RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'EDUCATION SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE
ÉCOLE NATIONALE POLYTECHNIQUE

Département d'Automatique

End of Studies Project

For the attainment of an Engineer degree in Control Engineering

---

Fuzzy Predictive Control of the Coupled Tanks Process

GUERAZEM Said & BENSLIMANE Tarik

Publicly presented and defended on (01/07/2024)

Composition of the jury:

President:	Dr. CHAKIR Messaoud	ENP
Examiner:	Pr. TADJINE Mohamed	ENP
Supervisor:	Pr. Djamel BOUKHETALA	ENP
Supervisor:	Dr. Hakim ACHOUR	ENP

ENP 2024

Control Engineering Department / ENP, 10 Rue des Frères OUDEK, El Harrach 16200 Alger Algérie.

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'EDUCATION SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE
ÉCOLE NATIONALE POLYTECHNIQUE

Département d'Automatique

End of Studies Project

For the attainment of an Engineer degree in Control Engineering

---

Fuzzy Predictive Control of the Coupled Tanks Process

GUERAZEM Said & BENSLIMANE Tarik

Publicly presented and defended on (01/07/2024)

Composition of the jury:

President:	Dr. CHAKIR Messaoud	ENP
Examiner:	Pr. TADJINE Mohamed	ENP
Supervisor:	Pr. Djamel BOUKHETALA	ENP
Supervisor:	Dr. Hakim ACHOUR	ENP

ENP 2024

Control Engineering Department / ENP, 10 Rue des Frères OUDEK, El Harrach 16200 Alger Algérie.

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'EDUCATION SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE
ÉCOLE NATIONALE POLYTECHNIQUE

Département d'Automatique

Mémoire de projet de fin d'études

pour l'obtention du diplôme d'Ingénieur d'État en Automatique

---

Commande Prédictive Floue d'un système à Quatre réservoires Couplés

---

GUERAZEM Said & BENSLIMANE Tarik

Présenté et soutenue publiquement le (01/07/2024)

Membres du jury:

Président:	Dr. CHAKIR Messaoud	ENP
Examinateur:	Pr. TADJINE Mohamed	ENP
Promoteur:	Pr. Djamel BOUKHETALA	ENP
Promoteur:	Dr. Hakim ACHOUR	ENP

ENP 2024

Département d'Automatique / ENP, 10 Rue des Frères OUDEK, El Harrach 16200 Alger Algérie.

الملخص

الهدف الرئيسي من هذا مشروع نهاية الدراسة هو التحكم في مستوى أنظمة الخزانات المترابطة. نبدأ بعرض المواد التعليمية المتاحة في مختبرنا قسم التحكم الآلي، التي تصف النظام المعني، ثم نطور نموذجًا رياضيًا تحليليًا للنظام المعني من خلاله تم وصف الخصائص الديناميكية للنظام. تم اعتماد منظم PI لامركزي، حيث تم تحسين تعديل معاييره بواسطة خوارزمية PSO. تم أخذ هذه التقنية كمرجع للمقارنة مع التقنيات التي طورناها لاحقًا. أولاً، نطور تقنيتين للتحكم التنبئي النموذجي (MPC) الخطي وغير الخطي. بالإضافة إلى ذلك، نقترح نهجًا آخر يعتمد على الشبكات العصبية التكرارية (RNN) للتنبؤ بمدخلات التحكم وتقليل وقت الحساب مقارنة بالطرق التقليدية. أخيرًا، استخدمنا أنظمة غامضة من نوع تاكاجي-سوجينو لوصف النموذج غير الخطي للتحكم متعدد النماذج مع تحليل الاستقرار وتتبع المسار. تم إجراء اختبارات الصلابة لتقييم أداء كل طريقة.

الكلمات المفتاحية: عملية الخزان الرباعي - التحكم PI اللامركزي - التحكم التنبئي النموذجي (MPC) - الشبكات العصبية التكرارية (RNNs) - نماذج تاكاجي-سوجينو الغامضة - التحكم PDC - متباينة المصفوفة الخطية (LMI) - تحسين سرب الجسيمات (PSO).

Résumé

L'Objectif principal de ce projet de fin d'études concerne la régulation de niveau dans un système à 04 réservoirs couplés. Nous commençons par une description du benchmark disponible au niveau du laboratoire du département d'automatiquen. Ensuite, nous élaborons un modèle mathématique analytique du système en question à travers lequel la description des caractéristiques dynamiques du système a été effectuée. Un régulateur PI décentralisé a été adopté dont l'ajustement de ses paramètres a été optimisé par l'algorithme PSO, cette technique de commande est prise comme référence de comparaison avec les techniques que nous avons développés par la suite. En premier lieu, nous développons deux techniques de commande prédictives (MPC) linéaire et non linéaire. Ensuite, nous proposons une autre approche se basant sur les réseaux de neurones récurrents (RNN) afin de prédire les entrées de commande et réduire le temps de calcul par rapport aux méthodes classiques. Enfin, nous avons utilisé les systèmes flous de type Takagi-Sugeno dans le but de décrire le modèle non-linéaire en vue d'une commande multi-modèle avec analyse de stabilité en poursuite de trajectoire. Des tests de robustesse ont été effectués pour évaluer les performances de chaque méthode.

Mots clés : - systèmes à 04 réservoirs couplés -commande PI décentralisée - Commande prédictive - réseaux de neurones récurrents (RNR) - Modèles flous de Takagi-Sugeno- Commande PDC - inégalité matricielle linéaire (IML) - Algorithme d'Optimisation par Essaim Particulaire (PSO) .

Abstract

The main objective of this End of Studies project concerns the level control of the coupled tank systems. We begin with a description of the benchmark available in the laboratory of the Control Engineering Department, then we develop an analytical mathematical model of the system in question through which the description of the dynamic characteristics of the system has been carried out. A decentralised PI controller was adopted whilst the adjustment of its parameters has been carried out using a PSO algorithm. This control technique was taken as a reference for comparison with the techniques that we developed subsequently. Firstly, we develop two control techniques , linear and non-linear model predictive control (MPC) techniques. Additionally, we propose another approach based on recurrent neural networks (RNN) to predict the control inputs and reduce the computation time compared with conventional methods. Finally, we used Takagi-Sugeno type fuzzy systems to describe the non-linear model for multi-model control with stability analysis and trajectory tracking. Robustness tests have been carried out to evaluate the performance of each method.

Keywords : Quadruple-Tank process- decentralized PI control -Model Predictive Control (MPC) -Recurrent Neural Networks (RNNs) - Takagi-Sugeno fuzzy models -PDC control - linear matrix inequality (LMI) - Particle Swarm Optimization (PSO).

Dedication

I never imagined that this day would come, where I would be writing this passage to leave a place I considered home for more than 5 years. These years taught me the meaning of failure, success, and growing up as a man, as I arrived here as a curious child wanting nothing but a glimmer of knowledge to feed my mind.

After this notable success, I would like to thank Allah, the best of planners, for providing me with the privilege of reaching this point in life, as a soldier of science, giving me a lifetime benefit of knowledge. I dedicate this work to my parents, for all their support from the start of my career to this present day. In the days when I saw nothing but darkness, I used your light as a candle to guide myself. I could never express how much gratitude I have for you!

My dedication extends to my five brothers and sisters: Tahar, Abdelrezzak, Khadija, Meriem, and Hamza. You were there for me when I felt like giving up and brought happiness to my life. I wish you all the best in making your aspirations come true. From the Ultramarathons all the way to the chess championships.

They say that your experiences are a mixture of all the encounters you have had. Speaking for myself, I think that making this work wouldn't have been possible without the support of my close friends, the Chakawi group, which started mainly for a scholarly task and ended up being one of the best blessings I could ask for in the last three years. Amine, Zaki, Abdelhak, Aghiles, and Yasser, you guys made me a better person, a happier one for sure, and I can't express how grateful I am while writing this. Pushing each other in the right direction, challenging ourselves like never before, and not forgetting to ease our hearts from time to time has made the level I have today the result of this encounter and much more. I will never forget the long, long nights either drowning with robots or winning a competition.

I would like to extend my thanks to the VIC, the IEEE Student branch and my committee friends: Mehdi, Abdou, Said, Bouchra, Sabrina and Feriel for making me feel more at home and letting me spread my wings after building up a wall around myself. A special dedication to Imed Ascem and Vniverse in the faraway land. The long nights will surely be remembered for ages, and I wish I could turn back time to relive them again.

Still in awe whilst writing this, living it like a dream or maybe a nightmare, they say that we shouldn't be sad that it's finished but should be happy that it happened. My final word is: Thank you Polytech for making me an Engineer!

Said

Dedication

A page in my life is being turned as I write this message after my three years at this prestigious school, which has seen us grow a little more each day, with all the challenges we have overcome and the failures that have contributed to our personal development.

I would like to start by thanking our creator Allah, who has guided me towards the path of knowledge and given me the strength, health and intelligence I needed to move forward, and who has never failed in his choices, illuminating my path and showing me the right way

I dedicate this modest work to my parents, without whom I would never have reached my goal, who guided me throughout my childhood, who supported me in good times and bad and who helped me grow up to become a man, there are no words that could express what I feel towards you

my dedication goes to my 02 brothers Rafik and Aris to whom I am very close and complicit which simply brings me every day joy and good humour and illuminates my life. I wish you all the best in your life. I dedicate it to all the members of my family, especially Uncle Mourad whom I consider to be like my father.

Finally, I dedicate this work to the whole family of the ENP, more precisely the department of automatic control and my dear comrades who have contributed a lot to my development.As they say, "the art of success consists in knowing how to surround yourself with the best people".

I can't end my thanks without thinking kindly of all the people who have supported me, if only with a smile. who have supported me, if only with a smile. THANK YOU FOR EVERYTHING

Tarik

Acknowledgements

We would like to thank everyone who contributed to the success of our study and who helped us during the writing of this thesis.

First, we would like to thank our thesis supervisors, Dr. ACHOUR and Pr. BOUKHETALA from the National Polytechnic School, for their availability and their advice, which helped sharpen our thinking.

We also thank the entire teaching team of the Polytechnic School, especially our dear professors who have greatly contributed to our development throughout our academic journey from preparatory class to our current institution.

Finally, We also want to thank the jury for reading this dissertation.

Contents

List of Figures

List of Tables

Acronyms

- **PI** : Proportional-Integral
- **PSO** : Particle Swarm Optimization
- **MPC** : Model Predictive Control
- **RHC** : Receding Horizon Control
- **LMPC** : Linear Model Predictive Controller
- **NMPC** : Nonlinear Model Predictive Controller
- **ANN** : Artificial Neural Network
- **RNN** : Recurrent Neural Network
- **BPTT** : Back Propagation Through Time
- **MIMO** : Multiple-Input Multiple-Output
- **QTP** : Quadruple Tank Process
- **SMC** : Sliding Mode Control
- **LMI** : Linear Matrix Inequalities
- **DTIS** :Discrete Time-Invariant Systems
- **OCP**: Optimal Control Problem
- **QP**: Quadratic Programming
- **T-S**: Takagi Sugeno
- **PDC**: Parallel distributed compensation

Chapter 1

Introduction

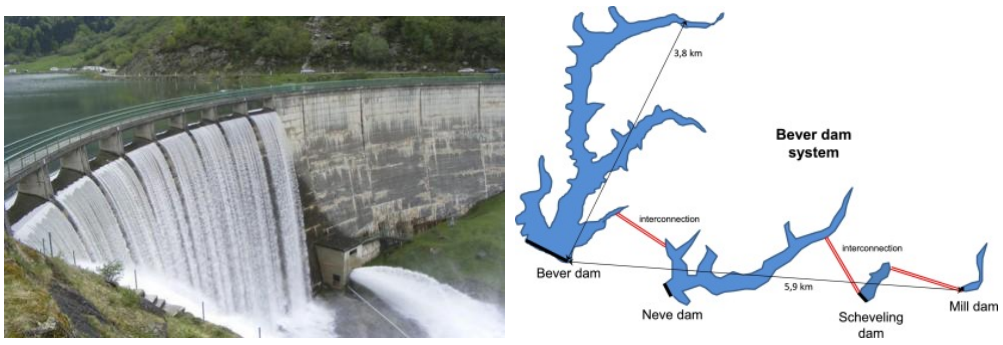
Motivation

Control theory applications are widely spread throughout our daily lives. In particular, process control is a traditional process engineering field that holds a great practical importance [28]. Usually it involves dynamic modelling, identification, diagnosis, etc [28] and it's mainly based on a linear controller, specifically a PID controller.

PID controllers have demonstrated significant effectiveness in regulating both linear and nonlinear systems. Their widespread adoption is attributed to their versatility and reliable performance across a diverse range of operating conditions. Furthermore, there exists a broad conceptual comprehension of the impact of the three terms P, I and D among control engineers, facilitating relatively straightforward manual tuning methods such as the Ziegler-Nichols rule (Ziegler & Nichols, 1942)[42]. Of course, from the theoretical point of view, the PID is used around a single specific operating point. If the operating region is small, we can effectively use this type of controller on a nonlinear process whilst modelling the system's inherent nonlinearities as model uncertainties[56]. However, most of real industrial processes exhibit intensive nonlinear characteristics that makes the operating region really large, thus making usage of the mentioned controller nearly impossible. It requires then different control techniques that could eventually make up for the locality problem of the PID while giving the best possible performances.

The implementation becomes even more complex when coupling is introduced in these systems. It is usually the case of most industrial MIMO systems where the outputs are dependant of the process's states. The problem is discussed first as a stability problem such as in [22] then it deals with reference tracking, observation and perturbation rejection problems [24].

Another type of highly spread nonlinear processes are interconnected large scale systems, these systems are characterised by a large numbers of variables, structure of interconnected subsystems, and other features that complicate the control models (time delay, uncertainties..etc) [29]. This notion could be introduced in many applications and surely one of them is the modelling of interconnected DAM systems [33].



Interconnected Dam System [57]

A Dam is a barrier that holds the flow of water in a certain place. It constitutes then a reservoir or a tank that is connected with a series of other tanks in a given region. Controlling the levels of such gigantic systems requires using a smaller version that could reproduce to a certain extent the behaviour of such systems. This is where the **Four Tanks coupled system** entered the scientific stage not only as a benchmark for nonlinear processes but also for large scale interconnected systems. Applying control techniques on this system might help us deal with larger plants, but we should deal with the controlling problem of the QTP first.

Dealing with the quadruple Tank process have been researched extensively by the scientific community. In this context, many control laws have been developed, the first one being the decentralized PI controller. The latter presented by the benchmark paper was proven to have some good performances. Additionally, the process was also used to provide a good testing platform for observers with their different types. This could be seen through classical high gain observers and even sliding mode observers. Intelligent modelling through Support Vector regressor have also been applied. Thus, the applications on this process are very diverse and different.

Another method that has shown a great impact among control engineers in the industry is **Predictive Control** and particularly **Model Predictive Control**. The latter has proven its effectiveness in the field while obtaining relatively good performances [28] i.e. MPC can deal with highly nonlinear and discontinuous plants as well as constraints on the inputs and the outputs. As its trend began in the late 90s, we're going to dedicate a whole chapter on this part solely demonstrating whether or not it will help us in dealing with our problem.

Dealing with nonlinear systems and their problems has also extended to the modelling problem. Can we effectively model our plant and consider all the nonlinearities dealing with it? This question was the main research topic of Tomohiro **Takagi** and Michio **Sugeno** in [51] as they have introduced a new benchmark in **Fuzzy Control Theory**. We will also dedicate a chapter to fuzzy control and we will try to show its advantages compared to other methods.

Thesis Scope

In this study, we will try to investigate different control techniques applied to the four tanks coupled system. We will start by introducing the system in question provided by National instruments available at the LCP laboratory in ENP. The system is considered adequate for educational purposes as it gives multiple options that we will explain further in this thesis. Then we will start by implementing a state of the art decentralised PI controller as a benchmark for the next implemented control strategies where we will be comparing their performances with it. The controllers in question will be divided into two parts; Predictive and Fuzzy

Controllers. Predictive control will emphasise the use of vanilla linear and nonlinear model predictive controllers as well as one of their variants. Additionally, Fuzzy controllers will be made given a Takagi-Sugeno modelling technique of our four tank plant that will be developed in this context. Everything will be validated first using simulation on Matlab/Simulink, to eventually lead to a possible implementation.

Chapter 2

State of the Art

2.1 Introduction

In this chapter,We will begin with the problem formulation of the quadruple tank process to where we will extend it by introducing the benchmark material available in our laboratory. Surely this will be done by defining the nonlinear process using nonlinear ordinary differential equations. These equations are based on the physical laws of mass balances and Bernoulli's fundamental equations. We will then proceed with an overview of the existing literature regarding the control strategies used for this particular problem. Additionally, we will set up a benchmark for the control of the plant by designing a decentralised PI controller to see its performances and compare it with the other control strategies that we're going to design.

