MPC strategies for Chemical Engineering Applications using the GAMs tool

Candidate: Olcay Dagdeviren

Supervisor: Prof Harvey Arellano-Garcia

The dissertation submitted in
fulfilment of the requirements for the degree of MSc in
Process System Engineering in
the Chemical and Process Engineering.

September 2018

Abstract

This work focuses on the line of developing MPC strategies for chemical engineering applications, using GAMs as an implementation tool. Two case studies have been selected for the analysis: the quadruple-tank process, and CSTR process.

Firstly a methodology for the selected quadruple-tank process was established to observe the operation of the central MPC in the GAMs program. This methodology aims at modelling the system with simple and understandable equations, and was based on work done by Alvarado et al (2011). To this aim, nonlinear equations are linearized around an operating point. In view of the linearized equations, the formulation of the system is defined as linear matrix equalities. The values in the work of Alvarado were used as boundary conditions and operating conditions. The objective function consists of the contribution of the inputs, the outputs, the steady states values, and the set point. They have been formulated as a nonlinear programming (NLP) problem in GAMs.

Secondly, the application of MPC to a CSTR has been adapted to the GAMs program with NLP. The methodology developed for this purpose is designed in consideration of the work of Limon et al (2008). The nonlinear process model has been linearized by choosing the fastest responding operating condition in the article. Similar to the quadruple-tank process, equations are defined as linear matrix equalities. Unlike the quadruple-tank process, the objective function depends on the input, output and setpoint values.

Finally, a new model has been created using CSTRs instead of the tanks in the quadruple-tank process. The resulting model integrates the above-mentioned processes. The new linear matrix equalities were created by combining existing matrices in quadruple-tank and CSTR processes. Boundary conditions and operating conditions have been selected from the articles like the other processes. However, the obtained equations and matrices could not be written in the GAMs program.

This study focuses on the centralised MPC approach only. In the future, oher MPC's strategies (the decentralized MPC and the cooperative MPC) will be considered. In addition, work will be done to improve the simulation time.

Table of Contents

Acknowledgments

Firstly, I would like to thank my supervisor Professor Harvey Arellano-Garcia and his assistant Bogdan Dorneanu for their help, their guidance, and their advice. I am grateful for the support of my father, who died before living his dream, and mother. I also would like to thank my family, friends, and fiancée for their support and their patience in this process. Finally, I would like to say that I owe much to Mustafa Kemal Ataturk, the head teacher who says, "The real mentor in life is the science." I could not do this without them.

Notation

The symbols and terms used throughout the thesis are listed below.

a_i	The cross-sectional area of the outlet hole in tank i (m^2)
γ	Valve constant
A, B, C	Discrete-time matrices
A_c, B_c, C_c	Continuous-time matrices for Quadruple-tank process
A_r, B_r, C_r	Continuous-time matrices for CSTR process
C_f	Input concentration (mol/l)
C	Concentration (for CSTR problems) (mol/l)
D	Specific matrix for CSTR process
E	Activation energy (j/mol)
g	Gravity force (9.81 m/s^2)
h_i^0	Linearization level of tank i (m)
h_i	The liquid level of the tank i (m)
$h_{i,max}$	The maximum level of tank i (m)
i	Index ($i = 1, 2, 3, 4$)
J	The performance index
k	Discrete-time ($second$)
$k+1$	Next sampling time ($second$)
K	Terminal control gain
K_0	Constants (min^{-1})
K_1	Constants
K_2	Constants (l^{-1})
K_3	Constants (l/min)
k_i	Pump constant
M_s	Matrix consisted of steady state (xr) and input (ur) values
N	Chosen prediction horizon
P, Q, R, T	Matrices for the objective function
q_a	Flowrates of the pump A (m^3/h)
${q_a}^0$	Linearization flow of pump A (m^3/h)
q_b	Flowrates of the pump B (m^3/h)
${q_b}^0$	Linearization flow of pump B (m^3/h)

q_c	The flow rate of coolant (m^3/h)
${q_c}^0$	The operating point of cooling flow (m^3/h)
q_f	The flow rate of product (m^3/h)
q_{in}	Volumetric flow rate into the tank (m^3/h)
q_{out}	Volumetric flow rate out of the tank (m^3/h)
$q_{a,max}$,	Maximum flow of qa (m^3/h)
$q_{b,max}$	Maximum flow of qb (m^3/h)
R	İdeal gas constant ($j/mol\ K$)
r_A	Reaction rate
r	Parameter vector
s	Setpoint values
S	The cross-sectional area of the whole tanks (m^2)
T	Reactor temperature (for CSTR) (K)
T_c	Coolant temperature (K)
T_f	Inlet temperature (K)
u	Input variable of the system
u_r	Input variables as decision variables (defined by variable r)
V	Volume of the CSTR (m^3)
V_N	Objective function
x	The steady-state variable of the system
x_r	Steady-state variables as decision variables (defined by variable r)
y	The output variable of the system

Abbreviations

The abbreviations used throughout the thesis are listed below.

CSTR	Continuous Stirred Tank Reactor
DMC	Dynamic Matrix Control
DMPC	Distributed Model Predictive Control
EHAC	Extended Horizon Adaptive Control
EPSAC	Extended Prediction Self-Adaptive Control
GAMs	General Algebraic Modelling System
LP	Linear Programming
LQR	Linear Quadratic Regulator
LRPC	Long-Range Predictive Control
LRQP	Long-Range Quadratic Programming
MAC	Model Algorithmic Control
Matlab	Matrix Laboratory
MIMO	Multiple Input and Multiple Output
MPC	Model Predictive Control
MV	Manipulated Variables
NLP	Nonlinear Programming
PID	Proportional Integral Differential
QP	Quadratic Programming
QDMC	Quadratic Dynamic Matrix Control
QTP	Quadruple-Tank Process
RGA	Relative Gain Array
SISO	Single Input and Single Output

List of Figures

List of Tables

1 INTRODUCTION

1.1 Introduction

Processes are the steps that are starting from raw materials to obtain products by using some chemical and physical operations (Seborg, et al., 2011). The process should be controlled to work under the desired conditions, increase product quality, and ensure it runs safely and efficiently. The basic controllable variables in the process are temperature, pressure, and compositions of the feed and products. The desired reference value of the controlled variable is the set point, and the controlled variables can be set to be kept at or near their set points. Also, the controlled variables are affected by the manipulated variables. The disturbance variables are depending on the operating conditions of the process (Seborg, et al., 2011 cited by Dagdeviren, 2018).

The process control systems can be developed using generally two approaches: traditional and model-based. The control strategy and the control system in the traditional method are chosen depending on experience and knowledge of the process. The chosen control system is adapted to the process after the control parameters are set. The model-based control strategy is one of the current subjects of control engineering (Haber, et al., 2012) and will be discussed in more detail in the following section (Dagdeviren, 2018).

Model predictive control is the most common control technology in the multivariable process industry. MPC refers to a family of control algorithms that use an explicit dynamic model of motion to solve the optimal control movements by predicting future behaviour and reducing the objective function based on an output estimate. Optimization is a component of the MPC, the objective function consists of various parameters such as transfer function, state space models and a time-dependent step and impulse responses. In addition, linear and non-linear programming approaches are used in MPC. The control input at the first control interval is applied to the process and the calculation procedures are repeated at the next interval to receive feedback from the process. Moreover, time delay and constraints can be rearranged in the MPC (Al-Gherwi, 2010 cited by Dagdeviren, 2018).

There are three basic components of an MPC design:

1) Prediction of the future change of an open-loop system. The productivity of an MPC is related to the correctness of the model.

2) There is a performance index on a limited horizon. This index is minimized due to certain constraints and sampling time.

3) There is a horizon scheme that provides feedback on the control law to remove modelling mistakes (Christofidesa, et al., 2013 cited by Dagdeviren, 2018).

Since the prediction is based on a model, the latter is the base stone of MPC and therefore the type of MPC algorithm to be used depends on the type of model chosen. MPC can be developed using different methods, and the centralized MPC and the decentralized MPC are two of the most common methods of the MPC. Centrality in these methods refers to the fact that there is a central part of the system to collect all information and spread them (Dagdeviren, 2018).

MPC is very popular as an advanced control technique in the chemical industry. There are many MPC devices used in the academy and in the industry, predicting future process behaviour using formulas to optimize the results. The MPC optimization problem is formulated so that the solution can be obtained in the shortest time possible. Therefore, problems are either written into linear programming (LP) or quadratic programming (QP) formulations. In the LP formulation, both the objective function and the constraints are linear, and in the QP formulation, the boundaries must be linear although the objective function is quadratic (Hovd, 2004).

The chemical industry is working with many sub-process and they interact by transferring or exchanging energy and material. These interactions are depending on the process dynamics. Therefore, the different MPC strategies had been needed to control multiprocessing plants (Al-Gherwi, 2010). To consider the subsystem interactions, two different distributed MPC strategies are encountered in the literature. The first strategy is based on the Nash equilibrium. According to Nash equilibrium, if many controllers take decisions at the same time and if the results are depending on the decision of others, what will happen can be predicted. Rather than analyse the decision one by one, it asks the decisions of others and what to do (Wikipedia, 2018). All MPCs share their estimates and exchange solutions with each other and can

coordinate the controllers. Moreover, they are resolved with local objective functions instead of a general objective function. Other distributed MPC strategy is known as the feasible cooperation-based MPC (Venkat, 2006) or MPC based on neighbourhood optimization (Zhang & Li, 2007). The main objective of this strategy is that even if the optimal result of the control performance cannot be obtained, the results meet operational objectives (Venkat, 2006).

The MPC uses the explicit model to obtain the control signal. Moreover, it can be operated without worker intervention for a long time. (Holkar & Waghmare, 2010)

In the mentioned strategies, there should be no uncertainty in the process models to achieve near-optimal or optimal performance in the closed-loop system. Generally, nonlinear models are linearized in MPC applications in industry, but these models are not accurate completely. (Al-Gherwi, 2010) To provide more accurate results, nonlinear models should be used in MPC. However, to solve nonlinear equations, and to show stability and coordination of them is not an easy task. To increase the accuracy of the MPC performances, distributed MPC strategies should be preferred.

1.2 Objective

The objective of this work is to implement a theoretical description of MPC in GAMs. The well-known nonlinear four-tank process is used for this purpose.

The key objectives of this research are:

- to implement a linear MPC algorithm for the four-tank process and investigate if the computational cost, time and complexity of the optimization problem can be reduced,
- to conduct performance assessments of existing MPC strategies for the four-tank benchmark process application,
- to select the best MPC strategy and best control structure,
- to analyse the application in GAMs of MPC strategies (Dagdeviren, 2018).

1.3 Outline

This thesis is arranged in different chapters according to the tasks. This section informs briefly about the content and chapters of the thesis.

In Chapter 1, the thesis is outlined. The background of the Model Predictive Control (MPC) and Benchmark process are given. The purpose of the dissertation, the steps to be followed and the structure of the thesis are explained.

In Chapter 2, the historical sub-structure of process control is given, and the period of development of MPC is explained. Then the introduction of MPC strategies is given. Finally, the theory, the structure and the working principle of MPC are explained and then the optimization elements are summarised.

In Chapter 3, the four-tank benchmark process and CSTR process are defined, and its mathematical formulation, which is accepted by many researchers, is also shown.

In Chapter 4, the methodology of the quadruple-tank process, CSTR process, and the quadruple CSTR process following in this work are shown. In addition, the steps followed are explained in detail.

In Chapter 5, the obtained results of the project are presented and discussed.

In Chapter 6, the conclusions of the project and targeted future work are shown.

2 MODEL PREDICTIVE CONTROL

In this chapter, the basic information about the MPC and the brief history of work on the MPC is presented. Firstly, the historical development of MPC is given. Then MPC theories and the structure of the controller are explained. The cost functions and constraints for these theories are summarized. In addition, information has been provided on the working principle of the MPC.

2.1 Literature Review

The concept of MPC has a long history. Process control is used in industrial systems to ensure consistency and safety in production and to control processes as economics. The development of the modern control concept dates back to the early 1960s (Ruchika, 2013). The first works are the designed linear quadratic regulators (LQR) that aim to minimize an unconstrained quadratic objective function of states and inputs. Zadeh and Whalen in 1962 accepted the connections between the minimum time-optimal control problem and Linear Programming (Zadeh & Whalen, 1962). Propoi (1963) suggested the moving horizon approach that is central to MPC algorithms, which is known as "Open Loop Optimal Feedback" (Propoi, 1963). In 1978's, the MPC has become more popular in the Chemical Process Industries (Garica, et al., 1989). The engineers working at Shell Oil developed the MPC technology for their company and first implemented at the beginning of the 1970's (Holkar & Waghmare, 2010). According to Garcia et al., Shell Oil has also used MPC technology for many systems like a fluid catalytic cracking unit and a highly non-linear batch reactor (1989).

The linear impulse response model and linear step response models were used by Richalet et al (1978) and by Cutler and Ramaker (1979), respectively. Even though both took into consideration an unconstrained quadratic function, a heuristic iterative algorithm in the first formulation and the solution of a least-squares problem in the second formulation were used to obtain the optimal inputs. In the 1970s, the articles about the successful applications of the MPC such as Model Algorithmic Control (MAC) and Dynamic Matrix Control (DMC) were written.

The constrained algorithm is used to specify the use of a quadratic objective function and is Quadratic Dynamic Matrix Control (QDMC) done by Cutler et al (1983). The convex of the resulting problem for QDMC means that the constraints are linear.

When creating a high-performance MPC model, the required number of coefficients, which are related to settling time, for the step and impulse response models are overabundant. Multi-variable processes require a large number of coefficients, so optimization calculations are very intensive. This situation can be the limitation for Dynamic Matrix Control and Quadratic Dynamic Matrix Control. The use of the state space models to overcome this limitation was proposed. These improved linear state-space models can be used to be simply numerical solutions and test the system's controllability, observability, and stability. A DMC algorithm based on the step response was shown by Li. et al. (1989). In the work done by Prett and Garcia (1988), the step response model with a general discrete state-space model was used to remove the effect of errors caused by using step coefficients. To control stable and unstable systems, state-space MPC models are improved by Muske and Rawlings (1993).

According to the paper by Matsko, the MPC was implemented in the pulp and paper industries (1985 cited in Garica, et al., 1989). Increasing application areas of MPC lead to the development of new software.

MPC was applied in a simple mixing tank and a heat exchanger (Arkun et al., 1986 cited in Garica, et al., 1989) and in a coupled distillation column system that separates a ternary mixture (Levien and Morari, 1987 cited in Garica, et al., 1989).

The studies, which were done by De Keyser et al. (1988), focus on self-adaptive long-range predictive control (LRPC) methods that are based on robustness with respect to un-modelled dynamics, parameter variances, process noise and changing dead-time (Holkar & Waghmare, 2010).

The second branch of MPC has been done that is independent of the first. The second set of MPC approaches aims to be adaptable to control and to develop new strategies for mono-variable processes, which are formulated using input-output models. The predictive controller developed by Peterka (1984), Extended Horizon Adaptive Control (EHAC) of Ydstie (1984), the Extended Prediction Self-Adaptive Control (EPSAC) algorithm developed by DeKeyser et al (1982), and generalized predictive control (GPC) algorithm is written by Clarke et al (1987) are in the second branch of MPC (Garica, et al., 1989). The EHAC method tries to keep the future output value, which is calculated with the Diophantine equation $(Ax + By = C)$, close to the reference at a

particular time after the processing delay and is suitable for use in different strategies. The EPSAC method uses a suboptimal predictor instead of using a solution of a Diophantine equation, and suggests a constant control signal starting from the present (Garica, et al., 1989) (Camacho & Bordons, 2007).

The quadratic dynamic matrix control (QDMC) is part of the second generation of MPC. QDMC consists of algorithms that provide systematic ways to use input and output constraints. The system is linear, and the QP has been used to solve it. The control and state constraints are also defined as linear inequalities. (Qin & Badgwell, 2003)

The studies by Garcia et. al (1989) include the relation between MPC and linear quadratic control, the effect on robustness, and application of MPC to nonlinear systems (Garica, et al., 1989). According to Clarke and Scattolini, constrained receding horizon predictive control is used to optimize a quadratic function over a costing horizon to stabilize general linear plants. However, the computation is more complex. Another way is to use finite-horizon methods, which are numerically highly sensitive (1991 cited in Holkar & Waghmare, 2010).

Some simple rules that are represented in terms of step response or impulse response to choose prediction horizon have been developed by R. Scattolini and S. Bittanti (1990). The rules are the basis of success for LRPC to assure stability in the closed-loop. (Holkar & Waghmare, 2010). Long-range (LR), long-range quadratic programming (LRQP) and quadratic programming (QP) methods were used with the MPC models by Sandoz et al in 2000. They aimed to consider constraints and their violations. According to their results, long range is trustable, effective and robust to use with MPC. To manage the input constraint, LRQP can be used but the linear quadratic method is effective for output constraint (Holkar & Waghmare, 2010).

Table 2-1 Chronological improvement of MPC strategies

MPC Algorithms	Year	Summary
Model Algorithmic Control (MAC)	1978	Finite impulse response model
Dynamic Matrix Control (DMC)	1980	Step response model Control calculation by using the least-squares method (without constraints) (Garcia & Morshedi, 1986)
Quadratic Dynamic Matrix Control (QDMC)	1984	Step response model Control calculation by using the least-squares method (without constraints) (Garcia & Morshedi, 1986)
IDCOM-M (the M means multi-input/ single output) (The solution software)	1988	Linear impulse response model Multi-objective function formulation (quadratic output objective followed by a quadratic input objective) (Qin & Badgwell, 2003)
Hierarchical Constraint Control (HIECON) (The solution software)	1990	(Qin & Badgwell, 2003)
Generalized Predictive Control (GPC)	1987	Transfer function model (Clarke, et al., 1987)
Nonlinear Model Predictive Control (NMPC)	1989	

2.1.1 The concept of MPC

As seen in the previous section, various MPC algorithms have been developed, with different ways of defining the mathematical formulation (equations, and cost functions desired to minimise.) Control actions used only in industries have recently been used in many areas of practice, such as medical studies and robots. The use of the MPC in these different areas leads to the development of new strategies (Camacho & Bordons, 2007).

2.1.1.1 MPC Elements

All the MPC strategies have basic elements such as prediction model, or objective function. Some elements are added depending on different algorithms (Camacho & Bordons, 2007). Specific mathematical formulations for each model allow the problem to be solved within a limited time by calculating the necessary manipulations in

response to disturbance. However, the important elements for acceptable models are defined as follows (Boom & Backx, 1999):

Process model: The models have been generally chosen as linear. When using nonlinear models are required, the linearization of the model is the best way to obtain a solution simply. Basically, the models have been used to predict the process signals for the specific horizon.

Performance index: The performance index is formulated based on the reference tracking error and the control actions.

Disturbance rejection: In a situation where there is a disturbance, a good performance is obtained by predicting the disturbance signals.

Constraints: The operating points are used to be limitations of equipment and cost. When the system is operating close to the limits, control, status and output signals will be subject to constraints.

Optimization: The optimization algorithm is used to calculate the control signals depending on the defined constraints.

Prediction Horizon: In predictive control, receding horizon principle is considered. The first control sample is used after the optimal control value is calculated. Then the horizon is shifted to the next value and the calculations are made again.

2.1.1.2 MPC Strategy

To calculate the optimum, a dynamic model of the process, a history of past control action, and a cost function are used by the Model Predictive Control (MPC). In general, what is the strategy of MPC, and which terms MPC depends on are explained below. The MPC strategy is represented in Figure 2-1.

For each interval, the process output values are calculated using the process model. The predicted output values are calculated based on past input and output values and future control signals.

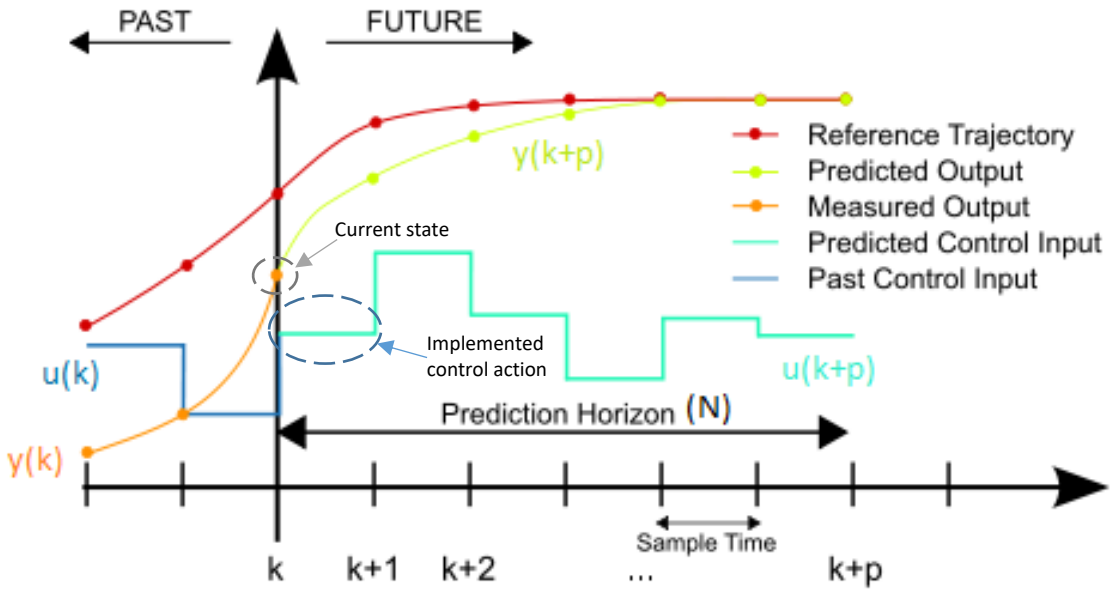


Figure 2-1 MPC Strategy

Future control signals are calculated by optimizing the set criteria. The goal here is to bring the process to the closest possible value. This set criterion is, in general, the difference between the recognized output and reference values and is a quadratic function. An iterative optimization model should be used when no explicit solution is found (Camacho & Bordons, 2007). MPC strategies are implemented like the following structure (Figure 2-2).

