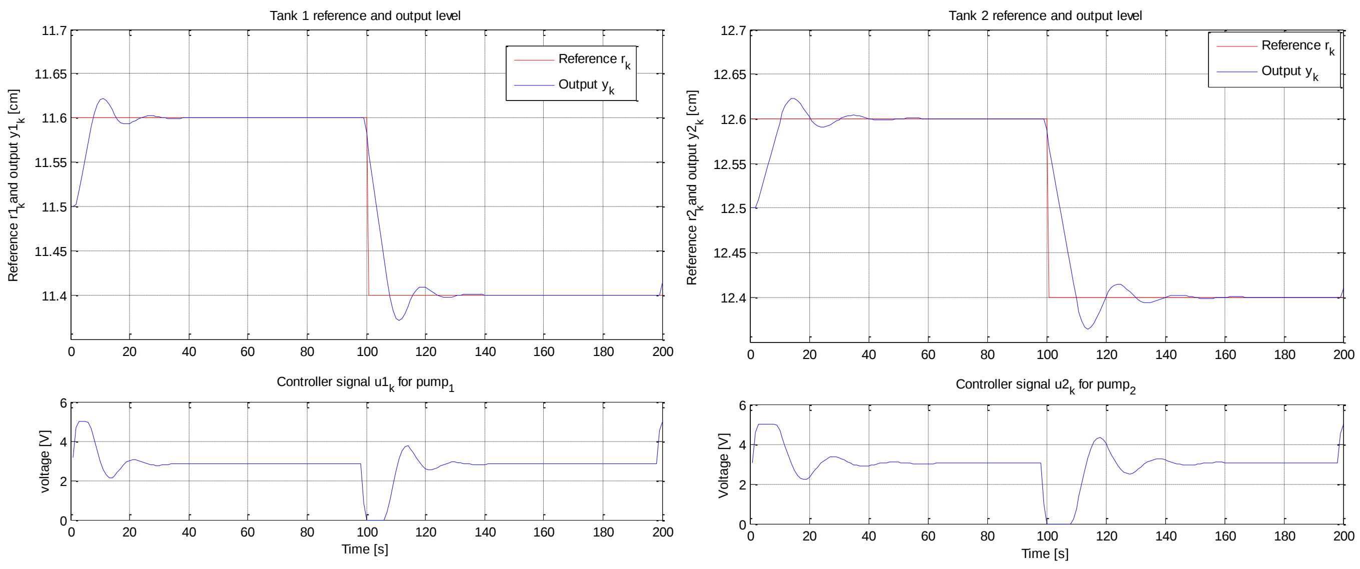


Figure 5.19: Simulation of four-tank minimum phase process with MPC with integral action in reducing control horizon. Control horizon Lu=2, Prediction horizon L=15

### 5.4.3.2 Non-minimum phase

Similar to the minimum phase process, three cases are tested in non-minimum phase process to compare the performance of the MPC with integral action in reducing the control horizon. The sampling time of 0.1 second and prediction horizon of 15 seconds kept constant for all the cases. In section 5.4.1, MPC with integral action using control horizon equal to prediction horizon already implemented, therefore it is not repeated here. The parameter values are taken from the Table 5.4 and constrained MPC with integral action is implemented. The initial level in tanks 1 and 2 are 11.5 and 12.5 [cm] respectively and small reference of ±0.1 [cm] is given using the MATLAB m-file script "*prbs1.m*". The simulation is performed by changing the set point every 100 seconds.

- **Case 1: Lu=10 and L=15**

  In this case, a control horizon of 10 seconds is used and the levels are achieved nicely as illustrated in Figure 5.20. It was interesting that MPC with integral using control horizon 10 give better performance as compared to the case when $Lu = L$ shown in the Figure 5.9. In the upper two plots, the red line and blue represent the set point and output level respectively. In lower two plots, the control signal for pumps 1 and 2 are shown.

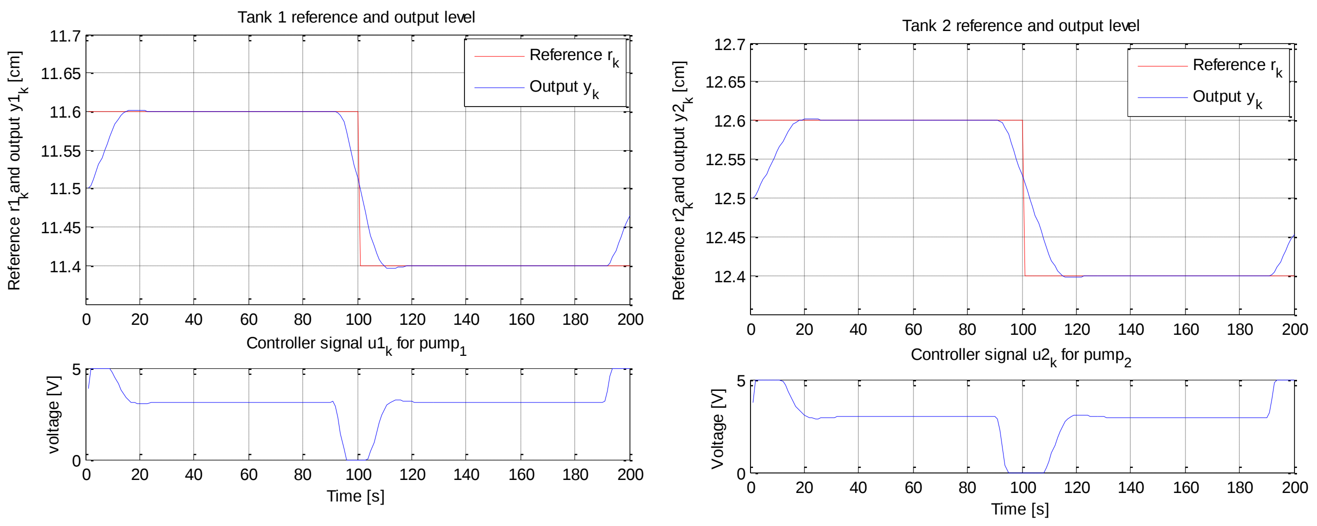


Figure 5.20: Simulation of four-tank non-minimum phase process with MPC with integral action in reducing control horizon. Control horizon Lu=10 and Prediction horizon L=15

- **Case 1: Lu=4 and L=15**

  In this case, a control horizon of 4 seconds is used and then simulation is performed. There are variations in the start for both the tanks as compared to the above case and after 40 seconds the output level reached the set point. The simulation results from implementing MPC with integral action using control horizon of 4 seconds are illustrated in the Figure 5.21, where the red line and blue represent the set point and output level in upper two plots respectively. In lower two plots, the control signal for pumps 1 and 2 are shown.

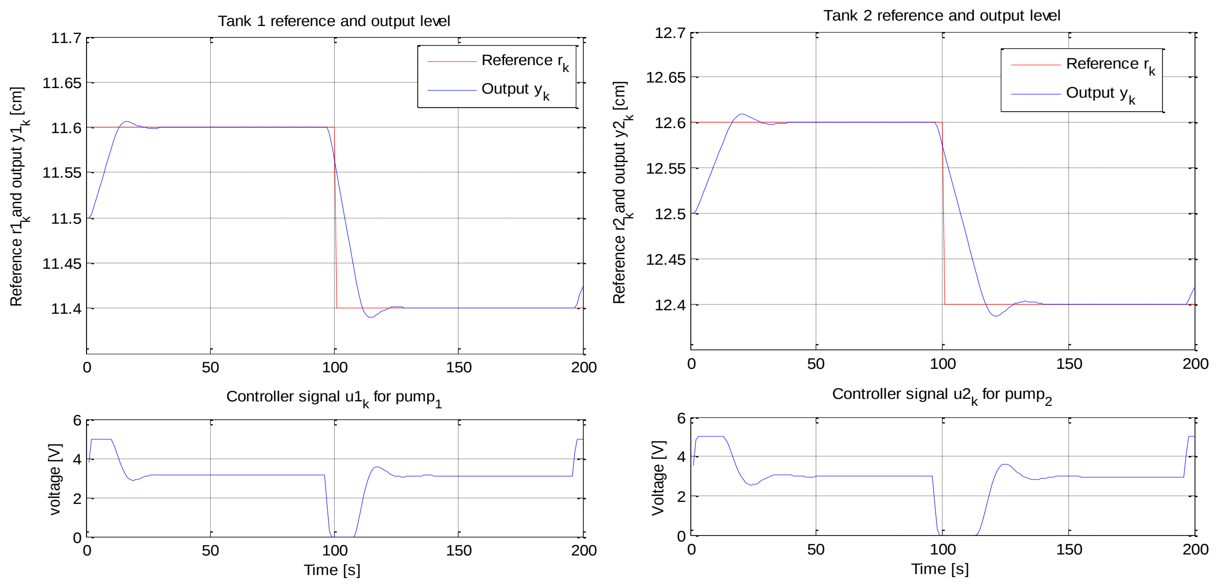


Figure 5.21: Simulation of four-tank non-minimum phase process with MPC with integral action in reducing control horizon. Control horizon Lu=4, Prediction horizon L=15

- **Case 1: Lu=2 and L=15**

  An extreme case when the control horizon of 2 seconds is used in MPC with the integral action algorithm. The output level for both tanks showed a lot of fluctuation around the set point. The controller achieves the set point after 65 seconds in tank 1. However it was not able to reach the desired reference point in tank 2. The simulation results from implementing MPC with integral action using control horizon of 2 seconds are illustrated in the Figure 5.22, where the red line and blue represent the set point and output level in upper two plots respectively. The control input signals also have a lot of variation between 0-5 [V] for pumps 1 and 2 as shown in lower two plots of the Figure 5.22.

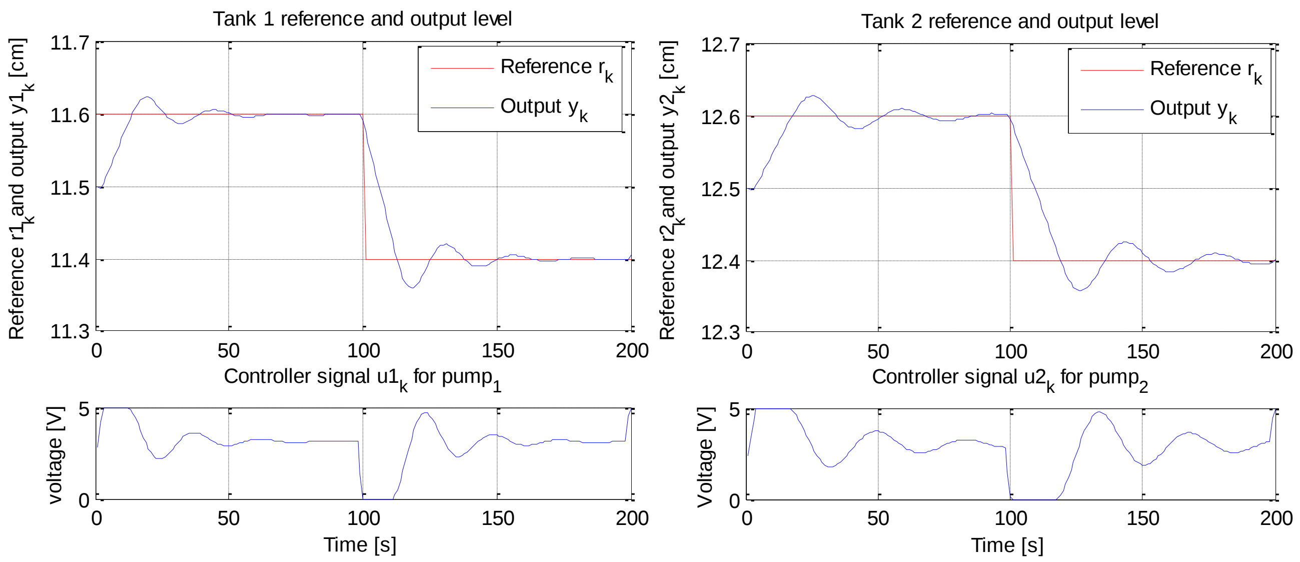


Figure 5.22: Simulation of four-tank non-minimum phase process with MPC with integral action in reducing control horizon. Control horizon Lu=2, Prediction horizon L=15

## 5.5 Implementation of PI controller

The PI classical controllers are widely used in control engineering practice for the last seventy years, and the reason for commonly used control technique is that they are easy to implement. Decentralized or multi-loop PI controllers are used to control MIMO[2] system [40, 41]. Such a decentralized PI controller is implemented to control the four-tank process, and the idea behind the implementation is to compare the result with the MPC optimal controller. In this method, two PI controllers $c_1$ and $c_2$ are controlling two pumps. The controller $c_1$ and $c_2$ take the feedback

[2] Multiple Input and Multiple Output

from the outputs $y_1$ and $y_2$ respectively and calculated the inputs $u_1$ and $u_2$. Then these input signals send to the process $G$. A decentralized PI control system for the four-tank process is illustrated in Figure 5.23. The controller output signal $u_1$ and $u_2$ are calculated as given in equations (5.2) and (5.3) respectively [35].

$$u_1 = K_{p1}e_1 + \frac{K_{p1}}{T_{i1}}t_s e_1 \tag{5.2}$$

$$u_2 = K_{p2}e_2 + \frac{K_{p2}}{T_{i2}}t_s e_2 \tag{5.3}$$

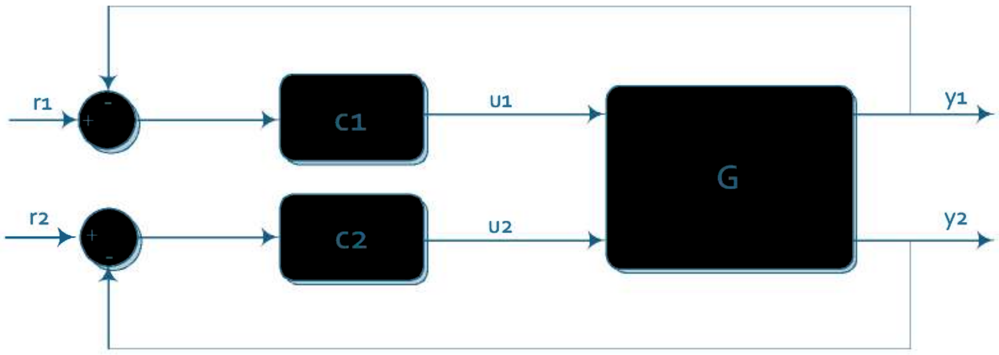


Figure 5.23: Structure of decentralized PI control with two PI controllers [35]

## 5.5.1 RGA analysis

The Relative Gain Array (RGA) is a powerful tool used as an interaction measure for control systems having multiple variables [42], hence it is used here to select input-output pairing. Johansson [35] has defined RGA matrix as,

$$\Lambda = G(0) * G^{-T}(0) \tag{5.4}$$

Where $\Lambda$ is RGA matrix, asterisk (*) is an element by element multiplication and $G$ is the transfer function of the four-tank process defined in section (4.1.3) as,

$$G(s) = \begin{bmatrix} \frac{\gamma_1 c_1}{(1+sT_1)} & \frac{(1-\gamma_2)c_1}{(1+sT_1)(1+sT_3)} \\ \frac{(1-\gamma_1)c_2}{(1+sT_2)(1+sT_4)} & \frac{\gamma_2 c_2}{(1+sT_2)} \end{bmatrix} \tag{5.5}$$

Where,

$$c_1 = \frac{T_1 k_1 k_c}{A_1}$$

$$c_2 = \frac{T_2 k_2 k_c}{A_2},$$

The transfer functions for minimum and non-minimum phase process are calculated by solving equation (5.5), and the parameter values are given in Tables 5.1, 5.2 and 5.4. The transfer function for minimum phase gives,

$$G_{-}(s) = \begin{bmatrix} \frac{2.6}{(1+62s)} & \frac{1.5}{(1+23s)(1+62s)} \\ \frac{1.4}{(1+30s)(1+90s)} & \frac{2.8}{(1.90s)} \end{bmatrix} \tag{5.6}$$

And then solving equation (5.5) for the non-minimum phase process gives,

$$G_{+}(s) = \begin{bmatrix} \frac{1.5}{(1+63s)} & \frac{2.5}{(1+39s)(1+63s)} \\ \frac{2.5}{(1+56s)(1+91s)} & \frac{1.6}{(1.91s)} \end{bmatrix} \tag{5.7}$$

Where $G_{-}(s)$ and $G_{+}(s)$ are transfer function matrices of minimum and non-minimum phase process. When $s = 0$ then the equations (5.6) and (5.7) becomes,

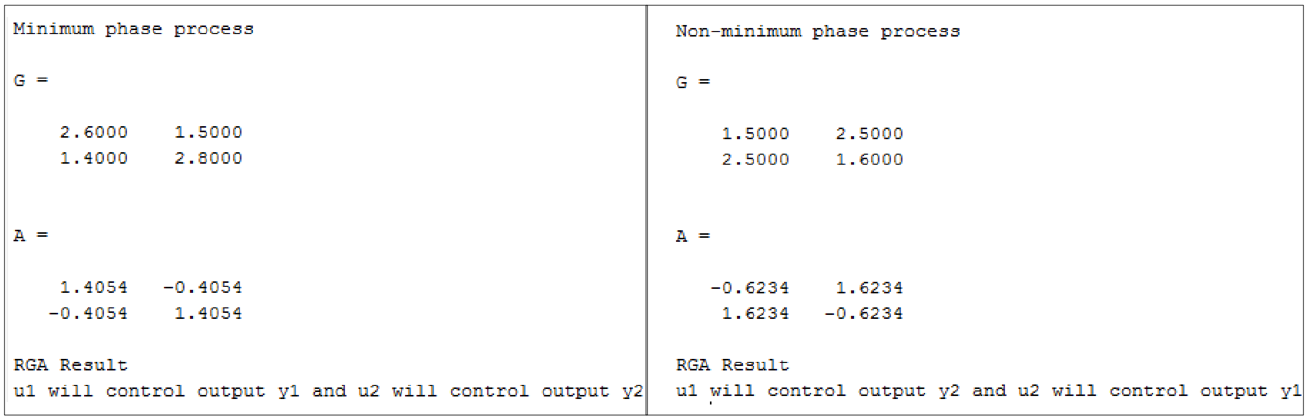
$$G_{-}(0) = \begin{bmatrix} 2.6 & 1.5 \\ 1.4 & 2.8 \end{bmatrix} \tag{5.8}$$

$$G_{+}(0) = \begin{bmatrix} 1.5 & 2.5 \\ 2.5 & 1.6 \end{bmatrix} \tag{5.9}$$

The pairing of variables based on the RGA analysis are described by Rusico [32]. According to him the starting point is the element $\lambda_{ij}$ in RGA matrix.

- Select the pairing $u_j \to y_i$ for which the corresponding element $\lambda_{ij}$ of RGA is positive and magnitude close to one as possible.
- The pairing $u_j \to y_i$ must be avoided if the element of RGA is negative such that $\lambda_{ij} < 0$.

Substituting the equation (5.8) and (5.9) separately and performed the RGA analysis by using MATLAB. The optimal input-output variable pairing based on the above rules is selected for the four-tank minimum and non-minimum phase process as shown in the Figure 5.24. The m-file script is provided in Appendix 8.



```
Minimum phase process

G =

    2.6000    1.5000
    1.4000    2.8000


A =

    1.4054   -0.4054
   -0.4054    1.4054

RGA Result
u1 will control output y1 and u2 will control output y2
```

```
Non-minimum phase process

G =

    1.5000    2.5000
    2.5000    1.6000


A =

   -0.6234    1.6234
    1.6234   -0.6234

RGA Result
u1 will control output y2 and u2 will control output y1
```

Figure 5.24: RGA analysis to determine the optimal input-output variable pairing for four-tank process.

## 5.5.2 Minimum phase process

According to RGA analysis of the minimum phase system, the input $u_1$ will control the output $y_1$ and input $u_2$ will control the output $y_2$, therefore a decentralized PI controller is implemented to achieve the target. The simulation is performed for 2000 seconds and the results are illustrated in Figure 5.25. In the upper two plots, the red and blue lines represent the set point and output level respectively. The initial level in tanks 1 and 2 are 12.3 and 12.8 [cm] respectively and step changes to the reference are given at 300 and 1100 seconds of simulation horizon. The upper two plots show that levels are controlled, but the PI controller takes a long time to follow the reference level as compared to MPC with integral action. In this case, the settling time is around 200 seconds whereas in the constrained MPC with integral was 15 seconds, which is presented in section 5.4.1. The PI tuning parameters are found by trial and error methods where $k_{p1} = 4$, $k_{p2} = 3.5$, $T_{i1} = 9$ and $T_{i2} = 10$ give better results. Similar to MPC with integral action using *"if-else"* method, constrained are handled and the control signals for pumps 1 and 2 are restricted to 0-5 [V] as shown in lower two plots of the Figure 5.25. A MATLAB m-file script is written for implementation of the PI controller in the four-tank minimum phase process and provided in Appendix 9.

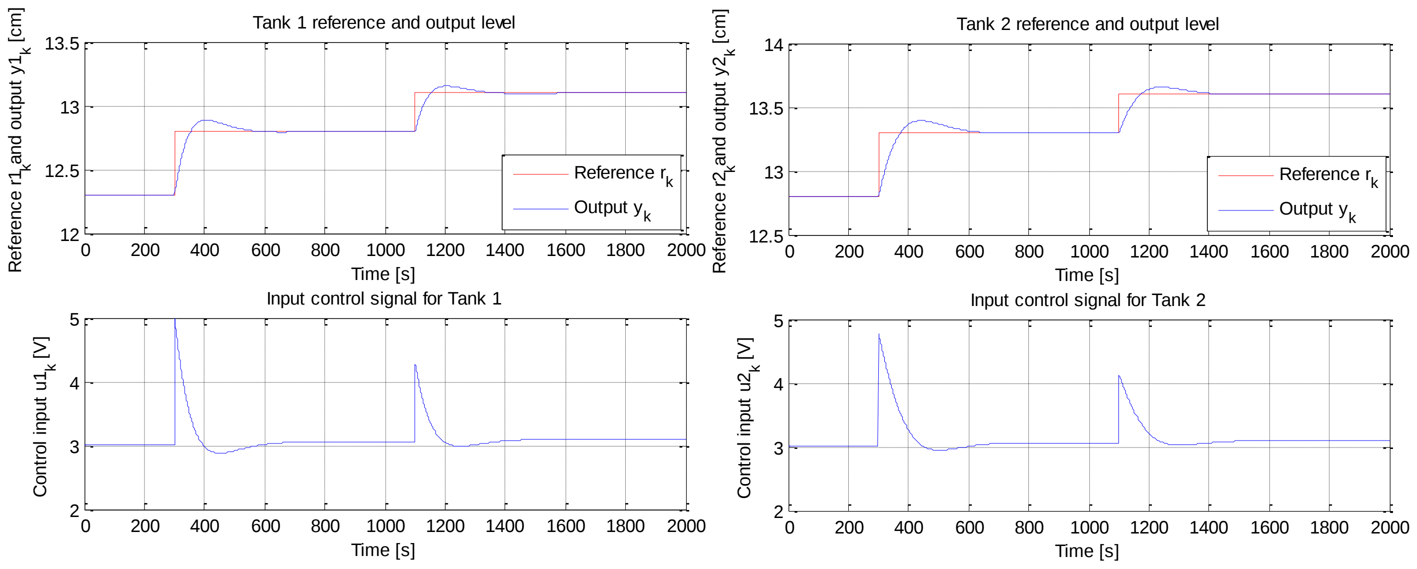


Figure 5.25: Result of PI controller implementation in four-tank minimum phase process. The upper two plots are the reference and the output levels in tanks 1 and 2. Lower two plots are the control signal for pumps 1 and 2.