2.2 Problem Formulation

The quadruple-tank process was introduced by Karl Henrik Johansson [37] in the year 2000 as a novel multivariable laboratory setup composed of four interconnected water tanks. The main idea behind the creation of such setup was to illustrate performance limitations due to the zero location in the multivariable control context. Indeed, it is well known that nonminimum-phase characteristics of a system impose limitations on the design of linear feedback controllers [9]. These limitations are the results of the constraints put on the sensitivity function by the zeros found in the right side of the s-plane. These limitations could be well controlled in the context of the four tanks coupled system as Johansson described in his paper.

Figure 1.1 shows a diagram of the discussed process. Its inputs are the voltages of two pumps and the outputs are the water levels of the two lower tanks shown in the figure. It is said in the paper that the process could be built easily by using two double-tank systems that are used in many laboratories. The paper is also titled "A Multivariable Laboratory Process with an Adjustable Zero" This is mainly due to the fact that the linearized model of the process in question has a multivariable-zero that could be located in either the left or the right half-plane (root locus graph) by changing the valve ratio γ that we're going to see further more in the model development section [37].

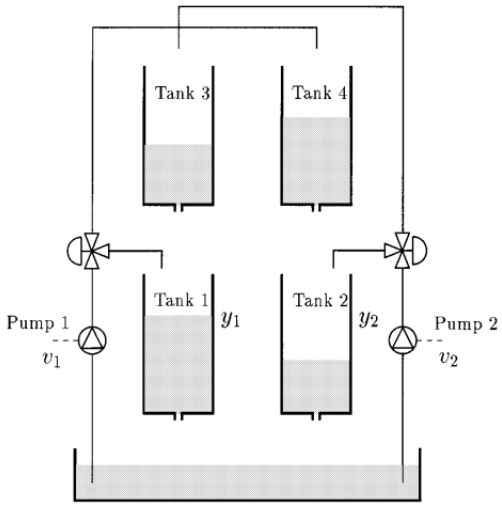


Figure 2.1: Schematic diagram of the quadruple-Tank process [37]

As mentionned before, The process inputs are u_1 and u_2 (input voltages to pumps), and the outputs are y_1 and y_2 (voltages from level measurement devices). The target is to control the level of the two lower tanks with the inlet flow rates.

As shown in figure 1.1, The output of each pump is split into two using a three-way valve. Pump 1 is shared by tank 1 and tank 3, while pump 2 is shared by tank 2 and tank 4. Thus, each pump output goes to two tanks, one lower and another upper diagonal tank, and the flow to these tanks is controlled by the position of the valve represented as γ. The position of the two valves determines whether the system is in the minimum phase or in the non-minimum phase. Let the parameter γ be determined by how the valves are set. Particularly, according to [37]:

The system is nonminimum phase for

$$0 < \gamma_1 + \gamma_2 < 1$$

And minimum phase for:

$$1 < \gamma_1 + \gamma_2 < 2$$

Of course, the system is much more difficult to control in the nonminimum phase setting than in it is in the minimum one [37].

Each tank has a discharge valve at the bottom. The discharge from tank 4 goes to tank 1 while the discharge of tank 3 goes to tank 2. This interaction creates a strong coupling between the tanks which makes it a multivariable control system. Due to its strong nonlinear behavior, the problem of identification and control of QTP is always a challenging task for control systems engineers. Discharge from tank 1 and tank 2 goes to the reservoir tank at the bottom[37].

2.2.1 Model Development

The QTP design is a well-known MIMO system suitable for analysis of various control schemes used in real-time which have nonlinear dynamics. Some systems cannot be represented by a linear model and require the use of nonlinear models. The nonlinearity in QTP is due to the square root term in mass flow relationship, between flow and level of the tank. The nonlinear models create more difficulty in controlling the system. The linearization of this type of system requires a stationary point around which the system operates. Taylor series expansion is one of

the methods used for linearization which approximates the system at a given stationary point. Generally any system can be represented by state-space or Input-output model. The NL model will be used to compute A, B, C and D of the state-space representation which are obtained using Jacobian matrices.[36]

Modelling of a process is necessary to investigate how the behaviour of a process changes with time under the influence of changes in the external disturbances and manipulated variables and to consequently design an appropriate controller. This uses two different approaches, one is experimental and the other is theoretical. In such cases, a representation of the process is required in order to study its dynamic behaviour. This representation is usually given in terms of a set of mathematical equations whose solution gives the dynamic behaviour of the process [35].

For each tank $i = 1 \ldots 4$, the mathematical modelling is done by consideration of mass balance equation and Bernoulli's law yields:

[Rate of Accumulation of Mass in system] = [Mass flow Rate into the system]−[Mass flow Rate out the system]

Before deriving the mathematical equations of the system, let's consider the follwing notation:

- The input to pump 1 is denoted by V_1, and for pump 2, it's denoted by V_2.
- The valve priority set for the flow is denoted by $\gamma_1$andγ_2 in the range [0, 1].
- The flow through pump 1 when voltage V_1 is applied is k_1V_1, and for pump 2 when voltage V_2 is applied is k_2V_2.
- The flow through the pump is directly proportional to the input voltage applied for the pump.
- The flow in tank 1 after crossing valve 1 is $\gamma_1 k_1 V_1$, and for tank 2 after crossing valve 2 is $\gamma_2 k_2 V_2$.
- The flow in tank 4 after crossing valve 1 is $(1 - \gamma_1)k_1V_1$, and for tank 3 after crossing valve 2 is $(1 - \gamma_2)k_2V_2$.

The non-linear model of the Quadruple tank process is given below. The mass balance equation states that

[Rate of accumulation] = [Rate of in-flow] − [Rate of out-flow]

Using the law of conservation of mass,

$$\frac{dm_T}{dt} = m_{\text{in}} - m_{\text{out}} \tag{2.1}$$

Where,

m_T represents the mass accumulated in the tank,

m_{in} represents the input mass flow rate, and

m_{out} represents the output mass flow rate.

- Mass accumulated, m_T = volume of tank (v) × density of liquid in the tank (ρ)
- Input mass flow rate (m_{in}) = volumetric flow rate (q_{in}) × density of liquid in the inlet stream (ρ_1)

- Output mass flow rate (m_{out}) = volumetric flow rate (q_{out}) × density of liquid in the outlet stream (ρ_2)

then we will have

$$\frac{d\rho v}{dt} = \rho \cdot q_{\text{in}} - \rho \cdot q_{\text{out}} \tag{2.2}$$

Since the liquid used is the same throughout the system, then $\rho = \rho_1 = \rho_2$.

Modelling of the non-linear Quadruple tank process is given by:

$$A_i \frac{dh_i}{dt} = q_{\text{in}_i} - q_{\text{out}_i} \tag{2.3}$$

Where, for $i = 1, \ldots, 4$

A_i denotes the cross-sectional area of the tank,

h_i represents the water level, and

q_{in_i} indicates the in-flow of the tank,

q_{out_i} signifies the out-flow of the tank

The term q_{in} only depends on the input voltage supplied to the pump .
q_{out}, depends on the acceleration due to gravity and the head of the water in the tank.
q_{out} can be determined using Bernoulli's equation and the flow rate of the liquid.

therefore:

$$\left.\begin{cases} q_{\text{in}_1} = \gamma_1 k_1 V_1 \\ q_{\text{in}_2} = \gamma_2 k_2 V_2 \\ q_{\text{in}_3} = (1-\gamma_2) k_2 V_2 \\ q_{\text{in}_4} = (1-\gamma_1) k_1 V_1 \end{cases}\right\} \tag{2.4}$$

Where:

k_1, k_2 are the pump constants,

V_1, V_2 are the velocities of the water flow through pumps 1 and 2,

γ_1, γ_2 are the valve ratios.

And:

$$q_{\text{out}_i} = a_i \sqrt{2gh_i} \tag{2.5}$$

Where, for $i = 1, \ldots, 4$

a_i cross sectional area of the outlet pipes,

g acceleration due to gravity,

h_i represents level of the water in each tanks,

2.2.1.1 Tank 01

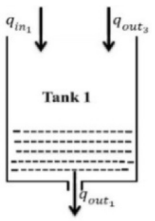


Figure 2.2: Mass-balance equation for Tank 1[36]

Using the law of conservation of mass,

$$[\text{Rate of accumulation}] = [\text{Rate of in-flow}] - [\text{Rate of out-flow}]$$

$$\begin{aligned} A_1 \frac{dh_1}{dt} &= q_{\text{in}_1} + q_{\text{out}_3} - q_{\text{out}_1} \\ &= \gamma_1 k_1 V_1 + a_3\sqrt{2gh_3} - a_1\sqrt{2gh_1} \end{aligned} \tag{2.6}$$

2.2.1.2 Tank 02

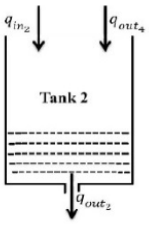


Figure 2.3: Mass-balance equation for Tank 2[36]

Using the law of mass conservation,

$$\begin{aligned} A_2 \frac{dh_2}{dt} &= q_{\text{in}_2} + q_{\text{out}_4} - q_{\text{out}_2} \\ &= \gamma_2 k_2 V_2 + a_2\sqrt{2gh_2} - a_4\sqrt{2gh_4} \end{aligned} \tag{2.7}$$

2.2.1.3 Tank 03

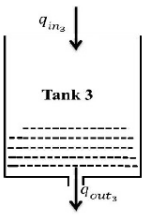


Figure 2.4: Mass-balance equation for Tank 3[36]

Using the law of conservation of mass,

$$[\text{Rate of accumulation}] = [\text{Rate of in-flow}] - [\text{Rate of out-flow}]$$

$$
\begin{aligned}
A_3 \frac{dh_3}{dt} &= q_{\text{in}_3} - q_{\text{out}_3} \\
&= (1-\gamma_2)k_2 V_2 - a_3\sqrt{2gh_3} \quad (8)
\end{aligned}
\tag{2.8}
$$

2.2.1.4 Tank 04

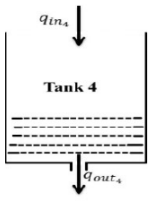


Figure 2.5: Mass-balance equation for Tank 4[36]

Using the law of conservation of mass,

$$\text{[Rate of accumulation]} = \text{[Rate of in-flow]} - \text{[Rate of out-flow]}$$

$$
\begin{aligned}
A_4 \frac{dh_4}{dt} &= q_{\text{in}_4} - q_{\text{out}_4} \\
&= (1-\gamma_1)k_1 V_1 - a_4\sqrt{2gh_4}
\end{aligned}
\tag{2.9}
$$

The Final equations:

$$
\left\{
\begin{array}{l}
\dfrac{dh_1(t)}{dt} = -\dfrac{a_1}{A_1}\sqrt{2gh_1(t)} + \dfrac{a_3}{A_1}\sqrt{2gh_3(t)} + \dfrac{\gamma_1 K_1}{A_1} v_1(t) \\
\dfrac{dh_2(t)}{dt} = -\dfrac{a_2}{A_2}\sqrt{2gh_2(t)} + \dfrac{a_4}{A_2}\sqrt{2gh_4(t)} + \dfrac{\gamma_2 K_2}{A_2} v_2(t) \\
\dfrac{dh_3(t)}{dt} = -\dfrac{a_3}{A_3}\sqrt{2gh_3(t)} + \dfrac{(1-\gamma_2)K_2}{A_3} v_2(t) \\
\dfrac{dh_4(t)}{dt} = -\dfrac{a_4}{A_4}\sqrt{2gh_4(t)} + \dfrac{(1-\gamma_1)K_1}{A_4} v_1(t)
\end{array}
\right\}
\tag{2.10}
$$

The above non-linear differential equation represents mathematical model of the four-tank system. The tank is being mathematically modelled by using law of conservation of mass. It is always enough to develop a controller for a particular process using its mathematical model. But here in QTP there is a challenge, that due to its non-linearity and uncertainty it is difficult to develop a controller which must take a proper control action[35].

Other characteristics that have been mentioned in [37] include that the system is controllable. However, the interactions should be considered in the choice of the control to output pairs by computing the Bristol Matrix. Furthermore, the experimental setup that we're going to present in the next section have four sensors for each and every single height. Thus, it is wise to consider that the QTP is fully observable for our study.

2.3 Experimental Setup Description

For the experimental setup part, we will be considering the use of the pilot plant coupled Tanks 33-230, from Feedback Instruments. This process was introduced as an educational control system lab machinery that is present in the "Laboratoire de Commandes des Processus" since two years ago.

The system in question has 4 translucent tanks each with a pressure sensor to measure the water level. The coupling between the tanks can be modified by the use of seven manual valves to change the dynamics of the system imposing the use of different controllers. Water is delivered to the tanks by two independently controlled, submersed pumps.

Step disturbances generation is provided by four manual valves. Drain flow rates can be modified using easy-to-change orifice caps.

The Coupled Tanks are controlled by using SIMULINK® and an Advantech PCI1711 Interface card.The user may build their own models or use the models supplied together with the curriculum. The process variables can be observed on-screen in plots. The product is supplied with a student manual that provides information about the physical behaviour of the system models and guides the student through the control tasks. Control algorithms are developed, tested on the models, and then implemented in a real-time application.[17]

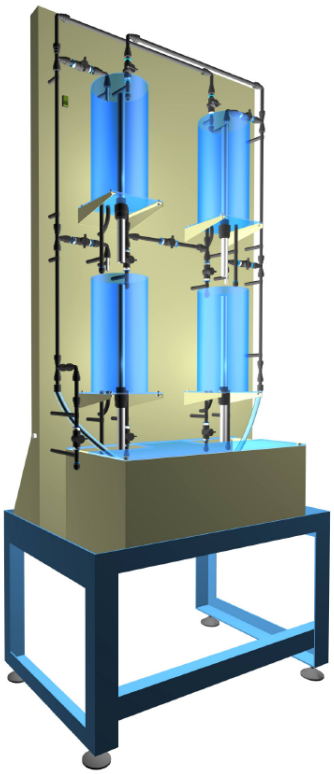


Figure 2.6: Coupled Tanks mechanical unit process [17]

A fifth reservoir tank is placed at the bottom. In the reservoir, two submersible pumps are placed, which pump the water on command to the tanks. The water flows freely to the bottom tanks through the configurable orifice. The way the water flows through the setup can be configured in many ways with manual valves labelled (MVA, MVB ... MVG, MV1, MV2...MV4). Configuration with valves allows for dynamic couplings introduction and step disturbances generation, giving vast possibilities of control. All of these are illustrated in the upper figure.

The model contains the following parts. All of this assembled according to Figure 1.7:

- 6 tank filling valves, linked 3 to each pump. Each pump connects with the two upper tanks and with the lower tank on its side
- 4 tank discharge valves, one per tank
- 4 discharge escapes without a valve, one per tank
- 4 area reducers, one per escape
- 1 connection valve between the two upper tanks
- 2 centrifugal pumps
- 6 T connections
- 6 90 degree elbows
- 4 tanks of 128.4mm^2 area discounting the discharge
- 4 overflow pipes in the tanks located at 25 and 27mm in height
- 2 diameter reducers, for the connection of the pipes
- 2 flexible pipes with an outer diameter of 18mm
- Rigid pipes of 12mm outer diameter and 10mm inner diameter[25]

Apart from the mechanical parts, the Coupled Tanks system is equipped with a Power Supply Unit and Power Amplifier (PSUPA) and the Cable Connector Box (Figure 1.7). The PSUPA unit amplifies the water pressure-level signals and passes them as analogue signals to the PCI1711 card. The pump control signal can be sent from the PC through the PCI1711 card and PSUPA unit.

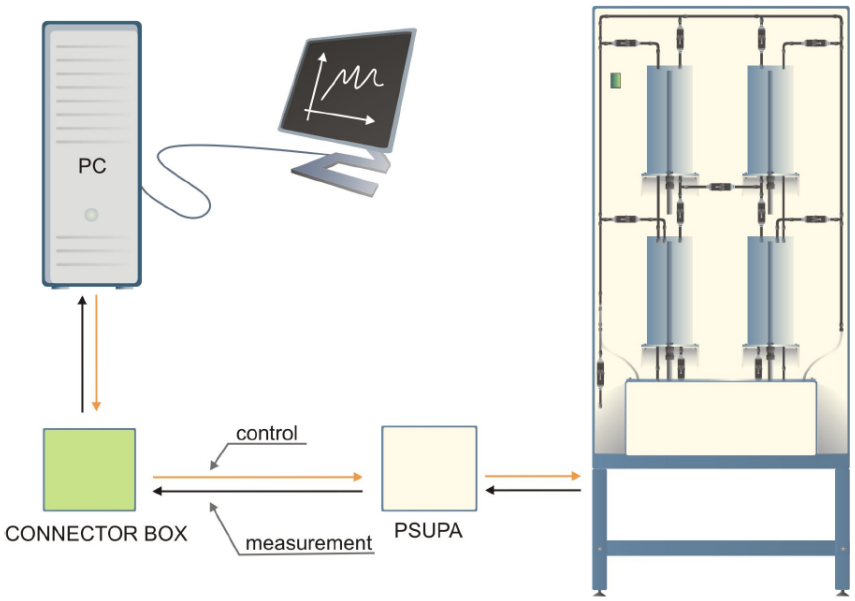


Figure 2.7: Coupled Tanks control system [1]

2.4 Literature Review

The quadruple-tank process is a well-known benchmark system in control theory, widely studied for its nonlinear, multivariable characteristics, and unique dynamics which include both minimum and non-minimum phase behaviors. This system provides an ideal platform to test various control strategies due to its flexibility and complexity.