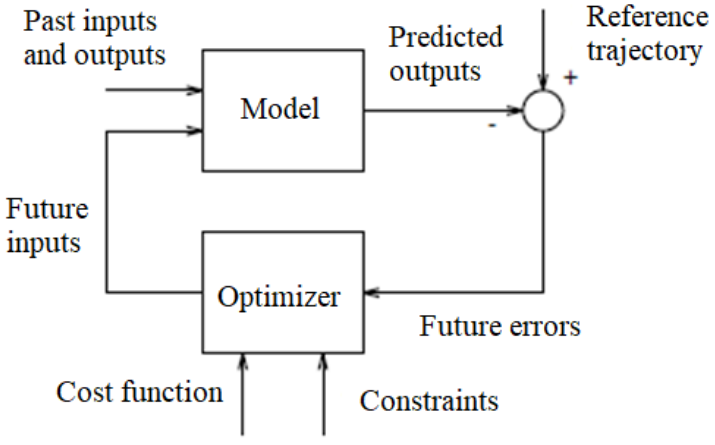


Figure 2-2 The structure of Model Prediction Control (Camacho & Bordons, 2007)

2.1.1.3 Advantages of MPC

When MPC strategies are compared with other control methods, MPC has a series of advantages. When conventional controllers make temporary decisions for error signals, the MPC predicts future error signals and gives an appropriate decision. The conventional controllers are proportional (P), integral (I), derivative (D), proportional-integral (PI), proportional-derivative (PD) and proportional-integral-derivative (PID) controllers. Considering the constraints, operating cost can be reduced to the minimum in MPC. Model predictive strategies are suitable for the multivariable case (for the great numbers of manipulated and controlled variables). The constraints can be defined for both manipulated and controlled variables. MPC can operate very close to the constraints. MPC can allow time delays, inverse responses, alteration of control objectives and sensor failure. To tolerate measurable disturbances, MPC provides feedforward control. It has certain basic principles that allow for future extensions (Camacho & Bordons, 2007) . The MPC is an easy closed-form controller when compared to an auto-tuned PID controller. Furthermore, the cost of solving the optimization problem is very low in the development of the MPC model. Process constraints in the MPC can be systematically considered by controllers. The applications of MPC are suitable for the linear, nonlinear systems in industries. When MPC is compared with other feedback controllers, it is based on the future predictions of the systems dynamics. (Kumar & Ahmad, 2012).

2.1.1.4 4. Industrial implementation

There are many industrial applications of MPC and the number of implementations is increasing each passing day. MPC was first applied to refineries. The application areas listed below are some of the most common areas where MPC is used:

- Superheater, steam generator, utility boiler,
- FCCU (Fluid Catalytic Cracking Unit),
- Pulp and paper industries.
- Hydrocracker reactors, Batch reactors,
- Distillation columns
- Polymer extruder, PVC plants
- Olefin plants

- The cement industry
- Robot arms
- Drying towers

2.2 Methods of Model Predictive Control

2.2.1 Centralized Model Predictive Control

As mentioned above, the sub-processes in the system are interacting with each other. Neglecting these interactions distract the system from the optimal result. However, a central coordinator that controls the process can be used to take these interactions to the account. The centralized MPC covers the achievement of optimal movements by minimizing a cost function involving the purpose of the system (Al-Gherwi, 2010). The schematic representation of the centralized MPC for a system with two subsystems is as in Figure 23 (Dagdeviren, 2018).

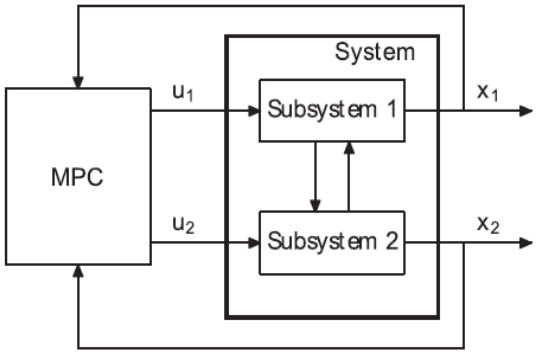


Figure 2-3 Diagram of Centralized MPC. A central coordinator that controls the process can add these interactions to the account.

The centralized MPC considers the interactions in the system, so it exhibits optimum performance. If a central MPC is used in a process with a large number of inputs and outputs, disadvantages may be observed, such as the model being sensitive to errors and having low flexibility in operational changes (Skogestad and Postlethwaite 2005 cited by Dagdeviren, 2018).

2.2.2 Decentralized Model Predictive Control

The decentralized MPC is developed for subsystems interacting with each other and subject to certain constraints. In the decentralized MPC, the problem of control is

divided into small sized local MPCs (Scattolini, 2009). Information sets, such as local state measurements, local decisions, and local estimates, are interrelated (Bemporad & Barcelli, 2010). In the decentralized MPC, there are several local control centres positioned to monitor the outputs and inputs. In a decentralized control system, there is no connection between local controls. The MPC structure shown in Figure 2-4where the system consists of two subsystems (S1, S2). The state, input and output variables of subsystems are (x1, u1, and y1) and (x2, u2, and y2), respectively. There is also the interaction between the states of subsystems (x1, x2) (Dagdeviren, 2018).

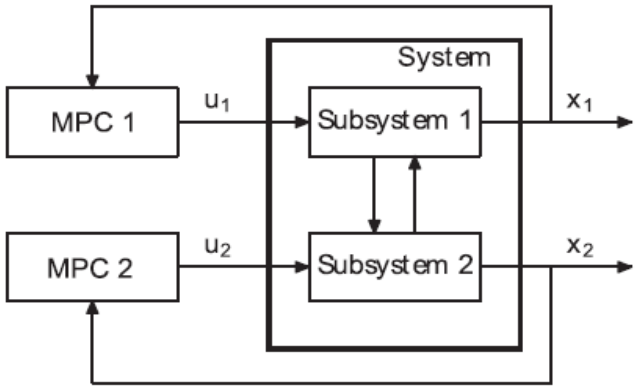


Figure 2-4 Diagram of Decentralized MPC. Each of the local supervisors checks the status of the subsystems and does not exchange information with each other.

In the decentralized MPC system, there are local cost functions and interactions can be ignored, either partially or completely. Therefore, the decentralized MPC has a weaker performance than the centralized MPC. In a decentralized MPC, it is assumed that the connections between the different subsystems are weak and can be considerably avoided as the controllers become unstable when these interactions are strong (Dagdeviren, 2018).

2.2.3 Distributed MPC based on a cooperative game

The controlling purpose of this controller is, on the one hand, to set the system to the specified set points, on the other hand, ensuring that a certain number of states and input constraints are met (Alvarado, et al., 2011). In the distributed schematic, there is a model for each subsystem and a controller with access to the state of the subsystem. Although the controllers do not know the dynamics of their neighbours, they can communicate with each other to understand the inputs applied to the system (Alvarado, et al., 2011). The strategy relies on negotiations between controllers based on the

global performance index. At each sampling time, delegates' present proposals to develop solutions based on local cost functions, state and model bases. From the first appropriate solution, optimal solutions and feedback gains are obtained of the two-time balancing controllers defined by the previous time step (Alvarado, et al., 2011 cited by Dagdeviren, 2018).

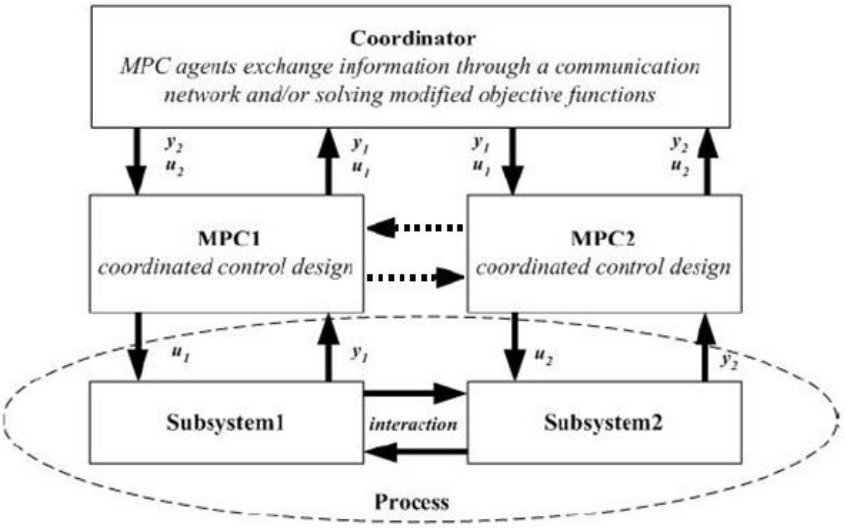


Figure 2-5 Coordinated distributed MPC configuration. Local controllers are responsible for a subsystem. They only arrive information of this subsystem. However, they can communicate with other local controllers.

There is no specific difference between the decentralized and distributed MPC, which is accepted by researchers. The decentralized MPC make independent decisions from the other controllers. The exchanges of information between controllers can be done before and after the decision process of controllers. Even if the decision-making process continues in the distributed MPC, the candidate decisions taken can be changed between the local controllers. This situation can be continuing until the local controllers agree on the same result (Bemporad & Barcelli, 2010).

2.3 Mathematical Models for MPC

In this study, the linear dynamic model is used in the development of the centralized MPC scheme, but the mathematical models of the decentralized and distributed MPC models will be discussed as well in the following Sections. Firstly, the following expressions are used to represent the state space model in continuous time.

$$\frac{dx}{dt} = A_c x + B_c u \quad (2.1)$$

$$y = C_c x \quad (2.2)$$

Where states $x \in R^{n_x}$, inputs $u \in R^{n_u}$, outputs $y \in R^{n_y}$, and $A_c \in R^{n_x \times n_x}$, $B_c \in R^{n_x \times n_u}$, $C_c \in R^{n_y \times n_x}$ are matrices for the state, input, and output, respectively (Alvarado, et al., 2011). Based on the continuous time model, a discrete time model can be obtained during a sampling period (Dagdeviren, 2018). Standard MPC formulation is shown following:

$$x(k+1) = Ax(k) + Bu(k) \quad (2.3)$$

$$y(k) = Cx(k) \quad (2.4)$$

Where A, B, and C are matrices for discrete time. These are calculated by using the Matlab program. The $k \in N^0$ represents the discrete time. The linear discrete-time model is used to design a centralized predictive controller.

A cost function is used for tracking of MPC. The function is based on the steady state (xr) and input (ur) variables defined by r variable, and the constraints (Alvarado, et al., 2011). The cost function is:

$$V_N(x, s, U, r) = \sum_{i=0}^{N-1} \|y(i) - r\|_Q^2 + \|u(i) - u_r\|_R^2 + \|x(N) - x_r\|_P^2 + \|r + h^0 - s\|_T^2 \quad (2.5)$$

where $Q, R, P,$ and T are matrices, U is a sequence of input values future, h^0 is the matrix obtained from operating values, s is the setpoint values, and $y(i)$ shows predicted output values of the system.

2.3.1 Centralized Model Predictive Control Strategies

The linearized predicted discrete-time-based centralized MPC must always meet constraints and monitor set point or reference sequence s (k). The MPC scheme is based on the use of modified cost functions and an extended terminal constraint (Alvarado, et al., 2011 cited by Dagdeviren, 2018).

$$minimize\ V_N \qquad \sum_{k=0}^{N-1}(x, s, U, r) \tag{2.6}$$

$$\begin{aligned} subject\ to \quad & x(k+1) = Ax(k) + Bu(k) \\ & y(k) = Cx(k) \\ & x(0) = x\,, \\ & (x, u) \in Z \\ & (x_r, u_r) = M_s r, \\ & U = \{u_r, \dots, u_{r+N-1}\} \end{aligned} \tag{2.7}$$

For each time-instant, x(r) is measured or estimated, optimal input sequence the prediction horizon (N) is found, and the first control action is implemented on the plant.

2.3.2 Decentralized Model Predictive Control Strategies

The second control technique is the decentralized predictive controller based on the decentralized model of the following systems (Alvarado, et al., 2011 cited by Dagdeviren, 2018).

$$x_{s1}(k) = A_{c1}\, x_{s1}(k) + B_{c1}^{(2)} u_{s1}(k), \tag{2.8}$$

$$y_{s1}(k) = C_{c1} x_{s1}(k), \tag{2.9}$$

And

$$x_{s2}(k) = A_{c2}\, x_{s2}(k) + B_{c2}^{(1)} u_{s2}(k), \tag{2.10}$$

$$y_{s2}(k) = C_{c2} x_{s2}(k). \tag{2.11}$$

The cost function for the subsystem j is

$$V_{N,j}(x_{sj}, s_j, U_j, r_j) = \sum_{i=0}^{N-1} \|y_{sj}(i) - r_j\|_{Q_j}^2 + \|u_{sj}(i) - u_{rj}\|_{R_j}^2 + \|x_{sj}(N) - x_{rj}\|_{P_j}^2 + \|r_j + h_j^0 - s_j\|_{T_j}^2$$

The optimization problem of MPC for each subsystem j is

$$minimize\ V_{N,j} \quad \sum_{k=0}^{N-1} (x_{sj}, s_j, U_j, r_j) \tag{2.12}$$

$$\begin{aligned} subject\ to \quad & x_{sj}(r+k+1) = A_j x_{sj}(r+k) + B_j u_j(r+k) \\ & y_j(r+k) = C_j x_{sj}(r+k) \\ & x_{sj}(0) = x_{sj}\ , \\ & (x_{sj}, u_j) \in Z \\ & \left(x_{rj}, u_{rj}\right) = M_{sj} r_j, \\ & U_{s,j} = \{u_{s,j}, \dots, u_{s,j+N-1}\} \end{aligned} \tag{2.13}$$

2.3.3 Distributed MPC based on a cooperative game strategy

This control scheme considers a distributed linear system consisting of subsystems coupled to the neighbouring subsystem via inputs. The following models are used to design the distributed controller (Alvarado, et al., 2011 cited by Dagdeviren, 2018).

$$x_{s1}(k) = A_{c1}\, x_{s1}(k) + B_{c1}^{(2)} u_{s1}(k) + B_{c1}^{(1)} v_{s1}(k), \tag{2.14}$$

$$y_{s1}(k) = C_{c1} x_{s1}(k), \tag{2.15}$$

And

$$x_{s2}(k) = A_{c2}\, x_{s2}(k) + B_{c2}^{(1)} u_{s2}(k) + B_{c2}^{(2)} v_{s2}(k), \tag{2.16}$$

$$y_{s2}(k) = C_{c2} x_{s2}(k) \tag{2.17}$$

3 THE FOUR TANK PROCESS and CSTR PROCESS

The Four Tank Benchmark Process and CSTR process are described in this chapter. The parameters of the processes are identified. The modelling of the non-linear MPC algorithm and the linearization of the model are given.

3.1 Background of the quadruple-tank process

In the past, there were one or more variables that had to be controlled in the processes used in the industry. Parallel to the developments in the industry, control mechanisms and plants have become more complex, and the number of variables increased. The manipulated variables can affect controlled variables. In the multiple inputs-multiple outputs systems, one input signal can affect more than one controlled variable. When MIMO is compared with the single input -single output system (SISO), it is the more complex system. Inputs and outputs should be matched in such a way that they can easily reach the desired control purpose. Since cross-coupling of input and output in MIMO systems poses a problem, the Quadruple-Tank Process (QTP) has been developed by Johansson et al. to develop the multivariable control theory (1996). The process, which is multivariable, has two inputs and two outputs. The process was built in HSN to achieve the same objective. Over the years, many linear and nonlinear control models have been designed to control the quadruple-tank process.

3.2 Description of the real system

The basic characteristic of the quadruple-tank process is that it can work in the right and left half planes and it shows the importance of the multivariate zeros. (Alvarado, et al., 2006) The quadruple-tank process is a system used to observe how different controllers work in a MIMO system. It is considered as two processes with the double tank.

The process consists of four tanks, which are interacting with each other, two pumps, and two valves. The multivariable process, originally designed for use in the control laboratory, is nonlinear.

Sensors measuring liquid levels are connected to the 1st and the 2nd tank. The inputs of the process are the voltages of the pumps ($u1, u2$) and the outputs are the voltages from the sensors ($y1, y2$). The four-tank process diagram is shown in Figure 3.1. The

feed of tank 1 and tank 2 are tank 3 and tank 4, respectively. Here, tank 1 and tank 2 are placed under tank 3 and tank 4 to use the gravity effect. In the bottom, there is a reservoir for storing the liquid outlet of tank 1 and tank 2.

The purpose of the pumps is to return from the reservoir container to the tanks. The valve opening is based on this operation. While pump 1 is feeding liquid from the reservoir to tank 1 and tank 4, pump 2 is delivering to tank 2 and tank 3. It is aimed to control the liquid levels of tank 1 and tank 2 in the system. That is, the controlled outputs are $h1$ and $h2$, while $\gamma 1$ and $\gamma 2$ are the valve parameters and show how the amount of fluid coming from the pump is divided between the upper and lower tanks. Differential equations (Eq 3.1) show the dynamics of the system (Prasad, 2016).

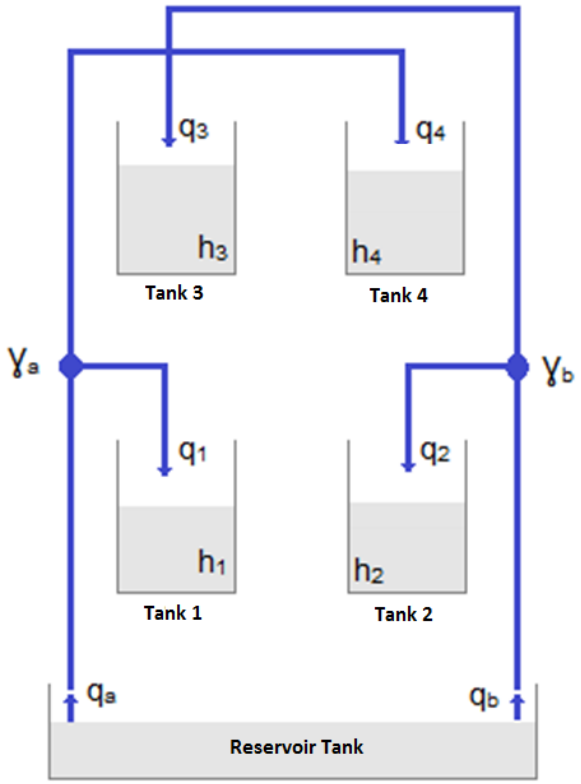


Figure 3-1 Quadruple-tank process diagram

The quadruple-tank process model is shown by the following:

$$\frac{dh_1}{dt} = \frac{-\alpha_1}{S}\sqrt{2gh_1} + \frac{\alpha_3}{S}\sqrt{2gh_3} + \frac{\gamma_a}{S}\,q_a$$
$$\frac{dh_2}{dt} = \frac{-\alpha_2}{S}\sqrt{2gh_2} + \frac{\alpha_4}{S}\sqrt{2gh_4} + \frac{\gamma_b}{S}\,q_b \qquad 3.1$$

$$\frac{dh_3}{dt} = \frac{-\alpha_3}{S}\sqrt{2gh_3} + \frac{(1-\gamma_b)}{S}q_b$$
$$\frac{dh_4}{dt} = \frac{-\alpha_4}{S}\sqrt{2gh_4} + \frac{(1-\gamma_a)}{S}q_a$$

Where γ_a, γ_b are the parameters of the 3 ways-valve, q_a, q_b are the flow rates of the pumps, α_i is the discharge constant of tank i, and h_i is the level of tank i. Also, S is the cross-sectional area of the tanks and g is gravity force.

After the linearization, the following discrete-time model ($Eq\ 3.3\ and\ Eq\ 3.4$) is defined depending on deviation variables ($Eq\ 3.2$):

$$x_i = h_i - h_i^0$$
$$u_1 = q_a - q_a^0 \qquad 3.2$$
$$u_2 = q_b - q_b^0$$

The discrete-time Model can be written as:

$$\frac{dx}{dt} = Ax - Bu \qquad 3.3$$

$$y = Cx \qquad 3.4$$

In the dynamic model, the liquid levels of the tanks are defined by assuming the following assumptions.

- The liquid levels of the tanks ($h1, h2, h3, h4$) are measured by means of a pressure sensor.
- *y* defines the output of the system depending on $x1\ and\ x2$.
- $u1\ and\ u2$ represent the stresses applied to the input terminals by the pumps ($1\ and\ 2$).

3.3 Description of the CSTR Process

The Continuous Stirred Tank Reactor (CSTR), which shows non-linear behaviour, is one of the processes commonly used in the industry. Reactions occurring in the reactor are endothermic or exothermic. Therefore, to provide the constant system temperature, the energy is given to the reactor or the energy is taken from the reactor. The CSTR operates in a generally stable condition and must be well mixed to ensure homogeneity.