### 5.5.3 Non-minimum phase process

In the four-tank non-minimum phase process, RGA analysis suggested that the input $u_1$ will control the output $y_2$ and input $u_2$ will control the output $y_1$. The simulation is performed for 2000 seconds by implementing the decentralized PI controller and the result are illustrated in Figure 5.26. In the upper two plots, the red and blue lines represent the set point and output level respectively. The initial level in tanks 1 and 2 are 12.4 and 13.2 [cm] respectively. The step change to the reference point are given at 300 seconds of simulation horizon and time scale is increased from 2000 to 5000 seconds. The settling time in this case is ten times larger than the previous case. It also can be seen from the upper two plots, it is difficult to control the four-tank non-minimum phase process however, the performance of constrained MPC with integral action (section 5.4.1) is much better than a decentralized PI controller in this particular case. The PI tuning parameters are found by trial and error methods where $k_{p1} = 1.4$, $k_{p2} = 0.22$, $T_{i1} = 100$ and $t_{i2} = 135$ give better results. Similar to MPC with integral action using *"if-else"* loop the control signals for pumps 1 and 2 are restrained to 0-5 [V] as shown in the Figure 5.26 lower two plots. A MATLAB m-script file is written for implementation of the PI controller in the four-tank non-minimum phase process and provided in Appendix 10.

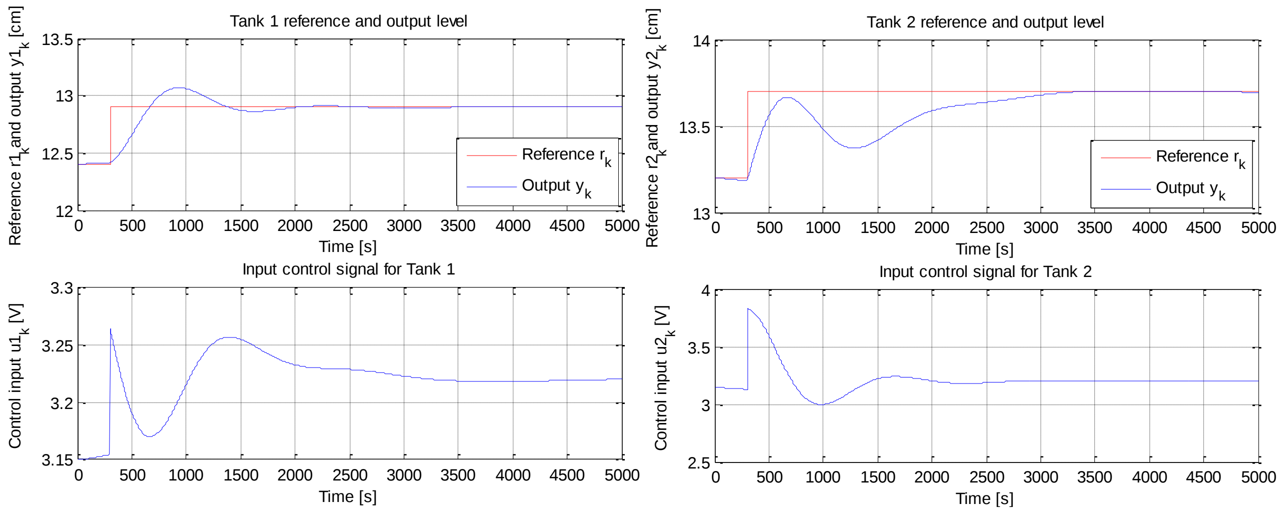


Figure 5.26: Result of PI controller implementation in the four-tank non-minimum phase process. The upper two plots are the reference and the output levels in tanks 1 and 2. Lower two plots are the control signals for pumps 1 and 2.

# 6. System identification and model free MPC

In this chapter, the system identification method is used to identify the model of the four-tank process and formulate a model free MPC algorithm. First of all, short overview of system identification algorithm as Deterministic and Stochastic System Identification and Realization '*DSR*' is introduced. The input-output data of the four-tank process is generated by simulation for both minimum and non-minimum phase setting. The collected data is used in the '*DSR*' algorithm to construct a linearized state space model. The identified model is validated and then used in MPC with integral action to control the four-tank process.

## 6.1 System identification algorithm as DSR

System identification can be defined as constructing a mathematical model of the dynamic systems from the measured input-output data. Process models, state space models, continuous and discrete time transfer functions can be identified by using the known data [43]. In this work, the system identification algorithm as Deterministic and Stochastic System Identification and Realization '*DSR*' is used to identify the linearized state space model for MPC. The '*DSR*' algorithm estimated the system order $n$, matrices $A, B, D, E, CF, F$ and initial state vector $x_0$ at discrete time combined with deterministic and stochastic dynamic model of innovation form as mention by Ruscio [24, 44]. The '*DSR*' algorithm synopsis as,

$$[A, B, D, E, CF, F, x_0] = dsr(Y, U, L, g, J, M, n) \tag{6.1}$$

The $Y, U$ and $L$ are the parameters on the input. The parameter $Y \in \mathbb{R}^{N \times m}$ is the output data matrix, where $N$ is the number of samples and $m$ denotes the number of the output variable. The input data matrix is $U \in \mathbb{R}^{N \times r}$ and $r$ is the number of input variables. $L$ is the future horizon used for predicting the order of the system.

Whereas $(g, J, M, n)$ are being the optional input parameters for advance use. The parameter $g$ is used to estimate $E$, if its value is equal to zero then $E = 0$ in the model. The past horizon $J$ is used to remove the future noise and $M$ is helpful for computing $CF$ and $F$ [24, 44].

The four-tank model presented in section 4.1.1 is simulated using the pseudo-random binary sequence (prbs) function to collect data. The sampling time for all the simulation is 0.1 seconds and initial levels in the tanks are assumed to be zero. The input and output data of 10000 samples are collected for minimum and non-minimum phase process separately. All the samples are used for system identification algorithm as '*DSR*' to identify a state space model. A totally new set of input-output data with 10000 samples is used to validate the identified model. A combined MATLAB m-file script is written to simulate the four-tank minimum and non-minimum phase process, and the m-file is provided in Appendix 11. In this program, the user can select either minimum or non-minimum phase process by typing "1" or "2" respectively.

## 6.2 Minimum phase process

The input and output data from the four-tank minimum phase simulation is saved as '*data_minimum.txt*'. The first two columns in the file represent the output level in tanks 1 and 2 whereas next two columns are the corresponding voltage to the pumps 1 and 2. The upper two plots show the simulated output level of tanks 1 and 2 in Figure 6.1 and lower two plots show the input voltage to the pumps 1 and 2. The MATLAB m-file for plotting the input-output data is provided in Appendix 12.

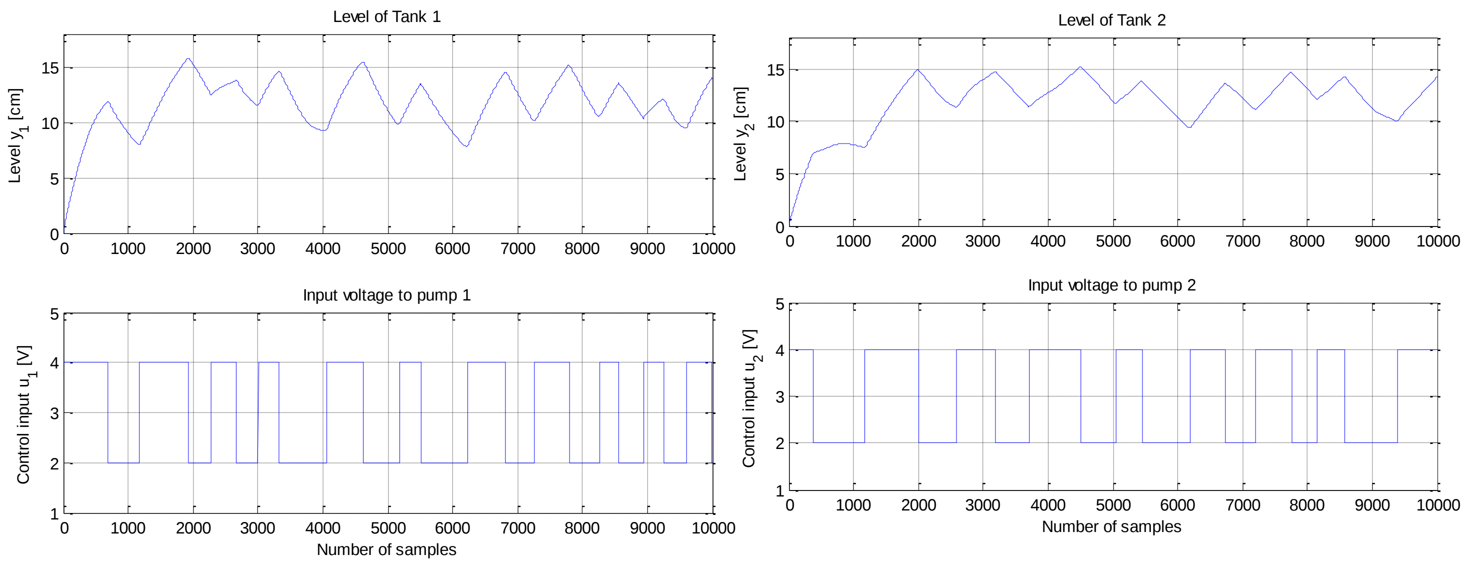


Figure 6.1: The four-tank minimum phase simulation data for system identification. Upper two plots show the simulated output level in tanks 1 and 2. Lower two plots show the input voltage to the pumps 1 and 2.

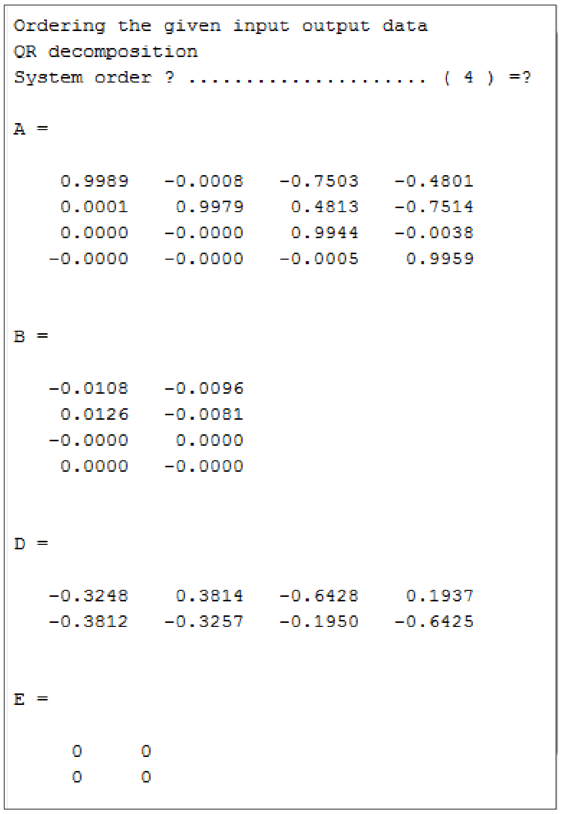
### 6.2.1 Identifying the model

The collected data of four-tank minimum phase process is centered before identify the linear model. In order to construct a more accurate model and remove offset the data is centered [45]. Therefore, the steady state values of output level 12.3 [cm] in tank 1 and 12.8 [cm] in tank 2 are subtracted from the output data. Similarly, the corresponding voltage 3 [V] for the steady state values are removed from the input data. The centered data are used to identify the state space model by '*DSR*' algorithm. The identified discrete time state space model is,

$$x_{k+1} = Ax_k + Bu_k \tag{6.2}$$

$$y_k = Dx_k + Eu_k \tag{6.3}$$

The model matrices $A, B, D$ and $E$ are given is the Figure 6.2.



```
Ordering the given input output data
QR decomposition
System order ? ..................... ( 4 ) =?

A =

    0.9989   -0.0008   -0.7503   -0.4801
    0.0001    0.9979    0.4813   -0.7514
    0.0000   -0.0000    0.9944   -0.0038
   -0.0000   -0.0000   -0.0005    0.9959


B =

   -0.0108   -0.0096
    0.0126   -0.0081
   -0.0000    0.0000
    0.0000   -0.0000


D =

   -0.3248    0.3814   -0.6428    0.1937
   -0.3812   -0.3257   -0.1950   -0.6425


E =

     0     0
     0     0
```

Figure 6.2: The identified model matrices A, B, D and E

The order of the system is investigated by condition numbers (CN) and the singular values (SV) such that it will be either the number of small CN or the number of large SV [24]. The estimated singular values and condition numbers for the system order by *'DSR'* algorithm are shown in Figure 6.3.

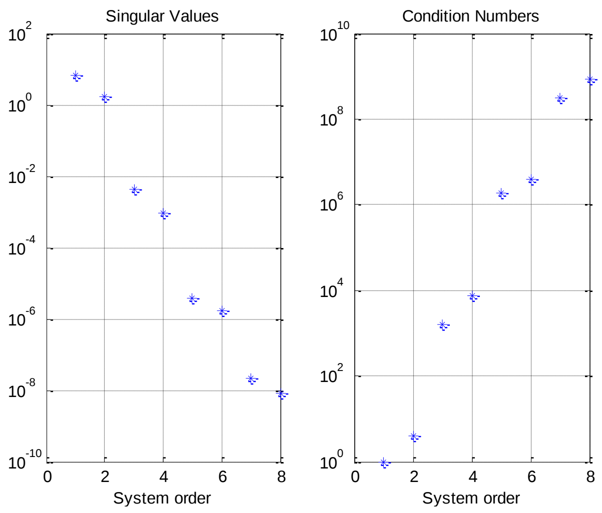
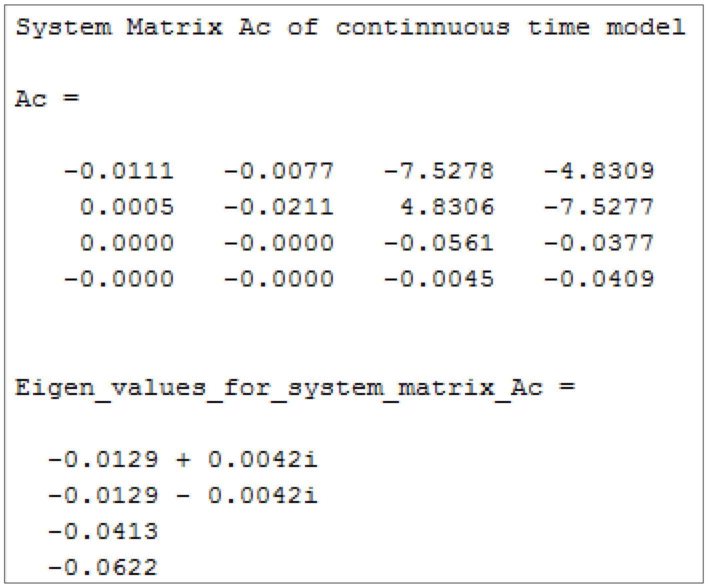


Figure 6.3: The singular values and condition number for system order of identified model in the four-tank minimum phase process.

The stability of the identified model can be analyzed by the Eigenvalues of the system matrix. First of all , the identified discrete time model is converted into a continuous time model and then the Eigenvalues for the system matrix *Ac* of the continuous model are computed. It can be seen from the Figure 6.4 real parts of the Eigenvalues are all negative, therefore the system is stable as presented in section 5.3. The MATLAB m-file to identify the model and Eigenvalues for the system matrix is provided in Appendix 13.



```
System Matrix Ac of continnuous time model

Ac =

   -0.0111   -0.0077   -7.5278   -4.8309
    0.0005   -0.0211    4.8306   -7.5277
    0.0000   -0.0000   -0.0561   -0.0377
   -0.0000   -0.0000   -0.0045   -0.0409


Eigen_values_for_system_matrix_Ac =

  -0.0129 + 0.0042i
  -0.0129 - 0.0042i
  -0.0413
  -0.0622
```

Figure 6.4: The four-tank minimum phase process, the Eigenvalues for system matrix $Ac$ of the continuous time model

### 6.2.2 Model validation

Validation is a method to test the performance of the developed model. There are different methods of validation such as Leverage correction, test set or cross validation [46]. Validating the model with a different data set is called cross validation. Although the same data can be used for identifying and validating the model. However, it will cause the over fitting of the data. [43]. Therefore, a new data set is collected with different '*prbs*' signal as input for the minimum phase setting. It is used to test the performance of the model and this data is also centered before using in the validation method. The identified state space model from '*DSR*' fits the validation data well. The simulation results of identified model along with validation data for minimum phase setting are shown in Figure 6.5. The green line shows the simulation output of the process and the blue line shows the simulation output from the '*DSR*' model. Noted here process output means simulation of a physical model of the four-tank minimum phase system. The MATLAB m-file script for the validation of identified model is provided in Appendix 14.

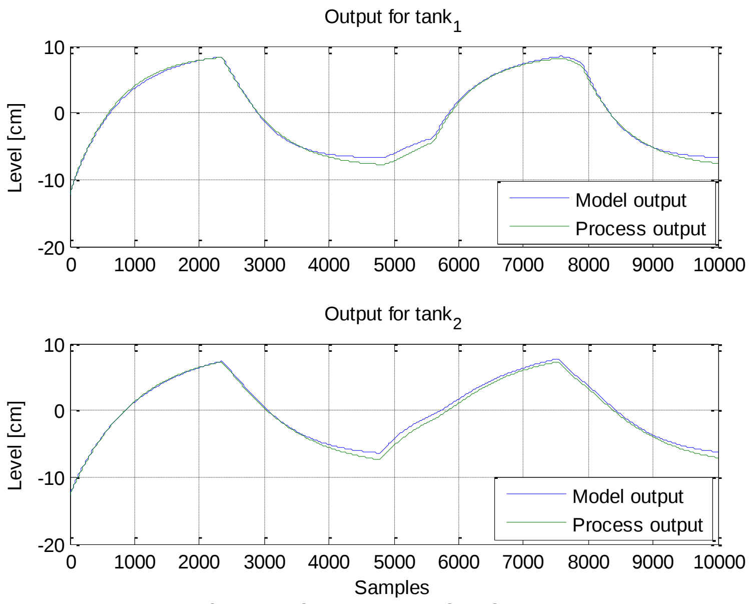


Figure 6.5: Validation of identified model for four-tank minimum phase process.

### 6.2.3 Implementations of MPC with integral action

The drawback of the MPC optimal controller is that it requires a model of the dynamic system. There are different techniques to construct a model of the process. One commonly used technique is to develop a mathematical model from the first principle based on the basic physic or chemistry laws that describe the behavior of the system [47]. However, sometimes we do not have explicit information about the physic of the system, then we can use the system identification method to identify the model based on the input-output data of the system. Such a method is used to find a linearized state space model for MPC method, and it is termed as model free MPC algorithm.

The model identified above will be used in MPC with integral algorithm and mathematical derivation of optimal controller with integral action is given in section 3.4. Appropriate values of the weighting matrices $Q$ and $R$ are assigned to weighted the output and input variables respectively. Where

$$Q = \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix},$$

$$R = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$$

The initial level in tanks 1 and 2 are 11.5 and 12.5 [cm] respectively and small reference of ±0.1 [cm] is given using the MATLAB m-file script "*prbs1.m*". The prediction horizon is 15 seconds similar to one used in section 5.4. The simulation is performed by changing the set point every 150 seconds.

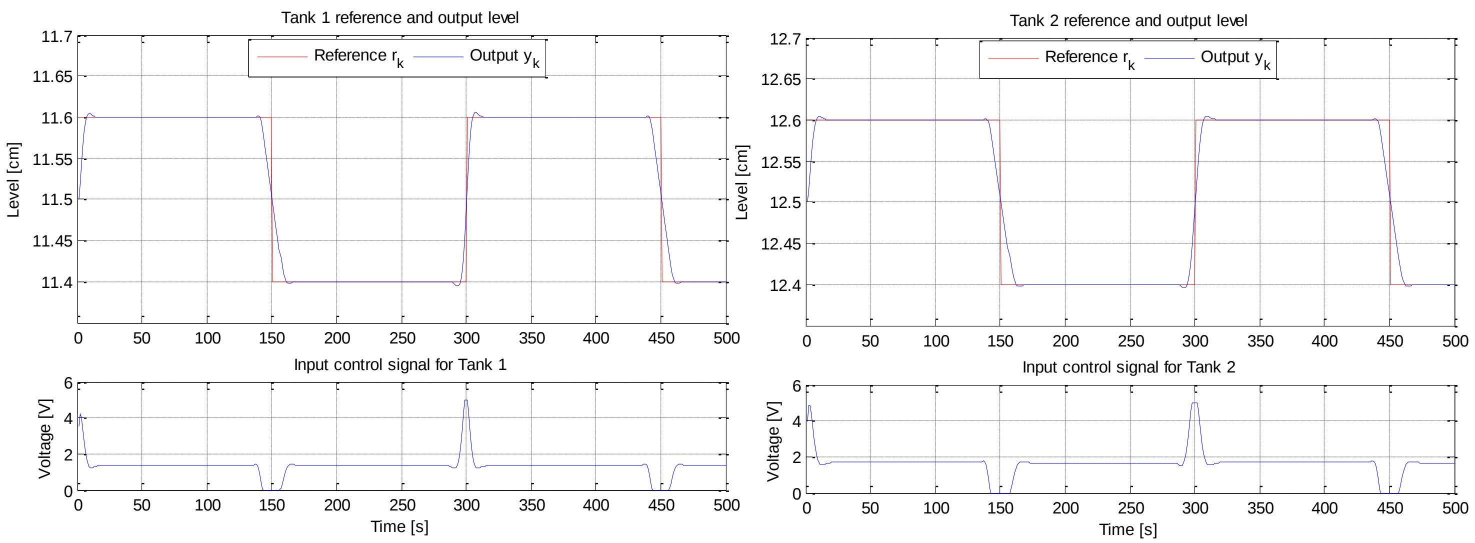


Figure 6.6: Simulation result by implementing the MPC with integral action in identified model of the four-tank minimum phase process. The upper two plots show the reference and the output levels in tanks 1 and 2. Lower two plots show the control signals for pumps 1 and 2.