Initial works on the quadruple-tank system focused on modelling and understanding the system's dynamics. As mentioned before, Johansson provided a comprehensive introduction to the system as the first pioneer that created this multivariable system. He was the first to highlight its decentralized PI controller design [37], which set the foundation for further research. Later, Numsomran et al. expanded on this by incorporating unknown disturbances into the model, providing a more realistic representation of the system [49].

Uçak and Öke introduced a novel approach using Support Vector Regression (SVR) for modelling the quadruple-tank system, which improved the system's generalisation due to its basic properties of structural risk minimization and ensuring global minima[55]. It was also a good introduction of SVRs as an intelligent modeling technique of nonlinear systems and in tuning of controller parameters based on this system model.

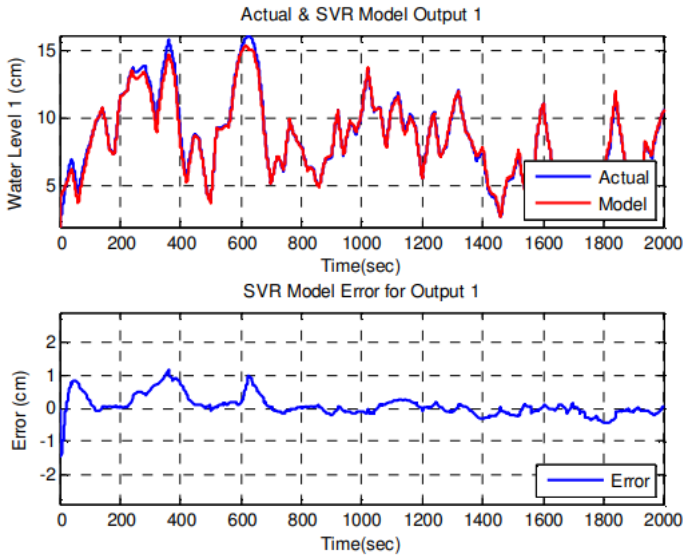


Figure 2.8: Results Presented by Modelling the QTP using SVR [55]

Following the same trend in these control strategies, decentralized control strategies have been extensively researched due to their simpler implementation and robustness. Haifen and Wen explored partially decentralized control where they demonstrated that it could achieve comparable performance to centralized control with a simpler structure and easier tuning [27]. Ferrese et al. proposed a decentralized control method for nonlinear interconnected large-scale systems. They employed Pontryagin's minimum principle for optimal control development [19]. Tunning the decentralized PI controller's parameters has been a hot topic in the research community and different Optimization algorithms alongside different Objective functions have been employed for this matter. For example, Abidin et al. used Particle Swarm Optimization (PSO) for optimizing PI controller parameters, demonstrating significant improvements in the control accuracy [3]. Similarly, Rao et al. applied the Grasshopper Optimization Algorithm to optimally tune PI controllers, achieving effective level control of the quadruple-tank system [45].

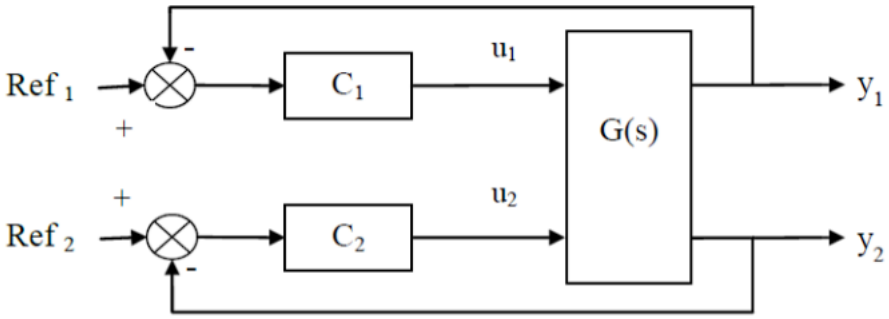


Figure 2.9: Decentralized control scheme [34]

High gain observer designs have also been a focal point in control strategies for the QTP. Gaaloul and M'Sahli developed a high gain output feedback control approach, achieving exponential stability and maintaining the separation principle [20]. Similarly, Turki et al. and Gouta et al. implemented back-stepping control combined with adaptive observers, ensuring global asymptotic stability and enhancing system robustness [53, 26].

Robust control techniques such as H_∞ loop shaping and Linear Matrix Inequalities (LMI) have been applied to the quadruple-tank system to address system uncertainties and ensure stability. Li et al. utilised H_∞ loop shaping for robust performance [41], while Hypiusová and Rosinová employed LMI for robust control design, ensuring quadratic stability and meeting H_2 and H_∞ performance specifications [30].

Adaptive control methods have also seen significant advancements. Sabura Banu and Lakshmanaprabu used Bat colony optimization for tuning multi-loop fractional order PID controllers, showing improved performance over traditional methods [50]. In another approach, Zhang and Li applied Gaussian Process-based Nonlinear Model Predictive Control, enhancing control precision through local linearization techniques [60].

Sliding mode control (SMC) has been explored to mitigate chattering and enhance system stability. Larguech et al. improved the first-order SMC by adding a term to reduce chattering and developed a stable adaptive tracking control for nonlinear MIMO systems [40]. El Hajjaji and Chadli used Takagi-Sugeno fuzzy models for system identification and control. They used parameter estimation with Levenberg-Marquardt method and least squares methods [5].

Model Predictive Control remains a popular choice for managing the multivariable nature of the quadruple-tank system. One of the pioneers to apply MPC on the QTP are Deepa et al. where they developed a discrete-time MPC for level control, demonstrating its effectiveness even with dead time considerations [15]. The advancements continued with Ghasemi et al. where they proposed a stabilized hybrid MPC, guaranteeing Lyapunov stability and robustness through an autonomous switched hybrid system model [23]. Furthermore, Ahmed et al. designed a decoupling neuro-fuzzy model predictive controller, combining the advantages of fuzzy logic and neural networks for improved dynamic modelling and control precision [4].

Recent advancements include the application of reinforcement learning for controller design. Kumar and Detroja introduced a reinforcement learning-based PI controller for nonlinear multivariable systems. They indeed could show the learning capabilities of neural networks to adapt to system dynamics and improve control performance [39].

To our knowledge, fuzzy TS based controllers haven't been researched thoroughly in the QTP context. Especially through the PDC control law based on the Lyapunov stability criterion. Our study will be first of its kind to propose this approach.

2.5 Base Method: Decentralised Proportional Integral Controller Design

In this section we will be tackling down the design of a decentralized PI control law for the quadruple Tank Process. As mentioned before in the literature review part, the first one that tackled down the design and the implementation of such control law was of course Johansson [37] The work then expanded to the research community as a series of papers dealt with the problem of tuning the parameters of such controller. For our study, we will focus on designing our control law based on a PSO optimization approach on a Pontriyagin principle of variation based objective function as it is introduced by [19].

The control laws developed would be considered for two operational modes. In a minimum phase and a non-minimum phase settings around given points. We will also perform a robustness test over the consideration of interconnections and a discharge coefficient that models the liquid behavior for higher viscosity fluids e.g. oil.

2.5.1 Presentation of the Solution

Recall the mathematical model of our four tanks coupled system 2.10 introduced previously:

$$\begin{aligned}
\frac{dh_1(t)}{dt} &= -\frac{a_1}{A_1}\sqrt{2gh_1(t)} + \frac{a_3}{A_1}\sqrt{2gh_3(t)} + \frac{\gamma_1 k_1}{A_1}v_1(t) \\
\frac{dh_2(t)}{dt} &= -\frac{a_2}{A_2}\sqrt{2gh_2(t)} + \frac{a_4}{A_2}\sqrt{2gh_4(t)} + \frac{\gamma_2 k_2}{A_2}v_2(t) \\
\frac{dh_3(t)}{dt} &= -\frac{a_3}{A_3}\sqrt{2gh_3(t)} + \frac{(1-\gamma_2)k_2}{A_3}v_2(t) \\
\frac{dh_4(t)}{dt} &= -\frac{a_4}{A_4}\sqrt{2gh_4(t)} + \frac{(1-\gamma_1)k_1}{A_4}v_1(t)
\end{aligned}$$

Let's consider now the following decentralized controller for our QTP:

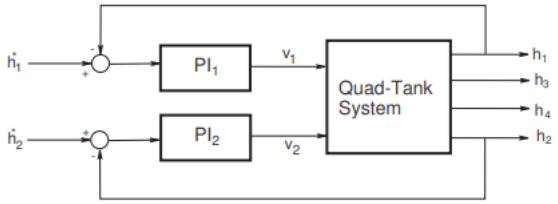


Figure 2.10: Decentralized controller scheme applied to the QTP [19]

As per [37], This configuration is correct only for the minimum phase setting. Details about the configuration would be explained further on in this section. The described closed-loop system incorporates two proportional-integral (PI) controllers. These controllers operate on decentralized feedback and are represented by the following equations [19]:

$$\text{PI1}: \quad v_1(t) = k_{p1}(h_1^* - h_1(t)) + k_{i1}\int_0^t (h_1^* - h_1(\tau))d\tau \tag{2.11}$$

$$\text{PI2}: \quad v_2(t) = k_{p2}(h_2^* - h_2(t)) + k_{i2}\int_0^t (h_2^* - h_2(\tau))d\tau \tag{2.12}$$

In these equations, k_{p1} and k_{p2} are the proportional gains, while k_{i1} and k_{i2} are the integral gains of

the respective controllers. The target water levels in tanks 1 and 2 are denoted by h_1^* and h_2^*. Notably, the control system does not account for the water levels in tanks 3 and 4.

To represent the complete closed-loop system in state-space form, we introduce two new state variables, z_1 and z_2 [19], defined as follows:

$$\dot{z}_1 = h_1^* - h_1 \tag{2.13}$$

$$\dot{z}_2 = h_2^* - h_2 \tag{2.14}$$

This yields the complete closed-loop system as follows:

$$\dot{h}_1 = -\frac{a_1}{A_1}\sqrt{2gh_1} + \frac{a_3}{A_1}\sqrt{2gh_3} + \frac{\gamma_1 k_1}{A_1}v_1 \tag{2.15}$$

$$\dot{h}_2 = -\frac{a_2}{A_2}\sqrt{2gh_2} + \frac{a_4}{A_2}\sqrt{2gh_4} + \frac{\gamma_2 k_2}{A_2}v_2 \tag{2.16}$$

$$\dot{h}_3 = -\frac{a_3}{A_3}\sqrt{2gh_3} + \frac{(1-\gamma_2)k_2}{A_3}v_2 \tag{2.17}$$

$$\dot{h}_4 = -\frac{a_4}{A_4}\sqrt{2gh_4} + \frac{(1-\gamma_1)k_1}{A_4}v_1 \tag{2.18}$$

$$\dot{z}_1 = h_1^* - h_1 \tag{2.19}$$

$$\dot{z}_2 = h_2^* - h_2 \tag{2.20}$$

The feedback controller is given by:

$$\text{PI1}: \quad v_1 = k_{p1}(h_1^* - h_1) + k_{i1}z_1 \tag{2.21}$$

$$\text{PI2}: \quad v_2 = k_{p2}(h_2^* - h_2) + k_{i2}z_2 \tag{2.22}$$

Substituting the feedback control law into the system model provides the system in the standard form:

$$\dot{x} = f(x, \alpha, t), \quad x(0) = x_0, \tag{2.23}$$

where $\alpha = [k_{p1}, k_{p2}, k_{i1}, k_{i2}]$ is the vector of the controller's unknown gains [19]. The goal of the control design is to determine the optimal α to maintain the desired water levels h_1^* and h_2^* in tanks 1 and 2, respectively. Note that the system model is nonlinear, precluding the use of design methods based on linear control theory [19]. Now we will use nonlinear optimization theory to design our controller.

2.5.2 Optimization Problem formulation and solution

The optimization problem defined above could be considered the same general class known as *parameter optimization* of dynamic systems. In classical optimal control problem, the solution is usually the best time varying control function that minimizes a given objective function. But in this case, we're sure that the controllable quantities are constant parameters rather than time varying functions [19].

Consider the system defined earlier in equation 1.15, where $x(t) \in \mathbb{R}^n$ is the state vector, and $\alpha \in \mathbb{R}^m$ is a constant parameter vector. The cost function is defined as follows

$$J(\alpha) = \Phi(x(t_f)) + \int_0^{t_f} L(x, \alpha, t)\, dt. \tag{2.24}$$

It is assumed that the functions Φ and L are positive, convex, and differentiable. The objective is to find the optimal parameter vector α such that the cost is minimized. We assume that the space of admissible parameters is unconstrained for now. This will be alternated of course in a further part [19].

Suppose α is the optimal parameter, and x is the corresponding trajectory. Also assume that $x + \delta x$ is a perturbed trajectory corresponding to a non-optimal parameter vector $\alpha + \delta\alpha$.

Defining the augmented cost function

$$\tilde{J}(\alpha) = \Phi(x(t_f)) + \int_0^{t_f} L(x, \alpha, t)\, dt + \int_0^{t_f} \langle \psi(t), f(x, \alpha, t) \rangle\, dt \tag{2.25}$$

and taking the first variation, we obtain

$$\delta\tilde{J} = \left\langle \frac{\partial \Phi(x(t_f))}{\partial x} - \psi(t_f), \delta x(t_f) \right\rangle + \int_{t_0}^{t_f} \left\langle \frac{\partial H}{\partial x}(t) - \dot{\psi}, \delta x(t) \right\rangle dt \tag{2.26}$$

$$+ \left\langle \int_0^{t_f} \frac{\partial H}{\partial \alpha}(t)\, dt, \delta\alpha \right\rangle \tag{2.27}$$

where H is the Hamiltonian defined by

$$H(x, \psi, \alpha, t) = L(x, \alpha, t) + \langle \psi(t), f(x, \alpha, t) \rangle. \tag{2.28}$$

Then, since the variations are arbitrary, we obtain the necessary conditions for optimality as

$$\dot{x} = f(x, u, t), \quad x(t_0) = x_0, \tag{2.29}$$

$$\dot{\psi} = -\frac{\partial H}{\partial x} = -\frac{\partial L}{\partial x} - \left(\frac{\partial f}{\partial x}\right)^\top \psi, \quad \psi(t_f) = \frac{\partial \Phi(x(t_f))}{\partial x}, \tag{2.30}$$

and

$$0 = \int_0^{t_f} \frac{\partial H}{\partial \alpha}\, dt = \int_0^{t_f} \left(\frac{\partial L}{\partial \alpha} + \left(\frac{\partial f}{\partial \alpha}\right)^\top \psi \right) dt. \tag{2.31}$$

The above equations are of course a set of necessary conditions for the optimal control. Furthermore, they form what we call a complex two-point-boundary-value problem (TPBVP) [19].

IN [19] they proposed a common iterative gradient method that minimizes the cost by an update on the control law at every iteration. This is done by considering the first variation of the augmented cost function and approximating the variations as Δx , $\Delta\alpha$ and $\Delta\hat{J}$. This will allow to define an update law for the parameters α over a gradient on the cost function. Such that:

$$\alpha_{i+1} = \alpha_i - \epsilon \int_{t_0}^{t_f} \frac{\partial H}{\partial \alpha}(x_i, \psi_i, \alpha_i)\, dt, \tag{2.32}$$

For our study we would rather use another method that will guarantee a global minima for the cost function over the control parameters. A **Particle Swarm Optimization algorithm** is used for our

problem. The same algorithm was used in [3] but with a different mathematical set up i.e. optimization criteria.

The cost function $J(u)$ is defined as:

$$\begin{aligned} J(u) =& \frac{1}{2}s_1(h_1^* - h_1(t_f))^2 + \frac{1}{2}s_2(h_2^* - h_2(t_f))^2 \\ &+ \frac{1}{2}\int_0^{t_f} \left\{ q_1(h_1^* - h_1(t))^2 + q_2(h_2^* - h_2(t))^2 + d_1 * v_1^2(t) + d_2 * v_2^2(t) \right\} dt \\ &+ R\{k_{p1}^2 + k_{p2}^2 + k_{i1}^2 + k_{i2}^2\}, \end{aligned} \tag{2.33}$$

where s_1 and s_2 are weighting factors for the final state errors, q_1 and q_2 are weighting factors for the transient errors, d_1 and d_2 are weighting factors for the control inputs,$v_1(t)$ and $v_2(t)$ are the control inputs, and R is a regularization parameter for the controller gains k_{p1}, k_{p2}, k_{i1}, and k_{i2}.

The system's parameters that were considered to conduct the simulation were the same as [32]. This is the only work available that has conducted the research on the same experimental setup as ours (the 33-230 coupled tanks system by Feedback Instruments) and they are defined in the following table:

Parameter	**Value**
A_i [cm^2]	138.9
a_i [cm^2]	0.5027
k_1 [cm^3/Vs]	26.00
k_2 [cm^3/Vs]	22.94
γ_1	0.836
γ_2	0.897
g [cm/s^2]	981

Table 2.1: System's parameters extracted by[32]

Important Note : As we can see, the parameters are made up such that the height will be in **centimeters**. For this reason, all the simulation figures that we will present in this work will have the heights expressed in **centimeter** and the time scale will be in **seconds**.