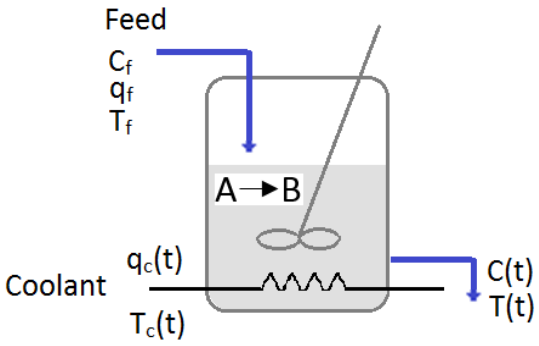


Figure 3-2 The continuous stirred-tank reactor diagram

The CSTR is modelled according to concentration, temperature, and reaction rate. $Figure\ 3-2$ represents the CSTR diagram. The mathematical modelling of the CSTR and methodology of the MPC for CSTR are presented in the following section.

The mass and energy balance of the CSTR is shown as follows.

$$\frac{q_f}{V}(C_f - C) - K_0 C e^{-\frac{E}{RT}} = \frac{dC}{dt} \qquad 3.5$$

$$\frac{dT}{dt} = \frac{q_f}{V}(T_f - T) + K_1 C e^{-\frac{E}{RT}} + K_2 q_c \left(1 - e^{-\frac{K_3}{q_c}}\right)(T_c - T) \qquad 3.6$$

The given mass and energy balances equations are nonlinear differential equations. These equations are linearized around operating points ($Table\ 4-2$) by using Taylor series with following $Eq\ 3.7$.

$$\begin{aligned} C &= C_0 \\ T &= T_0 \\ q_c &= q_{c_0} \end{aligned} \qquad 3.7$$

The linearization has occurred with the continuous-time matrices in next section.

4 METHODOLOGY

In this chapter, the calculation steps of the quadruple-tank process, the CSTR process, and the quadruple-CSTR process are shown. The mathematical formulations begin with mass and energy balances and continue with the steps necessary for optimization.

4.1 The Methodology of the Quadruple-Tank Process

The material balance is done for each tank.. Therefore, $Eq\ 4.2$ is obtained from the general mass balance formula ($Eq\ 4.1$).

$$Input - Output + Generation = Accummulation \quad 4.1$$

$$S\frac{dh_i}{dt} = -q_{i\,out} + q_{i\,in} \quad 4.2$$

Where S is the cross-sectional area for each tank, h is the liquid level, q is the flow rate of liquid entering and exiting the tank, "i" represents the number of the tank.

The inlet and outlet flow rates have been modelled using Bernoulli's Law. According to Bernoulli's law, the potential energy of the tank level should be equal to the kinetic energy of the liquid when there are no pressure changes ($Eq\ 4.3$).

$$mgh = \frac{1}{2}mv^2 \quad 4.3$$

$$v = \sqrt{2gh} \quad 4.4$$

$$q_{i\,out} = \alpha_i\, v = \alpha_i\sqrt{2gh_i} \quad 4.5$$

Where ai represents the cross-sectional area of the outlet hole of the tank, g is gravity force.

$$q_{i\,in} = \gamma k_i\ for\ i = 1{,}2$$
$$q_{i\,in} = (1-\gamma)k_i\ for\ i = 3{,}4 \quad 4.6$$

Where k_i denotes the pump constant but they are being assumed as "1". In this process, a nonlinear model of the system was created because there are inward and outward flows at the same time from all tanks.

$$S\frac{dh_1}{dt} = -q_{1out} + q_{3out} + q_{1in}$$
$$S\frac{dh_2}{dt} = -q_{2out} + q_{4out} + q_{2in}$$
$$S\frac{dh_3}{dt} = -q_{3out} + q_{3in} \qquad 4.7$$
$$S\frac{dh_4}{dt} = -q_{4out} + q_{4in}$$

The simplified model of the quadruple-tank process is based on for this model and differential equations are shown by the following.

$$\frac{dh_1}{dt} = \frac{-\alpha_1}{S}\sqrt{2gh_1} + \frac{\alpha_3}{S}\sqrt{2gh_3} + \frac{\gamma_a}{S}\,q_a$$
$$\frac{dh_2}{dt} = \frac{-\alpha_2}{S}\sqrt{2gh_2} + \frac{\alpha_4}{S}\sqrt{2gh_4} + \frac{\gamma_b}{S}\,q_b \qquad 4.8$$
$$\frac{dh_3}{dt} = \frac{-\alpha_3}{S}\sqrt{2gh_3} + \frac{(1-\gamma_b)}{S}\,q_b$$
$$\frac{dh_4}{dt} = \frac{-\alpha_4}{S}\sqrt{2gh_4} + \frac{(1-\gamma_a)}{S}\,q_a$$

Where $\boldsymbol{\gamma_a}, \boldsymbol{\gamma_b}$ are parameter of the 3 ways-valve (for a and b valves, respectively). In the design of controllers, both simple linearized models and nonlinear models can also be used for the analysis and design of the controller. For a linear model, the model is linearized around an operating point. The following equations are used for linearization. The given equations above are nonlinear and their solution is more difficult. The given equations are linearized by using Taylor series with following $Eq\ 4.9$.

$$x_i = h_i - h_i^0$$
$$u_1 = q_a - q_a^0 \qquad 4.9$$
$$u_2 = q_b - q_b^0$$

The above equations are written in $Eq\ 4.8$ and the following equations are obtained.

$$\frac{d(h_1-h_1^0)}{dt} \cong \frac{-\alpha_1}{S}\sqrt{2g}\frac{d\sqrt{h_1}}{dt}(h_1-h_1^0)+\frac{\alpha_3}{S}\sqrt{2g}\frac{d\sqrt{h_3}}{dt}(h_3-h_3^0)+\frac{\gamma_a}{S}(q_a-q_a^0)$$

$$\frac{d(h_2-h_2^0)}{dt} \cong \frac{-\alpha_2}{S}\sqrt{2g}\frac{d\sqrt{h_2}}{dt}(h_2-h_2^0)+\frac{\alpha_4}{S}\sqrt{2g}\frac{d\sqrt{h_4}}{dt}(h_4-h_4^0)+\frac{\gamma_b}{S}(q_b-q_b^0)$$

$$\frac{d(h_3-h_3^0)}{dt} = \frac{-\alpha_3}{S}\sqrt{2g}\frac{d\sqrt{h_3}}{dt}(h_3-h_3^0)+\frac{(1-\gamma_b)}{S}(q_b-q_b^0) \qquad 4.10$$

$$\frac{d(h_4-h_4^0)}{dt} = \frac{-\alpha_4}{S}\sqrt{2g}\frac{d\sqrt{h_4}}{dt}(h_4-h_4^0)+\frac{(1-\gamma_a)}{S}(q_a-q_a^0)$$

These equations ($Eq.4.11$) are arranged, and the forms below are obtained.

$$\frac{d(h_1-h_1^0)}{dt} \cong \frac{-\alpha_1}{S}\frac{\sqrt{g}}{\sqrt{2h_1^0}}(h_1-h_1^0)+\frac{\alpha_3}{S}\frac{\sqrt{g}}{\sqrt{2h_3^0}}(h_3-h_3^0)+\frac{\gamma_a}{S}(q_a-q_a^0)$$

$$\frac{d(h_2-h_2^0)}{dt} \cong \frac{-\alpha_2}{S}\frac{\sqrt{g}}{\sqrt{2h_2^0}}(h_2-h_2^0)+\frac{\alpha_4}{S}\frac{\sqrt{g}}{\sqrt{2h_4^0}}(h_4-h_4^0)+\frac{\gamma_b}{S}(q_b-q_b^0)$$

$$\frac{d(h_3-h_3^0)}{dt} = \frac{-\alpha_3}{S}\frac{\sqrt{g}}{\sqrt{2h_3^0}}(h_3-h_3^0)+\frac{(1-\gamma_b)}{S}(q_b-q_b^0) \qquad 4.11$$

$$\frac{d(h_4-h_4^0)}{dt} = \frac{-\alpha_4}{S}\frac{\sqrt{g}}{\sqrt{2h_4^0}}(h_4-h_4^0)+\frac{(1-\gamma_a)}{S}(q_a-q_a^0)$$

The following continuous-time linear model equations are obtained by writing $Eq.4.12$ in the above equations.

$$\tau_i = \frac{\alpha_i}{S}\frac{\sqrt{g}}{\sqrt{2h_i^0}} \quad i \in \{1,2,3,4\} \qquad 4.12$$

The definition of the terms in these equations is shown in the Notation section, and these parameters are shown in $Figure\ 3-1$. The nonlinear model is linearized with respect to the operating points given below. In addition, the given continuous-time matrices are calculated according to the values given in $Table\ 4-1$.

Table 4-1 Parameters of quadruple-tank process (Alvarado, et al., 2011)

γ_a= 0.3	γ_b=0.4
$q_{a,0}$ = 1.63 m^3/h	$q_{b,0}$ = 2.0 m^3/h
$q_{a,max}$ = 3.26 m^3/h	$q_{b,max}$ = 4.0 m^3/h
q_{min} = 0.0 m^3/h (for q_a and q_b)	h_{min} = 0.2 m (for all cases)
$h_{1,0}$ = 0.65 m	$h_{1,max}$ = 1.36 m
$h_{2,0}$ = 0.66 m	$h_{2,max}$ = 1.36 m
$h_{3,0}$ = 0.65 m	$h_{3,max}$ = 1.30 m
$h_{4,0}$ = 0.66 m	$h_{4,max}$ = 1.30 m
α_1 = 1.31*10^{-4} m^2	α_3 = 9.27*10^{-5} m^2
α_2 = 1.51*10^{-4} m^2	α_4 = 8.82*10^{-5} m^2
S = 0.06 m^2	$g = 9.81 m/s^2$

Continuous-time Linear Model

$$\frac{dx}{dt} = A_c x - B_c u \qquad 4.13$$

$$y = C_c x \qquad 4.14$$

Where $x = (x_1, x_2, x_3, x_4)$ $u = (u_1, u_2)$, and $y = (x_1, x_2)$. Used continuous-time matrices are given as follows.

$$A_c = \begin{bmatrix} -\tau_1 & 0 & \tau_3 & 0 \\ 0 & -\tau_2 & 0 & \tau_4 \\ 0 & 0 & -\tau_3 & 0 \\ 0 & 0 & 0 & -\tau_4 \end{bmatrix} \quad B_c = \begin{bmatrix} \frac{\gamma_a}{S} & 0 \\ 0 & \frac{\gamma_b}{S} \\ 0 & \frac{(1-\gamma_b)}{S} \\ \frac{(1-\gamma_a)}{S} & 0 \end{bmatrix} \quad C_c = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

These matrices are calculated by using the following parameters in $Table\ 2$. The matrices will be given in the next chapter.

Based on the linear continuous time model equations ($Eq\ 4.13\ and\ Eq\ 4.14$), the discrete-time model is got by using the Tustin method for chosen sampling time ($5\ seconds$).

The continuous- and discrete-time models should be matched, accurately. The model has also system dynamics. Therefore, the Tustin method is the best way to create correct matrices for these models.

Discrete-time model

$$x(k+1) = Ax(k) - Bu(k) \quad 4.15$$

$$y(k) = Cx(k) \quad 4.16$$

The discrete-time matrices ($A, B, and\ C$) are calculated with the Matlab code given in the $Appendix\ 1$.

Chosen Control Problem (Alvarado, et al., 2011)

The experiment was designed by Alvarado to follow reference changes in fluid levels of the first and second tanks (2011). The experiment is occurred by the qa and qb inlet flows manipulating according to the liquid levels.

- The first set-points are chosen as $s_1 = 0.65, and\ s_2 = 0.65\ m$. This selected set point is important to ensure that the plant operates at its operating point and to provide the same initial conditions for all controllers. When the system reaches the operating point, the reference point begins to operate for 300 seconds.
- Firstly, the set points are altered as to $s_1 = 0.3, and\ s_2 = 0.3$ m during the 3000 seconds.
- Secondly, the set points are altered as to $s_1 = 0.5, and\ s_2 = 0.75$ m during the 3000 seconds.
- Finally, the set points are altered as to $s_1 = 0.9, and\ s_2 = 0.75$ m during the 3000 seconds. For this change, tank 3 and tank 4 must be discharged and filled respectively (Alvarado, et al., 2011).

The designed MPC controller must meet all of the constraints, so it must track on the set point $s(k)$. The cost function is used for tracking of MPC. The function is based on the steady state (x_r) and input (u_r) variables defined by r variable, and the constraints (Alvarado, et al., 2011). The cost function is:

$$V_N(x,s,U,r) = \sum_{i=0}^{N-1} \|y(i) - r\|_Q^2 + \|u(i) - u_r\|_R^2 + \|x(N) - x_r\|_P^2 + \|r + h^0 - s\|_T^2 \quad 4.17$$

where $Q, R, P, and\ T$ are matrices which are created using appropriate dimensions according to the terms, U is a sequence of input values future, h^0 is a matrix which represents h^0 values for tank 1 and 2, s is the setpoint values, and $y(i)$ shows predicted output values of the system. $y(i)$ has been calculated by using $x(i+1) = Ax(i) + Bu(i)$ and $y(i) = Cx(i)$ equations and $x(0) = x$ initial condition.

The MPC optimization problem is given by

$$\begin{aligned} &minimize\ V_N && \sum_{k=0}^{N-1}(x,s,U,r) \\ &subject\ to && x(k+) = Ax(k) + Bu(k) \\ & && y(k) = Cx(k) \\ & && x(0) = x\,, \\ & && (x,u) \in Z \\ & && (x_r,u_r) = M_s r, \\ & && U = \{u_r, \dots, u_{r+N-1}\} \end{aligned} \quad 4.18$$

According to the given prediction model, M_s matrix can be calculated such as $Ms \cdot r = (xr, ur)$ (Alvarado, et al., 2011). Characterization of the input and steady-state values are done using follows equations (Limon, et al., 2008).

$$\begin{bmatrix} A - I_n & B & 0_{n,1} \\ C & D & -I_p \end{bmatrix} \begin{bmatrix} xr \\ ur \\ r \end{bmatrix} = \begin{bmatrix} 0_{n,1} \\ 0_{p,1} \end{bmatrix} \quad 4.19$$

The steady state (x_r) and input (u_r) decision variables are defined according to the reference output variable using the above equation (Muske & Rawlings, 1993 cited by Limon, et al., 2008). The solution obtained from the above equation can be parametrized as follows.

$$(x_r, u_r) = M_s r \qquad y_r = N_s r \qquad 4.20$$

Where r is the parameter vector that can be characterized. This parameter provides expression of calculations with the least variance by simplifying of the calculation. Acceptable steady-state and inputs can be limited by constraints (Limon, et al., 2008).

When the control law and $Eq\ 4.15\ and\ Eq\ 4.16$ are considered, the following invariant set is defined.

$$u = K(x - x_s)u_s = Kx + Lr \qquad 4.24$$

Also

$$L = [-K\ I_m]\ M_s \qquad 4.25$$

An admissible invariant set can be defined as the set, which is obtained from initial conditions, steady states (x_r) and inputs (u_r).

While choosing weight matrices for the controller, it is intended to lower the performance index. Therefore, $Q = I, R = 0.01I$ and $T = 100I$ (Offset weight matrix), and estimated horizon 5 have been selected (Alvarado, et al., 2011). The P matrix ($n\ x\ n$) is derived from the Riccati quadratic algebraic equation ($Eq\ 4.21$) (Kucera, 1973), using the *care* function in Matlab.

$$0 = -PBB^T P + PA + A^T P + C^T C \qquad 4.21$$

$$u(k) = -B^T P x(k) \qquad 4.22$$

Given the control law in the above equation ($Eq\ 4.22$), the terminal control gain K is calculated as follows.

$$K = -B^T P \qquad 4.23$$

4.2 The Methodology of the CSTR

The material balance equation ($Eq\ 4.27$) is done for each tank by using the general mass balance formula.

$$Input - Output + Generation = Accummulation \quad 4.26$$

$$\mathrm{q_f}C_f - q_f C + (-\mathrm{r_A})V = \mathrm{V}\frac{\mathrm{dC}}{\mathrm{dt}} \quad 4.27$$

The reaction rate is calculated by using the Arrhenius expression ($Eq\ 4.28$) (Pekar & Havlena, 2004).

$$r_A = K_0 C e^{-\frac{E}{RT}} \quad 4.28$$

Where K_0 is the reaction rate constant, E is the activation energy, R is the ideal gas constant and T is the reactor temperature. $Eq\ 4.28$ is substituted in $Eq\ 4.27$, and the following expression is obtained by arranging mass balance.

$$\frac{q_f}{V}\left(C_f - C\right) - K_0 C e^{-\frac{E}{RT}} = \frac{dC}{dt} \quad 4.29$$

Where V is the volume of CSTR, q_f is the flow rate of the product, C_f is the input concentration, C is the controlled variable. Other terms and their values are shown in $Table\ 4-2$.

The energy balance of the CSTR is shown as follows.

$$\frac{dT}{dt} = \frac{q_f}{V}\left(T_f - T\right) + K_1 C e^{-\frac{E}{RT}} + K_2 q_c \left(1 - e^{-\frac{K_3}{q_c}}\right)(T_c - T) \quad 4.30$$

Where T_f is the inlet temperature, T_c is the coolant temperature, K_1, K_2, and K_3 are constants, q_c is the flow rate of coolant, which is the manipulated variable.

Table 4-2 The operating conditions of CSTR

Variables	Values
q_f	$100\ (l/min)$
C_f	$1\ (mol/l)$
T_f	$350\ (K)$
q_c	$98.9\ (l/min)$
C	$0.0848\ (mol/l)$
T	$442.07\ (K)$
T_c	$350\ (K)$
E/R	$10000\ (K)$
V	$100\ (l)$
K_0	$7.2\ x\ 1010\ (1/min)$
K_1	$1.44\ x\ 1013$
K_2	$0.01\ (l-1)$
K_3	$700\ (l/min)$

The given mass and energy balances are modelled based on $Figure\ 4-1$. These equations are nonlinear differential equations and so to solve them is more difficult.

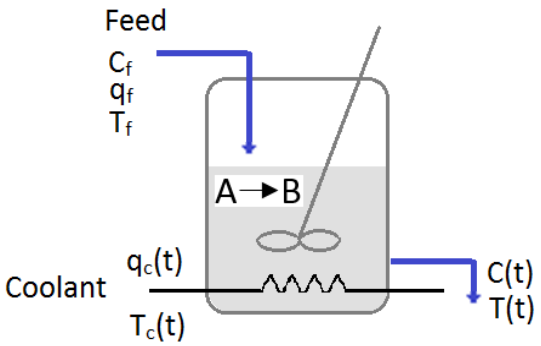


Figure 4-1 The continuous stirred-tank reactor diagram

They are linearized around operating points ($Table\ 4-2$) by using Taylor series with following $Eq\ 4.31$.

$$C = C_0$$
$$T = T_0 \qquad 4.31$$
$$q_c = q_{c_0}$$

The linearization has occurred with the following continuous-time matrices.

Continuous-time Linear Model

$$\frac{d\Delta x}{dt} = A_r \Delta x - B_r \Delta u \qquad 4.32$$

$$\Delta y = C_r \Delta x$$

Where $x = [T \ C]^T$, $\Delta x = (x - x_0)$, $\Delta u = (u - u_0)$, and $\Delta y = (y - y_0)$. Used continuous-time matrices are given as follows.

$$A_r = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \qquad B_r = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \qquad C_c = [0 \quad 1]$$

These matrices are calculated by following the below equations with parameters in $Table\ 4-2$.

$$a_{11} = -\frac{q_f}{V} + K_1 C \frac{E}{RT^2} e^{-\frac{E}{RT}} - K_2 q_c \left(1 - e^{-\frac{K_3}{q_c}}\right) \qquad a_{21} = -K_0 C \frac{E}{RT^2} e^{-\frac{E}{RT}}$$

$$a_{12} = K_1 e^{-\frac{E}{RT}} \qquad a_{22} = -\frac{q_f}{V} - K_0 e^{-\frac{E}{RT}} \qquad 4.33$$

$$b_1 = K_2 (T_c - T)\left(1 - e^{-\frac{K_3}{q_c}}\left(1 + \frac{K_3}{q_c}\right)\right) \qquad b_2 = 0$$

Based on the linear continuous time model, the discrete-time model is got by using the Tustin method for chosen sampling time (5 $seconds$).

Discrete-time model

$$x(k+1) = Ax(k) - Bu(k) \qquad 4.34$$

$$y(k) = Cx(k) \qquad 4.35$$

The discrete-time matrices ($A, B, and\ C$) are calculated using the Matlab code. The cost function is

$$V = \frac{1}{2} \sum (y - r)^T Q (y - r) + \Delta u^T R \, \Delta u \qquad 4.36$$

$$\Delta u = Du - u_0 \qquad 4.37$$

$$u_0 = [u(t_0) \ 0 \ \ldots 0]^T \qquad 4.38$$

where $Q\ and\ R$ are matrices with appropriate dimensions according to terms, D is a specific matrix.