The controller follows the reference point better than the one presented in section 5.4.1.1 with small undershoot and overshoot. The levels in Tanks 1 and 2 are obtained according to target set point. In the upper two plots, the red line and blue represents the set point and output level respectively. The input amplitude constraints are used in MPC algorithm, and the control signals for pumps 1 and 2 are confined to 0-5 [V] as shown in lower two plots of the Figure 6.6. The simulation result by using the identified model of four-tank minimum phase process in MPC with integral action algorithm is illustrated in Figure 6.6, and the MATLAB m-file for implementing the MPC with integral action is provided in Appendix 15.

## 6.3 Non-minimum phase process

The input and output data from the four-tank non-minimum phase simulation is saved as '*data_nonminimum.txt*'. The first two columns in the file represent the output level in tanks 1 and 2 whereas next two columns are the corresponding voltage to the pumps 1 and 2. The upper two plots show the simulated output level in tanks 1 and 2 in Figure 6.7 and lower two plots show the input voltage to the pumps 1 and 2. The MATLAB m-file for plotting the input-output data is provided in Appendix 16.

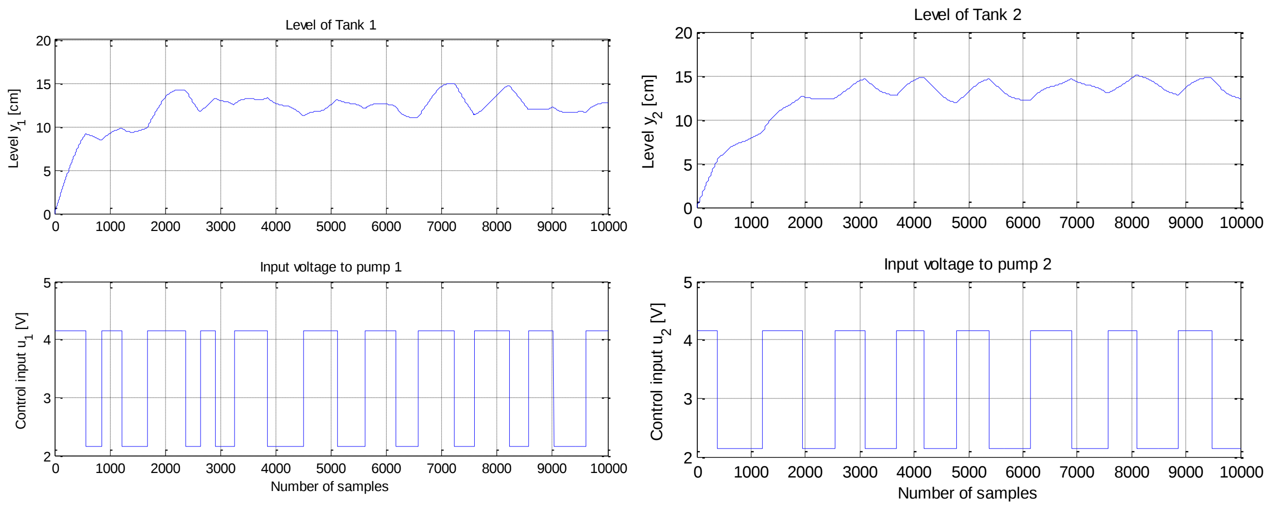
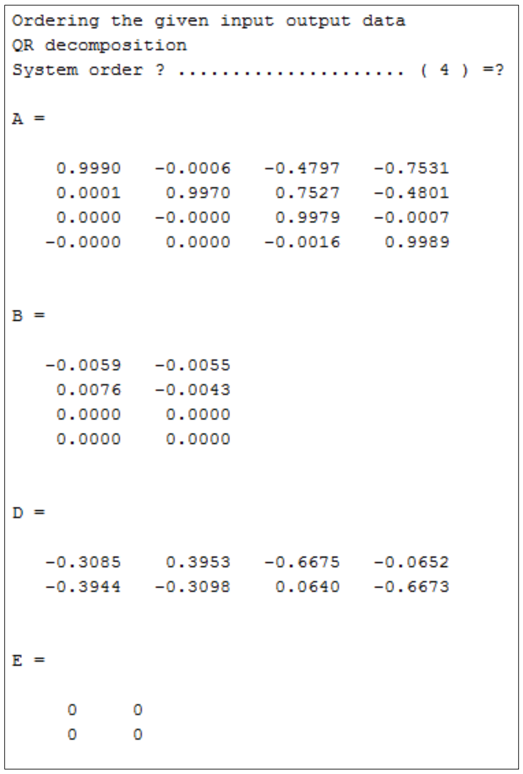


Figure 6.7: The four-tank non-minimum phase simulation data for system identification. Upper two plots show the simulated output level in tanks 1 and 2. Lower two plots show the input voltage to the pumps 1 and 2.

### 6.3.1 Identifying the model

The collected data of four-tank nonminimum phase process is centered before identify the linear model. The steady state values of output level 12.4 [cm] in tank 1 and 13.2 [cm] in tank 2 are removed from the output data. Similarly the corresponding voltage of 3 [V] for the steady state values are subtracted from the input data. The centered data are used to identify the state space model by *'DSR'* algorithm. The identified discrete time state space model is of the form given in in equation (6.1) and (6.2). The matrices $A, B, D$ and $E$ of identified model are given in Figure 6.8.



```
Ordering the given input output data
QR decomposition
System order ? ..................... ( 4 ) =?

A =

    0.9990   -0.0006   -0.4797   -0.7531
    0.0001    0.9970    0.7527   -0.4801
    0.0000   -0.0000    0.9979   -0.0007
   -0.0000    0.0000   -0.0016    0.9989


B =

   -0.0059   -0.0055
    0.0076   -0.0043
    0.0000    0.0000
    0.0000    0.0000


D =

   -0.3085    0.3953   -0.6675   -0.0652
   -0.3944   -0.3098    0.0640   -0.6673


E =

     0     0
     0     0
```

Figure 6.8: The non-minimum phase case the identified model matrices $A, B, D$ and $E$

The estimated singular values and condition numbers for the system order by '*DSR'* algorithm are shown in Figure 6.3. The MATLAB m-file to identify the model and Eigenvalues for the system matrix is provided in Appendix 17.

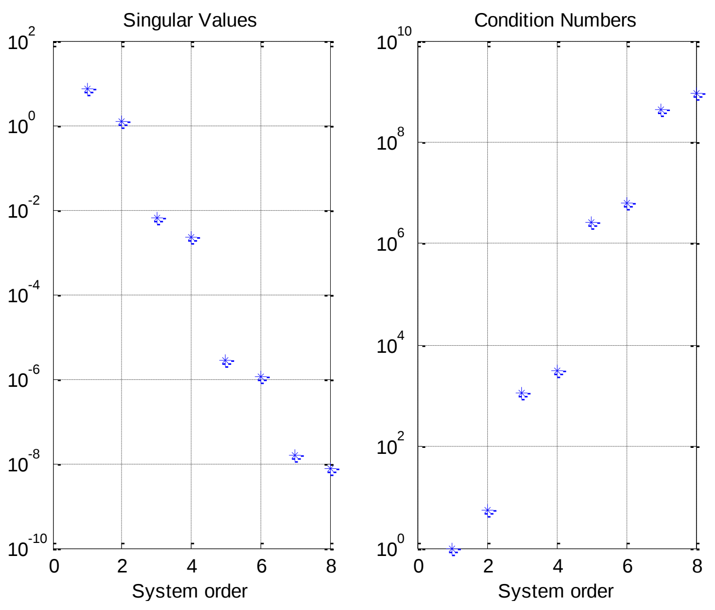
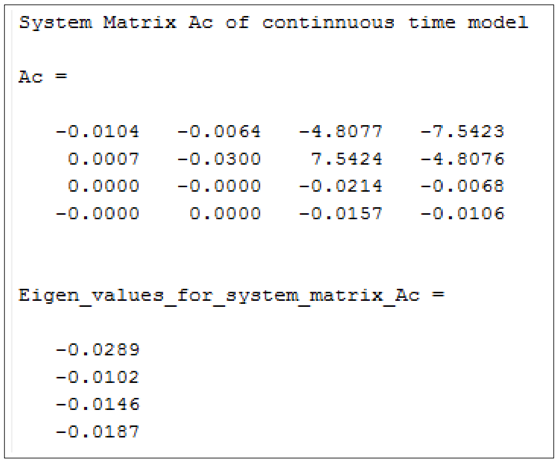


Figure 6.9: The singular values and condition number for system order of identifyed model in the four-tank non-minimum phase case.

The stability of the identified model can be analyzed by the Eigenvalues of the system matrix. First of all, the identified discrete time model is converted into a continuous time model and then the Eigenvalues for the system matrix $Ac$ are computed. It can be seen from the Figure 6.10 that the real parts of the Eigenvalues are all negative, therefore the system is stable.



```
System Matrix Ac of continnuous time model

Ac =

   -0.0104   -0.0064   -4.8077   -7.5423
    0.0007   -0.0300    7.5424   -4.8076
    0.0000   -0.0000   -0.0214   -0.0068
   -0.0000    0.0000   -0.0157   -0.0106


Eigen_values_for_system_matrix_Ac =

   -0.0289
   -0.0102
   -0.0146
   -0.0187
```

Figure 6.10: Four-tank non-minimum phase process, the Eigenvalues for system matrix Ac of the continuous time model.

## 6.3.2 Model validation

Similar to minimum phase process the cross validation is performed in this case. A new data set is collected with different ‘prbs’ signal as input for the non-minimum phase setting to test the performance of the model and this data was also centered before use in the validation method. The identified state space model from *DSR* fits the validation data especially in tank 1, however there is a small difference in tank 2. The simulation results of identified model along with validation data for non-minimum phase setting are shown in Figure 6.11. The green line shows the simulation output of the process and the blue line shows the simulation output from the 'DSR' model. Noted here process output means simulation of a physical model of the four-tank non-minimum phase system. The MATLAB m-file script for the validation of identified model for non-minimum phase case is provided in Appendix 18.

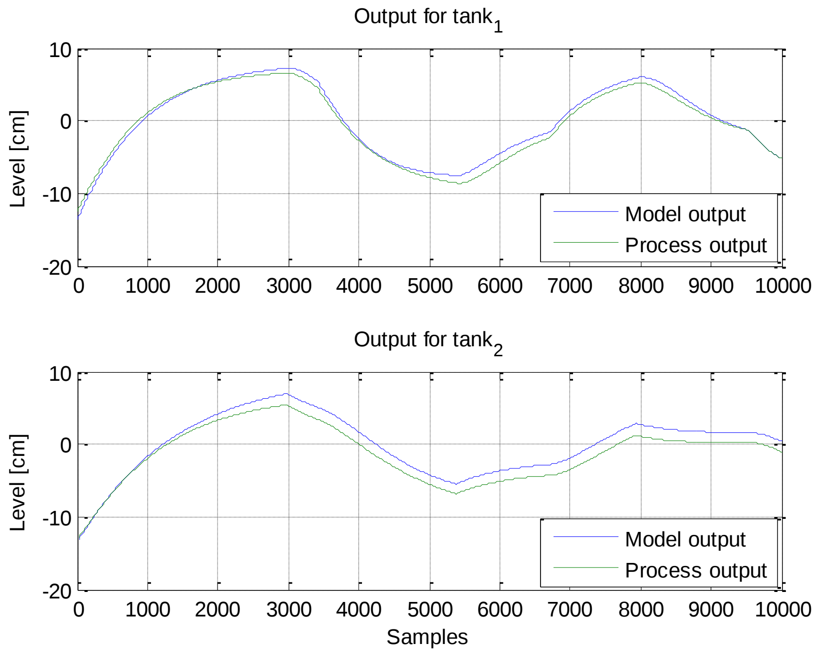


Figure 6.11: Validation of identified model for four-tank non-minimum phase process

### 6.3.3 Implementations of MPC with integral action

The model identified above is used in MPC with integral algorithm and mathematical derivation of optimal controller with integral action is given in section 3.4. Appropriate values of the weighting matrices $Q$ and $R$ were assigned to weighted the output and input variables respectively. Where

$$Q = \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix},$$

$$R = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$$

The initial level in tanks 1 and 2 are 11.5 and 12.5 [cm] respectively and small reference of ±0.1 [cm] is given using the MATLAB m-file script "*prbs1.m*". The prediction horizon is 15 seconds similar to the one used in section 5.4. The simulation is performed by changing the set point every 150 seconds.

The controller follows the reference point and interestingly the output response is faster as compared to the one presented in section 5.4.1.2. In the upper two plots, the red line and blue represents the set point and output level respectively. The input amplitude constraints are used in MPC algorithm, and the control signals for pumps 1 and 2 are restricted to 0-5 [V] as shown in lower two plots of the Figure 6.12. The simulation result by using the identified model of four-tank non-minimum phase process in MPC with integral action algorithm is illustrated in Figure 6.12. The MATLAB m-file for this case is provided in Appendix 19.

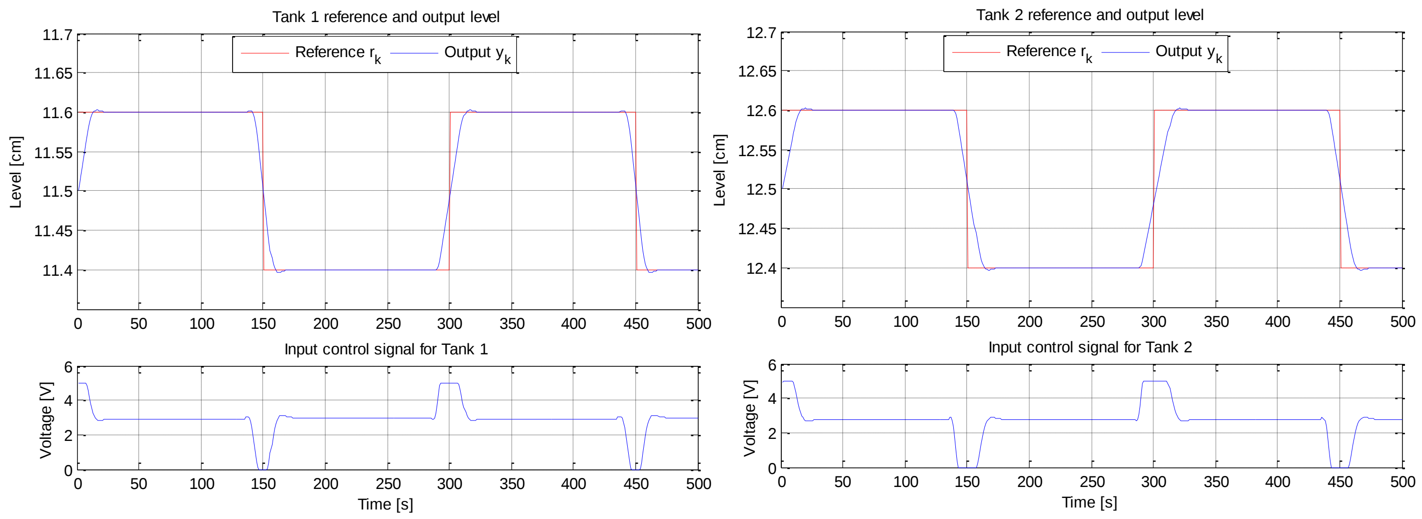


Figure 6.12: Simulation result by implementing the MPC with integral action in identified model of the four-tank non-minimum phase process. The upper two plots show the reference and the output levels in tanks 1 and 2. Lower two plots show the control signals for pumps 1

# 7. Discussions

In order to complete this work, I used the knowledge learned in the Model Predictive Control with implementation, System Identification and optimal estimation. These subjects are taught during the master's degree in Systems and Control Engineeing at Telemark University College, Norway. The mathematical derivation and programming in this work are referenced from these courses. MATLAB software (version: R2012a) developed by the MathWorks is used for simulation.

There are different nonlinear benchmark processes such as chemical reactors, the four-tank process, distillation columns and three-phase separator. The benchmark four-tank process is selected for control implementation as it is a non-linear, complex and interactive system. It has two operating conditions i.e., minimum and non-minimum phase that directly related to the valve position. The sum of valve constant is greater than one and less than two in a minimum phase process. In this case, the steady state level in the four tanks at constant voltage of 3 [V] are 12.3, 12.8, 1.63 and 1.41 [cm] respectively.

In non-minimum phase process, the sum of valve constant is less than one and greater than zero. In this case, the steady state level in four tanks at constant voltage of 3.15 [V] are 12.4, 13.2, 4.73 and 4.99 [cm] respectively. The four-tank is a nonlinear process, and nonlinear MPC can be used to control it, however this controller is not guaranteed to converge. Therefore, nonlinear model of the process is linearized and used in linear MPC.

MPC with integral is an effective technique to handle the offset problem, and there are several ways to achieve the integral action. The most common method is using the deviation variable to obtain the integral action as presented in the section 3.4. The aim of implementing the controller is to control the level in tanks 1 and 2. All the states are not measurable therefore Kalman filter is used to observe the states.

In all the experiments, sampling time of 0.1 seconds is used to discretize the model. Several simulations are performed, and optimal value of 15 seconds for prediction horizon and control horizon are used based on the time constant of the system. Larger values of prediction horizon and control horizon lead to unnecessary calculations at each sampling time. It is observed from simulation experiments in section 5.4.3 that using small control horizon cause the controller instability and behave more like a PI controller.

The constraints in the MPC optimal controller are handled using two different approaches i.e. "*quadprog*" function and *"if-else"* method. The voltage limits are 0-5 [V] and constraints are handled well in both methods. The executing time for the controller is smaller using the "*if-else*" method than "*quadprog*" function however, the performance of the controller decreased. The level in tanks 1 and 2 is controlled according to the specified set point, and the response in non-minimum phase setting is comparatively slower than minimum phase setting.

Using unconstrained MPC with integral action the required set point achieved with higher overshoot and undershoot, and the control signal varying from -8 to 15 [V]. Interestingly the output response is faster than constrained MPC with integral action.

A decentralized PI controller is also implemented to control the four-tank process. From the result, it shows that it is not an effective method to control the multivariable process. The set point is achieved in the minimum phase process, however the settling time is too large compared with the optimal controller. In non-minimum phase process, the controller is not able to achieve the target.

System identification algorithm as '*DSR*' is very useful tool to identify the linear state space model from input-output data, especially when the physical model of the process is not available. The identified model by using the simulated input-output of the four-tank process is used in MPC with integral action. The controller performed better in both cases as compared to the results presented in section 5.4.3.

## 7.1 Future works

The future work related to this topic can be listed as,

- Implementing the proposed controller in other benchmark processes such as chemical reactors, distillation columns and three-phase separator.
- Simulink in MATLAB can be used for graphical block presentation of the controllers.
- It would be interesting to implement Linear Quadratic (LQ) with integral action and compare the performance of the optimal controllers.
- Data from the four-tank real process can be used to identify the model and then used in an MPC method to control the real process.

# 8. Conclusions

In this work, the detailed study of Model Predictive Control (MPC) with integral action was described and implemented in benchmark process. Four-tank process used for simulation experiments. The nonlinear model of the process developed and then linearized it for using in the control algorithm. The integral action in MPC method was achieved by using the deviation variable. The experiment results showed that the proposed optimal controller worked very well for both operating conditions and responded to the set point changes in an optimal way. The constraints are handled using the '*quadprog*' function and '*if-else'* method, and input voltage was bound to 0-5 [V]. The '*quadprog*' function made the control work slower due to extra complex calculation. A large value of prediction horizon and control horizon lead to unnecessary calculation. Moreover, smaller control horizon caused the controller instability. A decentralized PI controller was developed and implemented in the four-tank process, however it was not able to control the process according to the desired set point. Comparing the results obtained by using an optimal controller with those from PI controller, it was found that the MPC with integral action is more robust and faster than traditional controller. It proved that MPC with integral action is an effective strategy to control MIMO system. The simulated input-output data were collected for system identification and validation of the model. System identification algorithm as '*DSR*' is used to identify the linearized state space model of the process. The identified model is used in the control algorithm, and this controller performed better compared with the one used in section 5.4. Using the system identification method model free MPC with integral action was formulated by just using the input-output data of the process. This is a very useful method when the model of the process is not available.

## References

[1] S. Bennett, "A brief history of automatic control," *Control Systems, IEEE,* vol. 16, pp. 17-25, 1996.

[2] Eng Boo Chin and W. A. T. S. Kwee Hong Meo, Yusof Khairiyah Mohd,, "Formulation Of Model Predictive Control Algorithm For Nonlinear Process," *Universiti Teknologi Malaysia,* 2006.

[3] L. Wang, "Model Predictive Control System Design and Implementation Using MATLAB," *Melbourne, Springer,* 2009.

[4] J. Richalet and J. L. T. A. Rault, J. Papon, "Model predictive heuristic control: Applications to industrial processes," *Automatica,* vol. Volume 14, pp. 413-426, 1978.

[5] G. C. Nunes, "Design and Analysis of Multivariable Predictive Control Applied to an oil-water-gas seperator: A Polynomial Approach," *University of Florida,* 2001.

[6] Qin S. Joe and Badgwell Thomas. A, "A survey of industrial model predictive control technology," *Control Engineering Practice,* vol. 11 (2003) 733-764, 2003.

[7] Ruscio David.Di, "Discrete LQ optimal control with integral action: A simple controller on incremental form for MIMO systems," *Modeling, Identication and Control, MIC Journal,* vol. 33, pp. 35-44, 2012.

[8] Garcia Carlos E, *et al.*, "Model predictive control: theory and practice—a survey," *Automatica,* vol. 25, pp. 335-348, 1989.