In addition, the various weights in the cost function are taken as follows [19]: $s_1 = 10$, $s_2 = 20$, $q_1 = 90$, $q_2 = 100$, $d_1 = 30$, $d_2 = 30$ and $R = 10$. The initial water levels are also taken from [19] where $h_1(0) = 12.4$[cm], $h_2(0) = 12.7$[cm], $h_3(0) = h_4(0) = 1.5$[cm]. The chosen references are two step signals with a respective values of 20 and 16 respectively .

The details related to the PSO algorithm and the code created specifically for this task could be found in Appendix A.

The algorithm was initialized with a total of 100 particles and around 20 iterations were used for that. The space search was reduced to the interval $[0.0001, 5]$ because we already had an idea on the parameters values of the corresponding decentralize controller [19].

2.5.2.1 Minimum Phase setting

This phase corresponds to

$$1 < \gamma_1 + \gamma_2 < 2$$

And according to the benchmark paper [37] it is way easier to find a decentralized controller for this particular setting. The control scheme is the same one as figure 1.11 and the corresponding results are the following:

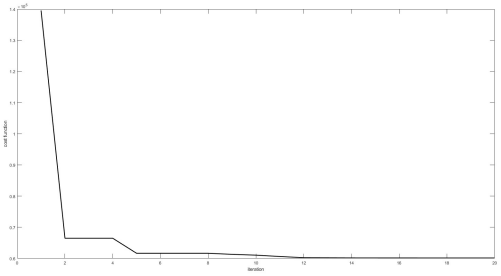


Figure 2.11: Cost Function Evolution with a PSO algorithm update

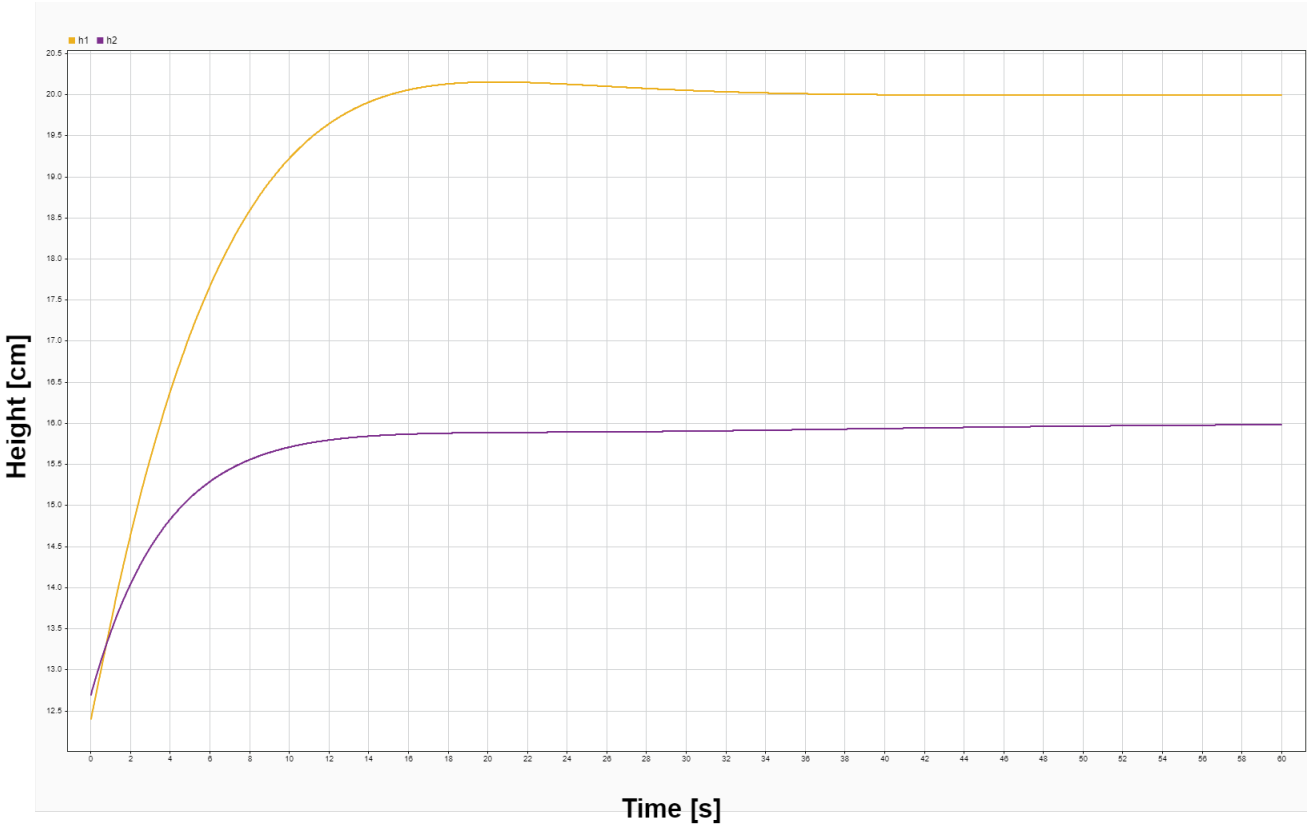


Figure 2.12: Optimized System Response with the mentioned conditions; h_1 (yellow) and h_2 (purple)

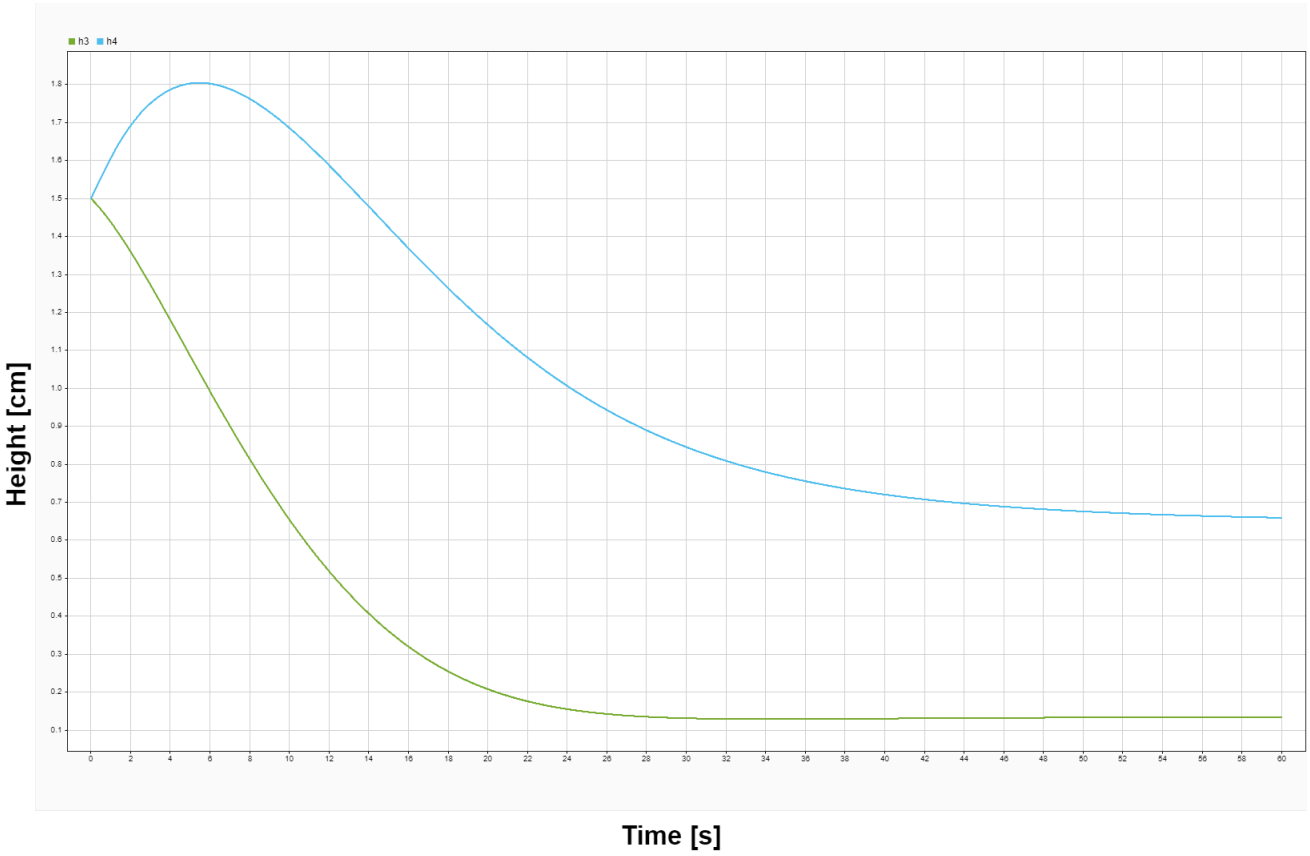


Figure 2.13: Corresponding h_3 (green) and h_4 (cyan) optimized response with the mentioned conditions

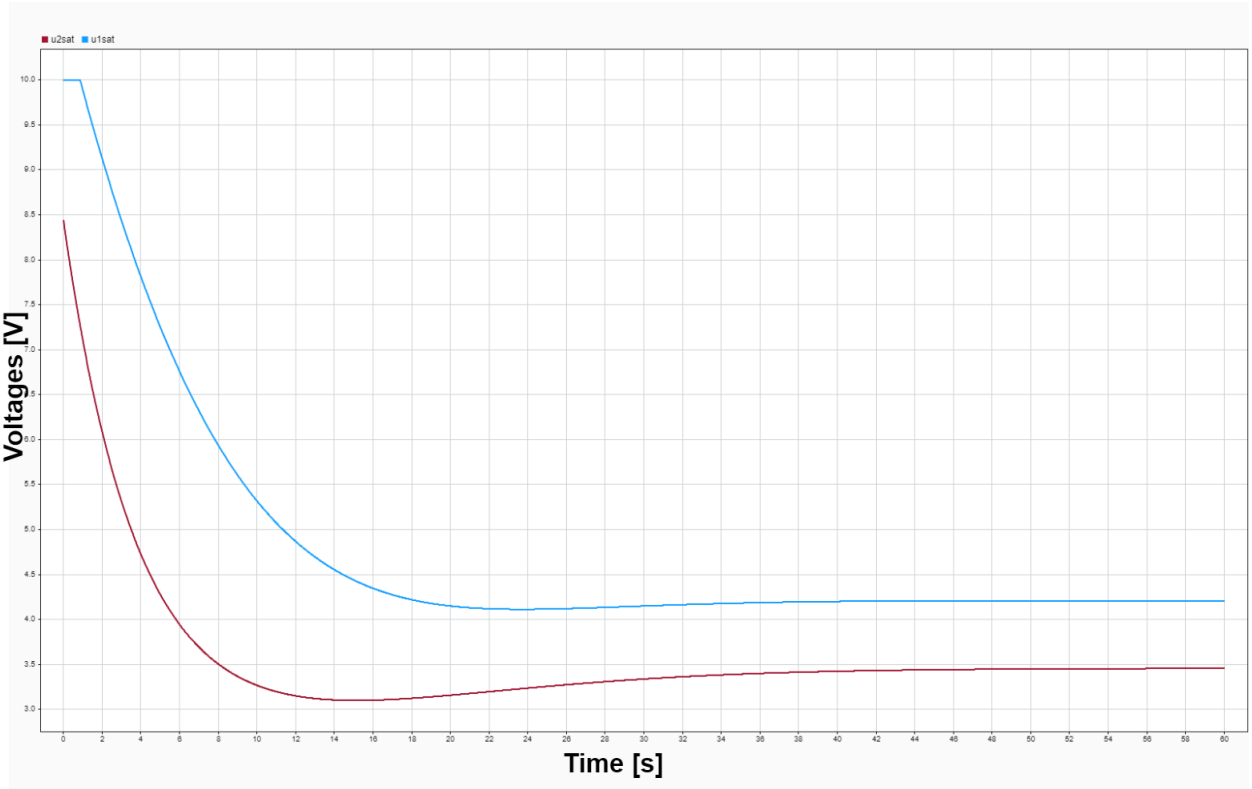


Figure 2.14: Corresponding Control signals with the mentioned conditions

The optimal parameters were found as follows:

$$\alpha^* = \begin{bmatrix} 1.4007 \\ 2.5549 \\ 0.1266 \\ 0.2133 \end{bmatrix} \tag{2.34}$$

The step response at 95% of the final value is found being at around 14 seconds for both outputs. It is clearly similar to the one designed in [19] whilst having a much larger Tanks to fill (comparing the system's parameters used). It is also worth mentioning that this response is optimized for this initial with respect to the final values discrepancy and could change depending on the gap between the desired and the actual output of our plant.

This would imply two things: First, if we made an optimization over a unitary step response for both outputs is just non usable for actual applications. A unitary step response represent 1[cm] and it is commonly not our desire to give that in operational conditions [37]. Second, the overshoot would increase even more (if we set it up with unitary step responses) for normal operational references for our outputs.

2.5.2.2 Non-minimum Phase setting

This phase corresponds to

$$0 < \gamma_1 + \gamma_2 < 1$$

This controller is much more difficult to design than the one in the minimum phase setting [37].

First of all, given the linear system's RGA matrix around a given operational point [37]:

$$\text{RGA} = \begin{bmatrix} \lambda & 1-\lambda \\ 1-\lambda & \lambda \end{bmatrix} \tag{2.35}$$

Such that:

$$\lambda = \frac{\gamma_1 \times \gamma_2}{\gamma_1 + \gamma_2 - 1} \tag{2.36}$$

So for $0 < \gamma_1 + \gamma_2 < 1$ The chosen control pair is inverted with respect to the original min-phase setting i.e. the chosen set up would be (u_1, h_2) and (u_2, h_1).

We will take $\gamma_1 = 0$ and $\gamma_2 = 0$. This would be equivalent to no coupling at all, seeming like it would be easier to do this task. Yet it is purely to see the non-minimum phase behavior as well as making a robustness test over the interconnections down below in this section. We will also choose the same initial conditions as before, to see the difference between the dynamics of the system.

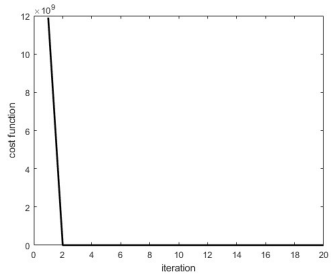


Figure 2.15: Cost Function Evolution with a PSO algorithm update

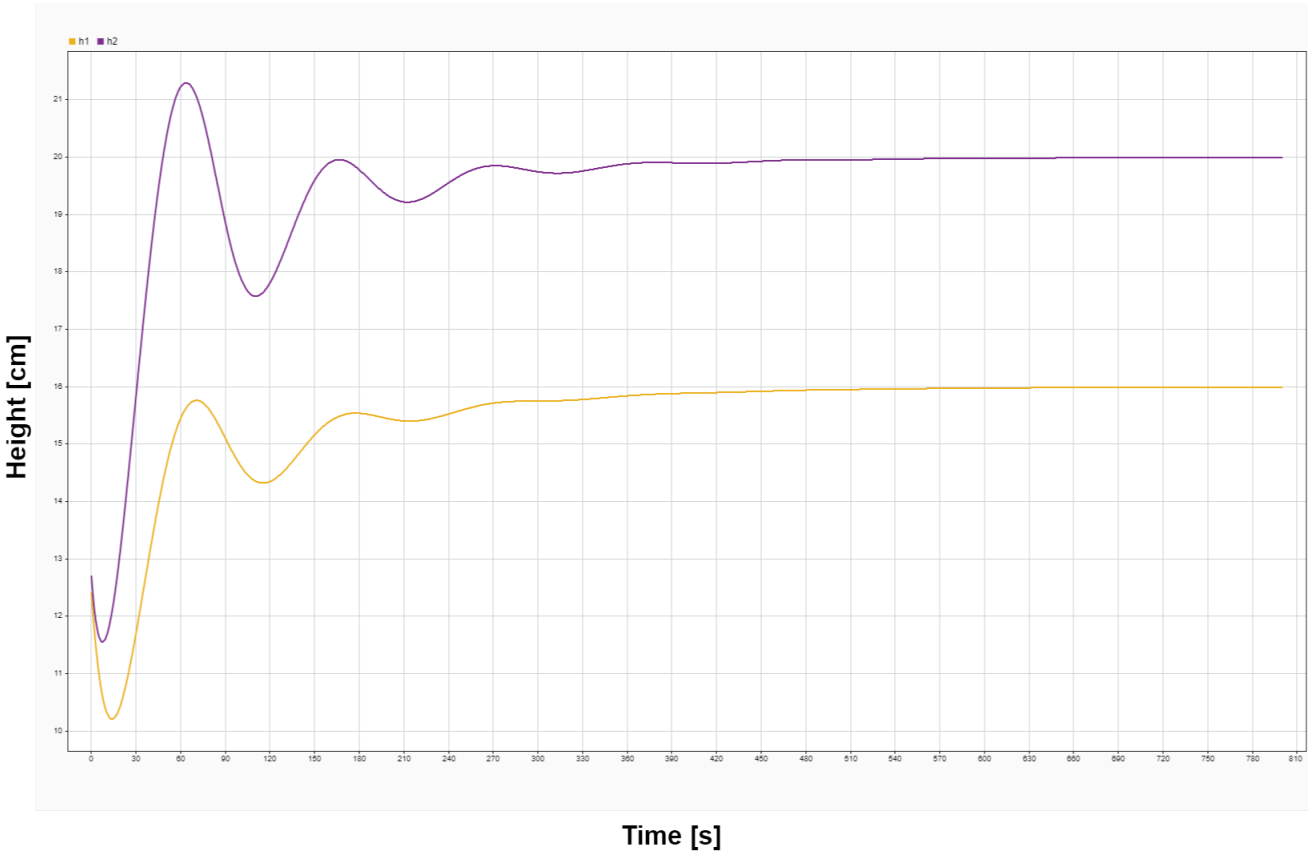


Figure 2.16: Optimized System Response with the mentioned conditions ; h_1 (yellow) and h_2 (purple)