4.3 The Methodology of the Quadruple-CSTR Process

As in the previous sections, the following equations are created with the aid of the mass and energy balances and $Figure\ 4-2$:

$$\begin{aligned}
\frac{dh_1}{dt} &= \frac{-\alpha_1}{S}\sqrt{2gh_1} + \frac{\alpha_3}{S}\sqrt{2gh_3} + \frac{\gamma_a}{S}\,q_a \\
\frac{dh_2}{dt} &= \frac{-\alpha_2}{S}\sqrt{2gh_2} + \frac{\alpha_4}{S}\sqrt{2gh_4} + \frac{\gamma_b}{S}\,q_b \\
\frac{dh_3}{dt} &= \frac{-\alpha_3}{S}\sqrt{2gh_3} + \frac{(1-\gamma_b)}{S}\,q_b \\
\frac{dh_4}{dt} &= \frac{-\alpha_4}{S}\sqrt{2gh_4} + \frac{(1-\gamma_a)}{S}\,q_a
\end{aligned} \qquad 4.39$$

$$\begin{aligned}
\frac{dC_1}{dt} &= \frac{q_a}{h_1 S}(C_f - C_1) - K_0 C_1 e^{-\frac{E}{RT_1}} \\
\frac{dC_2}{dt} &= \frac{q_b}{h_2 S}(C_f - C_2) - K_0 C_2 e^{-\frac{E}{RT_2}} \\
\frac{dC_3}{dt} &= \frac{q_b}{h_3 S}(C_f - C_3) - K_0 C_3 e^{-\frac{E}{RT_3}} \\
\frac{dC_4}{dt} &= \frac{q_a}{h_4 S}(C_f - C_4) - K_0 C_4 e^{-\frac{E}{RT_4}}
\end{aligned} \qquad 4.40$$

$$\begin{aligned}
&\frac{q_a}{h_1 S}(T_f - T_1) + K_1 C_1 e^{-\frac{E}{RT_1}} + K_2 q_{c_1}\left(1 - e^{-\frac{K_3}{q_{c_1}}}\right)(T_c - T_1) \\
&\frac{q_b}{h_2 S}(T_f - T_2) + K_1 C_2 e^{-\frac{E}{RT_2}} + K_2 q_{c_2}\left(1 - e^{-\frac{K_3}{q_{c_2}}}\right)(T_c - T_2) \\
&\frac{q_b}{h_3 S}(T_f - T_3) + K_1 C_3 e^{-\frac{E}{RT_3}} + K_2 q_{c_3}\left(1 - e^{-\frac{K_3}{q_{c_3}}}\right)(T_c - T_3) \\
&\frac{q_a}{h_4 S}(T_f - T_4) + K_1 C_4 e^{-\frac{E}{RT_4}} + K_2 q_{c_4}\left(1 - e^{-\frac{K_3}{q_{c_4}}}\right)(T_c - T_4)
\end{aligned} \qquad 4.41$$

Linearization is done for simplifying the solution of the equations. The following equations ($Eq\ 4.42\ and\ Eq\ 4.43$) are used for linearization:

$$x_i = h_i - h_i^0$$
$$u_1 = q_a - q_a^0 \quad 4.42$$
$$u_2 = q_b - q_b^0$$
$$C = C_0$$
$$T = T_0 \quad 4.43$$
$$q_c = q_{c_0}$$

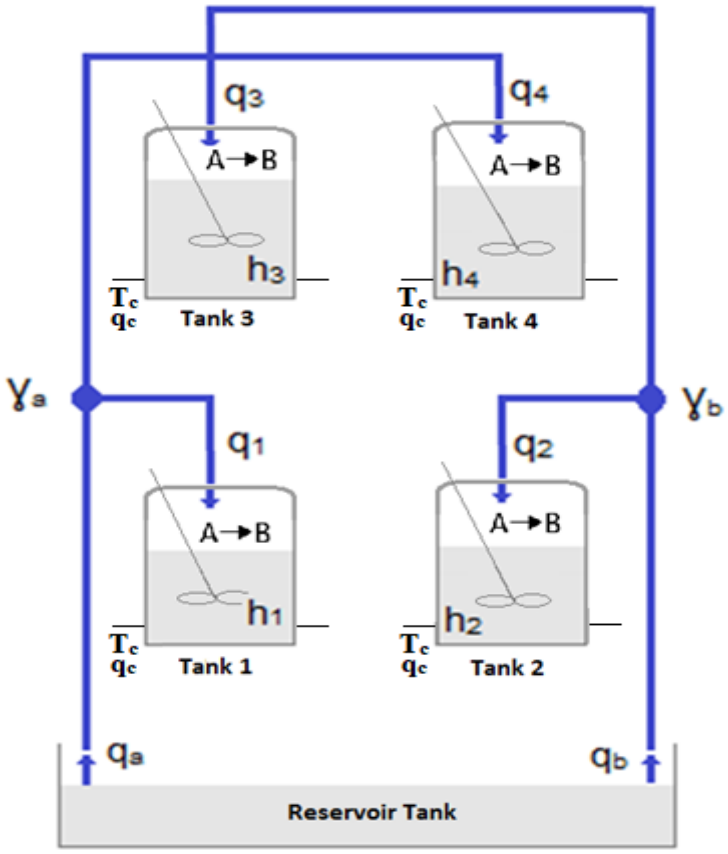


Figure 4-2 The quadruple CSTR process diagram

The following continuous-time linear model equations are obtained by using following equations.

$$\tau_i = \frac{\alpha_i}{S}\frac{\sqrt{g}}{\sqrt{2h_i^0}} \quad i \in \{1,2,3,4\}$$

$$a_{ij} = -\frac{q_f}{V} + K_1 C \frac{E}{RT^2} e^{-\frac{E}{RT}} - K_2 q_c \left(1 - e^{-\frac{K_3}{q_c}}\right)$$
$$(ij = 55, 66, 77, 88)$$

$$a_{ij} = K_1 e^{-\frac{E}{RT}}$$
$$(ij = 59, 610, 711, 812)$$

$$a_{21}{}^{i} = -K_0 C \frac{E}{RT^2} e^{-\frac{E}{RT}}$$
$$(ji = 95, 106, 117, 128)$$

$$a_{ij} = -\frac{q_f}{V} - K_0 e^{-\frac{E}{RT}}$$
$$(ij = 99, 1010, 1111, 1212)$$

$$a_{ij} = \frac{q_a}{S} T_f - \frac{q_a}{S}$$
$$(ij = 51, 84, 15, 48)$$

$$a_{ij} = \frac{q_a}{S} C_f - \frac{q_a}{S}$$
$$(ij = 91, 124, 19, 412)$$

$$a_{ij} = \frac{q_b}{S} T_f - \frac{q_b}{S}$$
$$(ij = 62, 73, 26, 37)$$

$$a_{ij} = \frac{q_b}{S} C_f - \frac{q_b}{S}$$
$$(ij = 102, 113, 210, 311)$$

$$b_{ij} = K_2 (T_c - T)\left(1 - e^{-\frac{K_3}{q_c}\left(1 + \frac{K_3}{q_c}\right)}\right)$$
$$(ij = 11,\ 22, 33, 44)$$

$$b_{ij} = 0$$
$$(ij = 51, 62, 73, 84)$$

$$b_{ij} = \frac{T_f - 1}{S}$$
$$(ij = 51,\ 81,\ 62,\ 72)$$

$$b_{ij} = \frac{C_f - 1}{S}$$
$$(ij = 91,\ 121,\ 102,\ 112)$$

Table 4-3 Parameters of the quadruple-tank process

γ_a= 0.3	γ_b=0.4	α_1 = 1.31*10^{-4} m^2	α_2 = 1.51*10^{-4} m^2
$q_{a,0}$ = 1.63 m^3/h	$q_{b,0}$ = 2.0 m^3/h	α_3 = 9.27*10^{-5} m^2	α_4 = 8.82*10^{-5} m^2
$q_{a,max}$ = 3.26 m^3/h	$q_{b,max}$ = 4.0 m^3/h	S = 0.06 m^2	$V = 100\ (l)$
q_{min} = 0.0 m^3/h (for q_a and q_b)	hmin = 0.2 m (for all cases)	$q_f = 100\ (l/min)$	$q_c = 98.9\ (l/min)$
		$C_f = 1\ (mol/l)$	$C = 0.0848\ (mol/l)$
$h_{1,0}$ = 0.65 m	$h_{1,max}$ = 1.36 m	$T_f = 350\ (K)$	$T = 442.07\ (K)$
$h_{2,0}$ = 0.66 m	$h_{2,max}$ = 1.36 m	$\frac{E}{R} = 10000\ (K)$	$T_c = 350\ (K)$
$h_{3,0}$ = 0.65 m	$h_{3,max}$ = 1.30 m	$K_0 = 7.2x1010\ (min^{-1})$	$K_1 = 1.44\ x\ 1013$
$h_{4,0}$ = 0.66 m	$h_{4,max}$ = 1.30 m	$K_2 = 0.01\ (l-1)$	$K_3 = 700\ (l/min)$

These matrices are calculated by following the below equations with parameters in $Table\ 4-3$.

$$A_c = \begin{bmatrix} -\tau_1 & 0 & \tau_3 & 0 & a_{15} & 0 & 0 & 0 & a_{19} & 0 & 0 & 0 \\ 0 & -\tau_2 & 0 & \tau_4 & 0 & a_{26} & 0 & 0 & 0 & a_{210} & 0 & 0 \\ 0 & 0 & -\tau_3 & 0 & 0 & 0 & a_{37} & 0 & 0 & 0 & a_{311} & 0 \\ 0 & 0 & 0 & -\tau_4 & 0 & 0 & 0 & a_{48} & 0 & 0 & 0 & a_{412} \\ a_{51} & 0 & 0 & 0 & a_{55} & 0 & 0 & 0 & a_{59} & 0 & 0 & 0 \\ 0 & a_{62} & 0 & 0 & 0 & a_{66} & 0 & 0 & 0 & a_{610} & 0 & 0 \\ 0 & 0 & a_{73} & 0 & 0 & 0 & a_{77} & 0 & 0 & 0 & a_{711} & 0 \\ 0 & 0 & 0 & a_{84} & 0 & 0 & 0 & a_{88} & 0 & 0 & 0 & a_{812} \\ a_{91} & 0 & 0 & 0 & a_{95} & 0 & 0 & 0 & a_{99} & 0 & 0 & 0 \\ 0 & a_{102} & 0 & 0 & 0 & a_{106} & 0 & 0 & 0 & a_{1010} & 0 & 0 \\ 0 & 0 & a_{113} & 0 & 0 & 0 & a_{117} & 0 & 0 & 0 & a_{1111} & 0 \\ 0 & 0 & 0 & a_{124} & 0 & 0 & 0 & a_{128} & 0 & 0 & 0 & a_{1212} \end{bmatrix}$$

$$B_c = \begin{bmatrix} \frac{\gamma_a}{S} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{\gamma_b}{S} & 0 & 0 & 0 & 0 \\ 0 & \frac{(1-\gamma_b)}{S} & 0 & 0 & 0 & 0 \\ \frac{(1-\gamma_a)}{S} & 0 & 0 & 0 & 0 & 0 \\ b_{51} & 0 & b_{53} & 0 & 0 & 0 \\ 0 & b_{62} & 0 & b_{64} & 0 & 0 \\ 0 & b_{72} & 0 & 0 & b_{75} & 0 \\ b_{81} & 0 & 0 & 0 & 0 & b_{86} \\ b_{91} & 0 & b_{93} & 0 & 0 & 0 \\ 0 & b_{102} & 0 & b_{104} & 0 & 0 \\ 0 & b_{112} & 0 & 0 & b_{115} & 0 \\ b_{121} & 0 & 0 & 0 & 0 & b_{126} \end{bmatrix}$$

$$C_c = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Discrete-time model

$$x(k+1) = Ax(k) - Bu(k) \qquad 4.47$$

$$y(k) = Cx(k) \qquad 4.48$$

The discrete-time matrices ($A, B, and\ C$) are calculated from the linear continuous time model by using the Tustin method for chosen sampling time ($5\ seconds$).The control problem given in $Section\ 4.1$ is used for this process.

The cost function is

$$V_N(x, s, U, r) = \sum_{i=0}^{N-1} \|y(i) - r\|_Q^2 + \|u(i) - u_r\|_R^2 + \|x(N) - x_r\|_P^2 + \|r + h^0 - s\|_T^2 \qquad 4.49$$

where $Q, R, P, and\ T$ are matrices which are created using appropriate dimensions according to terms, U is a sequence of input values future, h^0 is a matrix which represents h^0 values for tank 1 and 2, s is the setpoint values, and $y(i)$ shows predicted output values of the system. $y(i)$ has been calculated by using $x(i+1) = Ax(i) + Bu(i)$ and $y(i) = Cx(i)$ equations and $x(0) = x$ initial condition. The MPC optimization problem is given in the previous section ($Eq\ 4.18$).

5

5 RESULTS AND DISCUSSION

In this chapter, the results that are obtained using mathematical calculations, and Matlab and GAMs scripts are shown for each process.

5.1 The result of the Quadruple-Tank Process

Considering $Table\ 4-1$, the calculated continuous time matrices are shown in the following section. These matrices are calculated according to the methodology of $Ac\ and\ Bc$ matrices in $Section\ 4.1$.

$$A_c = \begin{bmatrix} 0.0081 & 0 & -0.0057 & 0 \\ 0 & 0.0093 & 0 & -0.0054 \\ 0 & 0 & 0.0057 & 0 \\ 0 & 0 & 0 & 0.0054 \end{bmatrix} \quad B_c = \begin{bmatrix} 5 & 0 \\ 0 & 10 \\ 0 & 11.67 \\ 6.67 & 0 \end{bmatrix} \quad C_c = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Based on the linear continuous-time matrices, the discrete-time matrices are calculated by using the Tustin method in Matlab for the chosen sampling time ($5\ seconds$) ($Appendix\ 1$).

$$A = \begin{bmatrix} 0.9713 & 0 & 0.0203 & 0 \\ 0 & 0.9671 & 0 & 0.0192 \\ 0 & 0 & 0.9797 & 0 \\ 0 & 0 & 0 & 0.9808 \end{bmatrix} \quad B = \begin{bmatrix} -12.5 & 0 \\ 0 & -25 \\ -29.17 & 0 \\ 0 & -16.7 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$V_N(x,s,U,r) = \sum_{i=0}^{N-1} ((y(i)-r(i))^2)^T Q((y(i)-r(i))^2) + ((u(i)-ur(i))^2)^T R((u(i)-ur(i))^2)$$
$$+ ((x(i)-xr(i))^2)^T P((x(i)-xr(i))^2)$$
$$+ ((r(i)+h^0-s)^2)^T T((r(i)+h^0-s)^2)$$

Where the first term $(y(i)-r)$ of the equation represents the output cost, the second term $(u(i)-u_r)$ is the input cost, the third $(x(N)-x_r)$ is the steady state cost and the final term $(r+h^0-s)$ shows the offset cost function. The representation of the sub-symbols (P,R,Q,T) in the above objective function is as follows.

The matrices $(R,Q,and\ T)$ used in this function have been determined by taking the article of Alvarado et al. work into consideration.

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad R = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix} \quad T = \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix}$$

P matrix ($n\ x\ n$) is derived from Riccati quadratic algebraic Equation by using discrete-time matrices ($A, B, and\ C$) in Matlab ($Appendix\ 2$).

$$P = \begin{bmatrix} 1434.8 & 0 & -613.03 & 0 \\ 0 & 2.7x10^{6} & 0 & 4.1x10^{6} \\ -613.03 & 0 & 261.94 & 0 \\ 0 & 4x10^{6} & 0 & 6.1x10^{6} \end{bmatrix}$$

According to the control law ($Eq\ 4.22$), the terminal control gain K is calculated by using B and P matrices with ($Eq\ 4.23$).

$$K = \begin{bmatrix} 52.8 & 0 & -22.1 & 0 \\ 0 & 2300.4 & 0 & 3445.4 \end{bmatrix}$$

Using $Eq\ 4.18$, the M_S matrix and the following equations are obtained to show relations between $r\ and\ xr/ur$ ($Appendix\ 3$).

$$M_s = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ -7x10^{-4} & 0 \\ 0 & -8x10^{-4} \end{bmatrix}$$

$$x_{r1} = r_1 \qquad x_{r3} = r_1 \qquad u_{r1} = -7x10^{-4} r_1$$

$$x_{r2} = r_2 \qquad x_{r4} = r_2 \qquad u_{r2} = -8x10^{-4} r_2$$

Considering the control law and the M_s matrix, the L matrix is calculated as $Eq\ 4.25$.

$$L = \begin{bmatrix} -30.71 & 0 \\ 0 & -1145 \end{bmatrix}$$

The boundary conditions, which are given in the work of Alvarado, are used to limit the calculated values.

$$h_{min} = 0.2$$

$$h_{1,max} = 1.36 \qquad h_{2,max} = 1.36 \qquad h_{3,max} = 1.3 \qquad h_{4,max} = 1.3$$

$$q_{a,max} = 3.26 \qquad q_{a,min} = 0 \qquad q_{b,max} = 4 \qquad q_{b,min} = 0$$

Following these steps (written GAMs model in $Appendix\ 4$), the optimization value for each set point is calculated as $-7.05x10^{6}$.

After the above calculations, firstly optimization models are written for each set point and the results are obtained separately. Then a model with all set points was created in the GAMS program and the results were found.

The following graphs show the input and output values of the system for the first setpoint.

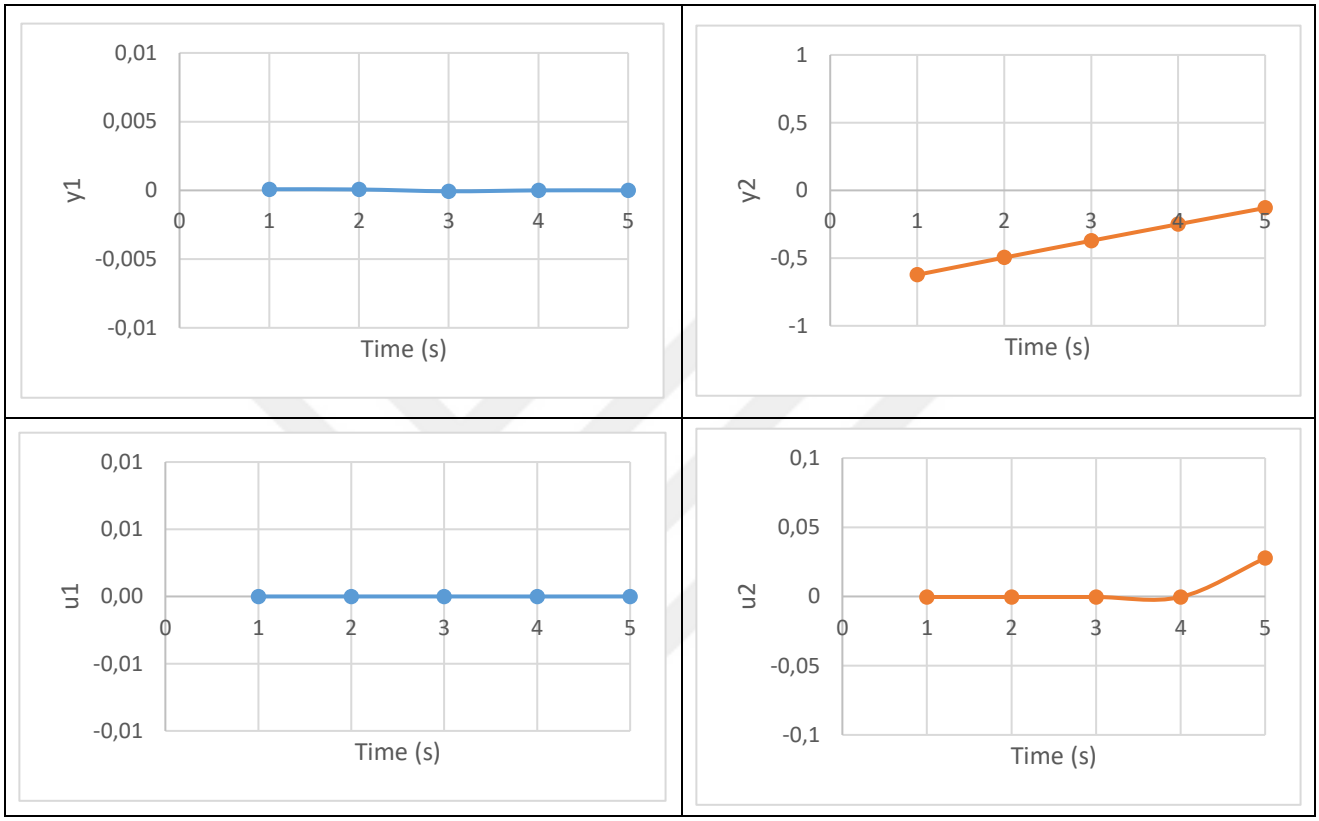


Figure-5-1 The evaluation of quadruple tank process for the conditions of the first control problem (set point 0.65)

Considering the graphs, it is observed that the inlet and outlet values of the first tank are close to zero as expected. Due to the unexpected increase in the outlet value of the second tank, the tank inlet appears to have moved away from zero at the end of the interval. These unexpected changes in output 2 and input 2 can occur because of the mistakes in modelling or the boundary conditions. Moreover, the interruption, which is done because the duration of the model is too long, of the program caused unexpected problems. The value of objective function for the first set points ($s1 = s2 = 0.65m$) is $-7.049x10^6$. The graphs obtained for the other three setpoints are quite similar to $Figure\ 5-1$, and the optimum value obtained from each model is equal. The graphics are given in the $Appendix\ 9$.

The following inputs and outputs were obtained with the help of the written model for the selected control problem (using the four setpoints). According to the obtained results, the output values are closer to stability than the input values are not stabile. The value of objective function for the control problem is found as -32.66 by using GAMs script in $Appendix\ 5$.