[9] R. Rouhani and R. K. Mehra, "Model algorithmic control (MAC); basic theoretical properties," *Automatica,* vol. 18, pp. 401-414, 1982.

[10] D. W. Clarke, *et al.*, "Generalized predictive control—Part I. The basic algorithm," *Automatica,* vol. 23, pp. 137-148, 1987.

[11] D. Clarke, *et al.*, "Generalized predictive control—part II extensions and interpretations," *Automatica,* vol. 23, pp. 149-160, 1987.

[12] K. Zhu, *et al.*, "Alternative algorithms for generalized predictive control," *Systems & Control Letters,* vol. 15, pp. 169-173, 1990.

[13] C. R. Cutler and B. Ramaker, "Dynamic matrix control-a computer control algorithm," in *Proceedings of the joint automatic control conference*, 1980, pp. Wp5-B.

[14] R. Soeterboek, *Predictive control: a unified approach*: Prentice-Hall, Inc., 1992.

[15] Ruscio David.Di, "Model predictive control and optimization, Lecture notes for Master's course (SCE 4106)," *Telemark University College, Norway,* 2012.

[16] H. Hans-Petter, "Model Predictive Control in LabVIEW," *Faculty of Technology, Telemark University College, Norway,* 30.06.2011.

[17] B. Kim and H.-S. Y. Gregory N Washington, "Hysteresis-reduced dynamic displacement control of piezoceramic stack actuators using model predictive sliding mode control.," *IOP Science,* 1 May 2012.

[18] X. Yong, "ROBUST COORDINATED MULTIPLE-AXIS MOTION CONTROL SYSTEMS AND APPLICATIONS," *SCHOOL OF ELECTRICAL & ELECTRONIC ENGINEERING, NANYANG TECHNOLOGICAL UNIVERSITY,* 2006.

[19] D. A. M. Dale E. Seborg, Thomas F. Edgar, Francis J. Doyle, III, "Process Dynamics and Control," *John Wiley & Sons,* vol. 3rd, 2011.

[20] J. Mattingley*, et al.*, "Receding horizon control," *Control Systems, IEEE,* vol. 31, pp. 52-65, 2011.

[21] P. T. Boggs and a. J. W. Tolle, "Sequential Quadratic Programming," *Acta Numerica,* vol. 4, pp. 1-51, 1995.

[22] C. B. Eduardo F. Camacho, "Model predictive control," *Springer London, Limited,* vol. 2nd Edition, 2004.

[23] Ruscio David. Di, "Model Based Predictive Control: An extended state space approach.," *In proceedings of the 36th Conference on Decision and Control, San Diego, California, December 6-14,* 1997.

[24] Ruscio David.Di, "System identification and optimal estimation, Lecture notes for Master's course (SCE 2206)," *Telemark University College, Norway,* 2012.

[25] J. Åkesson and P. Hagander, "Integral action–A disturbance observer approach," in *Proceedings of European Control Conference, Cambridge, UK (September 2003)*, 2003.

[26] K. R. Muske and T. A. Badgwell, "Disturbance modeling for offset-free linear model predictive control," *Journal of Process Control,* vol. 12, pp. 617-632, 2002.

[27] J. B. Rawlings*, et al.*, "Nonlinear model predictive control: A tutorial and survey," *Proceedings of ADCHEM'94,* pp. 203-214, 1994.

[28] G. Pannocchia and J. B. Rawlings, "Disturbance models for offset-free model-predictive control," *AIChE Journal,* vol. 49, pp. 426-437, 2003.

[29] D. M. Prett and C. E. García, "Fundamental process control," 1988.

[30] E. Davison and H. Smith, "Pole assignment in linear time-invariant multivariable systems with constant disturbances," *Automatica,* vol. 7, pp. 489-498, 1971.

[31] Hovd. Morten, "A brief introduction to Model Predictive Control," *URL=* *http://www. itk. ntnu. no/fag/TTK4135/viktig/MPCkompendium% 20HOvd. pdf,* 2004.

[32] Ruscio David. Di, "System theory, state space analysis and control theory, lecture notes in control theory for Master's course (SCE 1106)," *Telemark University College, Norway* 2011.

[33] F. Haugen, *Lecture Notes in Models, Estimation and Control*: Techteach, 2009.

[34] H. Hans-Petter, "State Estimation with Kalman Filter," *Systems control laboratory laboratory, Telemark University College, Norway,* vol. , 2012.

[35] Johansson Karl. Henrik, "The Quadruple-Tank Process: A Multivariable Laboratory Process with an Adjustable Zero," *IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY,* vol. 8, pp. 456-465, 2000.

[36] Ruscio David.Di, "State estimation of quadruple tank process, Exercise 8. System identification and optimal estimation. Telemark University College, Norway," 2012.

[37] Numsomran A. V. Tipsuwanporn. K Tirasesth, "Modeling of the Modified Quadruple-Tank Process," *SICE Annual Conference* 2008.

[38] Hardt David, "Understanding Poles and Zeros, Lecture notes," *Massachusetts Institute of Technology, Department of Mechanical Engineering.,* 2002.

[39] Katsuhiko Ogata, "Modern Control Engineering," *Prentice Hall, Pearson Education International,* vol. Fourth Edition, 2002.

[40] O'Dwyer  Aidan, "PI and PID controller tuning rules: an overview and personal perspective," *Proceedings of the IET Irish Signals and Systems Conference, Dublin Institute of Technology,* pp. 161-166, 2006.

[41] D. Chen and D. E. Seborg, "Design of decentralized PI control systems based on Nyquist stability analysis," *Journal of Process Control,* vol. 13, pp. 27-39, 2003.

[42] E. H. Bristol, "On a new measure of interaction for multivariable process control," *IEEE Transactions on Automatic Control - IEEE TRANS AUTOMAT CONTR* vol. 11, pp. 133-134, 1966.

[43] L. Ljung, "System Identification Toolbox," *The MathWorks, Inc*, 2013.

[44] David Di. Ruscio, "D-SR Toolbox for MATLAB. Copyright D. Di Ruscio. ," *Available upon request, david.di.ruscio@hit.no.,* 2013.

[45] R. A. Harshman and M. E. Lundy, "PARAFAC: Parallel factor analysis," *Computational Statistics & Data Analysis*, vol. 18, pp. 39-72, 1994.

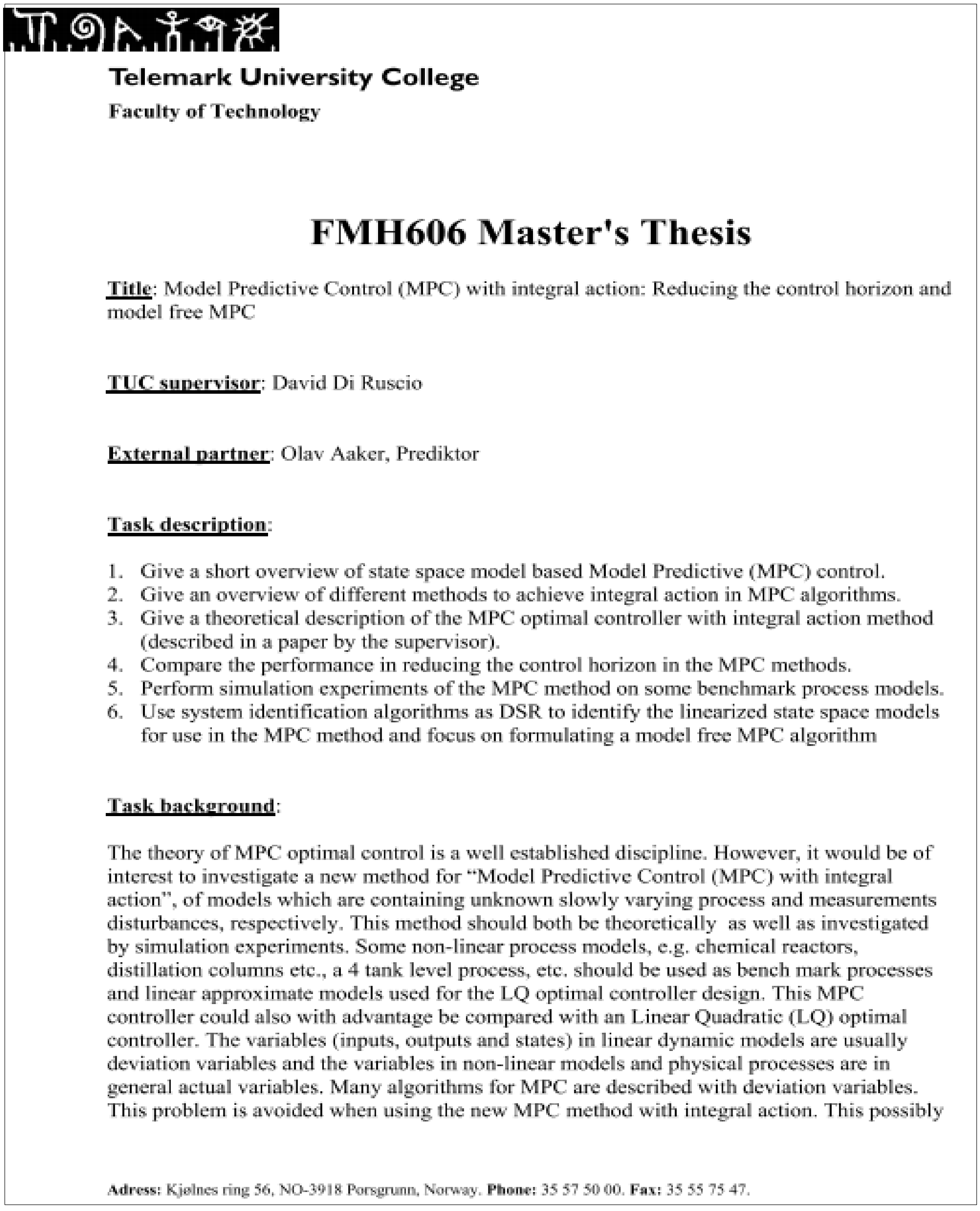
[46] K. H. Esbensen*, et al., Multivariate data analysis-in practice: An introduction to multivariate data analysis and experimental design*: Multivariate Data Analysis, 2002.

[47] J. Poshtan and H. Mojallali, "Subspace system identification," *Iranian Journal of Electrical & Electronic Engineering,* vol. 1, pp. 11-17, 2005.

## Appendices

### Appendix 1:

The scan copy of Master’s thesis task description.



**Telemark University College**

**Faculty of Technology**

## FMH606 Master's Thesis

**Title**: Model Predictive Control (MPC) with integral action: Reducing the control horizon and model free MPC

**TUC supervisor**: David Di Ruscio

**External partner**: Olav Aaker, Prediktor

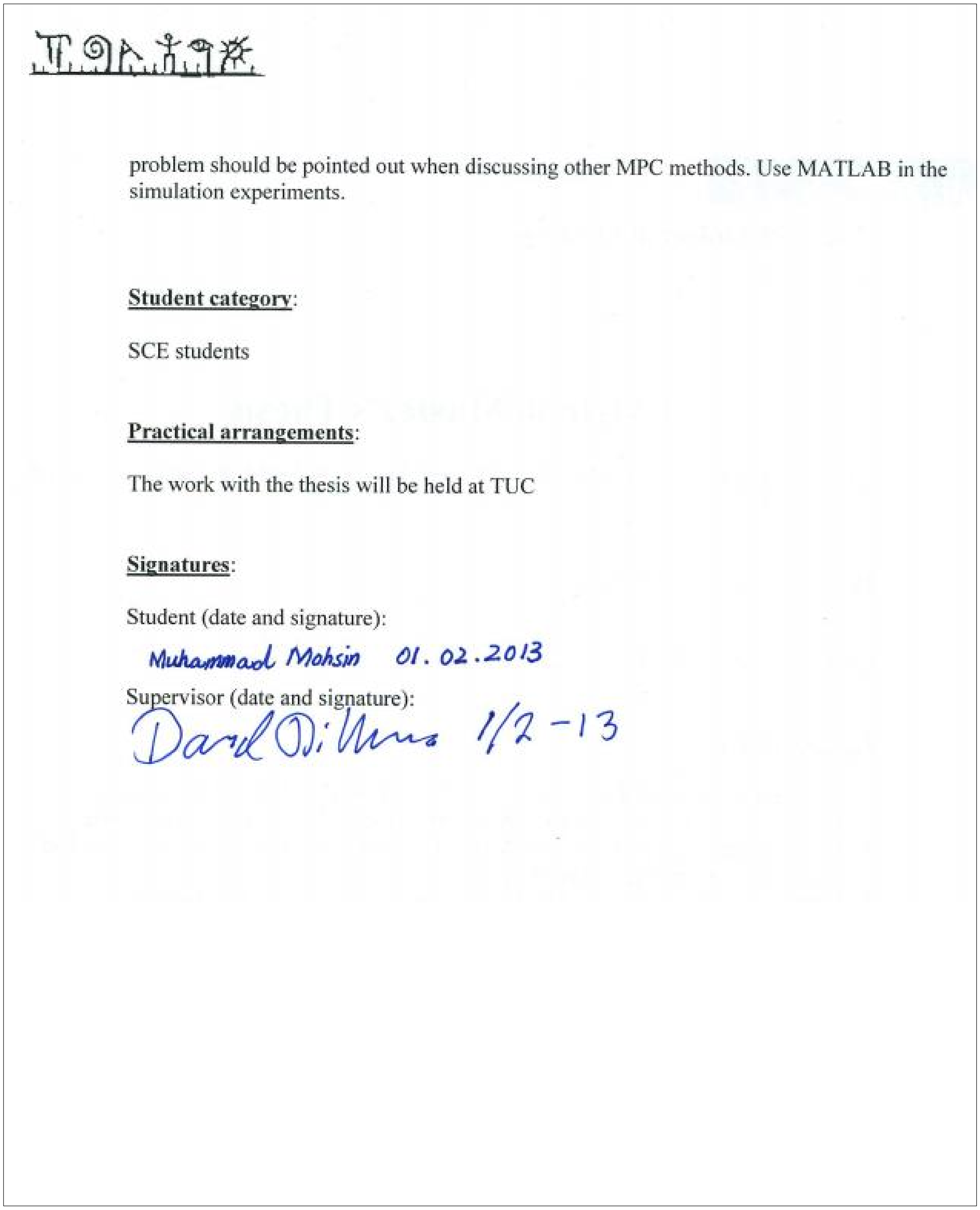
**Task description**:

1. Give a short overview of state space model based Model Predictive (MPC) control.
2. Give an overview of different methods to achieve integral action in MPC algorithms.
3. Give a theoretical description of the MPC optimal controller with integral action method (described in a paper by the supervisor).
4. Compare the performance in reducing the control horizon in the MPC methods.
5. Perform simulation experiments of the MPC method on some benchmark process models.
6. Use system identification algorithms as DSR to identify the linearized state space models for use in the MPC method and focus on formulating a model free MPC algorithm

**Task background**:

The theory of MPC optimal control is a well established discipline. However, it would be of interest to investigate a new method for “Model Predictive Control (MPC) with integral action”, of models which are containing unknown slowly varying process and measurements disturbances, respectively. This method should both be theoretically as well as investigated by simulation experiments. Some non-linear process models, e.g. chemical reactors, distillation columns etc., a 4 tank level process, etc. should be used as bench mark processes and linear approximate models used for the LQ optimal controller design. This MPC controller could also with advantage be compared with an Linear Quadratic (LQ) optimal controller. The variables (inputs, outputs and states) in linear dynamic models are usually deviation variables and the variables in non-linear models and physical processes are in general actual variables. Many algorithms for MPC are described with deviation variables. This problem is avoided when using the new MPC method with integral action. This possibly

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



problem should be pointed out when discussing other MPC methods. Use MATLAB in the simulation experiments.

**Student category**:

SCE students

**Practical arrangements**:

The work with the thesis will be held at TUC

**Signatures**:

Student (date and signature):

Muhammad Mohsin 01. 02. 2013

Supervisor (date and signature):

[illegible] 1/2 -13

## Appendix 2:

MATLAB script files for simulating the four-tank process with minimum and non-minimum phase setting.

**Appendix 2.1:** Nonlinear_model_simulation.m

```
% ===========================================================================
% MATLAB Script for:  Simulation of nonlinear model.
% Master's Thesis:   "MPC with Integral Action: Reducing the control horizon
%                     and model free MPC."
% Supervisior:        David Di Ruscio
% Author:             Muhammad Mohsin
% Script File:        Nonlinear_model_simulation.m
% Function file:      Fourtank_nonlinear_model.m
% ---------------------------------------------------------------------------
% Telemark University College, Porsgrunn, Norway [20-03-2013].
% ===========================================================================
clc;
close all;
clear all;

% System selection by the user
System_sel=input(' Type 1 for minimum phase system or\n Type 2 for non-
minimum phase system \n');

if  System_sel==1;
% Model will show minimum phase characteristics
% Initial level in the tanks [cm]
h1=0;
h2=0;
h3=0;
h4=0;
h=[h1,h2,h3,h4];
% Initial control inputs [V]
u1=3.0; u2=3.0;
u=[u1 u2];
disp('Minimum phase system')

elseif System_sel==2;
% Model will show non-minimum phase characteristics
% Initial level in the tanks [cm]
h1=0;
h2=0;
h3=0;
h4=0;
h=[h1,h2,h3,h4];
% Initial control inputs [V]
u1=3.15; u2=3.15;
u=[u1 u2];
disp('Non-minimum phase system')

else display('Number not in range, type 1 or 2')
end
```

```
% Simulation time
N=1000;
time=1:0.1:N;
[t,y]=ode45(@Fourtank_nonlinear_model, time,h,[],u,System_sel);

% Results for four tank simulation
figure(1)
subplot(2,2,1)
plot(t,y(:,1)),ylabel('Level [cm]'),grid on,legend('Tank 1')
subplot(2,2,2)
plot(t,y(:,2)),grid on,legend('Tank 2')
subplot(2,2,3)
plot(t,y(:,3)),xlabel('Time [s]'),ylabel('Level [cm]'),grid on,legend('Tank
3')
subplot(2,2,4)
plot(t,y(:,4)),xlabel('Time [s]'),grid on,legend('Tank 4')

% For super title
set(gcf,'NextPlot','add');
axes;
if  System_sel==1;
    label = title('Four-tank nonlinear model simulation for minimum phase
process');
elseif System_sel==2;
    label = title('Four-tank nonlinear model simulation for nonminimum phase
process');
end
set(gca,'Visible','off');
set(label,'Visible','on');
```

**Appendix 2.2:** Fourtank_nonlinear_model.m

```
%===========================================================================
% MATLAB Script for:  Function for simulating the nonlinear model of the
                      four-tank process.
% Master's Thesis:    "MPC with Integral Action: Reducing the control horizon
%                     and model free MPC."
% Supervisior:        David Di Ruscio
% Author:             Muhammad Mohsin
% Function file:      Fourtank_nonlinear_model.m
% -------------------------------------------------------------------------
% Telemark University College, Porsgrunn, Norway [20-03-2013]
%===========================================================================
function model=Fourtank_nonlinear_model(time,h,u,System_sel)
% Parameters values taken from the literature [Johansson] given in Table 5.1
% Cross-section area of tanks [cm^2]
A1=28; A3=28;
A2=32; A4=32;
% Cross-section area of the outlet holes[cm^2]
a1=0.071; a3=0.071;
a2=0.057; a4=0.057;
% Level in the tanks [cm]
h1=h(1); h2=h(2);
h3=h(3); h4=h(4);
```

```
% Control inputs [V]
u1=u(1);
u2=u(2);
% Pump gain [V/cm]
kc=0.5;
% Acceleration of gravity [cm/s^2]
g=981;

if System_sel==1
% Model will show minimum phase characteristics [Table 5.2]
% Pump constants [cm^3/Vs]
k1=3.33; k2=3.35;
% Valve constants
r1=0.70; r2=0.60;

elseif System_sel==2
% Model will show non-minimum phase characteristics, [Table 5.3]
% Pump constants [cm^3/Vs]
k1=3.14; k2=3.29;
% Flow constants
r1=0.43; r2=0.34;
end

% The nonlinear model. [Equation (4.10)-(4.13)]
dh(1)=( -a1*sqrt(2*g*h1) + a3*sqrt(2*g*h3) + r1*k1*u1 )/A1;
dh(2)=( -a2*sqrt(2*g*h2) + a4*sqrt(2*g*h4) + r2*k2*u2 )/A2;
dh(3)=( -a3*sqrt(2*g*h3) + (1-r2)*k2*u2 )/A3;
dh(4)=( -a4*sqrt(2*g*h4) + (1-r1)*k1*u1 )/A4;

model=[dh(1);dh(2);dh(3);dh(4)];
```

## Appendix 3

Controllability and observability analysis of linearized model.

**Appendix 3.1:** Linearized_model_anaylsis.m

```
%========================================================================
% MATLAB Script for:  Controllability and observability analysis of
                      linearized model.
% Master's Thesis:    "MPC with Integral Action: Reducing the control horizon
%                     and model free MPC."
% Supervisior:        David Di Ruscio
% Author:             Muhammad Mohsin
% Script file:        Linearized_model_anaylsis.m
% ----------------------------------------------------------------------
% Telemark University College, Porsgrunn, Norway [20-03-2013]
%========================================================================
clc;
close all;
clear all;

% System selection by the user
System_sel=input(' Type 1 for minimum phase system or\n Type 2 for non-
minimum phase system \n');

if  System_sel==1;
display('Minimum phase system')
% Model will show minimum phase characteristics
% Steady state level [cm] in the tanks after performing simulations
% [Table 5.3]
hs1=12.3;
hs2=12.8;
hs3=1.63;
hs4=1.41;
% Pump constants [cm^3/Vs]
k1=3.33; k2=3.35;
% Flow constants
r1=0.70; r2=0.60;

elseif System_sel==2;
display('Non-minimum phase system')
% Model will show non-minimum phase characteristics
% Steady state level [cm] in the tanks after performing simulations
% [Table 5.3]
hs1=12.4;
hs2=13.2;
hs3=4.73;
hs4=4.99;
% Pump constants [cm^3/Vs]
k1=3.14; k2=3.29;
% Flow constants
r1=0.43; r2=0.34;
else display('Number not in range, type 1 or 2')
end
```

```
% Parameter values taken from the literature [Johansson], Table [5.1].
% Cross-section area of tanks [cm^2]
A1=28; A3=28;
A2=32; A4=32;
% Cross-section area of the outlet holes[cm^2]
a1=0.071; a3=0.071;
a2=0.057; a4=0.057;
% Pump gain [V/cm]
kc=0.5;
% Acceleration of gravity [cm/s^2]
g=981;
% Time constants [Section 4.1.2]
T1=A1*(sqrt(2*hs1/g))/a1;
T2=A2*(sqrt(2*hs2/g))/a2;
T3=A3*(sqrt(2*hs3/g))/a3;
T4=A4*(sqrt(2*hs4/g))/a4;

% Linearized state space equation matrices A, B and D
% Equations (4.15) and (4.16)
A=[-1/T1,0,A3/(A1*T3),0;0,-1/T2,0,A4/(A2*T4);0,0,-1/T3,0;0,0,0,-1/T4];
B=[r1*k1/A1,0;0,r2*k2/A2;0,(1-r2)*k2/A3;(1-r1)*k1/A4,0];
D=[kc,0,0,0;0,kc,0,0];

% Results
% Eigen Vales of system matrix
Eigen_values_of_system_matrix_are=eig(A)
% Rank of the system
Rank_of_the_system=rank(A)
% Observability Matrix
obs_mat=obsv(A,D);
Rank_of_Observability_matrix_is=rank(obs_mat)
% Controllability Matrix
c_mat=ctrb(A,B);
Rank_of_controllability_matrix_is=rank(c_mat)
```

## Appendix 4

Implementation of constrained MPC with integral action using *'quadprog'* function in the four-tank process. The user can select either a minimum or non-minimum phase process by typing "1" or "2" respectively. The main script is in Appendix 4.1 and the supporting files (four_tank_model.m, eobsv.m, q2qt.m, prbs1.m, scmat.m, ss2h.m) are given in appendices 4.2 to 4.7.