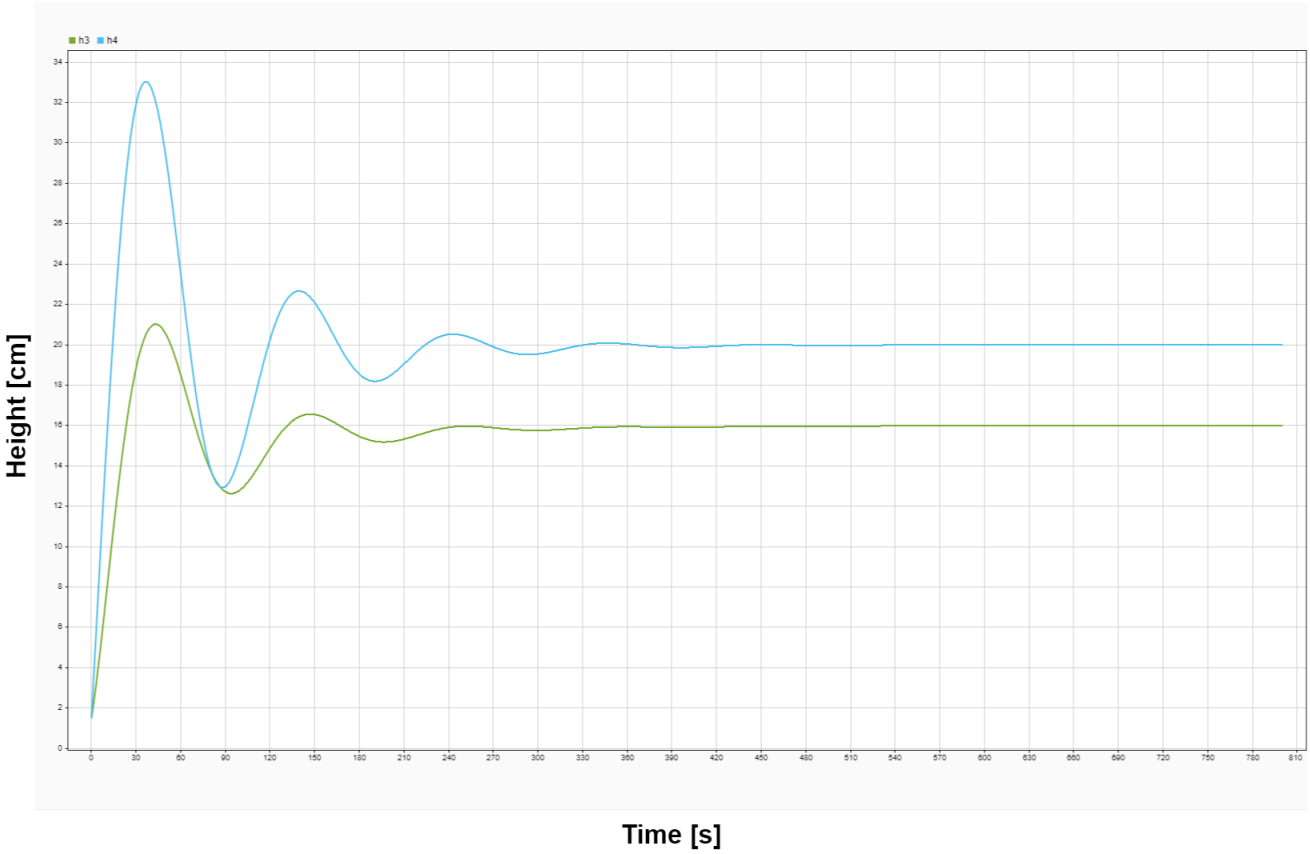


Figure 2.17: Corresponding h_3 (green) and h_4 (blue) optimized response with the mentioned conditions

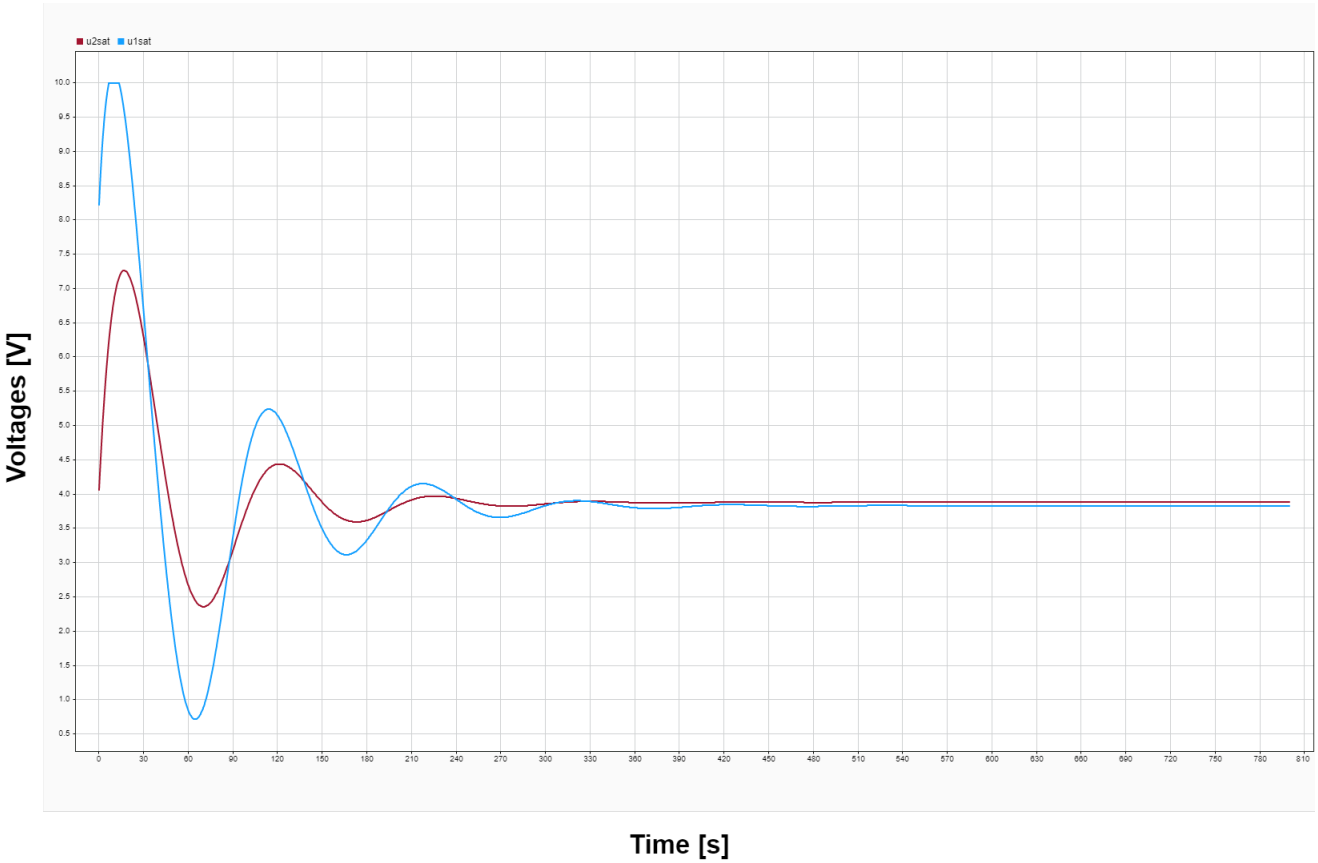


Figure 2.18: Corresponding Control signals with the mentioned conditions

The optimal parameters were found as follows:

$$\alpha^* = \begin{bmatrix} 1.1267 \\ 1.1267 \\ 0.0093 \\ 0.0093 \end{bmatrix} \tag{2.37}$$

The step response at 95% of the final value is found being around 340 seconds for both outputs. As astonishing as it can be, the benchmark for this type of setting has also a higher duration (as shown in the figure below). We can explain this physically by the fact that the system in this case depends on the gravitational force mainly to fill up the lower tanks whilst in the minimum case, it was relaying on the two lower nozzles that provided a flow with a given γ mainly for that task. In the chosen case, we can also see that the two upper tanks stabilize at the final values of the two lower ones (outputs). The inputs have also much bigger values than the case studied earlier. This is mainly due to the fact that the pump has to provide a constant flow and compensate the fluid that's being poured down by the gravitational force.

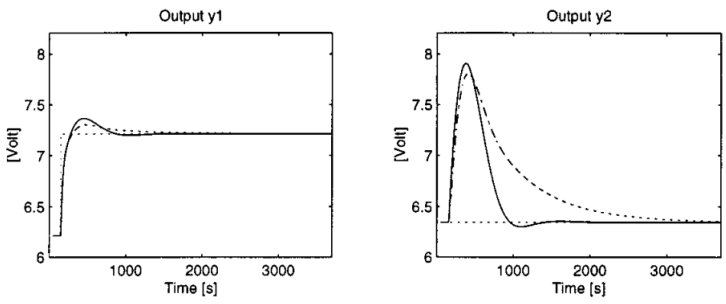


Figure 2.19: Benchmark Output for the non-minimum phase setting [37]

Even though we named h_3 and h_4 as hidden states. We assure that it is just an expression to distinct

them from the two known outputs. The system is completely observable as mentioned before.

2.5.3 Robustness tests and Validation of the controller

2.5.3.1 Robustness with respect to High viscosity fluids

High viscosity fluids have a different dynamics than lower ones like water. Meanwhile, if we look at our model we can clearly see that there isn't a single parameter that represents the characteristics of the used fluid in the QTP. The question remains now, how can we distinct between the different dynamics and see the effect of using a different fluid on the QTP.

We introduce the discharge coefficient C_d [16], a parameter that will model the ratio between the actual flow and the theoretical flow i.e:

$$C_d = \frac{Q_{actual}}{Q_{Th}} \tag{2.38}$$

given Torricelli equation for a fluid passing through a nozzle in our system:

$$Q_{Th} = a_i\sqrt{2gh_i} \tag{2.39}$$

Thus:

$$Q_{actual} = C_d a_i\sqrt{2gh_i} \tag{2.40}$$

We will also do the same thing for the two flows generated by the inputs, the revisited model after introducing this coefficient will be as follows:

$$\begin{aligned}
\frac{dh_1(t)}{dt} &= -\frac{C_d a_1}{A_1}\sqrt{2gh_1(t)} + \frac{C_d a_3}{A_1}\sqrt{2gh_3(t)} + \frac{C_d\gamma_1 k_1}{A_1}v_1(t)\\
\frac{dh_2(t)}{dt} &= -\frac{C_d a_2}{A_2}\sqrt{2gh_2(t)} + \frac{C_d a_4}{A_2}\sqrt{2gh_4(t)} + \frac{C_d\gamma_2 k_2}{A_2}v_2(t)\\
\frac{dh_3(t)}{dt} &= -\frac{C_d a_3}{A_3}\sqrt{2gh_3(t)} + \frac{(1-\gamma_2)C_d k_2}{A_3}v_2(t)\\
\frac{dh_4(t)}{dt} &= -\frac{C_d a_4}{A_4}\sqrt{2gh_4(t)} + \frac{(1-\gamma_1)C_d k_1}{A_4}v_1(t)
\end{aligned} \tag{2.41}$$

We will choose adequate values of C_d according to the literature [16] and we will see its effects on the step response of our system implemented alongside the decentralized controller in both the minimum and the non-minimum phase settings and the results were as shown in the figures 2.19 and 2.20.

At glance, we can clearly see that the coefficient induces a latency into the system. It is indeed expected with a higher viscosity fluid that tends to be more heavy, as we can say in the common language. The chosen values were $1, 0.8, 0.6, 0.4, 0.2$ and the distinction is clear seeing the representing figures.

In addition to the latency, a pseudo-sinusoidal behavior could also be seen. The overshoot increases every time we reduce the value of C_d and the system's response time at 95% from the final values does also increase. It is well logically following the physical behavior of higher viscosity fluids as the gravitational force takes a lot of time to change the value of the desired level.

Moreover, we can clearly say that the controller in the minimum phase setting is way more robust than the one in the non-minimum phase setting. For the critical value of $C_d = 0.2$ The states for the system with a zero on the right side of the plane is practically sinusoidal (even though the response will converge if we increase the simulation time by tenfold). The controller for the minimum phase can also go for even smaller values of C_d and converge in a reasonable timescale.

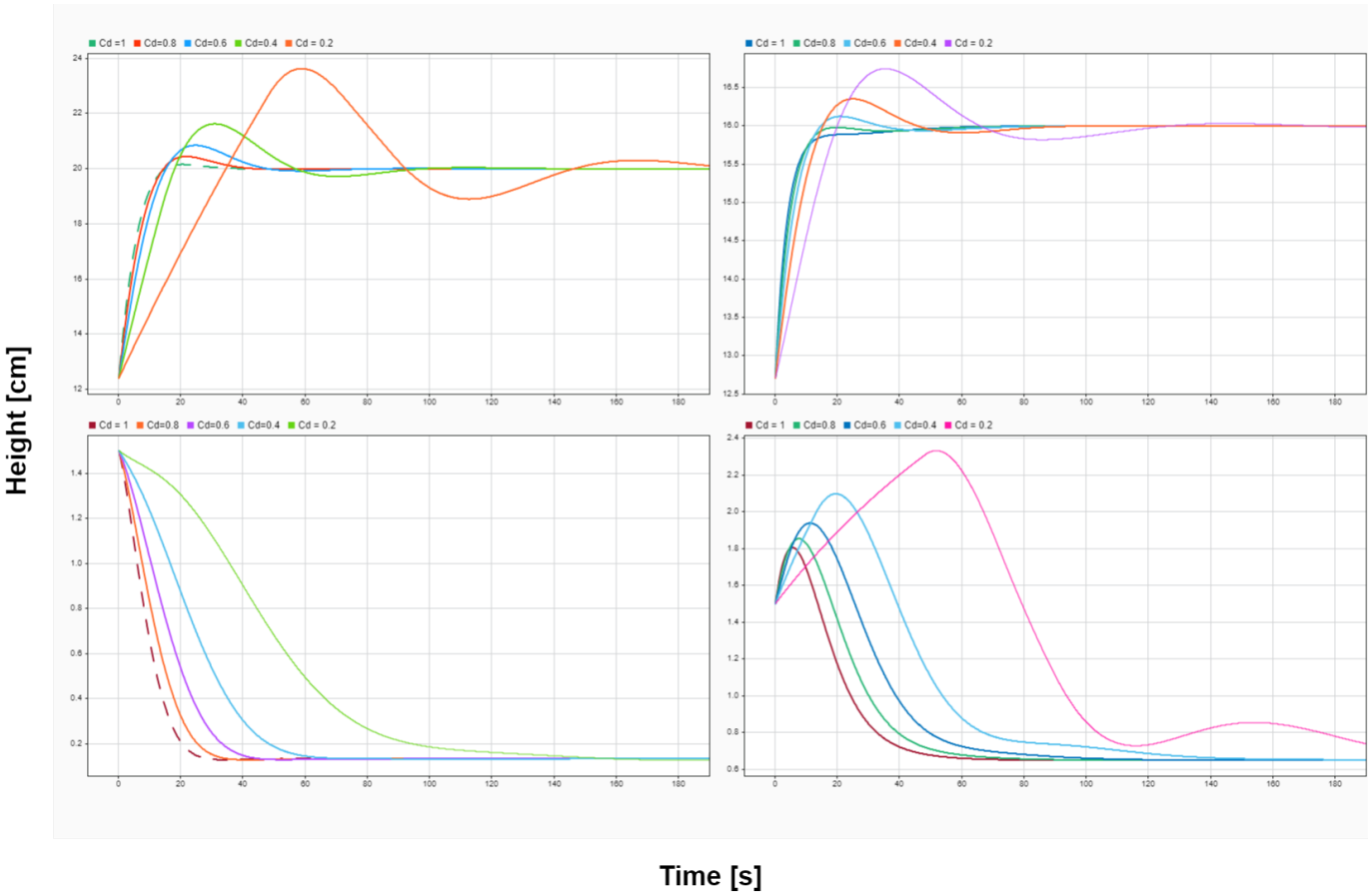


Figure 2.20: Evolution of the System's states under different values of C_d for the minimum phase setting, Upper left h_1, Upper right h_2, Lower left h_3, Lower right h_4