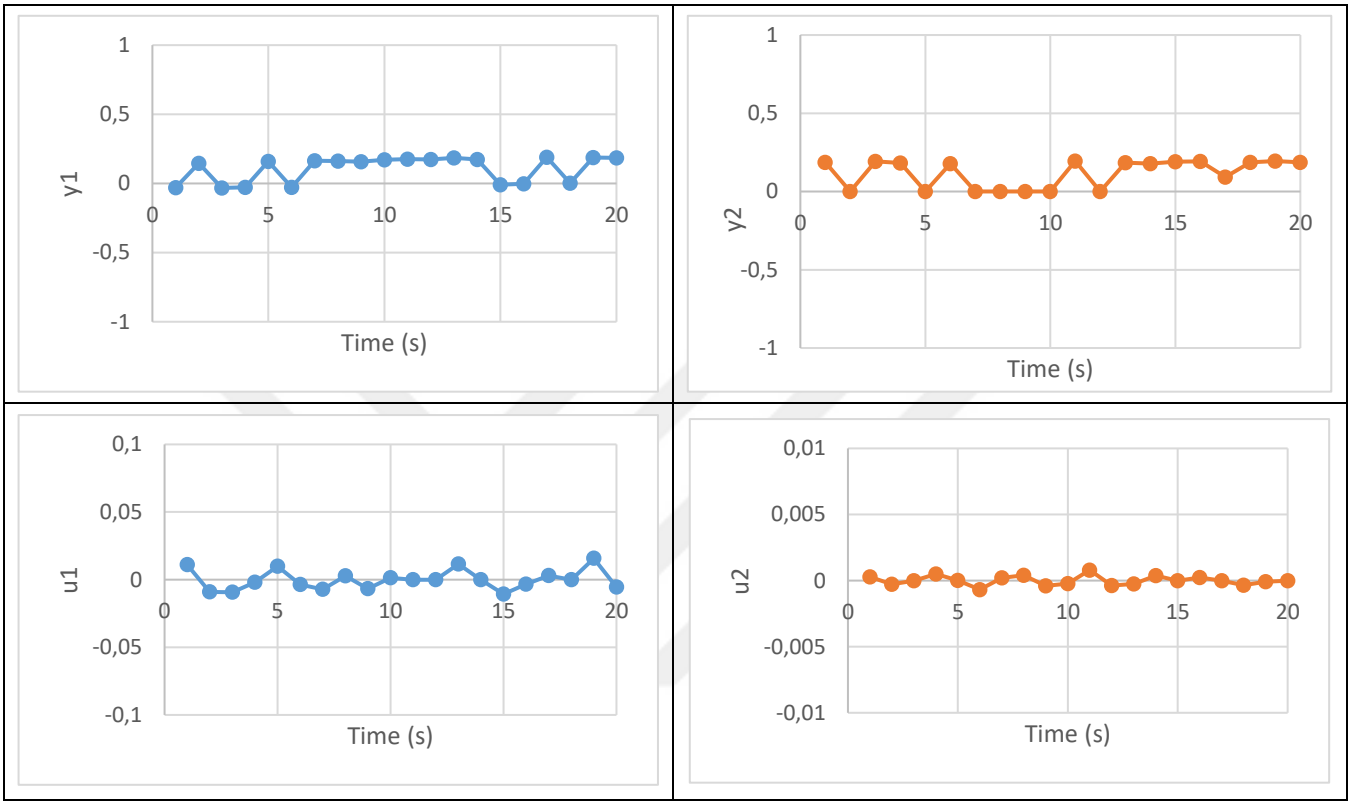


Figure-5-2 The evaluation of quadruple tank process for the conditions of the control problem

The input (flow rates) and output (level) variables are more clearly expressed in the graph below. The output values of the system and set points are shown in the first two graphs. When the desired set points and the output values of the system are compared, output values do not reach set point. Even if, the output value for the first tank is close to the set point in between 7th and 10th seconds, generally, it is not stable. The output value for the second tank does not reach to set point but it is more stable (after the 10th seconds) than the first tank.

The flow rates must be greater than zero to be physically possible. However, when the graph is examined it is seen that qa has values below zero. Finally, the value of qb is close to zero and stable.

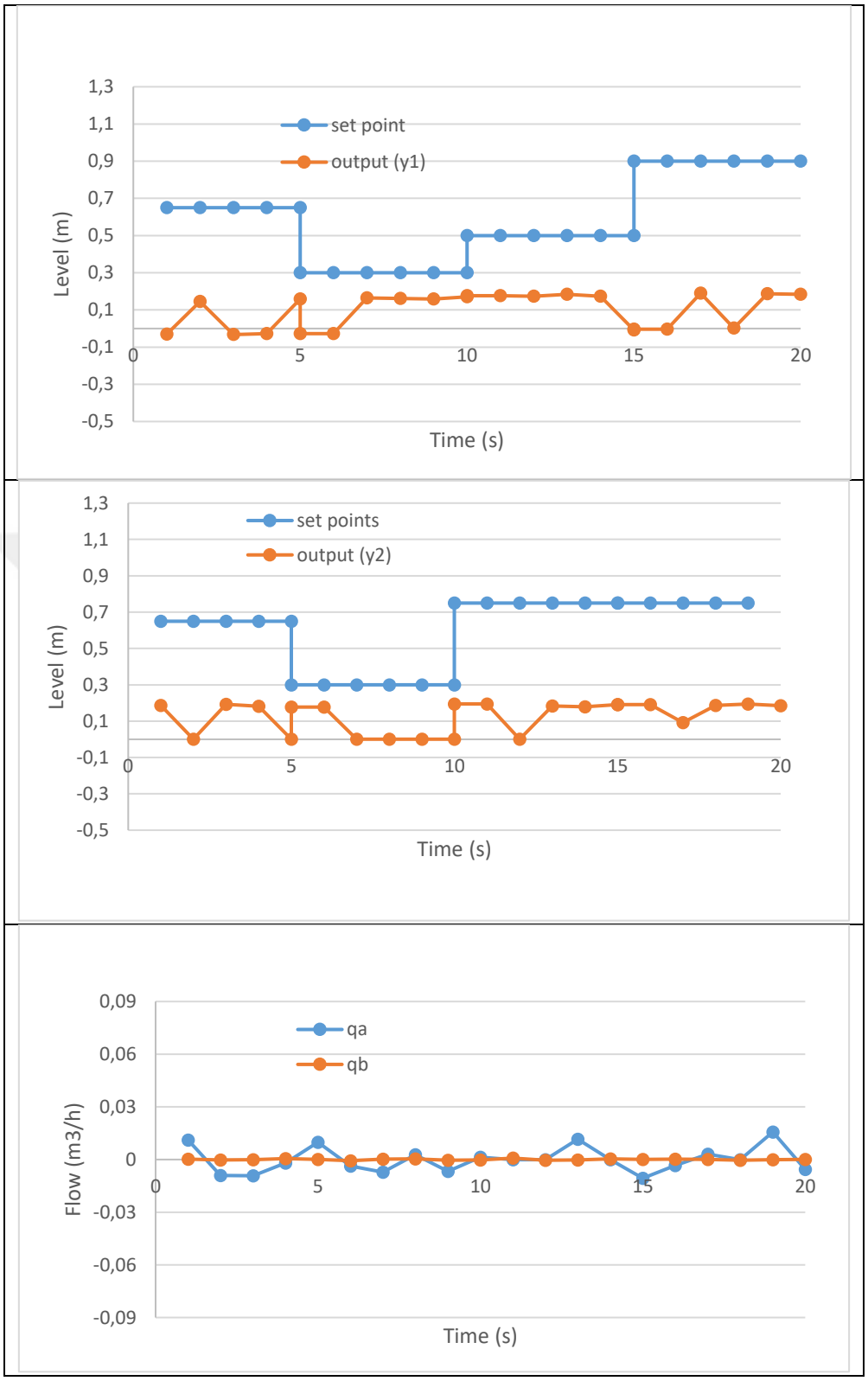


Figure 5-3 The evaluation of levels and flows for centralized MPC

Nevertheless, modelled system has not given the desired results. The reason for this can be that the model equations are correctly expressed in the GAMs program.

5.2 The result of the CSTR Process

Following continuous time matrices are calculated according to $Eq\ 4.33$, and $Table\ 4-2$. The operating points are selected as $qc = 98.9\ l/\ min$, $T = 442.07\ K$, and $C = 0.0848\ mol/l$.

$$A_r = \begin{bmatrix} 7.38 & 2158.99 \\ -0.0468 & -11.795 \end{bmatrix} \quad B_r = \begin{bmatrix} -0.914 \\ 0 \end{bmatrix} \quad C_r = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

Based on the linear continuous-time matrices, the discrete-time matrices have been calculated by using the Tustin method for the chosen sampling time (5 seconds) ($Appendix\ 6$).

$$A = \begin{bmatrix} -17.45 & -5397.5 \\ 0.117 & 30.49 \end{bmatrix} \quad B = \begin{bmatrix} 2.285 \\ 0 \end{bmatrix} \quad C = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

The objective function is

$$V = \frac{1}{2} \sum (y - r)^T Q (y - r) + \Delta u^T\ R\ \Delta u \qquad 4.36$$

Where the first term $(y(i) - r)$ of the equation represents the output cost, Δu is the input cost that is shown following.

$$\Delta u = D.u - u_0 \qquad 4.37$$

Where $u_0 = [u(t_0)\ 0\ \ldots 0]^T$ and D is a specific matrix. $Q\ and\ R$ are matrices which are appropriate dimensions according to terms.

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad R = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix} \quad D = [1]$$

By following these steps and values, the optimization value is found as $-9.06x10^{-6}$ ($Appendix\ 7$).

The inlet (coolant flow rate) and the outlet (concentration) values of the reactor process are given in the following graph. The system is modelled for 10 intervals. The given input and output boundaries are $0 - 150\ l/min$ and $0.001\ and\ 0.1\ mol/l$, respectively.

It can be said that the results are feasible considering the obtained optimization value and the stability of the graphics. However, it is not expected to be close to zero concentration. As mentioned in the previous section, this situation can cause from the

modelling error or incorrect choosing of boundary conditions or the interruption of the GAMs model.

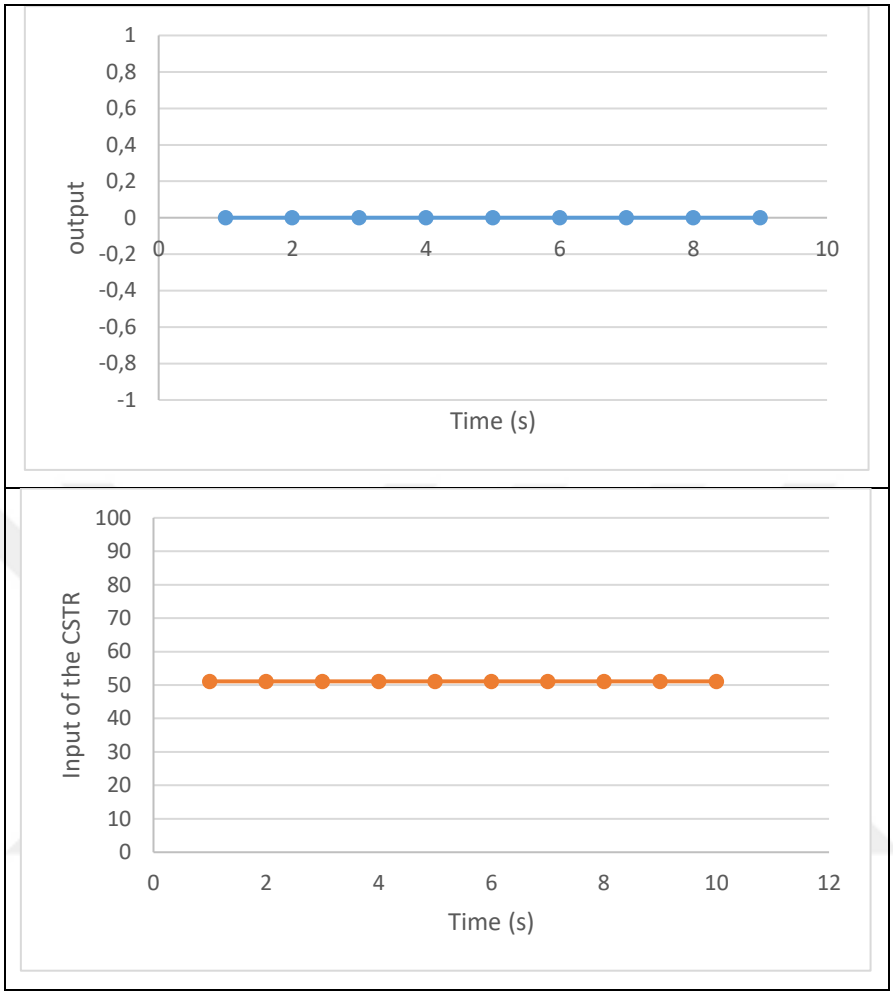


Figure 5-4 The evaluation of the output and input of CSTR for given operating point

When a manipulated variable (qc) has been increased by10 percent, the new operating points are selected as $qc = 108.8\, l/\, min$, $T = 442.07\, K$, and $C = 0.0848\, mol/l$. The continuous-time matrices are calculated as follows.

$$A_r = \begin{bmatrix} 7.28 & 2158.99 \\ -0.0468 & -11.795 \end{bmatrix} \quad B_r = \begin{bmatrix} -0.91 \\ 0 \end{bmatrix} \quad C_r = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

The discrete time matrices are calculated similarly to qc = 98.9 operating point. The calculated continuous-time matrices only are different from the above calculation.

$$A = \begin{bmatrix} -17.2 & -5397.5 \\ 0.117 & 30.49 \end{bmatrix} \quad B = \begin{bmatrix} 2.275 \\ 0 \end{bmatrix} \quad C = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

These different set point (qc) are used for each prediction horizon in the second model for CSTR. The set points are calculated above ($qc = 108.8\, l/min$ for the first set, $qc =$

$98.9\ l/\ min$ for the second set) and they are combined. The calculated value of the optimization function is found as $-8.354x10^{-6}$ using GAMs script in $Appendix\ 8$.

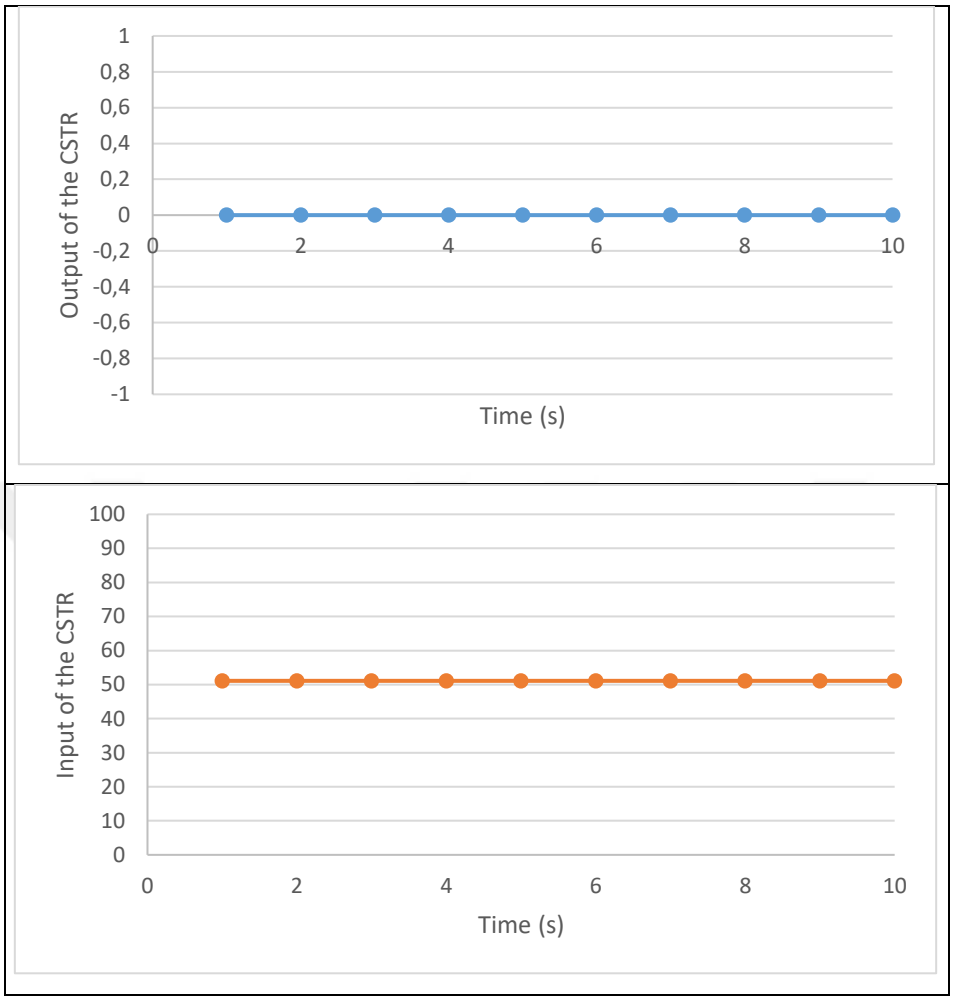


Figure 5-5 The evaluation of the given disturbance on the CSTR for given operating point

Although it has the disturbance, it seems that there is no difference between the first model. The small change in the value of the qc has not changed the solutions of calculations much. As can be seen from the matrices obtained with flow rates ($108.8\ l\ /\ min\ and\ 98.9\ l\ /\ min$), this situation depends on the place of qc in the equations used. Similar to the quadruple-tank process, errors in modelling and constraints may have caused this.

5.3 The result of the Quadruple CSTR Process

The modelling of the quadruple CSTR process given in the methodology section for the GAMs program did not be completed due to the short duration of the study. However, the model can be created using the methodologies for quadruple-CSTR process and following the quadruple-tank process steps. The process steps are listed below respectively.

- The continuous-time matrices are calculated using given matrices in previous section.

$$A_c = \begin{bmatrix} 0.0081 & 0 & -0.0057 & 0 & 9481.17 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.0093 & 0 & -0.0054 & 0 & 11633.33 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0057 & 0 & 0 & 0 & 11633.33 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0054 & 0 & 0 & 0 & 9481.17 & 0 & 0 & 0 & 0 \\ 9481.17 & 0 & 0 & 0 & 7.38 & 0 & 0 & 0 & 2158.99 & 0 & 0 & 0 \\ 0 & 11633.33 & 0 & 0 & 0 & 7.38 & 0 & 0 & 0 & 2158.99 & 0 & 0 \\ 0 & 0 & 11633.33 & 0 & 0 & 0 & 7.38 & 0 & 0 & 0 & 2158.99 & 0 \\ 0 & 0 & 0 & 9481.17 & 0 & 0 & 0 & 7.38 & 0 & 0 & 0 & 2158.99 \\ 0 & 0 & 0 & 0 & -0.0468 & 0 & 0 & 0 & -11.795 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.0468 & 0 & 0 & 0 & -11.795 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -0.0468 & 0 & 0 & 0 & -11.795 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.0468 & 0 & 0 & 0 & -11.795 \end{bmatrix}$$

$$B_c = \begin{bmatrix} 5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 11.67 & 0 & 0 & 0 & 0 \\ 6.67 & 0 & 0 & 0 & 0 & 0 \\ 5816.67 & 0 & 2.285 & 0 & 0 & 0 \\ 0 & 5816.67 & 0 & 2.285 & 0 & 0 \\ 0 & 5816.67 & 0 & 0 & 2.285 & 0 \\ 5816.67 & 0 & 0 & 0 & 0 & 2.285 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- The discrete-time matrices are calculated using Tustin method in Matlab.

$$A_c = \begin{bmatrix} 0.9797 & 0 & 0.0143 & 0 & -2.37x10^{-4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.9768 & 0 & 0.0135 & 0 & -2.91x10^{-4} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.9858 & 0 & 0 & 0 & -2.91x10^{-4} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.9865 & 0 & 0 & 0 & -2.37x10^{-4} & 0 & 0 & 0 & 0 \\ -2.37x10^{-4} & 0 & 0 & 0 & -17.45 & 0 & 0 & 0 & -5.4x10^{-3} & 0 & 0 & 0 \\ 0 & -2.91x10^{-4} & 0 & 0 & 0 & -17.45 & 0 & 0 & 0 & -5.4x10^{-3} & 0 & 0 \\ 0 & 0 & -2.91x10^{-4} & 0 & 0 & 0 & -17.45 & 0 & 0 & 0 & -5.4x10^{-3} & 0 \\ 0 & 0 & 0 & -2.37x10^{-4} & 0 & 0 & 0 & -17.45 & 0 & 0 & 0 & -5.4x10^{-3} \\ 0 & 0 & 0 & 0 & 0.117 & 0 & 0 & 0 & 30.49 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.117 & 0 & 0 & 0 & 30.49 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.117 & 0 & 0 & 0 & 30.49 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.117 & 0 & 0 & 0 & 30.49 \end{bmatrix}$$

$$B_c = \begin{bmatrix} -12.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & -25 & 0 & 0 & 0 & 0 \\ -29.18 & 0 & 0 & 0 & 0 & 0 \\ 0 & -16.68 & 0 & 0 & 0 & 0 \\ -1.45x10^{-4} & 0 & -5.71 & 0 & 0 & 0 \\ 0 & -1.45x10^{-4} & 0 & -5.71 & 0 & 0 \\ 0 & -1.45x10^{-4} & 0 & 0 & -5.71 & 0 \\ -1.45x10^{-4} & 0 & 0 & 0 & 0 & -5.71 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- The weight matrices ($P, Q, and\ T$) in the objective function are created based on the quadruple tank process.