**Appendix 4.1:** Const_MPC_integral_fourtankprocess.m

```
%==========================================================================
% MATLAB Script for:  Constrained MPC with integral action for the four-tank
                      process including Kalman Filter.
%                     1:Constrained MPC with integral action for minimum phase.
%                     2:Constrained MPC with integral action for non-minimum phase.
% Master's Thesis:    "MPC with Integral Action: Reducing the control horizon.
%                     and model free MPC."
% Supervisior:        David Di Ruscio
% Author:             Muhammad Mohsin
% Script file:        Const_MPC_integral_fourtankprocess.m
% Function files:     four_tank_model.m
%                     eobsv.m, prbs1.m, q2qt.m, scmat.m, ss2h.m by [David Di Ruscio]
% Reference           http://www2.hit.no/tf/fag/sce4106/ovinger/ovinger.html
% -------------------------------------------------------------------------
% Telemark University College, Porsgrunn, Norway [22-03-2013]
%==========================================================================
clc;
clear all;
close all;
System_sel= input  (' Type 1: For Constrainted MPC with integral action for
minimum phase \n or\n Type 2: For Constrainted MPC with integral action for
non-minimum phase \n');
% Table 5.1, Parameters values taken from the literature [Johansson].
% Cross-section area of tanks [cm^2]
A1=28; A3=28;
A2=32; A4=32;
% Cross-section area of the outlet holes[cm^2]
a1=0.071; a3=0.071;
a2=0.057; a4=0.057;
% Pump gain [V/cm]
% kc=0.5;
kc=1;
% Acceleration of gravity [cm/s^2]
g=981;

if  System_sel==1;
% Model will show minimum phase characteristics [Table 5.2 & 5.3]
% Steady state level [cm] in the tanks after performing simulations
hs1=12.3;
hs2=12.8;
hs3=1.63;
hs4=1.41;
% Pump constants [cm^3/Vs]
k1=3.33; k2=3.35;
% Flow constants
r1=0.70; r2=0.60;
```

```
elseif System_sel==2;
% Model will show non-minimum phase characteristics [Table 5.4 & 5.5]
% Steady state level [cm] in the tanks after performing simulations
hs1=12.4;
hs2=13.2;
hs3=4.73;
hs4=4.99;
% Pump constants [cm^3/Vs]
k1=3.14; k2=3.29;
% Flow constants
r1=0.43; r2=0.34;

else display('Number not in range, type 1 or 2')
end
% Time constants [section 4.1.2]
T1=A1*(sqrt(2*hs1/g))/a1;
T2=A2*(sqrt(2*hs2/g))/a2;
T3=A3*(sqrt(2*hs3/g))/a3;
T4=A4*(sqrt(2*hs4/g))/a4;
% Equation (4.15) and (4.16), Linearized state space matrics Ac, Bc and Dc
Ac=[-1/T1,0,A3/(A1*T3),0;0,-1/T2,0,A4/(A2*T4);0,0,-1/T3,0;0,0,0,-1/T4];
Bc=[r1*k1/A1,0;0,r2*k2/A2;0,(1-r2)*k2/A3;(1-r1)*k1/A4,0];
Dc=[kc,0,0,0;0,kc,0,0];
% Coverting model from continuous to discrete time
% Method 'zoh' assuming a zero order hold on the inputs.
% Sampling time [s]
ts=0.1;
[A,B,D]=c2dm(Ac,Bc,Dc,zeros(2),ts,'zoh');
% Making augmented state space model.
l=size(A,1);
m=size(B,2);
n=size(D,1);
% Matrics At, Bt and Dt
At=[A zeros(l,n); D eye(n,n)];
Bt=[B;zeros(n,m)];
Dt=[D eye(n,n)];
% Model predictive control algorithm,
% Prediction horizon
L=15;
% Weighting matrices
q=100;
r=0.1;
Q=[q 0; 0 q];
R=[r 0; 0 r];
% Observability matrix OL and Toeplitz matrix HdL calculation by ss2h
function.[David Di Ruscio]
[HdL,OL,OLB]=ss2h(At,Bt,Dt,zeros(n,m),L,0);
FL=[OLB HdL];
Qt=q2qt(Q,L);
Rt=q2qt(R,L);
% Make matrices S and c in the relationship, u(k,L) = S du(k,L) + c u(k-
1).[David Di Ruscio]
[S,c] = scmat(m,L);
% H as defined in equation (3.50)
H=FL'*Qt*FL+Rt;
% Simulation time
N=515;
```

```
% Defining inital values for simulation
Is=[11.5;12.5];
Hd=D*inv(eye(4)-A)*B;
u_ss=inv(Hd)*Is;
I=eye(4);
x_ss =inv(I-A)*B*u_ss;
% Assigning variables
u=u_ss;
h=x_ss;
u_old=u; h_old=h; y_old=D*h;
% Make the references by using the function prbs1.m [David Di Ruscio]
rand('seed',0),randn('seed',0);
step=150;
ref=[11.5*ones(N,1)+0.1*prbs1(N,step,step),12.5*ones(N,1)+0.1*prbs1(N,step,st
ep)];
% Used in Input amplitude constraint implementation.
umax=5;
umin=0;
% Making for variable storage
r1L=zeros(15,1);
Voltage=zeros(N-L,2);
Level_meas=zeros(N-L,4);
Level_est=zeros(N-L,4);
out_level=zeros(N-L,2);
% Kalman Gain calculation.
% A is Transition matrix, G is process noise gain matrix
% D is a measurement gain matrix, Q1 and R are processed and measurement
% auto-covariance matrices.
G=eye(4);
Q1=200*G;
r1=0.1;
R1=[r1 0; 0 r1];
[K,P,Z,E]=dlqe(A,G,D,Q1,R1);
h_est=h_old;

for k=1:N-L
% Estimated output y.
y=D*h_est;
% Kalman Filter Algorithm [Section 3.5]

hb=h_old;         % Apriori (Predicted) state estimate
yk=D*hb;          % Measurement model update
ek=y-yk;          % Estimator error
hb=h_old+K*ek;    % Aposteriori state estimate
h_est=[hb(1);hb(2);hb(3);hb(4)];
% Make the extended reference vector, r_(k,L). [David Di Ruscio]
rf =ref(k+1:k+L,:);
r1L=rf(1,:)';
for i=2:L
r1L=[r1L;rf(i,:)'];
end

% Computing MPC control
xk=[h_est-h_old;y_old];
pL=OL*At*xk;
```

```
% f as defined in equation (3.50)
f=FL'*Qt*(pL-r1L);
% For constrained MPC [Equation 2.8]
a=[S;-S];
b=[umax*ones(L*m,1)-c*u_old;-umin*ones(L*m,1)+c*u_old];
% quadprog function for input amplitude constraints. [Equation 2.9]
duf=quadprog(H,f,a,b);
u=u+duf(1:m);
u_old=u;

% For plotting purpose store the variables.
Voltage(k,:)=u';
Level_meas(k,:)=h';
Level_est(k,:)=h_est';
out_level(k,:)=y';

% Feed control to the four-tank process.
h_old=h_est;
y_old=y;
h=h+ts*four_tank_model(h,u,System_sel);             % Simulating model for
level measured.
h_est=h_est+ts*four_tank_model(h_est,u,System_sel);% Simulating model for
level estimated.
end

% Plotted Results
t=1:N-L;
% Level results for Tank 1.
figure(1)
subplot(3,1,[1 2])
plot(t, ref(1:N-L,1),'r');hold on
plot(t,out_level(:,1))
ylabel('Reference r1_kand output y1_k [cm]');
title('Tank 1 reference and output level')
legend('Reference r_k','Output y_k')
grid
% Input Control signal for Tank 1
subplot(3,1,3),
plot(t,Voltage(:,1)),
xlabel('Time [s]');
ylabel('voltage [V]')
title('Controller signal u1_k for pump_1')
grid
figure(2)
% Level results for Tank 2.
subplot(3,1,[1 2])
plot (t,ref(1:N-L,2),'r'); hold on
plot(t,out_level(:,2))
ylabel('Reference r2_kand output y2_k [cm]');
title('Tank 2 reference and output level')
legend('Reference r_k','Output y_k')
grid
% Input Control signal for Tank 2
subplot(3,1,3),
plot(t,Voltage(:,2)),
xlabel('Time [s]');
```

```
ylabel('Voltage [V]')
title('Controller signal u2_k for pump_2')
grid
% Results for measured and estimated states for the four - tank process
figure(3)
% Tank 1
subplot(2,2,1)
plot(t,Level_meas(:,1));hold on;
plot(t,Level_est(:,1),'g')
% xlabel('Time [s]');
ylabel('Level [cm]');
title('Tank 1');
legend('Measured level','Estimated level');
grid

% Tank 2
subplot(2,2,2)
plot(t,Level_meas(:,2));hold on;
plot(t,Level_est(:,2),'g')
% xlabel('Time [s]');
ylabel('Level [cm]');
title('Tank 2');
legend('Measured level','Estimated level');
grid

% Tank 3
subplot(2,2,3)
plot(t,Level_est(:,3),'g');
xlabel('Time [s]');
ylabel('Level [cm]');
title('Tank 3');
legend('Estimated level');
grid

% Tank 4
subplot(2,2,4)
plot(t,Level_est(:,4),'g');
xlabel('Time [s]');
ylabel('Level [cm]');
title('Tank 4');
legend('Estimated level');
grid
```

**Appendix 4.2:** Four_tank_Nonlinear_model.m

```
%=========================================================================
% MATLAB Script for:  Model of four-tank process.
% Master's Thesis:    "MPC with Integral Action: Reducing the control horizon
%                     and model free MPC."
% Supervisior:        David Di Ruscio
% Author:             Mohsin (113817)
% File name:          Four_tank_Nonlinear_model.m
% -----------------------------------------------------------------------
% Telemark University College, Porsgrunn, Norway [23-03-2013]
%=========================================================================
 function model=four_tank_model(h,u,System_sel)
```

```
% Table 5.1, parameter values taken from the literature [Johansson].
% Cross-section area of tanks [cm^2]
A1=28; A3=28;
A2=32; A4=32;
% Cross-section area of the outlet holes[cm^2]
a1=0.071; a3=0.071;
a2=0.057; a4=0.057;
% Level in the tanks [cm]
h1=h(1);
h2=h(2);
h3=h(3);
h4=h(4);
% Control inputs [V]
u1=u(1);
u2=u(2);
% Pump gain [V/cm]
kc=1;
% Acceleration of gravity [cm/s^2]
g=981;

if System_sel==1
% Model will show minimum phase characteristics [Table 5.3]
% Pump constants [cm^3/Vs]
k1=3.33; k2=3.35;
% Flow constants
r1=0.70; r2=0.60;

elseif System_sel==2
% Model will show non-minimum phase characteristics [Table 5.5]
% Pump constants [cm^3/Vs]
k1=3.14; k2=3.29;
% Flow constants
r1=0.43; r2=0.34;
end
% The four-tank model. [Equation (4.10)-(4.13)]
f(1)=( -a1*sqrt(2*g*h1) + a3*sqrt(2*g*h3) + r1*k1*u1 )/A1;
f(2)=( -a2*sqrt(2*g*h2) + a4*sqrt(2*g*h4) + r2*k2*u2 )/A2;
f(3)=( -a3*sqrt(2*g*h3) + (1-r2)*k2*u2 )/A3;
f(4)=( -a4*sqrt(2*g*h4) + (1-r1)*k1*u1 )/A4;

model=[f(1);f(2);f(3);f(4)];
```

## **Appendix 4.3:** eobsv.m

```
%========================================================================
% MATLAB Script for:  Function for Compute the Extended Observability matrix O_i
% Master's Thesis:    "MPC with Integral Action: Reducing the control horizon
%                     and model free MPC."
% Author:             David Di Ruscio
% Function file:      eobsv.m
% ----------------------------------------------------------------------
% Telemark University College, Porsgrunn, Norway [23-03-2013]
%========================================================================
function O=eobsv(A,D,i);
% Syntax:
% O_i=eobsv(A,D,i);
% ON INPUT
% A,D  System matrices, n x n system matrix and m x n output matrix.
% I    Number of block rows.
% ON OUTPUT
% O_I  Extended Observability matrix.

[ny,n]=size(D);
O=zeros(i*ny,n);
O(1:ny,:)=D;
w=D;
for j=2:i
    w=w*A;
    in=j*ny;
    O(in-ny+1:in,:)=w;
end
```

## **Appendix 4.4:** q2qt.m

```
%===========================================================================
% MATLAB Script for:  Make extended weight matrix qt=tilde(q),
%                     i.e. qt a block diagonal matrix with q on the diagonal.
% Master's Thesis:    "MPC with Integral Action: Reducing the control horizon
%                     and model free MPC."
% Author:             David Di Ruscio
% Function file:      q2qt.m
% -------------------------------------------------------------------------
% Telemark University College, Porsgrunn, Norway [23-03-2013]
%===========================================================================

function [qt] = q2qt(q,L);
[nc,nr]=size(q);
qt=zeros(nc*L,nr*L);

for i=1:L
  for j=1:L
    if i==j
     qt((i-1)*nc+1:i*nc,(j-1)*nr+1:j*nr)=q;
    end
  end
end
```

## **Appendix 4.5:** prbs1.m

```
%===========================================================================
% MATLAB Script for:  Make a Pseudo Random Binary Signal of length N samples.
% Master's Thesis:    "MPC with Integral Action: Reducing the control horizon
%                     and model free MPC."
% Author:             David Di Ruscio
% Function file:      prbs1.m
% -------------------------------------------------------------------------
% Telemark University College, Porsgrunn, Norway [23-03-2013]
%===========================================================================
function [u,t]=prbs1(N,Tmin,Tmax);
% The signal is constant for a random interval of T samples.
% The random interval T is bounded by a specified band, Tmin <= T <= Tmax.
% ON INPUT:
% N      INTEGER. Number of samples in input signal u, (u_t for all t=1,...,N).
% Tmin   INTEGER. Minimal interval for which u_t is constant.
% Tmax   INTEGER. Maximal interval for which u_t is constant.

% ON OUTPUT:
% u      REAL. The PRBS input signal of lenght N samples.
%-----------------------------------------------------------------------
is=1;
Tmin= Tmin-1;
dT  = Tmax-Tmin;
u=zeros(N,1);                 % Make space for input signal.
if is==0
 s=sign(randn);               % Sign of input (change) at time 0.
else
 s=1;
end
k=1;                           % Initialize integer counter for time to switch.
it=1;                          % Initialize integer counter for # of intervals.
while k < N+1
 T=Tmin+dT*rand; T=ceil(T);   % Compute random time horizon T in
 u(k:k+T-1)=s*ones(T,1);      % the interval [Tmin <= T <= Tmax].
 s=s*(-1);                    % Update sign variable s which is either -1 or 1.
 k=k+T;                       % Update time counter.
 t(it)=T;                     % Save intervals.
 it=it+1;
end
u=u(1:N);                     % Last interval T may be shorter than Tmin.
```

## **Appendix 4.6:** scmat.m

```
%=========================================================================
% MATLAB Script for:  Make matrices S and c in the relationship,
                      u(k,L) = S du(k,L) + c u(k-1)
% Master's Thesis:    "MPC with Integral Action: Reducing the control horizon
%                     and model free MPC."
% Author:             David Di Ruscio
% Function file:      scmat.m
% ------------------------------------------------------------------------
% Telemark University College, Porsgrunn, Norway [22-03-2013]
%=========================================================================
function [S,c] = scmat(nr,L);

S=zeros(nr*L,nr*L); c=zeros(nr*L,nr);
for i=1:L
  for j=1:i
    S((i-1)*nr+1:i*nr,(j-1)*nr+1:j*nr)=eye(nr);
  end
end

for i=1:L
  c((i-1)*nr+1:i*nr,:)=eye(nr);
end
```

**Appendix 4.7:** ss2h.m

```
%==========================================================================
% MATLAB Script for:  Compute the lower block triangular Toeplizt matrix
%                     H^d_L from a state space model given by the quadruple
%                     System matrices (A,B,D,E).
% Master's Thesis:    "MPC with Integral Action: Reducing the control horizon
%                     and model free MPC."
% Author:             David Di Ruscio
% Function file:      ss2h.m
% -------------------------------------------------------------------------
% Telemark University College, Porsgrunn, Norway [22-03-2013]
%==========================================================================
function [H,O,OB]=ss2h(A,B,D,E,L,g);
% SYNTAX
% [H^d_L,O_L,O_L B]=ss2h(A,B,D,E,L,g);
% ON INPUT
% A,B,D,E  System matrices.
% L        Number of block rows in H^d, O_L
% g        Define number of block columns, L+g-1, in H^d_L.
%          Olso g=0 when E=0 and g=1 when E.neq.0.
% ON OUTPUT
% H^D_L, O_L AND O_L B
[ny,nu]=size(E);

O = eobsv(A,D,L);            % The extended observability matrix.
OB= O*B;                     % DB, DAB, and so on.
hi=[E;OB(1:(L-1)*ny,:)];     % The impulse responses which are needed.
                             % The lower block triangular Toeplizt matrix.
for j=1:L+g-1
  H((j-1)*ny+1:L*ny,(j-1)*nu+1:j*nu) =hi(1:(L-j+1)*ny,:);
end
```

## Appendix 5

Implementation of constrained MPC with integral action using '*if-else'* method in the four-tank process. The user can select either a minimum or non-minimum phase process by typing "1" or "2" respectively. The main script is in Appendix 5.1 and the supporting files (four_tank_model.m, eobsv.m, q2qt.m, prbs1.m, scmat.m, ss2h.m) are given in appendices 4.2 to 4.7.