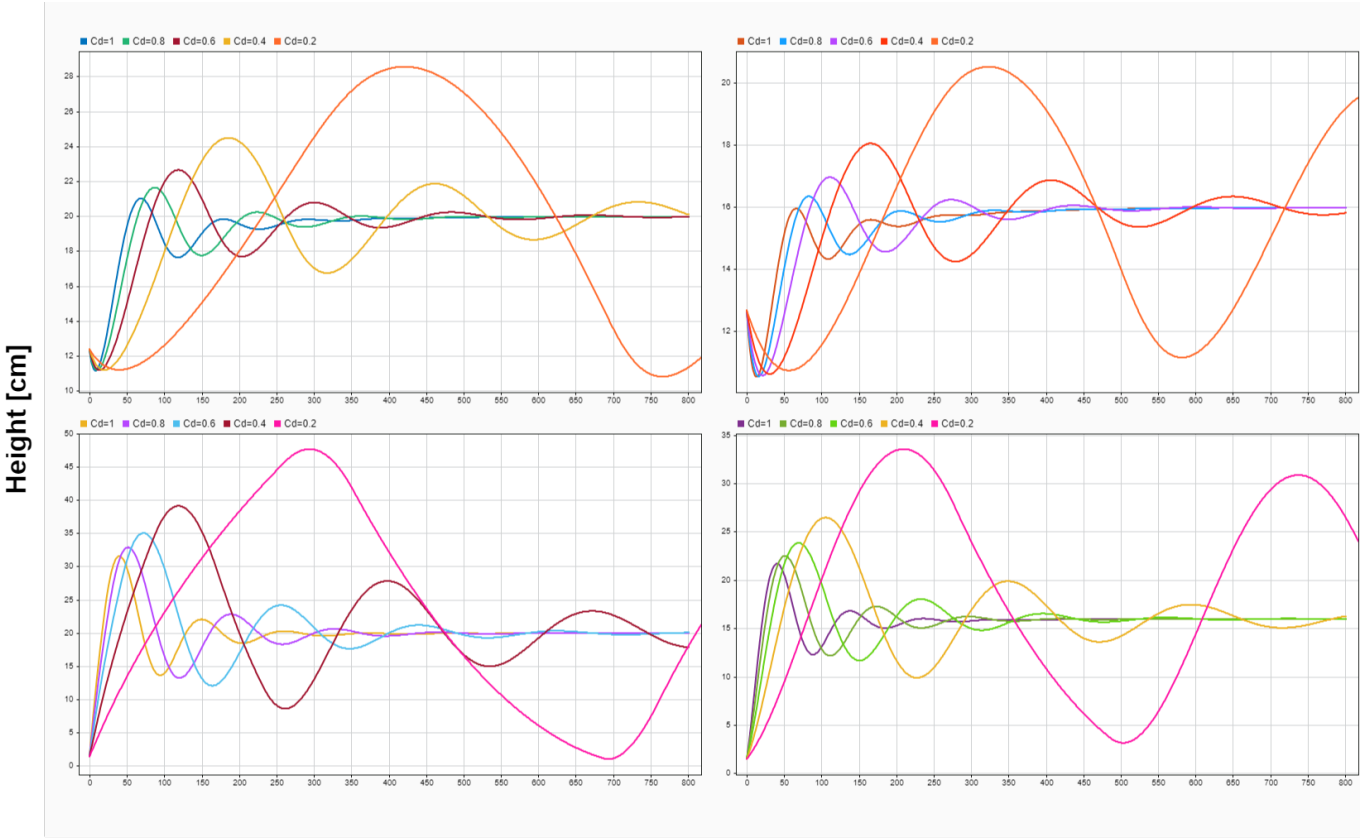


Figure 2.21: Evolution of the System's states under different values of C_d for the non-minimum phase setting, Upper left h_1, Upper right h_2, Lower left h_3, Lower right h_4

2.5.3.2 Robustness with respect to the interconnections

For this test, we will only consider the controller in the non-minimum phase setting. We designed that controller early on whilst considering $\gamma_1 = \gamma_2 = 0$. These values give us a setting where we will not consider the interconnections towards the two lower tanks. Making these two fill up completely using gravitational force and liquid descending from the two upper tanks. But how about we induce the designed controller with an interconnection i.e. $\gamma_1 = \gamma_2 = \gamma \neq 0$ and see their impact on our controller. This is shown through the figure down below.

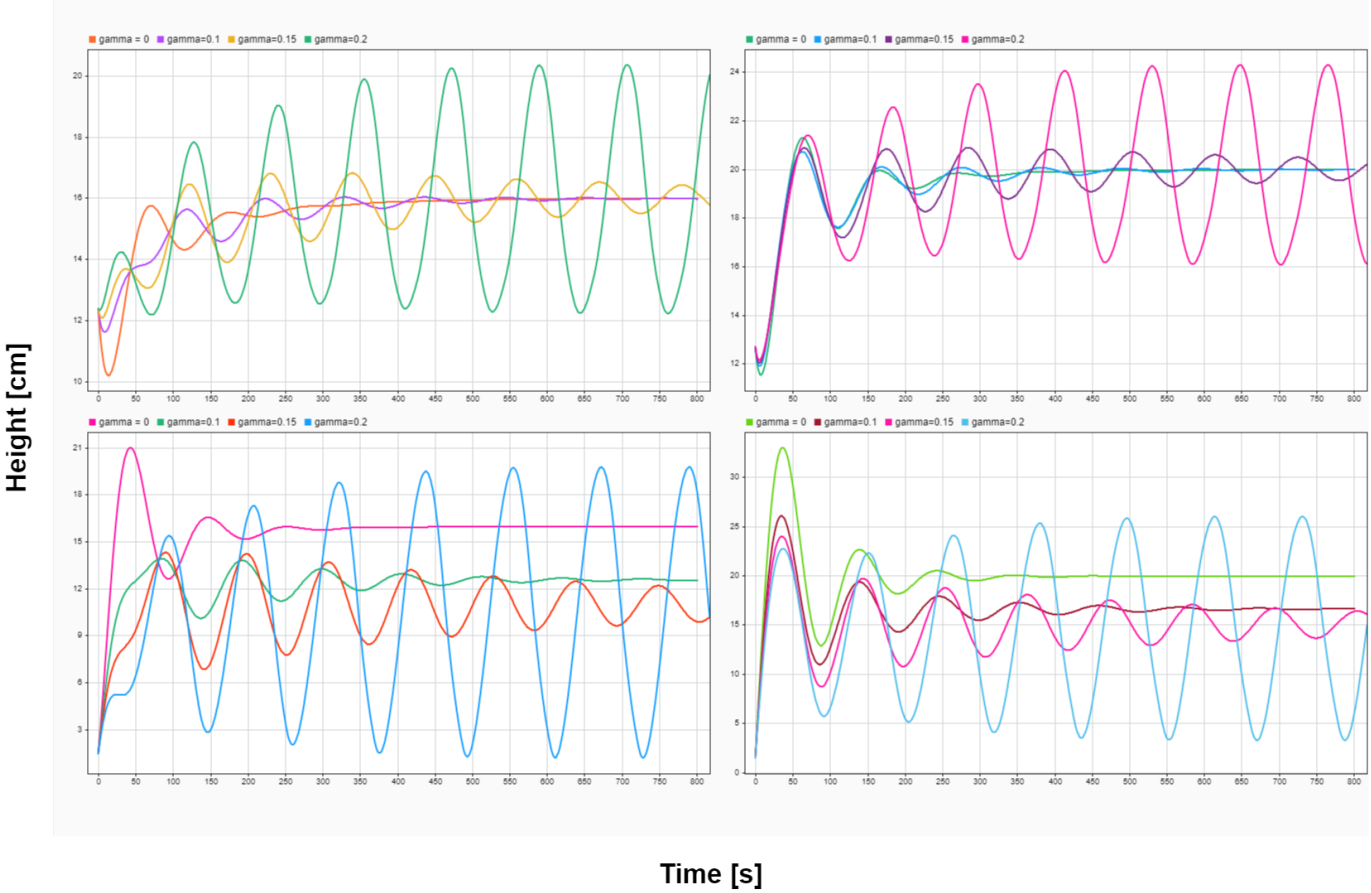


Figure 2.22: Evolution of the System's states under different values of γ for the non-minimum phase setting under a robustness of interconnections, Upper left h_1, Upper right h_2, Lower left h_3, Lower right h_4

The first thing we notice is that the interconnections induces a sinusoidal behavior over the system's state response. In fact, it does increase the step response time at 95% of the final value and the response is completely periodic for values $\gamma \geq 0.2$. The increased time is added up to the already long response of the non-minimum phase setting and we get a convergence time of over 1000[sec] for $\gamma = 0.15$.

The periodicity of the response also depends on the discrepancy between the desired reference signal and the output of the system. This means that in this case, if we decrease the difference between them, we could get a pseudo-periodic response, thus an exponential stability of the system.

This fact shows the relative robustness of our controller depending on external parameters related to the system which plays a huge deal in the common knowledge known about using linear control theory

on nonlinear systems. Basically, it shows some of the limitations of getting out of the linear range and trying to extend it to other nonlinear spaces for our system.

2.5.4 Conclusion

In this section, we explored the implementation and evaluation of a Particle Swarm Optimization (PSO) algorithm to optimize a decentralized controller for both minimum and non-minimum phase settings of our system. The specifics of the PSO algorithm and the related code are detailed in Appendix A A. By initializing the algorithm with 100 particles and running it for 20 iterations within the search space $[0.0001, 5]$, we did a good use of the prior knowledge of parameter values from a corresponding decentralized controller [19], which proved beneficial.

For the minimum phase setting, defined by $1 < \gamma_1 + \gamma_2 < 2$, the design of the decentralized controller was relatively straightforward. The optimized parameters yielded a system response similar to the benchmark results, with a 95% step response time of around 14 seconds. This demonstrates the efficiency of the PSO algorithm in achieving optimal control for this setting.

On the other hand, the non-minimum phase setting, characterized by $0 < \gamma_1 + \gamma_2 < 1$, presented more challenges. Here, the optimized parameters resulted in a longer step response time of approximately 340 seconds. This highlights the increased difficulty of controlling systems in non-minimum phase settings, where system dynamics heavily depend on gravitational forces, leading to slower stabilization and higher control effort.

Our robustness tests further validated the controller's performance under varying conditions. Introducing high viscosity fluids and different discharge coefficients resulted in increased latency and overshoot in the system response. Notably, the controller in the minimum phase setting showed greater robustness compared to the non-minimum phase setting, which showed more pronounced sinusoidal behavior under extreme conditions.

Additionally, testing the robustness with respect to interconnections revealed the impact of coupling parameters (γ values) on the system's behavior. Increased interconnections induced periodicity in the system's response, extending the step response time. This truly shows the challenges of maintaining stability and robustness in non-minimum phase systems with significant interconnections.

This being said, we can say that we have made our base control method for the four tanks coupled system. This method would be used to compare its performance with the performances of other controllers that are going to be designed in this thesis. The PSO algorithm ensured that we had our global optimum, Thus the parameters of the decentralized PI controller are relatively perfect with respect to the considered objective function.

Chapter 3

Predictive Control

3.1 Introduction

This chapter investigates the fundamentals and different techniques of predictive control known in the academia. We will first dive into the basics of Model Predictive Control (MPC) and how to implement an MPC Controller. The controller in question will be diveded into a Linear MPC and a nonlinear MPC. We will also see the different software tools that are used for this embedded optimization. Additionally, we will present a Predictive controller based on Recurrent Neural Networks (RNN) and classical vanilla MPC.

3.2 Background

Model Predictive Control (MPC), also referred to as Receding Horizon Control (RHC), is a method for deriving a feedback control law by learning open-loop control to assess current states and swiftly compute a control function for the open-loop system [32].

Also known as Model Based Predictive Control (MBPC), it emerged in the late 1970s and has significantly evolved since then. Rather than indicating a single control strategy, the term MPC encompasses a wide range of control methods that explicitly use a model of the process to derive the control signal by minimizing an objective function. These design methodologies result in linear controllers with similar structures and adequate degrees of freedom[12]. The core ideas prevalent in varying degrees across the predictive control family include:

- Explicit use of a model to forecast the process output at future time instances (horizon).
- Computation of a control sequence that minimizes an objective function.
- A receding horizon strategy, where the horizon moves forward at each instant, applying the first control signal of the sequence calculated at each step [12].

Different MPC algorithms, primarily differ in the model used to represent the process and noise, as well as the cost function to be minimized [12]. This control method has an open nature, having an extensive academic and industrial research and development. Presently, predictive control is successfully applied across various fields, from the process industry to diverse applications such as robot manipulators and clinical anesthesia [12].

3.2.1 Some Advantages

MPC offers several advantages over other control methods [12], notably:

- It is particularly appealing to personnel with limited control knowledge due to its intuitive concepts and relatively easy tuning.
- It can be applied to a wide variety of processes, from those with simple dynamics to more complex ones, including systems with long delay times, non-minimum phase systems, or unstable systems.
- It effectively handles multivariable cases.
- It inherently compensates for dead times.
- It naturally incorporates feedforward control to counteract measurable disturbances.
- Its extension to handle constraints is conceptually simple, allowing constraints to be systematically included during the design process.
- It is highly beneficial when future references (as in robotics or batch processes) are known.
- It is a completely open methodology based on fundamental principles that allow for future extensions.

3.2.2 Some Drawbacks

As with any control methodology, MPC has its drawbacks [12]. One notable issue is that, despite the resulting control law being straightforward to implement and requiring minimal computation, its derivation is more complex compared to classical PID controllers. If the process dynamics remain constant, the controller's derivation can be precomputed. However, in adaptive control scenarios, all computations must be performed at each sampling time [12]. This complexity increases further when constraints are involved. Although modern computing power reduces this issue to some extent, it is important to consider that many industrial process control computers may not have optimal computational capabilities. Additionally, much of the available processing time is often allocated to tasks other than the control algorithm itself, such as communications, operator interfaces, alarms, and data recording [12].

Moreover, the greatest drawback of MPC is the necessity for an accurate process model [12]. The design algorithm relies on prior knowledge of the model, and while the algorithm itself is independent of the specific model, the performance benefits are directly influenced by the accuracy of the model in representing the real process.

3.2.3 MPC strategy

The methodology of all controllers within the MPC family is characterized by the following strategy, represented in the figure 2.1 [12],[32]:

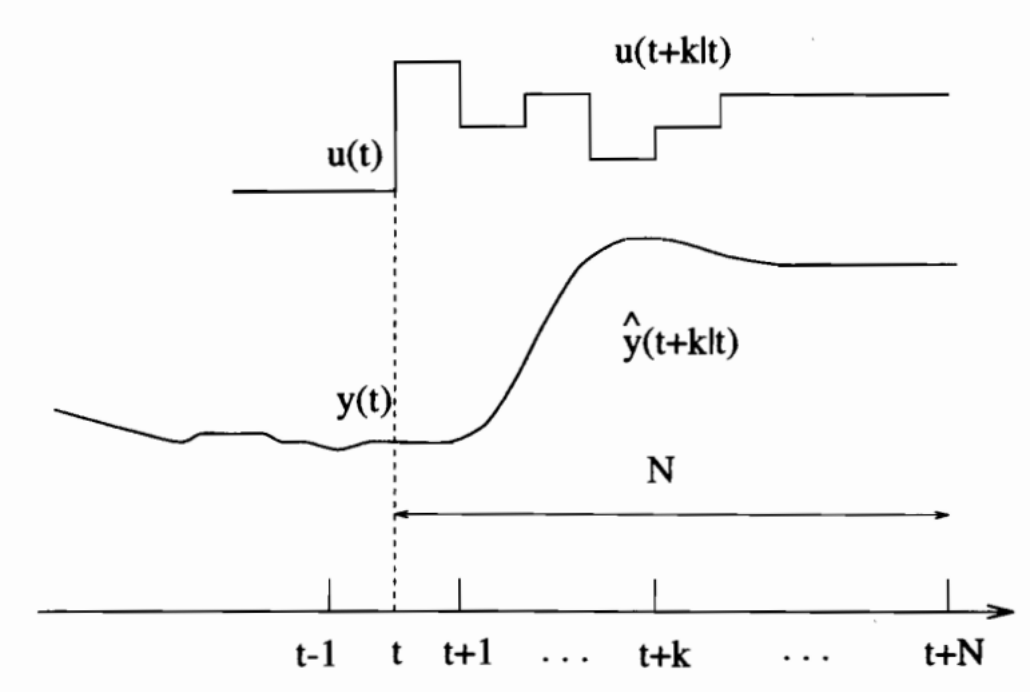


Figure 3.1: MPC Strategy [12]

1. At each instant t, the future outputs for a determined horizon N, called the prediction horizon, are predicted using the process model. These predicted outputs $y(t+k|t)$ for $k = 1, \ldots, N$ depend on the known values up to instant t (past inputs and outputs) and on the future control signals $u(t+k|t)$ for $k = 0, \ldots, N-1$, which are the signals to be calculated and sent to the system[32, 12].