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad R = \begin{bmatrix} 0.01 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.01 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.01 \end{bmatrix}$$

$$T = \begin{bmatrix} 100 & 0 & 0 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 & 0 \\ 0 & 0 & 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \end{bmatrix}$$

- P matrix (nxn) is derived from Riccati quadratic algebraic Equation by using discrete-time matrices ($A, B, and\ C$) in Matlab by using similar script with the quadruple-tank process.
- Considering to the control law, the terminal control gain K is calculated by using B and P matrices.
- Ms matrix is calculated like the quadruple tank process to show relations between $r\ and\ x_r/u_r$.
- The boundary conditions which are like values in $Section\ 5.1$, are used to limit the calculated values.

The following matrices are given in order to make the calculations more clear when the above steps are being performed. In the matrices, the which part is related to which process are shown.

Tank Process

$$A_c = \begin{bmatrix} -\tau_1 & 0 & \tau_3 & 0 & a_{15} & 0 & 0 & 0 & a_{19} & 0 & 0 & 0 \\ 0 & -\tau_2 & 0 & \tau_4 & 0 & a_{26} & 0 & 0 & 0 & a_{210} & 0 & 0 \\ 0 & 0 & -\tau_3 & 0 & 0 & 0 & a_{37} & 0 & 0 & 0 & a_{311} & 0 \\ 0 & 0 & 0 & -\tau_4 & 0 & 0 & 0 & a_{48} & 0 & 0 & 0 & a_{412} \\ a_{51} & 0 & 0 & 0 & a_{55} & 0 & 0 & 0 & a_{59} & 0 & 0 & 0 \\ 0 & a_{62} & 0 & 0 & 0 & a_{66} & 0 & 0 & 0 & a_{610} & 0 & 0 \\ 0 & 0 & a_{73} & 0 & 0 & 0 & a_{77} & 0 & 0 & 0 & a_{711} & 0 \\ 0 & 0 & 0 & a_{84} & 0 & 0 & 0 & a_{88} & 0 & 0 & 0 & a_{812} \\ a_{91} & 0 & 0 & 0 & a_{95} & 0 & 0 & 0 & a_{99} & 0 & 0 & 0 \\ 0 & a_{102} & 0 & 0 & 0 & a_{106} & 0 & 0 & 0 & a_{1010} & 0 & 0 \\ 0 & 0 & a_{113} & 0 & 0 & 0 & a_{117} & 0 & 0 & 0 & a_{1111} & 0 \\ 0 & 0 & 0 & a_{124} & 0 & 0 & 0 & a_{128} & 0 & 0 & 0 & a_{1212} \end{bmatrix}$$

CSTR Process

Tank Process

$$B_c = \begin{bmatrix} \frac{\gamma_a}{S} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{\gamma_b}{S} & 0 & 0 & 0 & 0 \\ 0 & \frac{(1-\gamma_b)}{S} & 0 & 0 & 0 & 0 \\ \frac{(1-\gamma_a)}{S} & 0 & 0 & 0 & 0 & 0 \\ b_{51} & 0 & b_{53} & 0 & 0 & 0 \\ 0 & b_{62} & 0 & b_{64} & 0 & 0 \\ 0 & b_{72} & 0 & 0 & b_{75} & 0 \\ b_{81} & 0 & 0 & 0 & 0 & b_{86} \\ b_{91} & 0 & b_{93} & 0 & 0 & 0 \\ 0 & b_{102} & 0 & b_{104} & 0 & 0 \\ 0 & b_{112} & 0 & 0 & b_{115} & 0 \\ b_{121} & 0 & 0 & 0 & 0 & b_{126} \end{bmatrix}$$

CSTR Process

Tank Process

$$C_c = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

CSTR Process

6

6 CONCLUSIONS AND FUTURE WORKS

6.1 Conclusion

In this study, it is aimed to model the centralized MPC strategy and to calculate the optimal result with the help of the GAMs program. These studies were carried out using quadruple-tank process and CSTR process. In addition, a model was built, which is a combination of the CSTR and the quadruple-tank process. The same studies were carried out for the process being created. The nonlinear process models were developed to control problems, and then the equations were linearized.

For the quadruple-tank process, and for the CSTR process, the same problems were encountered in this work. Although the results obtained are feasible, they are different from the expected results. The main reason for this is that the working periods of the models are long. Almost all models, mainly the model used for the control problem, were interrupted and the results given above were obtained. Since the process is interrupted before the optimization process is completed, the data obtained is not the final result of the model.

Although there are no problems with the creation of the methodology, there are problems with the integration of the models to the GAMs. Expression of constraints is the most important key to create a model in GAMs. Due to the fact that restrictions cannot be expressed correctly, error rates are high in the results.

The calculations for the Quadruple CSTR process have not been completed because of the computation of the matrix Ms and the insufficient time. The Ms matrix should be calculated manually. The Ms matrix which shows the relations of the input, output and the steady states of the system, can cause the errors in the results. This situation could have also caused the problems in the quadruple-tank process.

The models obtained in this study are not efficient. The plans for the development of the models are specified in the future work section.

6.2 Future work

In this section, future works are presented for the development of the model and process, and for the studies on other strategies.

- In the studies conducted on the central MPC, the time taken to obtain the results lasted longer than expected. So, further improvements are required to decrease the simulation time. To do this, changes can be made to equations. When the equations used are shorter, it may be easier for the system to calculate that equation and move to the next equation. The use of non-linear equations may also be causing the process time to lengthen. Moreover, different control horizon and prediction horizon can be selected.
- The results obtained are not close to the set points, which means that the error rate of the results is high. These mistakes resulting from modelling problems should be reduced with future work and the results should be closer to the set points.
- Calculation of the matrix Ms is one of the important steps in modelling the quadruple-tank process using for MPC strategies. As mentioned above, the calculations of the matrix Ms have been made manually. Work will be done to make these operations using computer programs.
- Finally, other MPC strategies, the Decentralized MPC and the Distributed MPC, should be implemented using the GAMs program. Once improvements have been made to the central MPC, the steps given in the modelling of the central MPC for these strategies can be followed.

7 REFERENCE LIST

Bemporad , A. & Barcelli, D., 2010. Decentralized Model Predictive Control. In: A. Bemporad, M. Heemels & M. Johansson, eds. *Networked Control Systems.* Berlin: Springer, pp. 149-178.

Johanssont, K. H., Horcht , A., Wijkt, O. & Hanssont, A., 1999. *Standard MPC formulation the Quadruple-Tank Process.* Arizona, Proceedings of the 38th Conference on Decision & Control.

Prasad, P., 2016. The Performance Analysis Of Four Tank System For Conventional Controller And Hybrid Controller. *International Research Journal of Engineering and Technology (IRJET),* pp. 2182-2187.

Zadeh , . A. & Whalen, B. H., 1962. On optimal control and linear programming. *IRE Trans. Automatic Control (Correspondesce),* Volume 4C-7, pp. 45-46.

Al-Gherwi, W., 2010. *Robust Distributed Model Predictive Control Strategies.* Ontario.

Alvarado, I. et al., 2006. An Educational Plant Based On the Quadruple-Tank Process. *Elsevier,* 39(6), pp. 82-87.

Alvarado, I. et al., 2011. A comparative analysis of distributed MPC techniques applied to the HD-MPC four-tank benchmark. *Journal of Process Control,* Issue 21, p. 800–815.

Boom, T. J. v. d. & Backx, . T. C., 1999. *Model Predictive Control*.

Camacho, E. F. & Bordons, C., 2007. *Model Predictive Control.* 2 ed. London: Springer.

Christofidesa, P. D., Scattoli, R., Pe˜nad, D. M. d. l. & Liue, J., 2013. Distributed model predictive control: A tutorial review and future esearch directions. *ELSEVIER,* Issue 51, pp. 21-41.

Clarke, D. W., Mohtadi, C. & Tuffs, P. S., 1987. Generalized Predictive Control Part I. The Basic Algorithm. *International Federation of Automatic Control,* 23(2), pp. 137-148.

Dagdeviren, O., 2018. *Report of Dissertation,* Surrey.

Garcia, C. E. & Morshedi, A. M., 1986. Quadratic Programming Solution of Dynamic Matrix Control (QDMC). *Chemical Engineering Communications ,* 46(1-3), pp. 73-87.

Garica, C. E., Morari, M. & Prett, D. M., 1989. *Model Predictive Control: Theory and Practice - a Survey,* Great Britain: International Federation of Automatic Control.

Goodwin, G. C., Graebe, S. F. & Salgado, M. E., 2000. *Control System Design.* International Edition ed. Valparaiso: Pearson.

Haber, R., Bars, R. & Schmitz, U., 2012. *Predictive Control in Process Engineering: From the Basics to the Applications.* s.l.:John Wiley & Sons, Incırporated.

Holkar, K. S. & Waghmare, L. M., 2010. An Overview of Model Predictive Control. *International Journal of Control and Automation,* pp. 47-64.

Hovd, M., 2004. *A brief introduction to Model Predictive Control,* Trondheim: The Norwegian University of Science and Technology.

Johansson, K. H., 2000. The Quadruple-Tank Process: A Multivariable Laboratory Process with an Adjustable Zero. *IEEE Transactions on Control Systems Technology,* Volume 8, pp. 456-465.

Kucera, V., 1973. A Review of the Matrix Riccati Equation. *Kybernetika,* Volume 9, pp. 42-61.

Kumar , A. S. & Ahmad, Z., 2012. Model Predictive Control (MPC) and Its Current Issues in Chemical Engineering. *Chemical Engineering Communications,* 4(199), pp. 472-511.

Limon, D., Alvarado, I., Alamo, . T. & Camacho, E., 2008. MPC for tracking piecewise constant references for constrained linear systems. *Automatica,* Issue 44, p. 2382–2387.

Pekar, J. & Havlena, V., 2004. Control of CSTR using Model Predictive Control based on mixture distribution. *IFAC Proceedings Volumes,* 37(13), pp. 793-798.

Propoi, A. I., 1963. Use of LP methods for synthesizing sampled-data. *Aut. Remote Control,* Volume 24, p. 837.

Qin, S. J. & Badgwell, T. A., 2003. A survey of industrial model predictive control technology. *Science Direct,* Issue 11, pp. 733-764.

Richalet, J. et al., 1987. *Predictive Functional Control - Application to Fast and Accurate Robots.* Munich, IFAC 10th Triennial World Congress.

Richalet, J., Rault, A., Testud, J. & Papon, J., 1978. model predictive heuristic control : applications. *Automatica,* Volume 14, pp. 413-428.

Richter, S., Mariethoz, S. & Morari, M., 2010. *High-Speed Online MPC Based on a Fast Gradient Method Applied to Power Converter Control.* Baltimore, American Control Conference.

Ruchika, N. R., 2013. Model Predictive Control: History and Development. *International Journal of Engineering Trends and Technology ,* Issue 4, p. 2600.

Scattolini, R., 2009. Architectures for distributed and hierarchical Model Predictive Control – A review. *Journal of Process Control,* Issue 19, pp. 723-731.

Seborg, D., Edgar, T., Mellichamp, D. & Doyle III, F. J., 2011. *Process Dynamics and Control.* 3 ed. :Jhon Wiley & Sons, Inc..

Shieh, L., Wang, H. & Yates, R., n.d. Discrete-c.

Venkat, A. N., 2006. *Distributed Model Predictive Control: Theory and Applications.* Madison.

Wikipedia, t. f. e., 2018. *Nash equilibrium: Revision history.* [Online]
Available at: https://en.wikipedia.org/wiki/Nash_equilibrium
[Accessed 10 08 2018].

Zhang, Y. & Li, S., 2007. Networked model predictive control based on neighbourhood optimization for serially connected large-scale processes. *Science Direct,* Volume 17, pp. 37-50.

8 APPENDICES

APPENDIX 1

Matlab script file shows to calculate the discrete-time matrices from the linear continuous-time matrices for the quadruple-tank process.

```
%dx/dt=Ax+Bu
%y=Cx
a1=1.31*10^-4; %Discharge constant of tank              %alpha
a2=1.51*10^-4;
a3=0.927*10^-4;
a4=0.882*10^-4;
g=9.81;         %Gravity force (m/s2)
S=0.06;         %Cross-sectional area(m2)
ga=0.3;         %Parameter of the 3 ways-valve          %gamma
gb=0.4;
h10=0.65;          %Linearization level of tank
h20=0.66;
h30=0.65;
h40=0.66;
t1=-(a1/S)*sqrt(9/(h10));                               %tau
t2=-(a2/S)*sqrt(9/(h20));
t3=-(a3/S)*sqrt(9/(h30));
t4=-(a4/S)*sqrt(9/(h40));
A = [-t1 0 t3 0; 0 -t2 0 t4; 0 0 -t3 0; 0 0 0 -t4];
B = [ga/S 0; 0 (1-gb)/S; (1-ga)/S 0; 0 gb/S];
C = [1 0 0 0; 0 1 0 0];
D = zeros(2,2);
Ts = 5;                                                 %sampling time
sys = ss(A,B,C,D);
[sysd2,G2] = c2d(sys,Ts,'tustin');
```

APPENDIX 2

Matlab script file shows that P matrix is derived from Riccati quadratic algebraic Equation for the quadruple-tank process.

```
A = [0.9713 0 0.0203 0; 0 0.9671 0 0.0192; 0 0 0.9797 0; 0 0 0 0.9808];
B = [-12.5 0; 0 -25; -29.17 0 ; 0 -16.7];
C= [1 0 0 0; 0 1 0 0];
[P,L,C] = care(A,B,C'*C)
```

APPENDIX 3

The steps used to calculate the relation between ur/xr and r are given below.

$$\begin{bmatrix} A - I_n & B & 0_{n,1} \\ C & D & -I_p \end{bmatrix} \begin{bmatrix} xr \\ ur \\ r \end{bmatrix} = \begin{bmatrix} 0_{n,1} \\ 0_{p,1} \end{bmatrix}$$

$$\begin{bmatrix} 0.9713-1 & 0 & 0.0203 & 0 & -12.5 & 0 & 0 & 0 \\ 0 & 0.9671-1 & 0 & 0.0192 & 0 & -25 & 0 & 0 \\ 0 & 0 & 0.9797-1 & 0 & 29.17 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.9808-1 & 0 & -16.7 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_r \\ u_r \\ r \end{bmatrix} = [0]_{6x6}$$

APPENDIX 4

GAMs script for the quadruple-tank process (set point $s_1 = s_2 = 0.65\,m$) is shown follows.

```
sets
   k prediction horizon /1*5/;
sets
   n number of states   /n1*n4/
   m number of inputs   /m1*m2/
   p number of outputs  /p1*p2/  ;

Alias(n,n1);
Alias(k,k1);
Alias(m,m1);
Alias(p,p1);

$CALL GDXXRW.EXE set_point_tank.xlsm par=s rng=A1:F3
Parameter s(p,k);
$GDXIN set_point_tank.gdx
$LOAD s
display s
$GDXIN

$CALL GDXXRW.EXE h_0_matrix_tank.xlsm par=h0 rng=A1:F5
Parameter h0(n,k);
$GDXIN h_0_matrix_tank.gdx
$LOAD h0
display h0
$GDXIN
```

```
*$CALL GDXXRW.EXE q_0_matrix_tank.xlsm par=q_0 rng=A1:F3
*Parameter q_0(m,k);
*$GDXIN q_0_matrix_tank.gdx
*$LOAD q_0
*display q_0
*$GDXIN

$CALL GDXXRW.EXE qa_0_matrix_tank.xlsm par=qa_0 rng=A1:F2
Parameter qa_0(m,k);
$GDXIN qa_0_matrix_tank.gdx
$LOAD qa_0
display qa_0
$GDXIN

$CALL GDXXRW.EXE qb_0_matrix_tank.xlsm par=qb_0 rng=A1:F2
Parameter qb_0(m,k);
$GDXIN qb_0_matrix_tank.gdx
$LOAD qb_0
display qb_0
$GDXIN

$CALL GDXXRW.EXE Q_matrix_tank.xlsm par=Q_I rng=A1:C3
Parameter Q_I(p,p);
$GDXIN Q_matrix_tank.gdx
$LOAD Q_I
display Q_I
$GDXIN

$CALL GDXXRW.EXE R_matrix_tank.xlsm par=R_I rng=A1:C3
Parameter R_I(m,m);
$GDXIN R_matrix_tank.gdx
$LOAD R_I
display R_I
$GDXIN

$CALL GDXXRW.EXE T_matrix_tank.xlsm par=T_I rng=A1:C3
Parameter T_I(p,p);
$GDXIN T_matrix_tank.gdx
$LOAD T_I
display T_I
$GDXIN

$CALL GDXXRW.EXE P_matrix_tank.xlsm par=P_I rng=A1:E5
Parameter P_I(n,n);
```

```
$GDXIN P_matrix_tank.gdx
$LOAD P_I
display P_I
$GDXIN

$CALL GDXXRW.EXE A_matrix_tank.xlsm par=A rng=A1:E5
Parameter A(n,n);
$GDXIN A_matrix_tank.gdx
$LOAD A
display A
$GDXIN

$CALL GDXXRW.EXE B_matrix_tank.xlsm par=B rng=A1:C5
Parameter B(n,m);
$GDXIN B_matrix_tank.gdx
$LOAD B
display B
$GDXIN

$CALL GDXXRW.EXE K_matrix_tank.xlsm par=K_matrix rng=A1:E3
Parameter K_matrix(m,n);
$GDXIN K_matrix_tank.gdx
$LOAD K_matrix
display K_matrix
$GDXIN

$CALL GDXXRW.EXE L_matrix_tank.xlsm par=L_matrix rng=A1:C3
Parameter L_matrix(m,p);
$GDXIN L_matrix_tank.gdx
$LOAD L_matrix
display L_matrix
$GDXIN

Variables
        x(n,k)
        u(m,k)
        y(p,k)
        xr(n,k)
        ur(m,k)
        r(p,k)

        y_1(p,k)
        y_2(p,k)
        y_3(k,p)
        y_4(k,p)
```

```
        y_value(k,k)
        y_sum

        u_1(m,k)
        u_2(m,k)
        u_3(k,m)
        u_4(k,m)
        u_value(k,k)
        u_sum

        x_1(n,k)
        x_2(n,k)
        x_3(k,n)
        x_4(k,n)
        x_value(k,k)
        x_sum

        h_0(p,k)

        offset_1(p,k)
        offset_2(p,k)
        offset_3(k,p)
        offset_4(k,p)
        offset_value(k,k)
        offset_sum

        V objective function

        const1_t1(n,k)
        const1_t2(n,k)

        const3_t1(m,k)
        const3_t2(m,k)

        h(n,k)
        qa(m,k)
        qb(m,k)   ;
Equation
        ur_value_1(m,k)
        ur_value_2(m,k)

        xr_value_1(n,k)
        xr_value_2(n,k)
        xr_value_3(n,k)
        xr_value_4(n,k)
```

```
equation_y_1(p,k)
equation_y_2(p,k)
equation_y_3(k,p)
equation_y_4(k,p)
equation_y(k,k)
sum_output

equation_u_1(m,k)
equation_u_2(m,k)
equation_u_3(k,m)
equation_u_4(k,m)
equation_u(k,k)
sum_input

equation_x_1(n,k)
equation_x_2(n,k)
equation_x_3(k,n)
equation_x_4(k,n)
equation_x(k,k)
sum_state

level_0_eqn1(p,k)
level_0_eqn2(p,k)

equation_offset_1(p,k)
equation_offset_2(p,k)
equation_offset_3(k,p)
equation_offset_4(k,p)
equation_offset(k,k)
sum_offset

objective_function

Constraint1_1(n,k)
Constraint1_2(n,k)
Constraint1(n,k)

Constraint3_1(m,k)
Constraint3_2(m,k)
Constraint3(m,k)

 x_boundry(n,k)
 u1_boundry(m,k)
 u2_boundry(m,k)  ;
```

```
ur_value_1(m,k).. ur('m1',k)=e=-0.0007*r('p1',k);
ur_value_2(m,k).. ur('m2',k)=e=-0.0008*r('p2',k);

xr_value_1(n,k).. xr('n1',k)=e=r('p1',k);
xr_value_2(n,k).. xr('n2',k)=e=r('p2',k);
xr_value_3(n,k).. xr('n3',k)=e=0.99*r('p1',k);
xr_value_4(n,k).. xr('n4',k)=e=0.69*r('p2',k);

equation_y_1(p,k).. y_1(p,k)=e=y(p,k)-r(p,k);
equation_y_2(p,k).. y_2(p,k)=e=sqr(y_1(p,k));
equation_y_3(k,p).. y_3(k,p)=e=y_2(p,k);
equation_y_4(k,p).. y_4(k,p)=e=sum(p1,(y_3(k,p)*Q_I(p1,p)));
equation_y(k,k).. y_value(k,k)=e=sum(p,(y_4(k,p)*y_2(p,k)));
sum_output.. y_sum=e=sum(k,y_value(k,k));

equation_u_1(m,k).. u_1(m,k)=e=u(m,k)-ur(m,k);
equation_u_2(m,k).. u_2(m,k)=e=sqr(u_1(m,k));
equation_u_3(k,m).. u_3(k,m)=e=u_2(m,k);
equation_u_4(k,m).. u_4(k,m)=e=sum(m1,(u_3(k,m)*R_I(m1,m)));
equation_u(k,k).. u_value(k,k)=e=sum(m,(u_4(k,m)*u_2(m,k)));
sum_input.. u_sum=e=sum(k,u_value(k,k));

equation_x_1(n,k).. x_1(n,k)=e=x(n,k)-xr(n,k);
equation_x_2(n,k).. x_2(n,k)=e=sqr(x_1(n,k));
equation_x_3(k,n).. x_3(k,n)=e=x_2(n,k);
equation_x_4(k,n).. x_4(k,n)=e=sum(n1,(x_3(k,n)*P_I(n1,n)));
equation_x(k,k).. x_value(k,k)=e=sum(n,(x_4(k,n)*x_2(n,k)));
sum_state.. x_sum=e=sum(k,x_value(k,k));

level_0_eqn1(p,k).. h_0('p1',k)=e=h0('n1',k);
level_0_eqn2(p,k).. h_0('p2',k)=e=h0('n2',k);

equation_offset_1(p,k).. offset_1(p,k)=e=r(p,k)+h_0(p,k)-s(p,k);
equation_offset_2(p,k).. offset_2(p,k)=e=sqr(offset_1(p,k));
equation_offset_3(k,p).. offset_3(k,p)=e=offset_2(p,k);
equation_offset_4(k,p).. offset_4(k,p)=e=sum(p1,(offset_3(k,p)*T_I(p,p1)));
equation_offset(k,k).. offset_value(k,k)=e=sum(p,(offset_4(k,p)*offset_2(p,k)));
sum_offset.. offset_sum=e=sum(k,offset_value(k,k));

objective_function.. V=e=y_sum+u_sum+x_sum+offset_sum;

Constraint1_1(n,k).. const1_t1(n,k)=e=sum(n1,A(n,n1)*x(n,k));
Constraint1_2(n,k).. const1_t2(n,k)=e=sum(m,B(n,m)*u(m,k));
Constraint1(n,k).. x(n,k+1)=e=const1_t1(n,k)+const1_t2(n,k);
```

```
Constraint3_1(m,k).. const3_t1(m,k)=e=sum(n,(K_matrix(m,n)*x(n,k)));
Constraint3_2(m,k).. const3_t2(m,k)=e=sum(p,(L_matrix(m,p)*r(p,k)));
Constraint3(m,k).. u(m,k)=e=const3_t1(m,k)+const3_t2(m,k);

x_boundry(n,k).. x(n,k)=e=h(n,k)-h0(n,k);
u1_boundry(m,k).. u('m1',k)=e=qa('m1',k)-qa_0('m1',k);
u2_boundry(m,k).. u('m2',k)=e=qb('m2',k)-qb_0('m2',k);

* Bounds on variables;
h.lo(n,k)=0.2;
h.up('n1',k)=1.36;
h.up('n2',k)=1.36;
h.up('n3',k)=1.3;
h.up('n4',k)=1.3;
qa.up('m1',k)=3.26;
qa.lo('m1',k)=0;
qb.up('m2',k)=4;
qb.lo('m2',k)=0;
*q.up('m1',k)=3.26;
*q.lo('m1',k)=0;
*q.up('m2',k)=4;
*q.lo('m2',k)=0

option optcr = 0.05;

option reslim = 1000000;
option iterlim = 1000000;
option limrow = 1000000;

Model obj_func /all/;
solve obj_func  using nlp minimizing V;

* Send results to Excel
Execute_Unload "result_x.gdx", x;
Execute 'GDXXRW.EXE result_x.gdx var=x rng=sheet1!a1';

Execute_Unload "result_u.gdx", u;
Execute 'GDXXRW.EXE result.gdx var=u rng=sheet1!a1';

Execute_Unload "result_y.gdx", y;
Execute 'GDXXRW.EXE result_y.gdx var=y rng=sheet1!a1';

Execute_Unload "result_xr.gdx", xr;
Execute 'GDXXRW.EXE result_xr.gdx var=xr rng=sheet1!a1';
```

```
Execute_Unload "result_ur.gdx", ur;
Execute 'GDXXRW.EXE result_ur.gdx var=ur rng=sheet1!a1';

Execute_Unload "result_r.gdx", r;
Execute 'GDXXRW.EXE result_r.gdx var=r rng=sheet1!a1';
```