**Appendix 5.1:** Const_MPC_integral_fourtankprocess.m

```
%===========================================================================
% MATLAB Script for:  Constrained MPC with integral action using if-else
%                     method for the four-tank process including Kalman Filter.
%                     1:Constrained MPC with integral action for minimum phase.
%                     2:Constrained MPC with integral action for non-minimum phase.
% Master's Thesis:    "MPC with Integral Action: Reducing the control horizon
%                     and model free MPC."
% Supervisior:        David Di Ruscio
% Author:             Muhammad Mohsin
% Script file:        Const_MPC_integral_ifelse.m
% Function files:     four_tank_model.m
%                     eobsv.m, q2qt.m, prbs1.m, scmat.m, ss2h.m by [David Di Ruscio]
% Reference           http://www2.hit.no/tf/fag/sce4106/ovinger/ovinger.html
% -------------------------------------------------------------------------
% Telemark University College, Porsgrunn, Norway [26-03-2013]
%===========================================================================
clc;
clear all;
close all;
System_sel= input  (' Type 1: For Constrained MPC with integral action for
minimum phase \n or\n Type 2: For Constrained MPC with integral action for
non-minimum phase \n');
% Table 5.1, Parameter values taken from the literature [Johansson].
% Cross-section area of tanks [cm^2]
A1=28; A3=28;
A2=32; A4=32;
% Cross-section area of the outlet holes[cm^2]
a1=0.071; a3=0.071;
a2=0.057; a4=0.057;
% Pump gain [V/cm]
% kc=0.5;
kc=1;
% Acceleration of gravity [cm/s^2]
g=981;

if  System_sel==1;
% Model will show minimum phase characteristics [Table 5.2 & 5.3]
% Steady state level [cm] in the tanks after performing simulations
hs1=12.3;
hs2=12.8;
hs3=1.63;
hs4=1.41;
% Pump constants [cm^3/Vs]
k1=3.33; k2=3.35;
% Flow constants
r1=0.70; r2=0.60;
```

```
elseif System_sel==2;
% Model will show non-minimum phase characteristics [Table 5.4 & 5.5]
% Steady state level [cm] in the tanks after performing simulations
hs1=12.4;
hs2=13.2;
hs3=4.73;
hs4=4.99;
% Pump constants [cm^3/Vs]
k1=3.14; k2=3.29;
% Flow constants
r1=0.43; r2=0.34;

else display('Number not in range, type 1 or 2')
end
% Time constants [section 4.1.2]
T1=A1*(sqrt(2*hs1/g))/a1;
T2=A2*(sqrt(2*hs2/g))/a2;
T3=A3*(sqrt(2*hs3/g))/a3;
T4=A4*(sqrt(2*hs4/g))/a4;
% Equation (4.15) and (4.16), Linearized state space matrices Ac, Bc and Dc
Ac=[-1/T1,0,A3/(A1*T3),0;0,-1/T2,0,A4/(A2*T4);0,0,-1/T3,0;0,0,0,-1/T4];
Bc=[r1*k1/A1,0;0,r2*k2/A2;0,(1-r2)*k2/A3;(1-r1)*k1/A4,0];
Dc=[kc,0,0,0;0,kc,0,0];
% Coverting model from continuous to discrete time
% Method 'zoh' assuming a zero order hold on the inputs.
% Sampling time [s]
ts=0.1;
[A,B,D]=c2dm(Ac,Bc,Dc,zeros(2),ts,'zoh');
% Making augmented state space model.
l=size(A,1);
m=size(B,2);
n=size(D,1);
% Matrics At, Bt and Dt
At=[A zeros(l,n); D eye(n,n)];
Bt=[B;zeros(n,m)];
Dt=[D eye(n,n)];
% Model predictive control algorithm,
% Prediction horizon
L=15;
% Weighting matrices
q=100;
r=0.1;
Q=[q 0; 0 q];
R=[r 0; 0 r];
% Observability matrix OL and Toeplitz matrix HdL calculation by ss2h
function.[David Di Ruscio]
[HdL,OL,OLB]=ss2h(At,Bt,Dt,zeros(n,m),L,0);
FL=[OLB HdL];
Qt=q2qt(Q,L);
Rt=q2qt(R,L);
% Make matrices S and c in the relationship, u(k,L) = S du(k,L) + c u(k-
1).[David Di Ruscio]
[S,c] = scmat(m,L);
% H as defined in equation (3.50)
H=FL'*Qt*FL+Rt;
```

```
% Simulation time
N=515;
% Defining inital values for simulation
Is=[11.5;12.5];
Hd=D*inv(eye(4)-A)*B;
u_ss=inv(Hd)*Is;
I=eye(4);
x_ss =inv(I-A)*B*u_ss;
% Assigning variables
u=u_ss;
h=x_ss;
u_old=u; h_old=h; y_old=D*h;
% Make the references by using the function prbs1.m[David Di Ruscio]
rand('seed',0),randn('seed',0);
step=150;
ref=[11.5*ones(N,1)+0.1*prbs1(N,step,step),12.5*ones(N,1)+0.1*prbs1(N,step,st
ep)];
% Used constraint implementation by using if-else statement.
umax=5;
umin=0;
umax=umax*ones(L*m,1);
umin=umin*ones(L*m,1);
% Making for variable storage
r1L=zeros(15,1);
Voltage=zeros(N-L,2);
Level_meas=zeros(N-L,4);
Level_est=zeros(N-L,4);
out_level=zeros(N-L,2);
% Kalman Gain calculation.
% A is Transition matrix, G is process noise gain matrix
% D is a measurement gain matrix, Q1 and R are process and measurement
% auto-covariance matrices.
G=eye(4);
Q1=200*G;
r1=0.1;
R1=[r1 0; 0 r1];
[K,P,Z,E]=dlqe(A,G,D,Q1,R1);
h_est=h_old;

for k=1:N-L
% Estimated output y.
y=D*h_est;
% Kalman Filter Algorithm [Section 3.5]

hb=h_old;         % Apriori (Predicted) state estimate
yk=D*hb;          % Measurement model update
ek=y-yk;          % Estimator error
hb=h_old+K*ek;    % Aposteriori state estimate
h_est=[hb(1);hb(2);hb(3);hb(4)];
% Make the extended reference vector, r_(k,L).[David Di Ruscio]
rf =ref(k+1:k+L,:);
r1L=rf(1,:)';
for i=2:L
r1L=[r1L;rf(i,:)'];
end
```

```
% Computing MPC control
xk=[h_est-h_old;y_old];
pL=OL*At*xk;
% f as defined in equation (3.50)
f=FL'*Qt*(pL-r1L);

% For constrained MPC by using if-else method [Equation 3.29]
duf=-inv(H)*f;
u=u+duf(1:m);
u_old=u;

if      u(1,1)>umax(1,1)
        u(1,1)=umax(1,1);
elseif u(1,1)<umin(1,1)
        u(1,1)=umin(1,1);
end
if      u(2,1)>umax(2,1)
        u(2,1)=umax(2,1);
elseif u(2,1)<umin(2,1)
        u(2,1)=umin(2,1);
end

% For plotting purpose store the variables.
Voltage(k,:)=u';
Level_meas(k,:)=h';
Level_est(k,:)=h_est';
out_level(k,:)=y';

% Feed control to the four-tank process.
h_old=h_est;
y_old=y;
h=h+ts*four_tank_model(h,u,System_sel);            % Simulating model for
level measured.
h_est=h_est+ts*four_tank_model(h_est,u,System_sel);% Simulating model for
level estimated.
end

% Plotted Results
t=1:N-L;
% Level results for Tank 1.
figure(1)
subplot(3,1,[1 2])
plot(t, ref(1:N-L,1),'r');hold on
plot(t,out_level(:,1))
ylabel('Reference r1_kand output y1_k [cm]');
title('Tank 1 reference and output level')
legend('Reference r_k','Output y_k')
grid
% Input Control signal for Tank 1
subplot(3,1,3),
plot(t,Voltage(:,1)),
xlabel('Time [s]');
ylabel('voltage [V]')
title('Controller signal u1_k for pump_1')
grid
```

```
figure(2)
% Level results for Tank 2.
subplot(3,1,[1 2])
plot (t,ref(1:N-L,2),'r'); hold on
plot(t,out_level(:,2))
ylabel('Reference r2_kand output y2_k [cm]');
title('Tank 2 reference and output level')
legend('Reference r_k','Output y_k')
grid
% Input Control signal for Tank 2
subplot(3,1,3),
plot(t,Voltage(:,2)),
xlabel('Time [s]');
ylabel('Voltage [V]')
title('Controller signal u2_k for pump_2')
grid
% Results for measured and estimated states for the four-tank process
figure(3)
% Tank 1
subplot(2,2,1)
plot(t,Level_meas(:,1));hold on;
plot(t,Level_est(:,1),'g')
% xlabel('Time [s]');
ylabel('Level [cm]');
title('Tank 1');
legend('Measured level','Estimated level');
grid

% Tank 2
subplot(2,2,2)
plot(t,Level_meas(:,2));hold on;
plot(t,Level_est(:,2),'g')
% xlabel('Time [s]');
ylabel('Level [cm]');
title('Tank 2');
legend('Measured level','Estimated level');
grid

% Tank 3
subplot(2,2,3)
plot(t,Level_est(:,3),'g');
xlabel('Time [s]');
ylabel('Level [cm]');
title('Tank 3');
legend('Estimated level');
grid

% Tank 4
subplot(2,2,4)
plot(t,Level_est(:,4),'g');
xlabel('Time [s]');
ylabel('Level [cm]');
title('Tank 4');
legend('Estimated level');
grid
```

## Appendix 6

Implementation of unconstrained MPC with integral action in the four-tank process. The user can select either a minimum or non-minimum phase process by typing "1" or "2" respectively. The main script is in Appendix 6.1 and the supporting files (four_tank_model.m, eobsv.m, q2qt.m, prbs1.m, scmat.m, ss2h.m) are given in appendices 4.2 to 4.7.

**Appendix 6.1:** Const_MPC_integral_fourtankprocess.m

```
%==========================================================================
% MATLAB Script for:  Unconstrained MPC with integral action for the four-
                      tank process including Kalman Filter.
%                    1:Unconstrained MPC with integral action for minimum phase.
%                     2:Unconstrained MPC with integral action for non-minimum phase.
% Master's Thesis:    "MPC with Integral Action: Reducing the control horizon
%                     and model free MPC."
% Supervisior:        David Di Ruscio
% Author:             Muhammad Mohsin
% Script file:        Unconst_MPC_integral_fourtankprocess.m
% Function files:     four_tank_model.m
%                     eobsv.m, q2qt.m, prbs1.m, scmat.m, ss2h.m by [David Di Ruscio]
% Reference           http://www2.hit.no/tf/fag/sce4106/ovinger/ovinger.html
% ------------------------------------------------------------------------
% Telemark University College, Porsgrunn, Norway [29-03-2013]
%==========================================================================
clc;
clear all;
close all;
System_sel= input (' Type 1: For uncnstrainted MPC with integral action for
minimum phase \n or\n Type 2: For unconstrainted MPC with integral action for
non-minimum phase \n');

% Table 5.1, Parameter values taken from the literature [Johansson].
% Cross-section area of tanks [cm^2]
A1=28; A3=28;
A2=32; A4=32;
% Cross-section area of the outlet holes[cm^2]
a1=0.071; a3=0.071;
a2=0.057; a4=0.057;
% Pump gain [V/cm]
Kc=0.5;
kc=1;
% Acceleration of gravity [cm/s^2]
g=981;
if  System_sel==1;
% Model will show minimum phase characteristics
% Table 5.3,Steady state level [cm] in the tanks after performing simulations
hs1=12.3;
hs2=12.8;
hs3=1.63;
hs4=1.41;
% Pump constants [cm^3/Vs]
k1=3.33; k2=3.35;
% Flow constants
r1=0.70; r2=0.60;
elseif System_sel==2;
```

```
% Model will show non-minimum phase characteristics
% Table 5.5,Steady state level [cm] in the tanks after performing simulations
hs1=12.4;
hs2=13.2;
hs3=4.73;
hs4=4.99;
% Pump constants [cm^3/Vs]
k1=3.14; k2=3.29;
% Flow constants
r1=0.43; r2=0.34;

else display('Number not in range, type 1 or 2')
end
% Time constants [section 4.1.2]
T1=A1*(sqrt(2*hs1/g))/a1;
T2=A2*(sqrt(2*hs2/g))/a2;
T3=A3*(sqrt(2*hs3/g))/a3;
T4=A4*(sqrt(2*hs4/g))/a4;
% Equation (4.15) and (4.16), Linearized state space matrices Ac, Bc and Dc
Ac=[-1/T1,0,A3/(A1*T3),0;0,-1/T2,0,A4/(A2*T4);0,0,-1/T3,0;0,0,0,-1/T4];
Bc=[r1*k1/A1,0;0,r2*k2/A2;0,(1-r2)*k2/A3;(1-r1)*k1/A4,0];
Dc=[kc,0,0,0;0,kc,0,0];
% Coverting model from continuous to discrete time
% Method 'zoh' assuming a zero order hold on the inputs.
% Sampling time [s]
ts=0.1;
[A,B,D]=c2dm(Ac,Bc,Dc,zeros(2),ts,'zoh');
% Making augmented state space model.
l=size(A,1);
m=size(B,2);
n=size(D,1);
% Matrics At, Bt and Dt
At=[A zeros(l,n); D eye(n,n)];
Bt=[B;zeros(n,m)];
Dt=[D eye(n,n)];
% Model predictive control algorithm,
% Prediction horizon
L=15;
% Weighting matrices
q=100;
r=0.1;
Q=[q 0; 0 q];
R=[r 0; 0 r];
% Observability matrix OL and Toeplitz matrix HdL calculation by ss2h
function.[David Di Ruscio]
[HdL,OL,OLB]=ss2h(At,Bt,Dt,zeros(n,m),L,0);
FL=[OLB HdL];
Qt=q2qt(Q,L);
Rt=q2qt(R,L);
% Make matrices S and c in the relationship,
% u(k,L) = S du(k,L) + c u(k-1).[David Di Ruscio]
[S,c] = scmat(m,L);
H=FL'*Qt*FL+Rt;
% Simulation time
N=515;
```

```
% Defining inital values for simulation
Is=[11.5;12.5];
Hd=D*inv(eye(4)-A)*B;
u_ss=inv(Hd)*Is;
I=eye(4);
x_ss =inv(I-A)*B*u_ss;
% Assigning variables
u=u_ss;
h=x_ss;
u_old=u; h_old=h; y_old=D*h;
% Make the references by using the function prbs.[David Di Ruscio]
rand('seed',0),randn('seed',0);
step=150;
ref=[11.5*ones(N,1)+0.2*prbs1(N,step,step),12.5*ones(N,1)+0.2*prbs1(N,step,st
ep)];
% Making for variable storage
r1L=zeros(15,1);
Voltage=zeros(N-L,2);
Level_meas=zeros(N-L,4);
Level_est=zeros(N-L,4);
out_level=zeros(N-L,2);
% Kalman Gain calculation.
% A is Transition matrix, G is process noise gain matrix
% D is a measurement gain matrix, Q1 and R are process and measurement
% auto-covariance.
G=eye(4);
Q1=100*G;
r1=0.1;
R1=[r1 0; 0 r1];
[K,P,Z,E]=dlqe(A,G,D,Q1,R1);
h_est=h_old;

for k=1:N-L
% Estimated output y.
y=D*h_est;
% Kalman Filter Algorithm [Section 3.5]
hb=h_old;         % Apriori (Predicted) state estimate
yk=D*hb;          % Measurement model update
ek=y-yk;          % Estimator error
hb=h_old+K*ek;    % Aposteriori (corrected) state estimate
h_est=[hb(1);hb(2);hb(3);hb(4)];
% Make the extended reference vector, r_(k,L),[David Di Ruscio]
rf =ref(k+1:k+L,:);
r1L=rf(1,:)';
for i=2:L
r1L=[r1L;rf(i,:)'];
end
% Computing MPC control
xk=[h_est-h_old;y_old];
pL=OL*At*xk;
% f as defined in equation (3.50)
f=FL'*Qt*(pL-r1L);
% For unconstrained MPC [Equation 3.9]
duf=-inv(H)*f;
u=u+duf(1:m);
uold=u;
```

```
% For plotting purpose store the variables.
Voltage(k,:)=u';
Level_meas(k,:)=h';
Level_est(k,:)=h_est';
out_level(k,:)=y';

% Feed control to the four-tank process.
h_old=h_est;
y_old=y;
h=h+ts*four_tank_model(h,u,System_sel);            % Simulating model for
level measured.
h_est=h_est+ts*four_tank_model(h_est,u,System_sel);% Simulating model for
level estimated.
end
% Plotted Results
t=1:N-L;
% Level results for Tank 1.
figure(1)
subplot(3,1,[1 2])
plot(t, ref(1:N-L,1),'r');hold on
plot(t,out_level(:,1))
ylabel('Reference r1_kand output y1_k [cm]');
title('Tank 1 reference and output level')
legend('Reference r_k','Output y_k')
grid
% Input Control signal for Tank 1
subplot(3,1,3),
plot(t,Voltage(:,1)),
xlabel('Time [s]');
ylabel('voltage [V]')
title('Controller signal u1_k for pump_1')
grid
figure(2)
% Level results for Tank 2.
subplot(3,1,[1 2])
plot (t,ref(1:N-L,2),'r'); hold on
plot(t,out_level(:,2))
ylabel('Reference r2_kand output y2_k [cm]');
title('Tank 2 reference and output level')
legend('Reference r_k','Output y_k')
grid
% Input Control signal for Tank 2
subplot(3,1,3),
plot(t,Voltage(:,2)),
xlabel('Time [s]');
ylabel('Voltage [V]')
title('Controller signal u2_k for pump_2')
grid

% Results for measured and estimated states for the four-tank process.
figure(3)
% Tank 1
subplot(2,2,1)
plot(t,Level_meas(:,1));hold on;
plot(t,Level_est(:,1),'g')
```

```
% xlabel('Time [s]');
ylabel('Level [cm]');
title('Tank 1');
legend('Measured level','Estimated level');
grid

% Tank 2
subplot(2,2,2)
plot(t,Level_meas(:,2));hold on;
plot(t,Level_est(:,2),'g')
% xlabel('Time [s]');
ylabel('Level [cm]');
title('Tank 2');
legend('Measured level','Estimated level');
grid

% Tank 3
subplot(2,2,3)
plot(t,Level_est(:,3),'g');
xlabel('Time [s]');
ylabel('Level [cm]');
title('Tank 3');
legend('Estimated level');
grid

% Tank 4
subplot(224)
plot(t,Level_est(:,4),'g');
xlabel('Time [s]');
ylabel('Level [cm]');
title('Tank 4');
legend('Estimated level');
grid
```

## Appendix 7

Performance comparison by reducing the control horizon in MPC with integral action algorithm. The user can select either a minimum or non-minimum phase process by typing "1" or "2" respectively. The main script is in Appendix 7.1 and the supporting files (four_tank_model.m, eobsv.m, q2qt.m, prbs1.m, scmat.m, ss2h.m) are given in appendices 4.2 to 4.7.

**Appendix 7.1:** Reduced_controlH_MPC_integral.m

```
%=========================================================================
% MATLAB Script for:  Reducing control horizon in MPC with integral action
                      for the four-tank process .
%                     1:Constrained MPC with integral action for minimum phase.
%                     2:Constrained MPC with integral action for non-minimum phase.
% Master's Thesis:    "MPC with Integral Action: Reducing the control horizon
%                     and model free MPC."
% Supervisior:        David Di Ruscio
% Author:             Muhammad Mohsin
% Script file:        Reduced_controlH_MPC_integral.m
% Function files:     four_tank_model.m
%                     eobsv.m, q2qt.m, prbs1.m, scmat.m, ss2h.m by [David Di Ruscio]
% Reference           http://www2.hit.no/tf/fag/sce4106/ovinger/ovinger.html
% ------------------------------------------------------------------------
% Telemark University College, Porsgrunn, Norway [20-04-2013]
%=========================================================================
clc;
clear all;
close all;
System_sel= input  (' Type 1: MPC with integral action for minimum phase \n
or\n Type 2: MPC with integral action for non-minimum phase \n');
% Table 5.1, Parameter values taken from the literature [Johansson].
% Cross-section area of tanks [cm^2]
A1=28; A3=28;
A2=32; A4=32;
% Cross-section area of the outlet holes[cm^2]
a1=0.071; a3=0.071;
a2=0.057; a4=0.057;
% Pump gain [V/cm]
% kc=0.5;
kc=1;
% Acceleration of gravity [cm/s^2]
g=981;

if  System_sel==1;
% Model will show minimum phase characteristics [Table 5.2 & 5.3]
% Steady state level [cm] in the tanks after performing simulations
hs1=12.3;
hs2=12.8;
hs3=1.63;
hs4=1.41;
% Pump constants [cm^3/Vs]
k1=3.33; k2=3.35;
% Flow constants
r1=0.70; r2=0.60;
elseif System_sel==2;
```

```
% Model will show non-minimum phase characteristics  [Table 5.4 & 5.5]
% Steady state level [cm] in the tanks after performing simulations
hs1=12.4;
hs2=13.2;
hs3=4.73;
hs4=4.99;
% Pump constants [cm^3/Vs]
k1=3.14; k2=3.29;
% Flow constants
r1=0.43; r2=0.34;
else display('Number not in range, type 1 or 2')
end
% Time constants [section 4.1.2]
T1=A1*(sqrt(2*hs1/g))/a1;
T2=A2*(sqrt(2*hs2/g))/a2;
T3=A3*(sqrt(2*hs3/g))/a3;
T4=A4*(sqrt(2*hs4/g))/a4;
% Equation (4.15) and (4.16), Linearized state space matrics Ac, Bc and Dc
Ac=[-1/T1,0,A3/(A1*T3),0;0,-1/T2,0,A4/(A2*T4);0,0,-1/T3,0;0,0,0,-1/T4];
Bc=[r1*k1/A1,0;0,r2*k2/A2;0,(1-r2)*k2/A3;(1-r1)*k1/A4,0];
Dc=[kc,0,0,0;0,kc,0,0];
% Coverting model from continuous to discrete time
% Method 'zoh' assuming a zero order hold on the inputs.
% Sampling time [s]
ts=0.1;
[A,B,D]=c2dm(Ac,Bc,Dc,zeros(2),ts,'zoh');
% Making augmented state space model.
l=size(A,1);
m=size(B,2);
n=size(D,1);
% Matrics At, Bt and Dt
At=[A zeros(l,n); D eye(n,n)];
Bt=[B;zeros(n,m)];
Dt=[D eye(n,n)];
% Model predictive control algorithm,
% Prediction horizon
L=15;
% Control horizon
 Lu=10;
% Lu=4;
% Lu=2;
% Weighting matrices
q=100;
r=0.1;
Q=[q 0; 0 q];
R=[r 0; 0 r];
% Observability matrix OL and Toeplitz matrix HdL calculation by ss2h
function.[David Di Ruscio]
[HdL,OL,OLB]=ss2h(At,Bt,Dt,zeros(n,m),Lu,0);
FL=[OLB HdL];
Qt=q2qt(Q,Lu);
Rt=q2qt(R,Lu);
% Make matrices S and c in the relationship, u(k,L) = S du(k,L) + c u(k-
1).[David Di Ruscio]
[S,c] = scmat(m,Lu);
% H as defined in equation (3.50)
H=FL'*Qt*FL+Rt;
```

```
% Simulation time
N=215;
% Defining inital values for simulation
Is=[11.5;12.5];
Hd=D*inv(eye(4)-A)*B;
u_ss=inv(Hd)*Is;
I=eye(4);
x_ss =inv(I-A)*B*u_ss;
% Assigning variables
u=u_ss;
h=x_ss;
u_old=u; h_old=h; y_old=D*h;
% Make the references by using the function prbs.[David Di Ruscio]
rand('seed',0),randn('seed',0);
step=100;
ref=[11.5*ones(N,1)+0.1*prbs1(N,step,step),12.5*ones(N,1)+0.1*prbs1(N,step,st
ep)];
% Used in Input amplitude constraint implementation.
umax=5;
umin=0;
% Making for variable storage
r1L=zeros(15,1);
Voltage=zeros(N-L,2);
Level_meas=zeros(N-L,4);
Level_est=zeros(N-L,4);
out_level=zeros(N-L,2);
% Kalman Gain calculation.
% A is Transition matrix, G is process noise gain matrix
% D is a measurement gain matrix, Q1 and R are process and measurement
% auto-covariance matrices.
G=eye(4);
Q1=200*G;
r1=0.1;
R1=[r1 0; 0 r1];
[K,P,Z,E]=dlqe(A,G,D,Q1,R1);
h_est=h_old;

for k=1:N-L
% Estimated output y.
y=D*h_est;
% Kalman Filter Algorithm [Section 3.5]

hb=h_old;        % Apriori (Predicted) state estimate
yk=D*hb;         % Measurement model update
ek=y-yk;         % Estimator error
hb=h_old+K*ek;   % Aposteriori state estimate
h_est=[hb(1);hb(2);hb(3);hb(4)];
% Make the extended reference vector, r_(k,L) .[David Di Ruscio]
rf =ref(k+1:k+Lu,:);
r1L=rf(1,:)';
for i=2:Lu
r1L=[r1L;rf(i,:)'];
end

% Computing MPC control
xk=[h_est-h_old;y_old];
```

```
pL=OL*At*xk;
f=FL'*Qt*(pL-r1L);

% For constrained MPC [Equation 2.8]
a=[S;-S];
b=[umax*ones(Lu*m,1)-c*u_old;-umin*ones(Lu*m,1)+c*u_old];
% quadprog function for input amplitude constraints. [Equation 2.29]
duf=quadprog(H,f,a,b);
u=u+duf(1:m);
u_old=u;

% For plotting purpose store the variables.
Voltage(k,:)=u';
Level_meas(k,:)=h';
Level_est(k,:)=h_est';
out_level(k,:)=y';

% Feed control to the four-tank process.
h_old=h_est;
y_old=y;
h=h+ts*four_tank_model(h,u,System_sel); % Simulating model for level measured.
h_est=h_est+ts*four_tank_model(h_est,u,System_sel);% Simulating model for
level estimated.
end
% Plotted Results
t=1:N-L;
% Level results for Tank 1.
figure(1)
subplot(3,1,[1 2])
plot(t, ref(1:N-L,1),'r');hold on
plot(t,out_level(:,1))
ylabel('Reference r1_kand output y1_k [cm]');
title('Tank 1 reference and output level')
legend('Reference r_k','Output y_k')
grid
% Input Control signal for Tank 1
subplot(3,1,3),
plot(t,Voltage(:,1)),
xlabel('Time [s]');
ylabel('voltage [V]')
title('Controller signal u1_k for pump_1')
grid
figure(2)
% Level results for Tank 2.
subplot(3,1,[1 2])
plot (t,ref(1:N-L,2),'r'); hold on
plot(t,out_level(:,2))
ylabel('Reference r2_kand output y2_k [cm]');
title('Tank 2 reference and output level')
legend('Reference r_k','Output y_k')
grid

% Input Control signal for Tank 2
subplot(3,1,3),
plot(t,Voltage(:,2)),
xlabel('Time [s]');
```

```
ylabel('Voltage [V]')
title('Controller signal u2_k for pump_2')
grid
% Results for measured and estimated states for four-tank process
figure(3)
% Tank 1
subplot(2,2,1)
plot(t,Level_meas(:,1));hold on;
plot(t,Level_est(:,1),'g')
% xlabel('Time [s]');
ylabel('Level [cm]');
title('Tank 1');
legend('Measured level','Estimated level');
grid

% Tank 2
subplot(2,2,2)
plot(t,Level_meas(:,2));hold on;
plot(t,Level_est(:,2),'g')
% xlabel('Time [s]');
ylabel('Level [cm]');
title('Tank 2');
legend('Measured level','Estimated level');
grid

% Tank 3
subplot(2,2,3)
plot(t,Level_est(:,3),'g');
xlabel('Time [s]');
ylabel('Level [cm]');
title('Tank 3');
legend('Estimated level');
grid

% Tank 4
subplot(224)
plot(t,Level_est(:,4),'g');
xlabel('Time [s]');
ylabel('Level [cm]');
title('Tank 4');
legend('Estimated level');
grid
```

## Appendix 8

RGA analysis to determine the optimal input-output variable pairing for the four-tank process. The user can select either a minimum or non-minimum phase process by typing "1" or "2" respectively.