2. The set of future control signals is calculated by optimizing a determined criterion to keep the process as close as possible to the reference trajectory $r(t+k)$ (which can be the setpoint itself or a close approximation of it). This criterion typically takes the form of a quadratic function of the errors between the predicted output signal and the predicted reference trajectory. In most cases, the control effort is also included in the objective function. An explicit solution can be obtained if the criterion is quadratic, the model is linear, and there are no constraints; otherwise, an iterative optimization method must be used. In some cases, assumptions about the structure of the future control law are made, such as the control law being constant from a given instant [32, 12].

3. The control signal $u(t|t)$ is sent to the process, while the subsequent control signals calculated are discarded. At the next sampling instant, $y(t+1)$ is known, and step 1 is repeated with this new value, updating all sequences. Thus, $u(t+1|t+1)$ is calculated (which, in principle, will differ from $u(t+1|t)$ due to the new information available) using the receding horizon concept [32, 12].

To implement this strategy, the basic structure depicted in figure 2.2 [12] is employed. A model is utilized to predict future plant outputs based on past and current values, as well as on the proposed optimal future control actions. These actions are calculated by the optimizer, taking into account the cost function (which considers future tracking error) and constraints.

The process model, consequently, plays a crucial role in the controller. The selected model must accurately capture the process dynamics to precisely predict future outputs while being simple to implement and comprehend. Since MPC encompasses a set of different methodologies rather than a singular technique, various types of models are used in different formulations[12].

One of the most popular models in industry is the Truncated Impulse Response Model. This model is straightforward to obtain, requiring only the measurement of the output when the process is excited

with an impulse input [12]. It is widely accepted in industrial practice due to its intuitive nature and applicability to multivariable processes. However, its main drawbacks include the large number of parameters required and its limitation to describing only open-loop stable processes. A closely related model is the Step Response Model, which is derived when the input is a step.

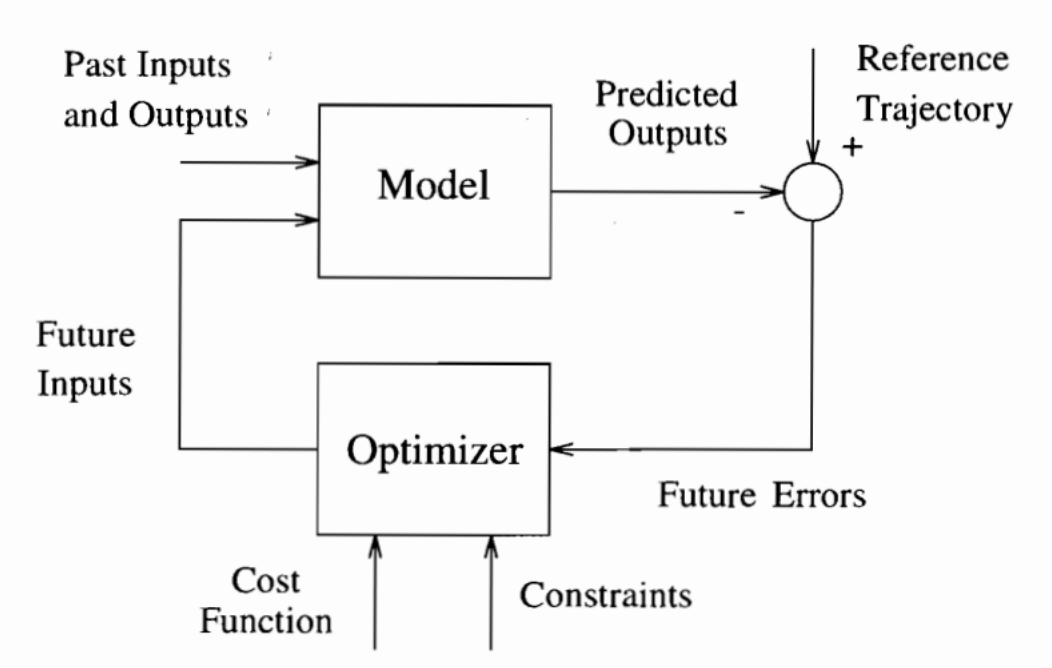


Figure 3.2: MPC Structure [12]

The Transfer Function Model is perhaps the most widespread in the academic community and is used in most control design methods. This model is favored because it requires only a few parameters and is valid for all types of processes. Additionally, the State-Space Model is utilized in some formulations, as it can easily describe multivariable processes [12].

The optimizer is another fundamental component of the MPC strategy as it provides the control actions. If the cost function is quadratic, its minimum can be determined as an explicit function (linear) of past inputs and outputs and the future reference trajectory. However, in the presence of inequality constraints, the solution must be obtained through more computationally intensive numerical algorithms. The size of the optimization problems depends on the number of variables and the prediction horizons used, and they typically result in relatively modest optimization problems that do not require sophisticated computer codes to solve. Nonetheless, the computation time needed for constrained and robust cases can be several orders of magnitude higher than that required for the unconstrained case, significantly reducing the bandwidth of the process to which constrained MPC can be applied[12, 32].

Notice that the MPC strategy is quite similar to the control strategy used in driving a car. The driver is aware of the desired reference trajectory for a finite control horizon and, by considering the car's characteristics (a mental model of the car), decides which control actions (accelerator, brakes, and steering) to take in order to follow the desired trajectory. Only the initial control actions are executed at each moment, and the procedure is repeated for the subsequent control decisions in a receding horizon manner [12]. In contrast, classical control schemes such as PIDs base their control actions on past errors. Extending the car driving analogy, as one commercial MPC vendor (SCAP) has done in their publicity, the PID approach to driving a car would be similar to driving using only the rear-view mirror, as depicted in the next figure. This analogy, however, is not entirely fair to PIDs, since MPC utilizes more information (the reference trajectory). If a future point in the desired reference trajectory is used as the setpoint for the PID, the differences between the two control strategies would not appear as drastic [12].

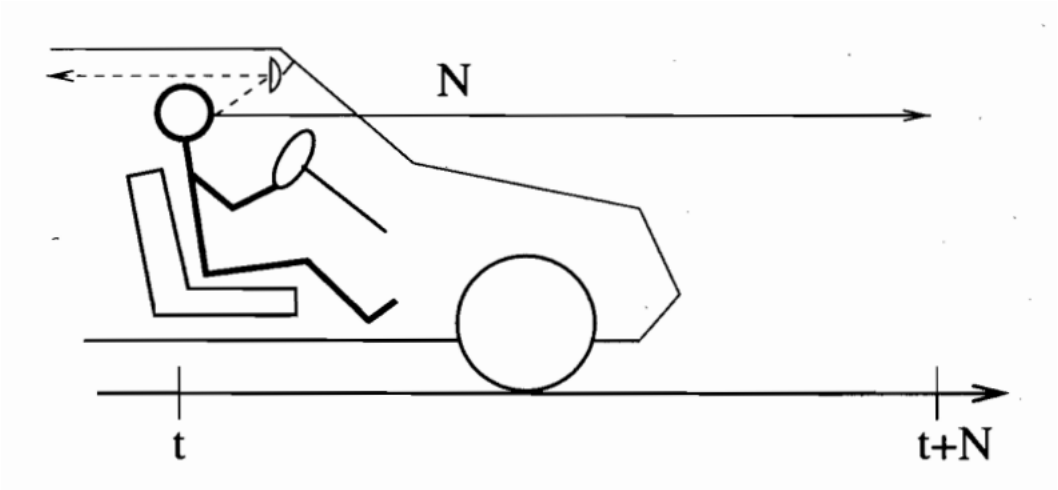


Figure 3.3: MPC Analogy [12]

3.2.4 Stability Analysis of MPC Controllers

The Receding Horizon Control strategy does not ensure the properties of optimal control based methods. The latter being the stability, the feasability and the optimality of the solution. More accurately, RHC does not guarentee that the system will evolve to an equilibrium (such as methods like dynamic programming does) nor to a terminal set. This will lead us to look for possible solutions to ensure the stability for the closed loop system.

We will now give one way to prove the asymptotic stability of a system under a closed loop interaction with an MPC.

Given the system $x(t+1) = f(x(t), u(t))$ with the origin as the equilibrium point ($f(0,0) = 0$). We assume the following [12]:

- The stage cost $l(x, u)$ is a positive-definite function, meaning it is zero only at $l(0, 0)$ and positive elsewhere.
- There exists a local control law $K(x)$ for which the terminal set X_f is invariant. This means that closed-loop trajectories starting within X_f remain in this set indefinitely, and state and input constraints are satisfied ($x \in X_f \implies K(x) \in U$).
- The terminal cost $l_f(x)$ is a Lyapunov function within X_f and satisfies $l_f(x(t+1)) - l_f(x(t)) \leq -l(x, K(x)) \, \forall x \in X_f$.

Before establishing stability, we first need to ensure the recursive feasibility of the MPC. Without this, there is no guarantee that the controller will not fail due to the optimization problem becoming infeasible during closed-loop operation.

To establish recursive feasibility, we assume that the initial state $x(t)$ is feasible, and consider $[u_0^*, u_1^*, \ldots, u_{N-1}^*]$ as the optimal control sequence for $x(t)$. For the next time step, i.e., $x(t+1)$, the sequence $[u_1^*, u_2^*, \ldots, K(x_N^*)]$ is also feasible due to the second assumption above. Thus, the MPC maintains recursive feasibility, meaning if we start at a feasible initial state $x(t)$, the MPC problem will remain feasible indefinitely.

To establish asymptotic stability, we need to show that the optimal value function $J^*(x(t))$ is a Lyapunov function satisfying the property [12]:

$$J^*(x(t+1)) - J^*(x(t)) < 0 \quad \forall x \neq 0.$$

For the optimal control sequence $[u_0^*, u_1^*, \ldots, u_{N-1}^*]$ calculated at $x(t)$, the optimal value function is defined as:

$$J^*(x(t)) = \sum_{k=0}^{N-1} l(x_k^*, u_k^*) + l_f(x_N^*),$$

whereas for the feasible, sub-optimal control sequence $[u_1^*, u_2^*, \ldots, K(x_N^*)]$ for $x(t+1)$, we have the sub-optimal value function as follows:

$$J'(x(t+1)) = \sum_{k=1}^{N} l(x_k^*, u_k^*) + l_f(\hat{x}_{N+1}),$$

where $\hat{x}_{N+1} = f(x_N^*, K(x_N^*))$. Since this is sub-optimal, we have:

$$J^*(x(t+1)) \leq J'(x(t+1)).$$

We can rewrite $J'(x(t+1))$ as follows:

$$\sum_{k=0}^{N-1} l(x_k^*, u_k^*) - l(x_0^*, u_0^*) + l_f(\hat{x}_{N+1}) + l(x_N^*, K(x_N^*)).$$

Adding and subtracting $l_f(x_N^*)$ to the above expression, we get:

$$\sum_{k=0}^{N-1} l(x_k^*, u_k^*) + l_f(x_N^*) - l_f(x_N^*) - l(x_0^*, u_0^*) + l_f(\hat{x}_{N+1}) + l(x_N^*, K(x_N^*)).$$

Notice that the first two terms above equal $J^*(x(t))$, thus we can rewrite the expression as follows:

$$J^*(x(t)) - l(x_0^*, u_0^*) - l_f(x_N^*) + l_f(\hat{x}_{N+1}) + l(x_N^*, K(x_N^*)).$$

From the third assumption, we know that $l_f(x)$ is a Lyapunov function satisfying $l_f(x(t+1)) - l_f(x(t)) \leq -l(x, K(x)) \, \forall x \in X_f$. Therefore, for the last three terms in the above expression, we have:

$$-l_f(x_N^*) + l_f(\hat{x}_{N+1}) + l(x_N^*, K(x_N^*)) \leq 0,$$

leaving us with the expression:

$$J^*(x(t+1)) - J^*(x(t)) \leq -l(x_0^*, u_0^*).$$

In the first assumption, we stated that $l(x, u)$ is strictly positive and is zero only for $l(0, 0)$. Thus, we have:

$$J^*(x(t+1)) - J^*(x(t)) < 0 \quad \forall x \neq 0,$$

indicating that the optimal value function $J^*(x(t))$ is strictly decreasing along closed-loop trajectories. Therefore, these trajectories will converge to the origin, establishing the asymptotic stability of the MPC at the origin.

Important Note : In the above development of the expressions, we assumed that the terminal cost and the terminal constraint are already given. They can be precomputed offline with the mentioned properties to establish stability.

For our case, we will not precompute them but we will base our study on previously adopted cost functions that have been shown to give stability for the coupled tank process [63] [32]. The latter being only a quadratic cost function similar to the classical LQR function with Q and R as positive defined matrices such that:

$$J = \min \sum_{i=1}^{p} ||x_{k+i|k} - x_r||_Q + \sum_{i=0}^{c} ||u_{k+i|k}||_R \tag{3.1}$$

This function will not only be used for the linear MPC formulation but also for the nonlinear one. The details are given in the section down below.

3.3 Controller design and Application on the QTP

The first discussion is about the dynamics models used for the MPC context. Given the following difference equation:

$$x_{k+1} = f_k(x_k, u_k), \tag{3.2}$$

$$y_k = h(x_k, u_k), \tag{3.3}$$

$$x(k_0) = x_0, \tag{3.4}$$

where $x_k \in \mathbb{R}^{n_x}$ are the states, $u_k \in \mathbb{R}^{n_u}$ are the inputs and $y_k \in \mathbb{R}^p$ are the outputs. The initial condition in 3.4 specifies the state value at $k = k_0$, normally the initial time will be at zero as a corresponding initial condition for the system [32].

In the following we will only consider the Discrete Time-Invariant Systems (DTIS) case. This means that a DTIS does not depend explicitly on time, and this fact occurs as long as the problem formulation does not depend explicitly on time, i.e., the model of the system, cost function, and constraints need to be independent of time [32]. This means that would be very accurate with respect to the system's model depicted in 2.10.

3.3.0.1 Linear Model Predictive Control

Recall the general formulation for any dynamics model from 3.2, a linear model (using state representation) formulation could be presented as the following [32]:

$$x_{k+1} = Ax_k + Bu_k, \tag{3.5}$$

$$y_k = Cx_k + Du_k, \tag{3.6}$$

$$x_{k_0} = x_0, \tag{3.7}$$

where $A \in \mathbb{R}^{n_x \times n_x}$, $B \in \mathbb{R}^{n_x \times n_u}$, $C \in \mathbb{R}^p \times n_x$, and $D \in \mathbb{R}^p \times n_u$ are time-invariant matrices defining the dynamics of the system.

With the aim of applying linear control theory, the model described by 3.2 has to be linearized into the form of linear equations. Therefore, the variables $x_i := h_i - h_i^0$ and $u_i := v_i - v_i^0$ are introduced, where h_i^0 and v_i^0 are the operating points around which the linear model is considered valid. The linearized continuous-time state-space representation is then [32]

$$\frac{dx(t)}{dt} = \begin{pmatrix} -\frac{1}{T_1} & 0 & \frac{A_3}{A_1 T_3} & 0 \\ 0 & -\frac{1}{T_2} & 0 & \frac{A_4}{A_2 T_4} \\ 0 & 0 & -\frac{1}{T_3} & 0 \\ 0 & 0 & 0 & -\frac{1}{T_4} \end{pmatrix} x(t) + \begin{pmatrix} \frac{\gamma_1 k_1}{A_1} & 0 \\ 0 & \frac{\gamma_2 k_2}{A_2} \\ 0 & \frac{(1-\gamma_2)k_2}{A_3} \\ \frac{(1-\gamma_1)k_1}{A_4} & 0 \end{pmatrix} u(t), \tag{3.8}$$

$$y(t) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} x(t), \tag{3.9}$$

where:

$$T_i = \frac{A_i a_1 s^2 h_i^0}{g}, i = 1, \ldots, 4$$

The two chosen operating points for this model will be the values $h_{1*} = 10$[cm] and $h_{2*} = 10$[cm] [32] in all the LMPCs designed down in this thesis. The results were very satisfactory.

From now on, the plant will be assumed to be a discrete linear model, and the cost function and constraints from the MPC formulation are quadratic and linear inequalities, respectively. Recovering the MPC formulation used in previously, the LMPC problem implemented is:

$$\begin{aligned} &\min_{u_{k|k},\ldots,u_{k+H_p-1|k}} \sum_{i=1}^{H_p} ||x_{k+i|k} - x_r||_Q + \sum_{i=0}^{H_c-1} ||u_{k+i|k}||_R \\ &\text{subject to} \\ &x_{k+i+1|k} = Ax_{k+i|k} + Bu_{k+i|k}, \\ &u_{k+i|k} \in U, \\ &x_{k+i|k} \in X, \end{aligned} \tag{3.10}$$

where $x_r \in \mathbb{R}^{n_x}$ is a constant desired set-point. Matrices $Q \in \mathbb{R}^{n_x \times n_x}$ and $R \in \mathbb{R}^{n_u \times n_u}$ are penalization weights assigning prioritization for the control objectives. Both states and inputs are subject to some physical and operating constraints defined as $X = \{x \in \mathbb{R}^{n_x} : \underline{x} \leq x_k \leq \overline{x}\, \forall k\}$ and $U = \{u \in \mathbb{R}^{n_u} : \underline{u} \leq u_k \leq \overline{u}\, \forall k\}$, respectively, where $\underline{x}$ and $\overline{x}$ correspond to the lower and upper limits for the states. Similarly, $\underline{u}$ and $\overline{u}$ are the lower and upper limits for the control signals, respectively. Moreover, H_p and H_c refer to the prediction and control horizons. It is assumed that each state is measurable; if not, a state observer should be implemented. Regardless of this fact, we sure know that the experimental plant has indeed 4 sensors that gives the real time liquid heights in the tanks without any problem.