APPENDIX 5

GAMs script of the quadruple-tank process for control problem is shown follows.

```
sets
   k prediction horizon /1*20/;

sets
   n number of states   /n1*n4/
   m number of inputs   /m1*m2/
   p number of outputs  /p1*p2/
   q number of (n+m)    /q1*q6/      ;

Alias(n,n1);
Alias(k,k1);
Alias(m,m1);
Alias(p,p1);
Alias(q,q1);

$CALL GDXXRW.EXE s_setpoint.xlsm par=s rng=A1:U3
Parameter s(p,k);
$GDXIN s_setpoint.gdx
$LOAD s
display s
$GDXIN

$CALL GDXXRW.EXE h_0_matris.xlsm par=h_0 rng=A1:U5
Parameter h_0(n,k);
$GDXIN h_0_matris.gdx
$LOAD h_0
display h_0
$GDXIN

$CALL GDXXRW.EXE Q_unit_matris.xlsm par=Q_I rng=A1:C3
Parameter Q_I (p,p1);
$GDXIN Q_unit_matris.gdx
$LOAD Q_I
display Q_I
$GDXIN

$CALL GDXXRW.EXE R_unit_matris.xlsm par=R_I rng=A1:C3
Parameter R_I (m,m1);
$GDXIN R_unit_matris.gdx
$LOAD R_I
display R_I
$GDXIN

*calculated from riccati equation
$CALL GDXXRW.EXE P_matris.xlsm par=P_I rng=A1:E5
Parameter P_I(n,n1);
$GDXIN P_matris.gdx
```

```
$LOAD P_I
display P_I
$GDXIN

$CALL GDXXRW.EXE T_unit_matris.xlsm par=T_I rng=A1:C3
Parameter T_I (p,p1);
$GDXIN T_unit_matris.gdx
$LOAD T_I
display T_I
$GDXIN

$CALL GDXXRW.EXE qa_s_matris.xlsm par=qa_s rng=A1:U2
Parameter qa_s(m,k);
$GDXIN qa_s_matris.gdx
$LOAD qa_s
display qa_s
$GDXIN

$CALL GDXXRW.EXE qb_s_matris.xlsm par=qb_s rng=A1:U2
Parameter qb_s(m,k);
$GDXIN qb_s_matris.gdx
$LOAD qb_s
display qb_s
$GDXIN

$CALL GDXXRW.EXE K_matris.xlsm par=K_matris rng=A1:E3
Parameter K_matris(m,n);
$GDXIN K_matris.gdx
$LOAD K_matris
display K_matris
$GDXIN
*$offtext

$CALL GDXXRW.EXE L_matris.xlsm par=L_matris rng=A1:C3
Parameter L_matris(m,p);
$GDXIN L_matris.gdx
$LOAD L_matris
display L_matris
$GDXIN

$CALL GDXXRW.EXE mat_A_disc_GAMS.xlsm par=A rng=A1:E5
Parameter A(n,n1);
$GDXIN mat_A_disc_GAMS.gdx
$LOAD A
display A
$GDXIN

$CALL GDXXRW.EXE mat_B_disc_GAMS.xlsm par=B rng=A1:C5
Parameter B(n,m);
$GDXIN mat_B_disc_GAMS.gdx
$LOAD B
display B
$GDXIN

*$CALL GDXXRW.EXE mat_C_disc_GAMS.xlsm par=C rng=A1:E3
*Parameter C(p,n);
*$GDXIN mat_C_disc_GAMS.gdx
*$LOAD C
*display C
*$GDXIN


Variables
        x(n,k)
        u(m,k)
        y(p,k)
        xr(n,k)
```

```
        ur(m,k)
        r(p,k)

        y_1(p,k)
        y_2(p,k)
        y_3(k,p)
        y_4(k,p)
        y_value(k,k)
        y_sum

        u_1(m,k)
        u_2(m,k)
        u_3(k,m)
        u_4(k,m)
        u_value(k,k)
        u_sum

        x_1(n,k)
        x_2(n,k)
        x_3(k,n)
        x_4(k,n)
        x_value(k,k)
        x_sum

        h0(p,k)

        offset_1(p,k)
        offset_2(p,k)
        offset_3(k,p)
        offset_4(k,p)
        offset_value(k,k)
        offset_sum

        V

        const1_t1(n,k)
        const1_t2(n,k)
        const3_t1(m,k)
        const3_t2(m,k)

        h(n,k)
        qa(m,k)
        qb(m,k)  ;

Equations
        ur_value_1(m,k)
        ur_value_2(m,k)

        xr_value_1(n,k)
        xr_value_2(n,k)
        xr_value_3(n,k)
        xr_value_4(n,k)

        equation_y_1(p,k)
        equation_y_2(p,k)
        equation_y_3(k,p)
        equation_y_4(k,p)
        equation_y(k,k)
        sum_output

        equation_u_1(m,k)
        equation_u_2(m,k)
        equation_u_3(k,m)
        equation_u_4(k,m)
        equation_u(k,k)
        sum_input
```

```
        equation_x_1(n,k)
        equation_x_2(n,k)
        equation_x_3(k,n)
        equation_x_4(k,n)
        equation_x(k,k)
        sum_state

        level_0_eqn1(p,k)
        level_0_eqn2(p,k)

        equation_offset_1(p,k)
        equation_offset_2(p,k)
        equation_offset_3(k,p)
        equation_offset_4(k,p)
        equation_offset(k,k)
        sum_offset

        objective_function

        Constraint1_1(n,k)
        Constraint1_2(n,k)
        Constraint1(n,k)

        Constraint3_1(m,k)
        Constraint3_2(m,k)
        Constraint3(m,k)

         x_boundry(n,k)
         u1_boundry(m,k)
         u2_boundry(m,k)  ;

ur_value_1(m,k).. ur('m1',k)=e=-0.0007*r('p1',k);
ur_value_2(m,k).. ur('m2',k)=e=-0.0008*r('p2',k);

xr_value_1(n,k).. xr('n1',k)=e=r('p1',k);
xr_value_2(n,k).. xr('n2',k)=e=r('p2',k);
xr_value_3(n,k).. xr('n3',k)=e=r('p1',k);
xr_value_4(n,k).. xr('n4',k)=e=r('p2',k);

equation_y_1(p,k).. y_1(p,k)=e=y(p,k)-r(p,k);
equation_y_2(p,k).. y_2(p,k)=e=sqr(y_1(p,k));
equation_y_3(k,p).. y_3(k,p)=e=y_2(p,k);
equation_y_4(k,p).. y_4(k,p)=e=sum(p1,(y_3(k,p)*Q_I(p1,p)));
equation_y(k,k).. y_value(k,k)=e=sum(p,(y_4(k,p)*y_2(p,k)));
sum_output.. y_sum=e=sum(k,y_value(k,k));

equation_u_1(m,k).. u_1(m,k)=e=u(m,k)-ur(m,k);
equation_u_2(m,k).. u_2(m,k)=e=sqr(u_1(m,k));
equation_u_3(k,m).. u_3(k,m)=e=u_2(m,k);
equation_u_4(k,m).. u_4(k,m)=e=sum(m1,(u_3(k,m)*R_I(m1,m)));
equation_u(k,k).. u_value(k,k)=e=sum(m,(u_4(k,m)*u_2(m,k)));
sum_input.. u_sum=e=sum(k,u_value(k,k));

equation_x_1(n,k).. x_1(n,k)=e=x(n,k)-xr(n,k);
equation_x_2(n,k).. x_2(n,k)=e=sqr(x_1(n,k));
equation_x_3(k,n).. x_3(k,n)=e=x_2(n,k);
equation_x_4(k,n).. x_4(k,n)=e=sum(n1,(x_3(k,n)*P_I(n1,n)));
equation_x(k,k).. x_value(k,k)=e=sum(n,(x_4(k,n)*x_2(n,k)));
sum_state.. x_sum=e=sum(k,x_value(k,k));

level_0_eqn1(p,k).. h0('p1',k)=e=h_0('n1',k);
level_0_eqn2(p,k).. h0('p2',k)=e=h_0('n2',k);

equation_offset_1(p,k).. offset_1(p,k)=e=r(p,k)+h0(p,k)-s(p,k);
equation_offset_2(p,k).. offset_2(p,k)=e=sqr(offset_1(p,k));
equation_offset_3(k,p).. offset_3(k,p)=e=offset_2(p,k);
equation_offset_4(k,p).. offset_4(k,p)=e=sum(p1,(offset_3(k,p)*T_I(p1,p)));
```

```
equation_offset(k,k).. offset_value(k,k)=e=sum(p,(offset_4(k,p)*offset_2(p,k)));
sum_offset.. offset_sum=e=sum(k,offset_value(k,k));

objective_function.. V=e=y_sum+u_sum+x_sum+offset_sum;

Constraint1_1(n,k).. const1_t1(n,k)=e=sum(n1,A(n,n1)*x(n,k));
Constraint1_2(n,k).. const1_t2(n,k)=e=sum(m,B(n,m)*u(m,k));
Constraint1(n,k).. x(n,k+1)=e=const1_t1(n,k)+const1_t2(n,k);

Constraint3_1(m,k).. const3_t1(m,k)=e=sum(n,(K_matris(m,n)*x(n,k)));
Constraint3_2(m,k).. const3_t2(m,k)=e=sum(p,(L_matris(m,p)*r(p,k)));
Constraint3(m,k).. u(m,k)=e=const3_t1(m,k)+const3_t2(m,k);

x_boundry(n,k).. x(n,k)=e=h(n,k)-h_0(n,k);
u1_boundry(m,k).. u('m1',k)=e=qa('m1',k)-qa_s('m1',k);
u2_boundry(m,k).. u('m2',k)=e=qb('m2',k)-qb_s('m2',k);

* Bounds on variables;
h.lo(n,k)=0.2;
h.up('n1',k)=1.36;
h.up('n2',k)=1.36;
h.up('n3',k)=1.3;
h.up('n4',k)=1.3;
qa.up('m1',k)=3.26;
qa.lo('m1',k)=0;
qb.up('m2',k)=4;
qb.lo('m2',k)=0;

option optcr = 0.05;

option reslim = 1000000;
option iterlim = 1000000;
option limrow = 1000000;

Model obj_func /all/;
solve obj_func  using nlp minimizing V;

* Send results to Excel
Execute_Unload "result_x_reg1.gdx", x;
Execute 'GDXXRW.EXE result_x_reg1.gdx var=x rng=sheet1!a1';

Execute_Unload "result_u_reg1.gdx", u;
Execute 'GDXXRW.EXE result_u_reg1.gdx var=u rng=sheet1!a1';

Execute_Unload "result_y_reg1.gdx", y;
Execute 'GDXXRW.EXE result_y_reg1.gdx var=y rng=sheet1!a1';

Execute_Unload "result_xr_reg1.gdx", xr;
Execute 'GDXXRW.EXE result_xr_reg1.gdx var=xr rng=sheet1!a1';

Execute_Unload "result_ur_reg1.gdx", ur;
Execute 'GDXXRW.EXE result_ur_reg1.gdx var=ur rng=sheet1!a1';

Execute_Unload "result_r_reg1.gdx", r;
Execute 'GDXXRW.EXE result_r_reg1.gdx var=r rng=sheet1!a1';
```

APPENDIX 6

Matlab script file shows to calculate the discrete-time matrices from the linear continuous-time matrices for the CSTR process (for $qc = 98.9\ l/\ min$).

```
%operating conditions qc=98.9 l/min
A = [7.38 2158.99; -0.0468 -11.795];
B = [-0.914 ; 0];
C = [0 1];
D = zeros(1,1);

Ts = 5;                                          %sampling time
sys = ss(A,B,C,D);
[sysd2,G2] = c2d(sys,Ts,'tustin');
```

APPENDIX 7

GAMs script for the CSTR process (for $qc = 98.9\ l/\ min$) is shown follows.

```
sets
   k prediction horizon /1*10/;
sets
   n number of states   /n1*n2/
   m number of inputs   /m1/
   p number of outputs  /p1/   ;

Alias(n,n1);
Alias(k,k1);
Alias(m,m1);
Alias(p,p1);

$CALL GDXXRW.EXE A2_matrix.xlsm par=A rng=A1:C3
Parameter A(n,n1);
$GDXIN A2_matrix.gdx
$LOAD A
display A
$GDXIN

$CALL GDXXRW.EXE B2_matrix.xlsm par=B rng=A1:B3
Parameter B(n,m);
$GDXIN B2_matrix.gdx
$LOAD B
display B
$GDXIN

*$CALL GDXXRW.EXE C_matrix.xlsm par=C rng=A1:C2
*Parameter C(p,n);
*$GDXIN C_matrix.gdx
*$LOAD C
*display C
*$GDXIN

$CALL GDXXRW.EXE Q_unit_matris.xlsm par=Q_I rng=A1:B2
Parameter Q_I (p,p1);
$GDXIN Q_unit_matris.gdx
$LOAD Q_I
display Q_I
$GDXIN
```

```
$CALL GDXXRW.EXE R_unit_matris.xlsm par=R_I rng=A1:B2
Parameter R_I (m,m1);
$GDXIN R_unit_matris.gdx
$LOAD R_I
display R_I
$GDXIN

$CALL GDXXRW.EXE D_matris.xlsm par=D rng=A1:B2
Parameter D (m,m1);
$GDXIN D_matris.gdx
$LOAD D
display D
$GDXIN

$CALL GDXXRW.EXE qc_0_matris.xlsm par=qc_0 rng=A1:K2
Parameter qc_0(m,k);
$GDXIN qc_0_matris.gdx
$LOAD qc_0
display qc_0
$GDXIN

$CALL GDXXRW.EXE T_0_matris.xlsm par=T_0 rng=A1:K2
Parameter T_0(n,k);
$GDXIN T_0_matris.gdx
$LOAD T_0
display T_0
$GDXIN

$CALL GDXXRW.EXE C_0_matris.xlsm par=C_0 rng=A1:K2
Parameter C_0(n,k);
$GDXIN C_0_matris.gdx
$LOAD C_0
display C_0
$GDXIN

Variables
        x(n,k)
        u(m,k)
        y(p,k)
        r(p,k)
        Du(m,k)
        u_0(m,k)

        y_1(p,k)
        y_2(k,p)
        y_3(k,p)
        y_value(k,k)
        y_sum

        Du_term1(m,k)

        Du_1(k,m)
        Du_2(k,m)
        Du_value(k,k)
        Du_sum

        J

        const1_t1(n,k)
        const1_t2(n,k)


        qc(m,k)
        T(n,k)
        C(n,k)   ;
```

```
Equations

        equation_y_1(p,k)
        equation_y_2(k,p)
        equation_y_3(k,p)
        equation_y(k,k)
        sum_output

        equation1_delta_u(m,k)
        delta_u(m,k)

        equation_Du_1(k,m)
        equation_Du_2(k,m)
        equation_Du(k,k)
        sum_input

        objective_function

        Constraint1_1(n,k)
        Constraint1_2(n,k)
        Constraint1(n,k)

        u_boundry(m,k)
        x1_boundry(n,k)
        x2_boundry(n,k)  ;

equation_y_1(p,k).. y_1(p,k)=e=y(p,k)-r(p,k);
equation_y_2(k,p).. y_2(k,p)=e=y_1(p,k);
equation_y_3(k,p).. y_3(k,p)=e=sum(p1,(y_2(k,p)*Q_I(p1,p)));
equation_y(k,k).. y_value(k,k)=e=sum(p,(y_3(k,p)*y_1(p,k)));
sum_output.. y_sum=e=sum(k,y_value(k,k));

equation1_delta_u(m,k).. Du_term1(m,k)=e=sum(m1,D(m,m1)*u(m1,k));
delta_u(m,k).. Du(m,k)=e=Du_term1(m,k)-u_0(m,k);

equation_Du_1(k,m).. Du_1(k,m)=e=Du(m,k);
equation_Du_2(k,m).. Du_2(k,m)=e=sum(m1,(Du_1(k,m)*R_I(m1,m)));
equation_Du(k,k).. Du_value(k,k)=e=sum(m,(Du_2(k,m)*Du(m,k)));
sum_input.. Du_sum=e=sum(k,Du_value(k,k));

objective_function.. J=e=y_sum+Du_sum ;

Constraint1_1(n,k).. const1_t1(n,k)=e=sum(n1,A(n,n1)*x(n,k));
Constraint1_2(n,k).. const1_t2(n,k)=e=sum(m,B(n,m)*u(m,k));
Constraint1(n,k).. x(n,k+1)=e=const1_t1(n,k)+const1_t2(n,k);

u_boundry(m,k).. u(m,k)=e=qc(m,k)-qc_0(m,k);
x1_boundry(n,k).. x('n1',k)=e=T('n1',k)-T_0('n1',k);
x2_boundry(n,k).. x('n2',k)=e=C('n2',k)-C_0('n2',k);

qc.up('m1',k)=150;
qc.lo('m1',k)=0;
T.up('n1',k)=700;
T.lo('n1',k)=350;
C.up('n2',k)=0.1;
C.lo('n2',k)=0.0001;

option optcr = 0.05;

option reslim = 1000000;
option iterlim = 1000000;
option limrow = 1000000;

Model obj_func /all/;
solve obj_func  using nlp minimizing J;

* Send results to Excel
```

```
Execute_Unload "result_x2.gdx", x;
Execute 'GDXXRW.EXE result_x2.gdx var=x rng=sheet1!a1';

Execute_Unload "result_u2.gdx", u;
Execute 'GDXXRW.EXE result_u2.gdx var=u rng=sheet1!a1';

Execute_Unload "result_y2.gdx", y;
Execute 'GDXXRW.EXE result_y2.gdx var=y rng=sheet1!a1';

Execute_Unload "result_Du2.gdx", Du;
Execute 'GDXXRW.EXE result_Du2.gdx var=Du rng=sheet1!a1';

Execute_Unload "result_r2.gdx", r;
Execute 'GDXXRW.EXE result_r2.gdx var=r rng=sheet1!a1';
```