**Appendix 8.1:** RGA_analysis.m

```
%=========================================================================
% MATLAB Script for:  RGA analysis to determine the optimal input-output
%                     variable pairing for
%                     four tank process.
% Master's Thesis:    "MPC with Integral Action: Reducing the control horizon
%                     and model free MPC."
% Supervisior:        David Di Ruscio
% Author              Muhammad Mohsin
% Script file:        RGA_analysis.m
% -----------------------------------------------------------------------
% Telemark University College, Porsgrunn, Norway [21-04-2013]
%=========================================================================
clc;
clear all;
close all;

% Minimum or non-minimum phase selection
System_sel= input  (' Type 1 for minimum phase system or\n Type 2 for non-
minimum phase system \n');

% RGA analysis for minimum phase
if  System_sel==1;
display('Minimum phase process')
% Equation (5.8)
G=[2.6 1.5;1.4 2.8]

% RGA analysis for non-minimum phase
elseif System_sel==2;
display('Non-minimum phase process')
% Equation (5.9)
G=[1.5 2.5; 2.5 1.6]
else display('Number not in range, type 1 or 2')
end
% Results
A=G.*(inv(G))'
disp('RGA Result');
% After the RGA analysis
if  System_sel==1;
disp('u1 will control output y1 and u2 will control output y2');
elseif System_sel==2;
disp('u1 will control output y2 and u2 will control output y1')
end
```

## Appendix 9

Implementation of the PI controller on the four-tank minimum phase process. The main script and supporting file are given in appendices 9.1 to 9.2 respectively.

**Appendix 9.1:** PI_fourtank_minimum_phase.m

```
%=========================================================================
% MATLAB Script for:  PI controller implementation in four-tank minimum phase
process
% Master's Thesis:    "MPC with Integral Action: Reducing the control horizon
%                     and model free MPC."
% Supervisior:        David Di Ruscio
% Author              Muhammad Mohsin
% Script file:        PI_fourtank_minimum_phase.m
% Function file:      four_tank_model.m, prbs1.m
% -----------------------------------------------------------------------
% Telemark University College, Porsgrunn, Norway [21-04-2013]
%=========================================================================


clc;
clear all;
close all;
% Table 5.1, Parameter values taken from the literature [Johansson].
% Cross-section area of tanks [cm^2]
A1=28; A3=28;
A2=32; A4=32;
% Cross-section area of the outlet holes[cm^2]
a1=0.071; a3=0.071;
a2=0.057; a4=0.057;
% Pump gain [V/cm]
% kc=0.5;
kc=1;
% Acceleration of gravity [cm/s^2]
g=981;
% Model will show minimum phase characteristics
% Table 5.3,Steady state level [cm] in the tanks after performing simulations
hs1=12.3;
hs2=12.8;
hs3=1.63;
hs4=1.41;
hs=[hs1,hs2,hs3,hs4]';
% Control inputs [V]
u1=3.0; u2=3.0;
u_inp=[u1; u2];
% Pump constants [cm^3/Vs]
k1=3.33; k2=3.35;
% Flow constants
r1=0.70; r2=0.60;

% Time constants [section 4.1.2]
T1=A1*(sqrt(2*hs1/g))/a1;
T2=A2*(sqrt(2*hs2/g))/a2;
T3=A3*(sqrt(2*hs3/g))/a3;
T4=A4*(sqrt(2*hs4/g))/a4;
```

```
% Equation (4.15) and (4.16), Linearized state space matrics Ac, Bc and Dc
Ac=[-1/T1,0,A3/(A1*T3),0;0,-1/T2,0,A4/(A2*T4);0,0,-1/T3,0;0,0,0,-1/T4];
Bc=[r1*k1/A1,0;0,r2*k2/A2;0,(1-r2)*k2/A3;(1-r1)*k1/A4,0];
Dc=[kc,0,0,0;0,kc,0,0];
% Coverting model from continuous to discrete time
% Method 'zoh' assuming a zero order hold on the inputs.
% Sampling time [s]
ts=0.1;
[A,B,D]=c2dm(Ac,Bc,Dc,zeros(2),ts,'zoh');
% Setpoint
ref=[hs(1);hs(2)];
% PI controller tuning parameters
kp1=4; kp2=3.5;
Ti1=9; Ti2=10;
% Initial conditions
hs_old=hs;
h=hs;
u_old=u_inp;
y_old=D*hs;
e_old=ref-y_old;
% Control signal range 0-5[V]
umin=0*ones(2,1);
umax=5*ones(2,1);

% Simulation time
T=2000;
% Simulation
i=300;
j=1100;
for k=1:T
    if      k<i;         ref=[hs(1);hs(2)];
    elseif  k>i && k<j;  ref=[hs(1)+0.5; hs(2)+0.5];
    elseif  k>j;         ref=[hs(1)+0.8;hs(2)+0.8];
    end
y=D*h;
e=ref-y;
% Equation (5.2) and (5.3)
u1=u_old(1)+kp1*(e(1)-e_old(1))+kp1*ts*e(1)/Ti1;
u2=u_old(2)+kp2*(e(2)-e_old(2))+kp2*ts*e(2)/Ti2;
u=[u1;u2];

if      u(1,1)>umax(1,1)
        u(1,1)=umax(1,1);
elseif u(1,1)<umin(1,1)
        u(1,1)=umin(1,1)
end
if      u(2,1)>umax(2,1)
        u(2,1)=umax(2,1);
elseif u(2,1)<umin(2,1)
        u(2,1)=umin(2,1)
end
e_old=e;
hs_old=h;
u_old=u;
y_old=y;
```

```
% Storing variables
Reference(k,:)=ref';
Voltage(k,:)=u';
Output(k,:)=y';
h=h+ts*four_tank_model(h,u);
end
% Results
% Plotted Results
t=1:T;
% Level results for Tank 1.
figure(1)
subplot(2,1,1)
plot(t,Reference(:,1),'r'), hold on
plot(t,Output(:,1));
xlabel('Time [s]');
ylabel('Reference r1_kand output y1_k [cm]');
title('Tank 1 reference and output level')
legend('Reference r_k','Output y_k')
grid on
% Input Control signal for Tank 1
subplot(2,1,2),
plot(t,Voltage(:,1)),
xlabel('Time [s]');
ylabel('Control input u1_k [V]')
title('Input control signal for Tank 1')
grid on
% Level results for Tank 2.
figure(2)
subplot(2,1,1)
plot(t, Reference(:,2),'r'), hold on
plot(t,Output(:,2))
xlabel('Time [s]');
ylabel('Reference r2_kand output y2_k [cm]');
title('Tank 2 reference and output level')
legend('Reference r_k','Output y_k')
grid on
% Input Control signal for Tank 2
subplot(2,1,2),
plot(t,Voltage(:,2)),
xlabel('Time [s]');
ylabel('Control input u2_k [V]')
title('Input control signal for Tank 2')
grid on
```

**Appendix 9. 2:** four_tank_model.m

```
%==========================================================================
% MATLAB Script for:  Model of four tank process.
% Master's Thesis:    "MPC with Integral Action: Reducing the control horizon
%                     and model free MPC."
% Supervisior:        David Di Ruscio
% Author              Muhammad Mohsin
% File name:          four_tank_model.m
% -------------------------------------------------------------------------
% Telemark University College, Porsgrunn, Norway [21-04-2013]
%==========================================================================
function model=four_tank_model(h,u)
% Table 5.1, parameters values taken from the literature [Johansson].
% Cross-section area of tanks [cm^2]
A1=28; A3=28;
A2=32; A4=32;
% Cross-section area of the outlet holes[cm^2]
a1=0.071; a3=0.071;
a2=0.057; a4=0.057;
% Level in the tanks [cm]
h1=h(1);
h2=h(2);
h3=h(3);
h4=h(4);
% Control inputs [V]
u1=u(1);
u2=u(2);
% Pump gain [V/cm]
kc=1;
% Acceleration of gravity [cm/s^2]
g=981;
% Model will show minimum phase characteristics [Table 5.3]
% Pump constants [cm^3/Vs]
k1=3.33; k2=3.35;
% Flow constants
r1=0.70; r2=0.60;
% The four-tank model. [Equation (4.10)-(4.13)]
f(1)=( -a1*sqrt(2*g*h1) + a3*sqrt(2*g*h3) + r1*k1*u1 )/A1;
f(2)=( -a2*sqrt(2*g*h2) + a4*sqrt(2*g*h4) + r2*k2*u2 )/A2;
f(3)=( -a3*sqrt(2*g*h3) + (1-r2)*k2*u2 )/A3;
f(4)=( -a4*sqrt(2*g*h4) + (1-r1)*k1*u1 )/A4;
model=[f(1);f(2);f(3);f(4)];
```

## Appendix 10

Implementation of the PI controller on the four-tank non-minimum phase process. The main script and supporting file are given in appendices 10.1 to 10.2 respectively.

**Appendix 10.1:** PI_fourtank_nonminimum_phase.m

```
%==========================================================================
% MATLAB Script for:  PI controller implementation in four-tank non-minimum
phase process
% Master's Thesis:    "MPC with Integral Action: Reducing the control horizon
%                     and model free MPC."
% Supervisior:        David Di Ruscio
% Author              Muhammad Mohsin
% Script file:        PI_fourtank_nonminimum_phase.m
% Function file:      four_tank_model.m
% --------------------------------------------------------------------------
% Telemark University College, Porsgrunn, Norway [21-04-2013]
%==========================================================================


clc;
clear all;
close all;
% Table 5.1, Parameter values taken from the literature [Johansson].
% Cross-section area of tanks [cm^2]
A1=28; A3=28;
A2=32; A4=32;
% Cross-section area of the outlet holes[cm^2]
a1=0.071; a3=0.071;
a2=0.057; a4=0.057;
% Pump gain [V/cm]
% kc=0.5;
kc=1;
% Acceleration of gravity [cm/s^2]
g=981;
% Model will show non-minimum phase characteristics
% Table 5.5, Steady state level [cm] in the tanks after performing
simulations
hs1=12.4;
hs2=13.2;
hs3=4.73;
hs4=4.99;
hs=[hs1,hs2,hs3,hs4]';
% Control inputs [V]
u1=3.15; u2=3.15;
u_inp=[u1; u2];
% Pump constants [cm^3/Vs]
k1=3.14;k2=3.29;
% Flow constants
r1=0.43; r2=0.34;
% Time constants [section 4.1.2]
T1=A1*(sqrt(2*hs1/g))/a1;
T2=A2*(sqrt(2*hs2/g))/a2;
T3=A3*(sqrt(2*hs3/g))/a3;
T4=A4*(sqrt(2*hs4/g))/a4;
```

```
% Equation (4.15) and (4.16), Linearized state space matrices Ac, Bc and Dc
Ac=[-1/T1,0,A3/(A1*T3),0;0,-1/T2,0,A4/(A2*T4);0,0,-1/T3,0;0,0,0,-1/T4];
Bc=[r1*k1/A1,0;0,r2*k2/A2;0,(1-r2)*k2/A3;(1-r1)*k1/A4,0];
Dc=[kc,0,0,0;0,kc,0,0];
% Coverting model from continuous to discrete time
% Method 'zoh' assuming a zero order hold on the inputs.
% Sampling time [s]
ts=0.1;
[A,B,D]=c2dm(Ac,Bc,Dc,zeros(2),ts,'zoh');
% Setpoint
ref=[hs(1);hs(2)];
% PI controller tuning parameters
kp1=1.4; kp2=0.22;
Ti1=100; Ti2=135;
% Initial conditions
hs_old=hs;
h=hs;
u_old=u_inp;
y_old=D*hs;
e_old=ref-y_old;
% Control signal range 0-5[V]
umin=0*ones(2,1);
umax=5*ones(2,1);
% Simulation time
T=5000;
% Simulation
i=300;
for k=1:T
    if      k<i;         ref=[hs(1);hs(2)];
    elseif  k>i;         ref=[hs(1)+0.5; hs(2)+0.5];
    end
y=D*h;
e=ref-y;
u1=u_old(1)+kp2*(e(2)-e_old(2))+kp2*ts*e(2)/Ti2;
u2=u_old(2)+kp1*(e(1)-e_old(1))+kp1*ts*e(1)/Ti1;
u=[u1;u2];
% Constraints implementation using if-else statement
if      u(1,1)>umax(1,1)
        u(1,1)=umax(1,1)
elseif u(1,1)<umin(1,1)
        u(1,1)=umin(1,1)
end
if      u(2,1)>umax(2,1)
        u(2,1)=umax(2,1)
elseif u(2,1)<umin(2,1)
        u(2,1)=umin(2,1)
end
e_old=e;
hs_old=h;
u_old=u;
y_old=y;
% Storing variables
Reference(k,:)=ref';
Voltage(k,:)=u';
Output(k,:)=y';
h=h+ts*four_tank_model(h,u);
end
```

```
% Results
% Plotted Results
t=1:T;
% Level results for Tank 1.
figure(1)
subplot(2,1,1)
plot(t, Reference(:,1),'r');hold on
plot(t,Output(:,1))
xlabel('Time [s]');
ylabel('Reference r1_kand output y1_k [cm]');
title('Tank 1 reference and output level')
legend('Reference r_k','Output y_k')
grid on
% Input Control signal for Tank 1
subplot(2,1,2),
plot(t,Voltage(:,1)),
xlabel('Time [s]');
ylabel('Control input u1_k [V]')
title('Input control signal for Tank 1')
grid on

% Level results for Tank 2.
figure(2)
subplot(2,1,1)
plot(t, Reference(:,2),'r');hold on
plot(t,Output(:,2))
xlabel('Time [s]');
ylabel('Reference r2_kand output y2_k [cm]');
title('Tank 2 reference and output level')
legend('Reference r_k','Output y_k')
grid on
% Input Control signal for Tank 2
subplot(2,1,2),
plot(t,Voltage(:,2)),
xlabel('Time [s]');
ylabel('Control input u2_k [V]')
title('Input control signal for Tank 2')
grid on
```

## **Appendix 10.2:** four_tank_model.m

```
%==========================================================================
% MATLAB Script for:  Model of four tank process.
% Master's Thesis:    "MPC with Integral Action: Reducing the control horizon
%                     and model free MPC."
% Supervisior:        David Di Ruscio
% Author              Muhammad Mohsin
% File name:          four_tank_model.m
%--------------------------------------------------------------------------
% Telemark University College, Porsgrunn, Norway [23-04-2013]
%==========================================================================
function model=four_tank_model(h,u)
% Table 5.1, parameter values taken from the literature [Johansson].
% Cross-section area of tanks [cm^2]
A1=28;A3=28;
A2=32;A4=32;
```

```
% Cross-section area of the outlet holes[cm^2]
a1=0.071; a3=0.071;
a2=0.057; a4=0.057;
% Level in the tanks [cm]
h1=h(1);
h2=h(2);
h3=h(3);
h4=h(4);
% Control inputs [V]
u1=u(1); u2=u(2);
% Pump gain [V/cm]
kc=1;
% Acceleration of gravity [cm/s^2]
g=981;
% Model will show non-minimum phase characteristics
% Pump constants [cm^3/Vs]
k1=3.14;
k2=3.29;
% Flow constants
r1=0.43; r2=0.34;
% The four-tank model. [Equation (4.10)-(4.13)]
f(1)=( -a1*sqrt(2*g*h1) + a3*sqrt(2*g*h3) + r1*k1*u1 )/A1;
f(2)=( -a2*sqrt(2*g*h2) + a4*sqrt(2*g*h4) + r2*k2*u2 )/A2;
f(3)=( -a3*sqrt(2*g*h3) + (1-r2)*k2*u2 )/A3;
f(4)=( -a4*sqrt(2*g*h4) + (1-r1)*k1*u1 )/A4;
model=[f(1);f(2);f(3);f(4)];
```

## Appendix 11

MATLAB script for simulating the four-tank minimum and non-minimum phase process. The user can select either minimum or non-minimum phase process by typing “1” or “2” respectively. The supporting function files (four_tank_model.m and prbs1.m) are given in appendix 4.2 and 4.5.

**Appendix 11.1:** simulation_fordata_generation

```
% ==========================================================================
% MATLAB Script for:  Simulation of four tank process for data generation.
% Master's Thesis:   "MPC with Integral Action: Reducing the control horizon
%                     and model free MPC."
% Supervisior:        David Di Ruscio
% Author              Muhammad Mohsin
% Script File:        simulation_fordata_generation.m
% Function file:      four_tank_model.m
%                     prbs1.m by [David Di Ruscio]
% --------------------------------------------------------------------------
% Telemark University College, Porsgrunn, Norway [02-05-2013].
% ==========================================================================
clc;
close all;
clear all;
% Table 5.1, Parameter values taken from the literature [Johansson].
% Cross-section area of tanks [cm^2]
A1=28; A3=28;
A2=32; A4=32;
% Cross-section area of the outlet holes[cm^2]
a1=0.071; a3=0.071;
a2=0.057; a4=0.057;
% Pump gain [V/cm]
% kc=1;
kc=0.50;
% Acceleration of gravity [cm/s^2]
g=981;
% Time interval and sampling time [s]
ts=0.1; t=0; N=10000;
% System selection by the user
System_sel=input(' Type 1 for minimum phase system or\n Type 2 for non-
minimum phase system \n');
if  System_sel==1;
% Model will show minimum phase characteristics
% Table 5.3, Steady state level [cm] in the tanks after performing
simulations
hs1=12.3;
hs2=12.8;
hs3=1.63;
hs4=1.41;
% Parameter values from Table 5.2
% Pump constants [cm^3/Vs]
k1=3.33; k2=3.35;
% Flow constants
r1=0.7; r2=0.6;
% Initial control inputs [V]
```

```
u1=3; u2=3;
elseif System_sel==2;
% Model will show non-minimum phase characteristics
% Table 5.5, Steady state level [cm] in the tanks after performing
simulations
hs1=12.4;
hs2=13.2;
hs3=4.73;
hs4=4.99;
% Parameter values from Table 5.4
% Pump constants [cm^3/Vs]
k1=3.14; k2=3.29;
% Flow constants
r1=0.43; r2=0.34;
% Initial control inputs [V]
u1=3.15; u2=3.15;
end

% Time constants [section 4.1.2]
T1=A1*(sqrt(2*hs1/g))/a1;
T2=A2*(sqrt(2*hs2/g))/a2;
T3=A3*(sqrt(2*hs3/g))/a3;
T4=A4*(sqrt(2*hs4/g))/a4;

% Equation (4.15) and (4.16), Linearized state space matrices Ac, Bc and Dc
Ac=[-1/T1,0,A3/(A1*T3),0;0,-1/T2,0,A4/(A2*T4);0,0,-1/T3,0;0,0,0,-1/T4];
Bc=[r1*k1/A1,0;0,r2*k2/A2;0,(1-r2)*k2/A3;(1-r1)*k1/A4,0];
Dc=[kc,0,0,0;0,kc,0,0];
% Coverting model from continuous to discrete time
% Method 'zoh' assuming a zero order hold on the inputs.
[A,B,D]=c2dm(Ac,Bc,Dc,zeros(2,2),ts,'zoh');
h=[hs1;hs2;hs3;hs4];
h_old=h;
y_old=D*h_old;
u=[u1;u2];
u_old=u;
% For simulation initialization
f=zeros(4,1);
T=zeros(N,1);
U=zeros(N,2);
X=zeros(N,4);

% Make input signal for model development.
input1=prbs1(N,250,800);
input2=prbs1(N,350,850);

% Make input signal for validation data.
% input1=prbs1(N,2000,3500);
% input2=prbs1(N,2100,3600);

U_input=[u1+input1, u2+input2];
% Initial level in the tanks is zero.
  h=h*0;
% Simulating close loop system
for i=1:N
   y=D*h;
   u=U_input(i,:)';
  % Storing for plotting purposes
   U(i,:)=u';
   X(i,:)=h';
   T(i)=t;
   f=four_tank_model(h,u,System_sel);
   h=h+ts*f;
   % Time update.
   t=t+ts;
end

% Save data for system identification algorithm
if System_sel==1
 Data=[X(:,1:2),U];
 save('data_minimum.txt','Data','-ASCII')
 % for saving validation data
 % save('valdata_minimum.txt','Data','-ASCII')
elseif System_sel==2
Data=[X(:,1:2),U];
save('data_nonminimum.txt','Data','-ASCII')
% for saving validation data
% save('valdata_nonminimum.txt','Data','-ASCII')
end
```

## Appendix 12

MATLAB script for plotting the simulation data of four-tank minimum phase process.

**Appendix 12.1:** data_plot.m

```
%=========================================================================
% MATLAB Script for:  Plotting the collected simulated data of the four-tank
                      Minimum phase process.
% Master's Thesis:    "MPC with Integral Action: Reducing the control horizon
%                     and model free MPC."
% Supervisior:        David Di Ruscio
% Author              Muhammad Mohsin
% Script file:        data_plot.m
% -------------------------------------------------------------------------
% Telemark University College, Porsgrunn, Norway [02-05-2013]
%=========================================================================
clc;
clear all;
close all;
load data_minimum.txt
% The  first two columns in file  represent  the output level in tanks 1 and
2
% Next two columns are the corresponding voltage to the pumps 1 and 2
Y=[data_minimum(:,1)  data_minimum(:,2)];
U=[data_minimum(:,3)  data_minimum(:,4)];

% Plotting process data
figure(1)
subplot(2,1,1)
plot(Y(:,1))
% xlabel('Number of samples')
ylabel('Level y_1 [cm]')
title ('Level of Tank 1')
grid on
subplot(2,1,2)
plot(U(:,1))
xlabel('Number of samples')
ylabel('Control input u_1 [V]')
title ('Input voltage to pump 1')
grid on
figure(2)
subplot(2,1,1)
plot(Y(:,2))
% xlabel('Number of samples')
ylabel('Level y_2 [cm]')
title ('Level of Tank 2')
grid on
subplot(2,1,2)
plot(U(:,2))
xlabel('Number of samples')
ylabel('Control input u_2 [V]')
title ('Input voltage to pump 2')
grid on
```

# Appendix 13

The MATLAB script for identifying  model and Eigenvalues of the system matrix.

## **Appendix 13.1:** sysID_4tank.m

```
%==========================================================================
% MATLAB Script for:  System identification algorithm as dsr to identify the
linearized
%                     state space model for use in MPC method.
% Master's Thesis:    "MPC with Integral Action: Reducing the control
hortizon
%                     and model free MPC."
% Supervisior:        David Di Ruscio
% Author              Muhammad Mohsin
% Script file:        sysID_4tank.m
% ------------------------------------------------------------------------
% Telemark University College, Porsgrunn, Norway [02-05-2013]
%==========================================================================
clear all;
clc;
close all;
% Load the simulated process data
load data_minimum.txt
n=size(data_minimum);
i=1;
t=i:n;
% The  first two columns in file  represent  the output level in tanks 1 and
2
% Next two columns are the corresponding voltage to the pumps 1 and 2
Y=[data_minimum(t,1) data_minimum(t,2)];
U=[data_minimum(t,3) data_minimum(t,4)];

% System order
L=4;
% Number of samples
N=10000;
% Trending the data
U=[U(i:N,1)-3.0   U(i:N,2)-3.0 ];
Y=[Y(i:N,1)-12.3  Y(i:N,2)-12.8];
% Deterministic and Stochastic system identification and Realization
[A,B,D,E,CF,F,x0]=dsr(Y,U,L,0);
% converting discrete time to continuous time model
ts=0.1;
[Ac,Bc,Dc,Ec]=d2cm(A,B,D,E,ts,'zoh');
display('System Matrix Ac of continuous time model ')
Ac
% Eigenvalues of system matrix Ac
Eigen_values_for_system_matrix_Ac=eig(Ac)
```

## Appendix 14

MATLAB file for validating the identified model of four-tank minimum phase process.

**Appendix 14.1:** validation_of_minimum.m

```
%======================================================================
% MATLAB Script for:  Validating the identified model of four-tank minimum
%                     phase process
% Master's Thesis:    "MPC with Integral Action: Reducing the control horizon
%                     and model free MPC."
% Supervisior:        David Di Ruscio
% Author              Muhammad Mohsin
% Script file:        validation_of_minimum.m
% --------------------------------------------------------------------
% Telemark University College, Porsgrunn, Norway [02-05-2013]
%======================================================================
clc;
clear all;
close all;

% Load the simulated process data
load data_minimum.txt
data=data_minimum(1:end,:);
n=size(data);
i=1;
t=i:n;
Uid=[data(t,3) data(t,4)];
Yid=[data(t,1) data(t,2)];
% System order
L=4;

% Trending the data
Uid=[Uid(:,1)-3.0    Uid(:,2)-3.0 ];
Yid=[Yid(:,1)-12.3   Yid(:,2)-12.8];
% Deterministic and Stochastic system identification and Realization
[A,B,D,E,CF,F,x0]=dsr(Yid,Uid,L,0)

% validating the model
% Load the validation data
load data_valid.txt
data_val=data_valid(1:end,:);
m=size(data_val);
j=1;
k=j:m;
Uval=[data_val(k,3) data_val(k,4)];
Yval=[data_val(k,1) data_val(k,2)];

% Trending the data
Um=mean([data_val(k,3) data_val(k,4)]);
Ym=mean([data_val(k,1) data_val(k,2)]);
Uval=[Uval(:,1)-Um(1)      Uval(:,2)-Um(2)];
Yval=[Yval(:,1)-Ym(1)      Yval(:,2)-Ym(2)];

% Simulation of discrete-time linear systems
Y_sim=dsrsim(A,B,D,E,Uval,x0);
```

```
figure(2),
subplot(2,1,1)
plot([Yval(:,1) Y_sim(:,1)])
legend('Model output','Process output')
title('Output for tank_1');
ylabel('Level [cm]');grid on

subplot(2,1,2)
plot([Yval(:,2) Y_sim(:,2)])
legend('Model output','Process output')
title('Output for tank_2')
xlabel('Samples'); ylabel('Level [cm]');grid on
```

## Appendix 15

MATLAB m-file for implementing the MPC with integral action in identified model of four-tank minimum phase process. The main script is given in appendix 15.1 and supporting function files (eobsv.m, q2qt.m, prbs1.m, scmat.m, ss2h.m) are attached in appendix 4.3 to 4.7.

**Appendix 15.1:**mpc_syid.m

```
%==========================================================================
% MATLAB Script for:  Implementation of MPC with integral action in
                      linearized state space model developed by the system
                      identification algorithm as dsr.
% Master's Thesis:    "MPC with Integral Action: Reducing the control horizon
%                     and model free MPC."
% Supervisior:        David Di Ruscio
% Author:             Muhammad Mohsin
% Script file:        mpc_syid.m
% Function files:     data_minimum.txt
%                     eobsv.m, prbs1.m, q2qt.m, scmat.m, ss2h.m by [David Di
Ruscio]
% Reference           http://www2.hit.no/tf/fag/sce4106/ovinger/ovinger.html
% ------------------------------------------------------------------------
% Telemark University College, Porsgrunn, Norway [06-05-2013]
%==========================================================================
clc;
clear all;
close all;
% Load the simulated process data
load data_minimum.txt
data=data_minimum(1:end,:);
nn=size(data);
i1=1;
t=i1:nn;
Uid=[data(t,3) data(t,4)];
Yid=[data(t,1) data(t,2)];
% System order
LL=4;
% Trending the data
Uid=[Uid(:,1)-3.0    Uid(:,2)-3.0 ];
Yid=[Yid(:,1)-12.3   Yid(:,2)-12.8];
% % Deterministic and Stochastic system identification and Realization
[A,B,D,E,CF,F,x0]=dsr(Yid,Uid,LL,0);
% Making augmented state space model.
l=size(A,1);
m=size(B,2);
n=size(D,1);
% Matrics At, Bt and Dt
At=[A zeros(l,n); D eye(n,n)];
Bt=[B;zeros(n,m)];
Dt=[D eye(n,n)];
% Model predictive control algorithm,
% Prediction horizon
L=15;
% Simulation horizon
N=515;
```

```
% Weighting matrices
q=100;
r=0.1;
Q=[q 0; 0 q];
R=[r 0; 0 r];
% Observability matrix OL and Toeplitz matrix HdL calculation by ss2h
function.[David Di Ruscio]
[HdL,OL,OLB]=ss2h(At,Bt,Dt,zeros(n,m),L,0);
FL=[OLB HdL];
Qt=q2qt(Q,L);
Rt=q2qt(R,L);
% Make matrices S and c in the relationship, u(k,L) = S du(k,L) + c u(k-
1).[David Di Ruscio]
[S,c] = scmat(m,L);
% H as defined in equation (3.50)
H=FL'*Qt*FL+Rt;
% Defining inital values for simulation
Is=[11.5;12.5];
Hd=D*inv(eye(4)-A)*B;
u_ss=inv(Hd)*Is;
I=eye(4);
x_ss =inv(I-A)*B*u_ss;
% Assigning variables
u=u_ss;
h=x_ss;
u_old=u; h_old=h; y_old=D*h;
% Make the references by using the function prbs1.m[David Di Ruscio]
rand('seed',0),randn('seed',0);
step=150;
ref=[11.5*ones(N,1)+0.1*prbs1(N,step,step),12.5*ones(N,1)+0.1*prbs1(N,step,st
ep)];

% Used in Input amplitude constraints implementation.
umin=0;
umax=5;
% Making for variables storage
r1L=zeros(15,1);
U=zeros(N-L,2);
Y=zeros(N-L,2);

for k=1:N-L
y=D*h; % output equation
% Make the extended reference vector, r_(k,L) .[David Di Ruscio]
rf =ref(k+1:k+L,:);
r1L=rf(1,:)';
for i=2:L
r1L=[r1L;rf(i,:)'];
end
% Computing MPC control
xk=[h-h_old;y_old];
pL=OL*At*xk;
% f as defined in equation (3.50)
f=FL'*Qt*(pL-r1L);
% For constrained MPC [Equation 2.8]
a=[S;-S];
b=[umax*ones(L*m,1)-c*u_old;-umin*ones(L*m,1)+c*u_old];
```

```
% quadprog function for input amplitude constraints. [Equation 2.29]
duf=quadprog(H,f,a,b);
u=u+duf(1:m);
u_old=u;
% For plotting purpose store the variables.
U(k,:)=u';
Y(k,:)=y';
% Feed control to process.
h_old=h;
y_old=y;
h=A*h+B*u;
end
% Plotted Results
t=1:N-L;
% Level results for Tank 1.
figure(1)
subplot(3,1,[1,2])
plot(t, ref(1:N-L,1),'r');hold on
plot(t,Y(:,1))
ylabel('Level [cm]');
title('Tank 1 reference and output level')
legend('Reference r_k','Output y_k')
grid on
% Input Control signal for Tank 1
subplot(3,1,3),
plot(t,U(:,1)),
xlabel('Time [s]');
ylabel('Voltage [V]')
title('Input control signal for Tank 1')
grid on

% Level results for Tank 2.
figure(2)
subplot(3,1,[1,2])
plot(t, ref(1:N-L,2),'r');hold on
plot (t,Y(:,2))
ylabel('Level [cm]');
title('Tank 2 reference and output level')
legend('Reference r_k','Output y_k')
grid on
% Input Control signal for Tank 2
subplot(3,1,3),
plot(t,U(:,2)),
xlabel('Time [s]');
ylabel('Voltage [V]')
title('Input control signal for Tank 2')
grid on
```

## Appendix 16

MATLAB script for plotting the simulation data of four-tank non-minimum phase process.

**Appendix 16.1:** data_plot.m

```
%===========================================================================
% MATLAB Script for:  Ploting the collected simulated data of four-tank non-
                      minimum phase process.
% Master's Thesis:    "MPC with Integral Action: Reducing the control horizon
%                     and model free MPC."
% Supervisior:        David Di Ruscio
% Author              Muhammad Mohsin
% Script file:        data_plot.m
% -------------------------------------------------------------------------
% Telemark University College, Porsgrunn, Norway [02-05-2013]
%===========================================================================
clc;
clear all;
close all;
load data_nonminimum.txt
%  The   first  two  columns  in  file   represent   the  output  level  in
tanks 1 and 2  next two columns are the corresponding voltage to the pumps 1
and 2.
Y=[data_nonminimum(:,1) data_nonminimum(:,2)];
U=[data_nonminimum(:,3) data_nonminimum(:,4)];
% Plotting simulated data
figure(1)
subplot(2,1,1)
plot(Y(:,1))
% xlabel('Number of samples')
ylabel('Level y_1 [cm]')
title ('Level of Tank 1')
grid on
subplot(2,1,2)
plot(U(:,1))
xlabel('Number of samples')
ylabel('Control input u_1 [V]')
title ('Input voltage to pump 1')
grid on
figure(2)
subplot(2,1,1)
plot(Y(:,2))
% xlabel('Number of samples')
ylabel('Level y_2 [cm]')
title ('Level of Tank 2')
grid on
subplot(2,1,2)
plot(U(:,2))
xlabel('Number of samples')
ylabel('Control input u_2 [V]')
title ('Input voltage to pump 2')
grid on
```

## Appendix 17

The MATLAB script for identifying  the model and Eigenvalues of the system matrix.

**Appendix 16.1:** sysID_4tank.m

```
%==========================================================================
% MATLAB Script for:  System identification algorithm as dsr to identify the
                      linearized state space model for use in MPC method.
% Master's Thesis:    "MPC with Integral Action: Reducing the control horizon
%                     and model free MPC."
% Supervisior:        David Di Ruscio
% Author              Muhammad Mohsin
% Script file:        sysID_4tank.m
% -------------------------------------------------------------------------
% Telemark University College, Porsgrunn, Norway [02-05-2013]
%==========================================================================
clear all;
clc;
close all;
% Load the simulated process data
load data_nonminimum.txt
n=size(data_nonminimum);
i=1;
t=i:n;
% The  first two columns in file  represent  the output level in tanks 1 & 2
% Next two columns are the corresponding voltage to the pumps 1 and 2
Y=[data_nonminimum(t,1) data_nonminimum(t,2)];
U=[data_nonminimum(t,3) data_nonminimum(t,4)];

% System order
L=4;
% Number of samples
N=10000;
% Trending the data
Uid=[U(i:N,1)-3.15   U(i:N,2)-3.15 ];
Yid=[Y(i:N,1)-12.4   Y(i:N,2)-13.2];
% Deterministic and Stochastic system identification and Realization
[A,B,D,E,CF,F,x0]=dsr(Yid,Uid,L,0)
% converting discrete time to continuous time model
ts=0.1;
[Ac,Bc,Dc,Ec]=d2cm(A,B,D,E,ts,'zoh');
display('System Matrix Ac of continnuous time model ')
Ac
% Eigenvalues of system matrix Ac
Eigen_values_for_system_matrix_Ac=eig(Ac)
```

## Appendix 18

MATLAB file for validating the identified model of four-tank non-minimum phase process.

**Appendix 18.1:** validation_of_nonminimum.m

```
%===========================================================================
% MATLAB Script for:  Validating the identified model of four-tank non-
                       minimum phase process
% Master's Thesis:    "MPC with Integral Action: Reducing the control horizon
%                     and model free MPC."
% Supervisior:        David Di Ruscio
% Author              Muhammad Mohsin
% Script file:        validation_of_nonminimum.m
% -------------------------------------------------------------------------
% Telemark University College, Porsgrunn, Norway [05-05-2013]
%===========================================================================
clc;
clear all;
close all;
% Load the simulated process data
load data_nonminimum.txt
data=data_nonminimum(1:end,:);
n=size(data);
i=1;
t=i:n;
Uid=[data(t,3) data(t,4)];
Yid=[data(t,1) data(t,2)];
% System order
L=4;
% Trending the data
Uid=[Uid(:,1)-3.15   Uid(:,2)-3.15 ];
Yid=[Yid(:,1)-12.4   Yid(:,2)-13.2];
% % Deterministic and Stochastic system identification and Realization
[A,B,D,E,CF,F,x0]=dsr(Yid,Uid,L,0);
% validating the model
% Load the validation data
load data_valid.txt
data_val=data_valid(1:end,:);
m=size(data_val);
j=1;
k=j:m;
Uval=[data_val(k,3) data_val(k,4)];
Yval=[data_val(k,1) data_val(k,2)];
% Trending the data
Um=mean([data_val(k,3) data_val(k,4)]);
Ym=mean([data_val(k,1) data_val(k,2)]);
Uval=[Uval(:,1)-Um(1)      Uval(:,2)-Um(2)];
Yval=[Yval(:,1)-Ym(1)      Yval(:,2)-Ym(2)];
% Simulation of discrete-time linear systems
Y_sim=dsrsim(A,B,D,E,Uval,x0);
figure(2)
subplot(2,1,1)
plot([Yval(:,1) Y_sim(:,1)])
legend('Model output','Process output')
title('Output for tank_1')
ylabel('Level [cm]')
```

```
grid on
subplot(2,1,2)
plot([Yval(:,2) Y_sim(:,2)])
legend('Model output','Process output')
title('Output for tank_2')
xlabel('Samples');
ylabel('Level [cm]')
grid on
```

## Appendix 19

MATLAB m-file for implementing the MPC with integral action in identified model of four-tank non-minimum phase process. The main script is given in appendix 19.1 and supporting function files (eobsv.m, q2qt.m, prbs1.m, scmat.m, ss2h.m) are attached in appendix 4.3 to 4.7.

**Appendix 19.1:** mpc_syid.m

```
%==========================================================================
% MATLAB Script for:  Implementation of MPC with integral action in
                      linearized state space model developed by the system
                      identification algorithm as dsr.
% Master's Thesis:    "MPC with Integral Action: Reducing the control horizon
%                     and model free MPC."
% Supervisior:        David Di Ruscio
% Author:             Muhammad Mohsin
% Script file:        mpc_syid.m
% Function files:     data_nonminimum.txt
%                     eobsv.m, prbs1.m, q2qt.m, scmat.m, ss2h.m by [David Di
Ruscio]
%                     http://www2.hit.no/tf/fag/sce4106/ovinger/ovinger.html
% -------------------------------------------------------------------------
% Telemark University College, Porsgrunn, Norway [06-05-2013]
%==========================================================================

clc;
clear all;
close all;
% Load the simulated process data
load data_nonminimum.txt
data=data_nonminimum(1:end,:);
n1=size(data);
i1=1;
t=i1:n1;
Uid=[data(t,3) data(t,4)];
Yid=[data(t,1) data(t,2)];
% System order
LL=4;
% % Deterministic and Stochastic system identification and Realization
[A,B,D,E,CF,F,x0]=dsr(Yid,Uid,LL,0);

% Making augmented state space model.
l=size(A,1);
m=size(B,2);
n=size(D,1);
% Matrics At, Bt and Dt
At=[A zeros(l,n); D eye(n,n)];
Bt=[B;zeros(n,m)];
Dt=[D eye(n,n)];
% Model predictive control algorithm,
% Prediction horizon
L=15;
% Simulation horizon
N=515;
% Weighting matrices
q=100;
```

```
r=0.1;
Q=[q 0; 0 q];
R=[r 0; 0 r];
% Observability matrix OL and Toeplitz matrix HdL calculation by ss2h
function.[David Di Ruscio]
[HdL,OL,OLB]=ss2h(At,Bt,Dt,zeros(n,m),L,0);
FL=[OLB HdL];
Qt=q2qt(Q,L);
Rt=q2qt(R,L);
% Make matrices S and c in the relationship, u(k,L) = S du(k,L) + c u(k-
1).[David Di Ruscio]
[S,c] = scmat(m,L);
% H as defined in equation (3.50)
H=FL'*Qt*FL+Rt;

% Defining inital values for simulation
Is=[11.5;12.5];
Hd=D*inv(eye(4)-A)*B;
u_ss=inv(Hd)*Is;
I=eye(4);
x_ss =inv(I-A)*B*u_ss;
% Assigning variables
u=u_ss;
h=x_ss;
u_old=u; h_old=h; y_old=D*h;
% Make the references by using the function prbs.[David Di Ruscio]
rand('seed',0),randn('seed',0);
step=150;
ref=[11.5*ones(N,1)+0.1*prbs1(N,step,step),12.5*ones(N,1)+0.1*prbs1(N,step,st
ep)];

% Used in Input amplitude constraint implementation.
umin=0;
umax=5;
% Making for variables storage
r1L=zeros(15,1);
U=zeros(N-L,2);
Y=zeros(N-L,2);

for k=1:N-L
y=D*h; % output equation
% Make the extended reference vector, r_(k,L) .[David Di Ruscio]
rf =ref(k+1:k+L,:);
r1L=rf(1,:)';
for i=2:L
r1L=[r1L;rf(i,:)'];
end
% Computing MPC control
xk=[h-h_old;y_old];
pL=OL*At*xk;
% f as defined in equation (3.50)
f=FL'*Qt*(pL-r1L);
% For constrained MPC [Equation 2.8]
a=[S;-S];
b=[umax*ones(L*m,1)-c*u_old;-umin*ones(L*m,1)+c*u_old];
```

```
% quadprog function for input amplitude constraints. [Equation 2.29]
duf=quadprog(H,f,a,b);
u=u+duf(1:m);
u_old=u;
% For plotting purpose store the variables.
U(k,:)=u';
Y(k,:)=y';
% Feed control to the process.
h_old=h;
y_old=y;
h=A*h+B*u;
end
% Plotted Results
t=1:N-L;
% Level results for Tank 1.
figure(1)
subplot(3,1,[1,2])
plot(t, ref(1:N-L,1),'r');hold on
plot(t,Y(:,1));

ylabel('Level [cm]');
title('Tank 1 reference and output level')
legend('Reference r_k','Output y_k')
grid on
% Input Control signal for Tank 1
subplot(3,1,3),
plot(t,U(:,1)),
xlabel('Time [s]');
ylabel('Voltage [V]')
title('Input control signal for Tank 1')
grid on

% Level results for Tank 2.
figure(2)
subplot(3,1,[1,2])
plot(t, ref(1:N-L,2),'r');hold on
plot(t,Y(:,2))
ylabel('Level [cm]');
title('Tank 2 reference and output level')
legend('Reference r_k','Output y_k')
grid on
% Input Control signal for Tank 2
subplot(3,1,3),
plot(t,U(:,2)),
xlabel('Time [s]');
ylabel('Voltage [V]')
title('Input control signal for Tank 2')
grid on
```