APPENDIX 8

GAMs script of the CSTR process with disturbance (for $qc = 98.9\ l/\ min$ and $qc = 108.8\ l/\ min$) is shown follows.

```
sets
   k prediction horizon /0*10/;
sets
   n number of states   /n1*n2/
   m number of inputs   /m1/
   p number of outputs  /p1/ ;

Alias(n,n1);
Alias(k,k1);
Alias(m,m1);
Alias(p,p1);

$CALL GDXXRW.EXE A1_matrix.xlsm par=A1 rng=A1:C3
Parameter A1(n,n1);
$GDXIN A1_matrix.gdx
$LOAD A1
display A1
$GDXIN

$CALL GDXXRW.EXE B1_matrix.xlsm par=B1 rng=A1:B3
Parameter B1(n,m);
$GDXIN B1_matrix.gdx
$LOAD B1
display B1
$GDXIN

$CALL GDXXRW.EXE A2_matrix.xlsm par=A2 rng=A1:C3
Parameter A2(n,n1);
$GDXIN A2_matrix.gdx
$LOAD A2
display A2
$GDXIN

$CALL GDXXRW.EXE B2_matrix.xlsm par=B2 rng=A1:B3
Parameter B2(n,m);
$GDXIN B2_matrix.gdx
$LOAD B2
display B2
$GDXIN

*$CALL GDXXRW.EXE C_matrix.xlsm par=C rng=A1:C2
*Parameter C(p,n);
*$GDXIN C_matrix.gdx
```

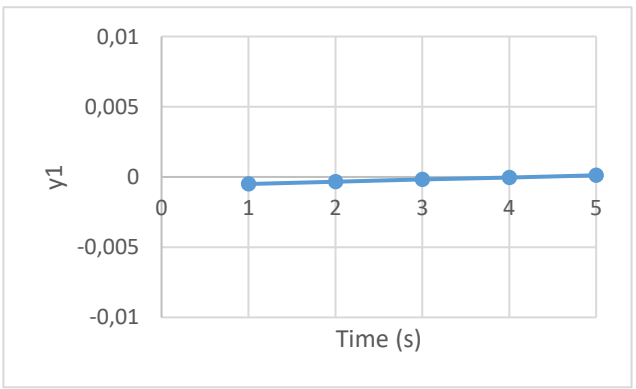
```
*$LOAD C
*display C
*$GDXIN

$CALL GDXXRW.EXE Q_unit_matris.xlsm par=Q_I rng=A1:B2
Parameter Q_I (p,p1);
$GDXIN Q_unit_matris.gdx
$LOAD Q_I
display Q_I
$GDXIN

$CALL GDXXRW.EXE R_unit_matris.xlsm par=R_I rng=A1:B2
Parameter R_I (m,m1);
$GDXIN R_unit_matris.gdx
$LOAD R_I
display R_I
$GDXIN

$CALL GDXXRW.EXE D_matris.xlsm par=D rng=A1:B2
Parameter D (m,m1);
$GDXIN D_matris.gdx
$LOAD D
display D
$GDXIN

$CALL GDXXRW.EXE qc_0_matris.xlsm par=qc_0 rng=A1:P2
Parameter qc_0(m,k);
$GDXIN qc_0_matris.gdx
$LOAD qc_0
display qc_0
$GDXIN

$CALL GDXXRW.EXE T_0_matris.xlsm par=T_0 rng=A1:P2
Parameter T_0(n,k);
$GDXIN T_0_matris.gdx
$LOAD T_0
display T_0
$GDXIN

$CALL GDXXRW.EXE C_0_matris.xlsm par=C_0 rng=A1:P2
Parameter C_0(n,k);
$GDXIN C_0_matris.gdx
$LOAD C_0
display C_0
$GDXIN

Variables
        x(n,k)
        u(m,k)
        y(p,k)
        r(p,k)
        Du(m,k)
        u_0(m,k)

        y_1(p,k)
        y_2(k,p)
        y_3(k,p)
        y_value(k,k)
        y_sum

        Du_term1(m,k)

        Du_1(k,m)
        Du_2(k,m)
        Du_value(k,k)
        Du_sum
```

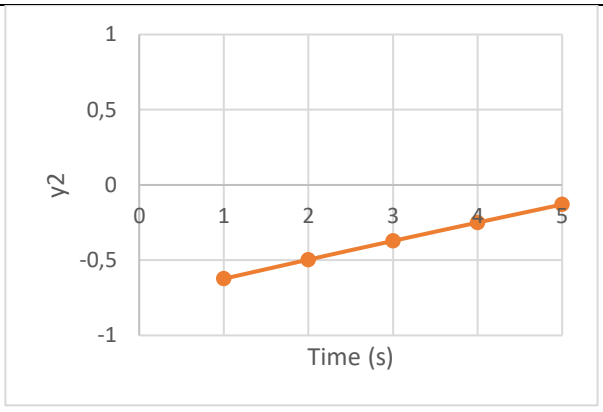
```
        J

        const1_t11(n,k)
        const1_t12(n,k)
        const1_t13(n,k)
        const1_t14(n,k)
        const1_t15(n,k)

        const1_t21(n,k)
        const1_t22(n,k)
        const1_t23(n,k)
        const1_t24(n,k)
        const1_t25(n,k)

        const1_t16(n,k)
        const1_t17(n,k)
        const1_t18(n,k)
        const1_t19(n,k)
        const1_t20(n,k)

        const1_t26(n,k)
        const1_t27(n,k)
        const1_t28(n,k)
        const1_t29(n,k)
        const1_t30(n,k)

        qc(m,k)
        T(n,k)
        C(n,k)  ;

Equations

        equation_y_1(p,k)
        equation_y_2(k,p)
        equation_y_3(k,p)
        equation_y(k,k)
        sum_output

        equation1_delta_u(m,k)
        delta_u(m,k)

        equation_Du_1(k,m)
        equation_Du_2(k,m)
        equation_Du(k,k)
        sum_input

        objective_function

        Constraint1_11(n,k)
        Constraint1_12(n,k)
        Constraint1_13(n,k)
        Constraint1_14(n,k)
        Constraint1_15(n,k)

        Constraint1_21(n,k)
        Constraint1_22(n,k)
        Constraint1_23(n,k)
        Constraint1_24(n,k)
        Constraint1_25(n,k)

        Constraint1_16(n,k)
        Constraint1_17(n,k)
        Constraint1_18(n,k)
        Constraint1_19(n,k)
        Constraint1_20(n,k)

        Constraint1_26(n,k)
```

```
        Constraint1_27(n,k)
        Constraint1_28(n,k)
        Constraint1_29(n,k)
        Constraint1_30(n,k)

        Constraint11(n,k)
        Constraint12(n,k)
        Constraint13(n,k)
        Constraint14(n,k)
        Constraint15(n,k)

        Constraint16(n,k)
        Constraint17(n,k)
        Constraint18(n,k)
        Constraint19(n,k)
        Constraint20(n,k)

        u_boundry(m,k)
        x1_boundry(n,k)
        x2_boundry(n,k) ;

equation_y_1(p,k).. y_1(p,k)=e=y(p,k)-r(p,k);
equation_y_2(k,p).. y_2(k,p)=e=y_1(p,k);
equation_y_3(k,p).. y_3(k,p)=e=sum(p1,(y_2(k,p)*Q_I(p1,p)));
equation_y(k,k).. y_value(k,k)=e=sum(p,(y_3(k,p)*y_1(p,k)));
sum_output.. y_sum=e=sum(k,y_value(k,k));

equation1_delta_u(m,k).. Du_term1(m,k)=e=sum(m1,D(m,m1)*u(m1,k));
delta_u(m,k).. Du(m,k)=e=Du_term1(m,k)-u_0(m,k);

equation_Du_1(k,m).. Du_1(k,m)=e=Du(m,k);
equation_Du_2(k,m).. Du_2(k,m)=e=sum(m1,(Du_1(k,m)*R_I(m1,m)));
equation_Du(k,k).. Du_value(k,k)=e=sum(m,(Du_2(k,m)*Du(m,k)));
sum_input.. Du_sum=e=sum(k,Du_value(k,k));

objective_function.. J=e=y_sum+Du_sum ;

Constraint1_11(n,k).. const1_t11(n,'1')=e=sum(n1,A2(n,n1)*x(n,'1'));
Constraint1_12(n,k).. const1_t12(n,'2')=e=sum(n1,A2(n,n1)*x(n,'2'));
Constraint1_13(n,k).. const1_t13(n,'3')=e=sum(n1,A2(n,n1)*x(n,'3'));
Constraint1_14(n,k).. const1_t14(n,'4')=e=sum(n1,A2(n,n1)*x(n,'4'));
Constraint1_15(n,k).. const1_t15(n,'5')=e=sum(n1,A2(n,n1)*x(n,'5'));

Constraint1_21(n,k).. const1_t21(n,'1')=e=sum(m,B2(n,m)*u(m,'1'));
Constraint1_22(n,k).. const1_t22(n,'2')=e=sum(m,B2(n,m)*u(m,'2'));
Constraint1_23(n,k).. const1_t23(n,'3')=e=sum(m,B2(n,m)*u(m,'3'));
Constraint1_24(n,k).. const1_t24(n,'4')=e=sum(m,B2(n,m)*u(m,'4'));
Constraint1_25(n,k).. const1_t25(n,'5')=e=sum(m,B2(n,m)*u(m,'5'));

Constraint1_16(n,k).. const1_t16(n,'6')=e=sum(n1,A1(n,n1)*x(n,'6'));
Constraint1_17(n,k).. const1_t17(n,'7')=e=sum(n1,A1(n,n1)*x(n,'7'));
Constraint1_18(n,k).. const1_t18(n,'8')=e=sum(n1,A1(n,n1)*x(n,'8'));
Constraint1_19(n,k).. const1_t19(n,'9')=e=sum(n1,A1(n,n1)*x(n,'9'));
Constraint1_20(n,k).. const1_t20(n,'10')=e=sum(n1,A1(n,n1)*x(n,'10'));

Constraint1_26(n,k).. const1_t26(n,'6')=e=sum(m,B1(n,m)*u(m,'6'));
Constraint1_27(n,k).. const1_t27(n,'7')=e=sum(m,B1(n,m)*u(m,'7'));
Constraint1_28(n,k).. const1_t28(n,'8')=e=sum(m,B1(n,m)*u(m,'8'));
Constraint1_29(n,k).. const1_t29(n,'9')=e=sum(m,B1(n,m)*u(m,'9'));
Constraint1_30(n,k).. const1_t30(n,'10')=e=sum(m,B1(n,m)*u(m,'10'));

Constraint11(n,k).. x(n,k+1)=e=const1_t11(n,'1')+const1_t21(n,'1');
Constraint12(n,k).. x(n,k+1)=e=const1_t12(n,'2')+const1_t22(n,'2');
Constraint13(n,k).. x(n,k+1)=e=const1_t13(n,'3')+const1_t23(n,'3');
Constraint14(n,k).. x(n,k+1)=e=const1_t14(n,'4')+const1_t24(n,'4');
Constraint15(n,k).. x(n,k+1)=e=const1_t15(n,'5')+const1_t25(n,'5');
```

```
Constraint16(n,k).. x(n,k+1)=e=const1_t16(n,'6')+const1_t26(n,'6');
Constraint17(n,k).. x(n,k+1)=e=const1_t17(n,'7')+const1_t27(n,'7');
Constraint18(n,k).. x(n,k+1)=e=const1_t18(n,'8')+const1_t28(n,'8');
Constraint19(n,k).. x(n,k+1)=e=const1_t19(n,'9')+const1_t29(n,'9');
Constraint20(n,k).. x(n,k+1)=e=const1_t20(n,'10')+const1_t30(n,'10');

u_boundry(m,k).. u(m,k)=e=qc(m,k)-qc_0(m,k);
x1_boundry(n,k).. x('n1',k)=e=T('n1',k)-T_0('n1',k);
x2_boundry(n,k).. x('n2',k)=e=C('n2',k)-C_0('n2',k);

qc.up('m1',k)=110;
qc.lo('m1',k)=0;
T.up('n1',k)=500;
T.lo('n1',k)=350;
C.up('n2',k)=0.1;
C.lo('n2',k)=0.01;

option optcr = 0.05;

option reslim = 1000000;
option iterlim = 1000000;
option limrow = 1000000;

Model obj_func /all/;
solve obj_func  using nlp minimizing J;

* Send results to Excel
Execute_Unload "result_x2.gdx", x;
Execute 'GDXXRW.EXE result_x2.gdx var=x rng=sheet1!a1';

Execute_Unload "result_u2.gdx", u;
Execute 'GDXXRW.EXE result_u2.gdx var=u rng=sheet1!a1';

Execute_Unload "result_y2.gdx", y;
Execute 'GDXXRW.EXE result_y2.gdx var=y rng=sheet1!a1';

Execute_Unload "result_Du2.gdx", Du;
Execute 'GDXXRW.EXE result_Du2.gdx var=Du rng=sheet1!a1';

Execute_Unload "result_r2.gdx", r;
Execute 'GDXXRW.EXE result_r2.gdx var=r rng=sheet1!a1';
```

APPENDIX 9





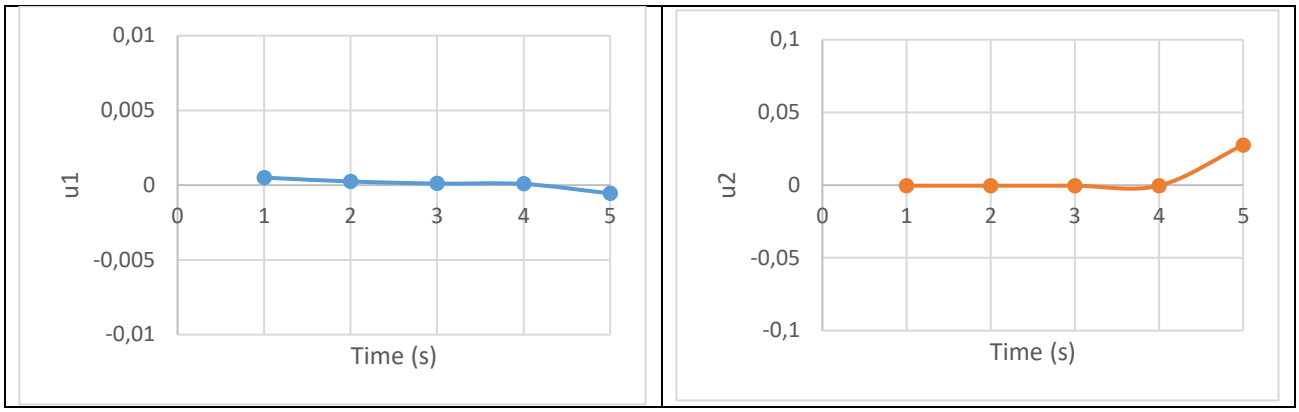


Figure-8-1 The evaluation of quadruple tank process for the conditions of the second control problem (set point 0.3)

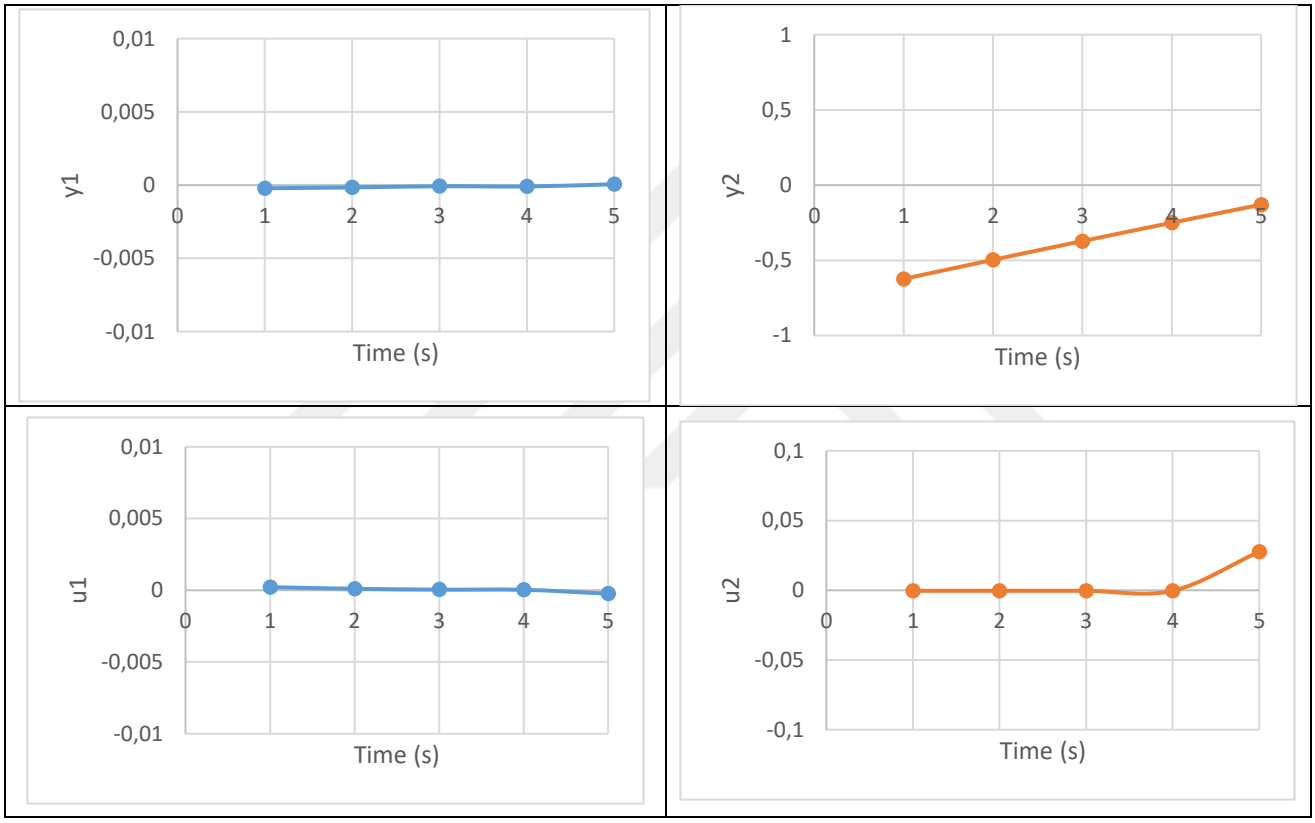
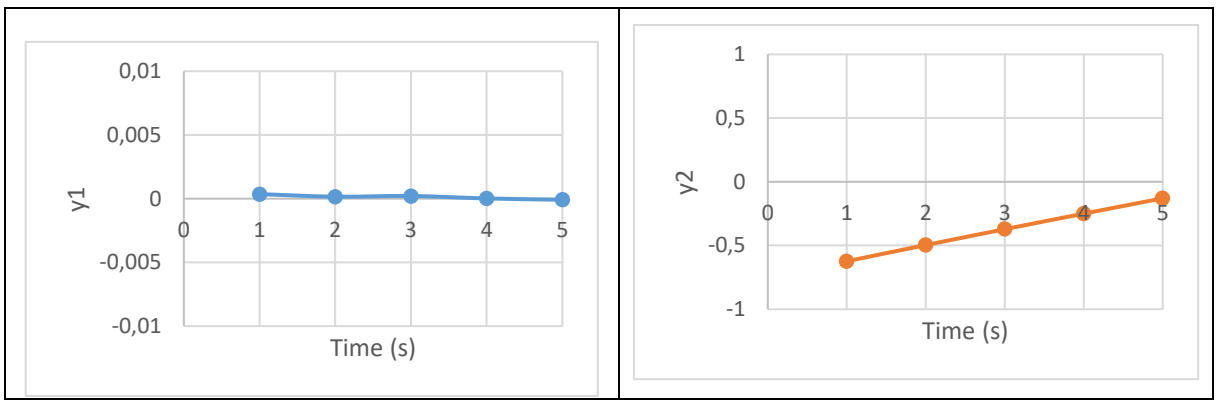


Figure-8-2 The evaluation of quadruple tank process for the conditions of the third control problem (set point 0.5 &0.75)



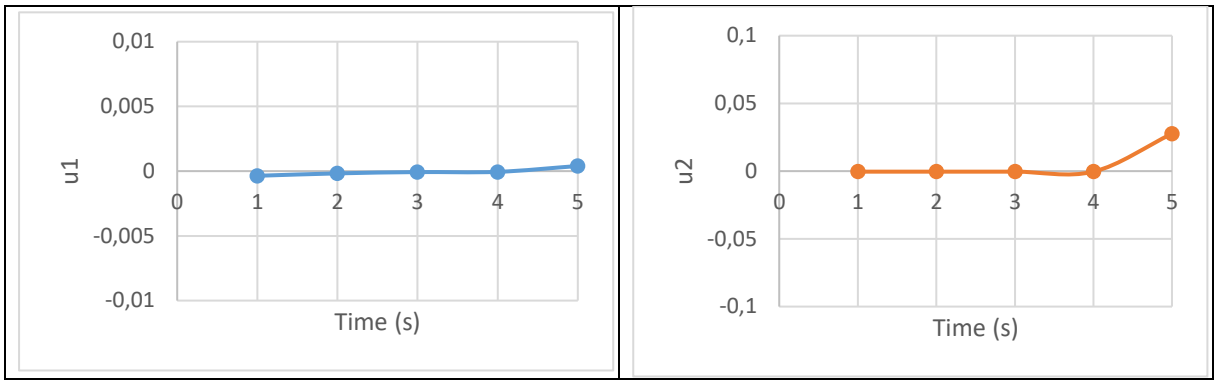


Figure-8-3 The evaluation of quadruple tank process for the conditions of the forth control problem (set point 0.9 &0.75)