Normally, for implementation purposes, it is preferred to have the predictions expressed in terms of $\Delta u_{k+i|k}$ (slew rate) rather than $u_{k+i|k}$, where $\Delta u_{k+i|k}$ is defined as

$$\Delta u_{k+i|k} = u_{k+i|k} - u_{k-1}. \tag{3.11}$$

This means that the change in the input will be penalized, from the previous iteration, rather than penalizing the input itself. The OCP, therefore, will also be defined as:

$$
\begin{aligned}
&\min_{u_{k|k},\ldots,u_{k+H_p-1|k}} \sum_{i=1}^{H_p} ||x_{k+i|k} - x_r||_Q + \sum_{i=0}^{H_c-1} ||\Delta u_{k+i|k}||_R \\
&\text{subject to} \\
&x_{k+i+1|k} = Ax_{k+i|k} + Bu_{k+i|k}, \\
&u_{k+i|k} \in U, \\
&x_{k+i|k} \in X,
\end{aligned}
\tag{3.12}
$$

where all the variables retain their previously defined meanings, and $\Delta u_{k+i|k} \in \mathbb{R}^{n_u}$.

It is important to notice that this OCP, with some mathematical transformations, can be expressed as a Quadratic Programming (QP) problem of the form,

$$
\begin{aligned}
&\min_z \frac{1}{2} z^T H z + g^T z \\
&\text{subject to} \\
&\underline{\alpha} \leq \Lambda z \leq \overline{\alpha},
\end{aligned}
\tag{3.13}
$$

where $H \in \mathbb{R}^{(n_x+n_u)\times(n_x+n_u)}$ is the Hessian matrix defined as positive semi-definite, i.e., it defines a convex optimization problem, and therefore, any locally optimal solution is also globally optimal [32]. The decision vector $z \in \mathbb{R}^{n_x+n_u}$ is defined as $z = [x^T\, u^T]^T$. Moreover, $\underline{\alpha}, \overline{\alpha} \in \mathbb{R}^{n_x}$ and $\Lambda \in \mathbb{R}^{n_x\times(n_x+n_u)}$ define the feasible set, and $g \in \mathbb{R}^{n_x+n_u}$ is the gradient vector. Depending on how the Hessian matrix H is defined, the problem will be either convex or not. To ensure the convexity of the problem, H must be defined as positive semi-definite. This is crucial since any locally optimal point of a convex problem is (globally) optimal [32]. Specialized solvers are going to be used in the next sections to actually solve that OCP without any issues.

3.3.1 Nonlinear Model Predictive Control

Nonlinear optimal control algorithms are at the center of all Nonlinear Model Predictive Control (NMPC). In the previous section, it has been shown that for the linear problem, mostly convex quadratic programs (QP) are solved exactly at each sampling time. Conversely, NMPC algorithms should deal with two possibilities: solving the problem until a specified convergence criterion is reached (this procedure may introduce delays between the numerical algorithms and the system to be controlled), or stopping the algorithm prematurely, which means only approximate solutions are used. In the following section, the NMPC problem has been solved using the nonlinear programming library CasAdi available on Matlab and Python. This library is considered one of the main libraries to implement OCPs in general.

If the general formulation for a dynamic system from 3.2 and the formulation for the MPC design are recovered, the nonlinear optimization problem behind the MPC can be defined in the following way:

$$
\begin{aligned}
&\min_{u_{k|k},\ldots,u_{k+N-1|k}} \sum_{i=0}^{N-1} l_i(x_{k+i|k}, u_{k+i|k}) \\
&\text{subject to} \\
&x_{k+i+1|k} = f(x_{k+i|k}, u_{k+i|k}), \quad k = 0,\ldots,N-1, \\
&x_{k|k} = x_k, \quad k = 0,\ldots,N-1, \\
&(x_{k+i|k} \in X, \quad k = 0,\ldots,N, \\
&u_{k+i|k} \in U, \quad k = 0,\ldots,N-1), \\
&x_{k+Hp|k} \in X_f.
\end{aligned}
\tag{3.14}
$$

The OCP in 3.14 can be solved using sequential or simultaneous approaches. In the former approach, at each optimization iteration the two steps, simulation and optimization, are performed sequentially, i.e., one after the other. Using a simultaneous approach, the optimization problem is addressed by a Newton-type optimization algorithm (it computes at each step a linearization of the system) by which the optimization and the simulation are performed simultaneously. This type of approach usually involves direct collocation methods as well as direct multiple shooting to construct the nonlinear programming (NLP) problem to be solved. The two best-known Newton-type optimization algorithms are interior-point (IP) methods and sequential quadratic programming (SQP).

Single shooting algorithms consider only the control signal as a decision variable while the states is considered part of the system. This would induce nonlinearities on the states due to propagating the state function $f(x)$ over our prediction horizon i.e. $x_{N+1} = f(x_N) = f(f(x_{N-1})) = ... = f^N(x_0)$.

Given f is a nonlinear function, the effects of doing such **single shooting** beginning from a point x_0 would be drastic. This is why considering the states x also as a decision variable in addition to the controls vector u is the most recommended. The state system equation would then be considered a constraint for our OCP problem and it is verified every single sampling time for our algorithm.

3.4 Software tools for Controller design

3.4.1 MPC Toolbox

The Model Predictive Control Toolbox is a Matlab based library that provides functions, Simulink blocks and reference examples through a Gui-based user end product. It supports the design of implicit, explicit, adaptive and gain-scheduled MPC. It does also provide ways to implement NMPCs, notably by single and multi-stage nonlinear MPC. Additionally, it provides deployable optimization solvers and enables the use of custom solvers.

This simplified toolbox is based on the formulas presented in 3.13 and 3.14. The controller is then designed automatically after providing the linear or the nonlinear plant model to the Toolbox. It is arguably the easiest way to design an MPC on Matlab compared to the other tools that we're going to introduce further on in this section.

3.4.1.1 Simulation results

The following figures shows the simulation results of using the LMPC and the NMPC respectively designed using the corresponding Toolbox.

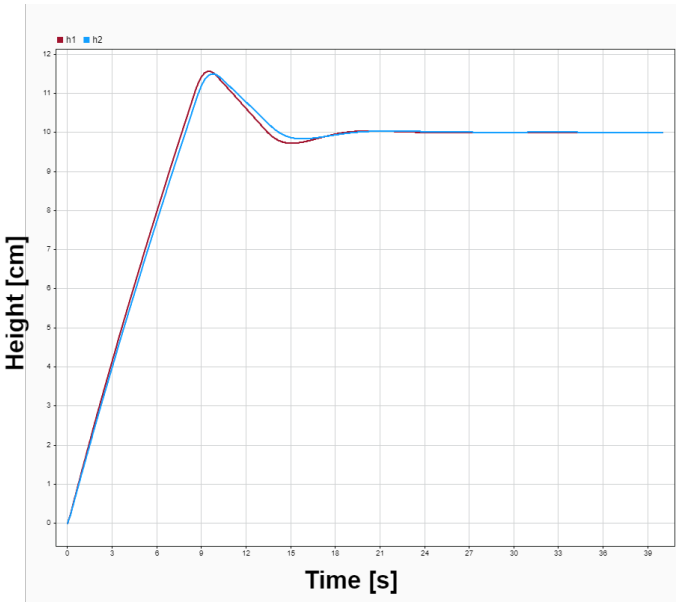


Figure 3.4: System's response h_1 (red) and h_2 (blue) to two step reference signals of value 10 in the minimum phase setting using the Matlab Toolbox LMPC

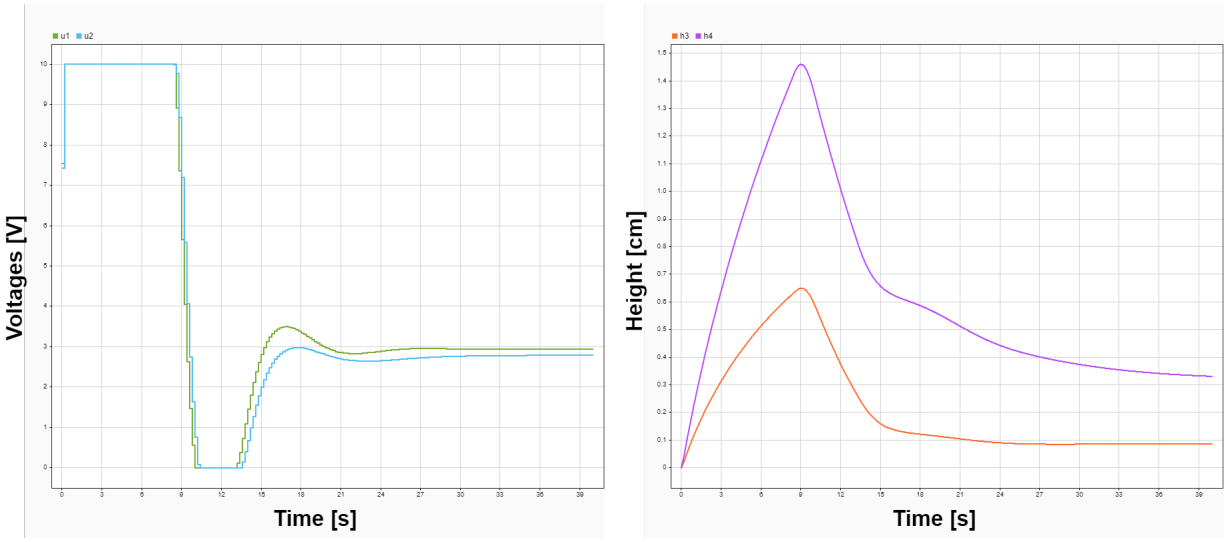


Figure 3.5: Corresponding control signals (left) and h_3 (orange) and h_4 (purple) (right) for the system's response to two step signals of value 10 using the Matlab Toolbox LMPC

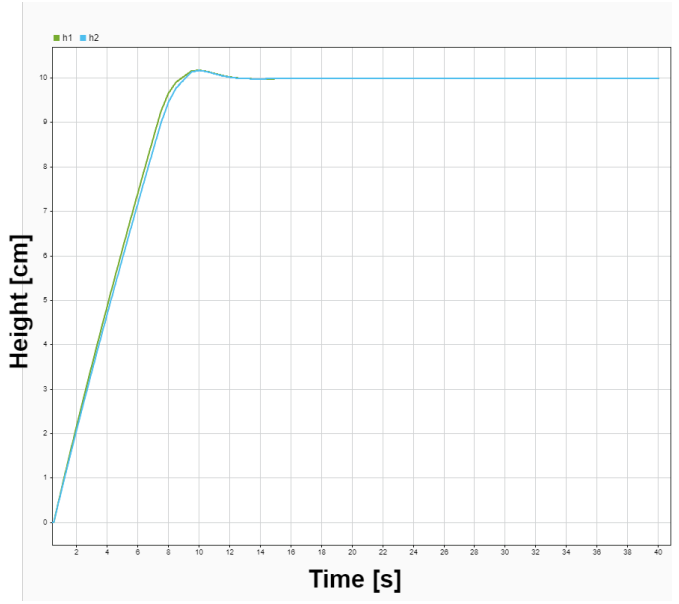


Figure 3.6: System's response h_1 (green) and h_2 (blue) to two step reference signals of value 10 in the minimum phase setting using the Matlab Toolbox NLMPC

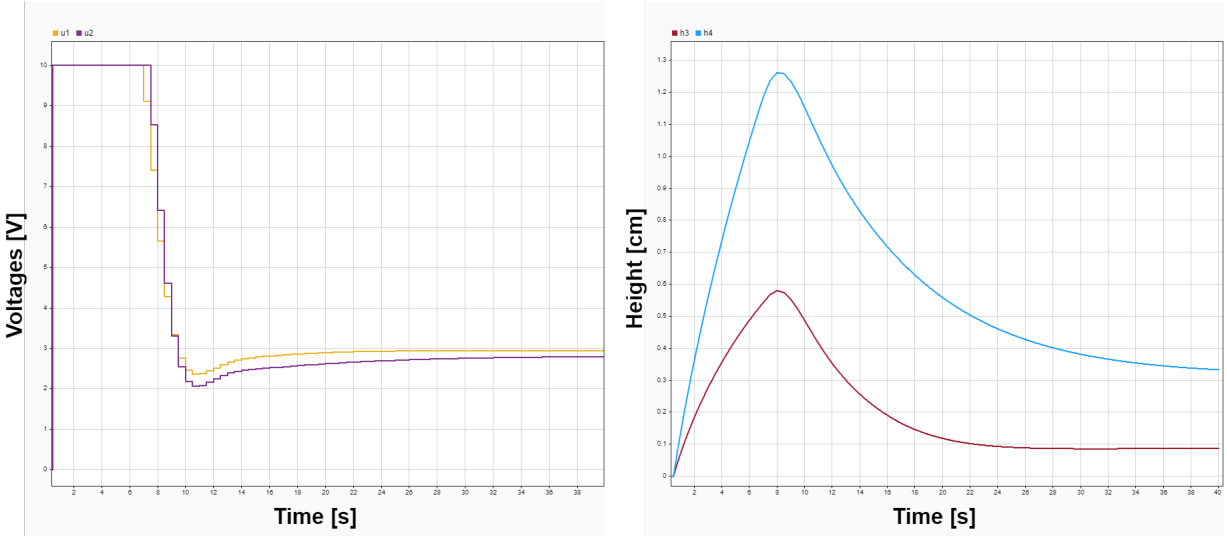


Figure 3.7: Corresponding control signals (left) and h_3 (brown) and h_4 (blue) (right) for the system's response to two step signals of value 10 using the Matlab Toolbox NLMPC

3.4.2 FiOrdOs

FiOrdOs is a versatile code generator toolbox developed by Fabian Ullmann and Stefan Richter at ETH Zurich [54], designed to interface with MATLAB for solving parametric convex problems using first-order methods. This toolbox is particularly useful for handling optimization problems formulated as follows:

$$\begin{aligned} \min_{x} \quad & \frac{1}{2} x^T H x + g^T x + c \\ \text{subject to} \quad & x \in X, \\ & Ax = b, \end{aligned} \tag{3.15}$$

where $x \in \mathbb{R}^n$ is the decision variable, $H \in S^n$ is the Hessian matrix (which is positive semidefinite, ensuring convexity), $g \in \mathbb{R}^n$, $c \in \mathbb{R}$, $A \in \mathbb{R}^{m \times n}$, and $b \in \mathbb{R}^m$ are matrices and vectors that define the cost function and constraints [54].

The analogy is then clear between the latter formula and the OCP defined in 3.13. We will use the same basic concept to define a Linear MPC using the FiOrdOs library.

To solve these optimization problems, FiOrdOs uses the Fast Gradient Method, a first-order method introduced by Yurii Nesterov in [47]. The latter is a variation of the classical gradient method with additional terms to give faster convergence properties.

3.4.2.1 LMPC design and Code Generation

As we described earlier, the similarities between our OCP for the linear MPC problem and vanilla convex optimization problems could let us use first order methods to design that.

When the optimization problem is specified and translated into something that could be interpreted by FiOrdOs, the library uses a user defined solver that gives the required solution. In this case, we will pre-compute the matrices needed for solving the OCP as the linear model around the chosen operating point is fixed through time.

The question now is how do we translate that mathematically, the answer is given in the following ; Let us consider a problem with box constraints on the inputs [54]:

$$U = \{u \in \mathbb{R}^{n_u} \mid 0 \leq u \leq 10\},$$

but no state constraints, i.e.,

$$\mathcal{X} = \mathbb{R}^n.$$

To obtain a stable closed loop, we take $P = \text{dlyap}(A^*, Q)$ and $\mathcal{X}_f = \mathbb{R}^n$ [54].

The typical procedure is to eliminate the states via :

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \underbrace{\begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}}_{A} x_0 + \underbrace{\begin{bmatrix} B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & AB & \cdots & B \end{bmatrix}}_{B} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix},$$

Such that the optimization problem can be written in condensed form (where the decision variable is the sequence of inputs U only) as [54]

$$\begin{aligned} \min_{U} \quad & \frac{1}{2} U^T H U + g(x_0)^T U + c(x_0) \\ \text{s.t.} \quad & U \in \mathcal{U}^N \end{aligned}$$

where

$$\begin{aligned} H &= (B^T Q B + R), \\ g(x_0) &= B^T Q A x_0, \\ c(x_0) &= \frac{1}{2} x_0^T (A^T Q A + Q) x_0, \end{aligned}$$

with

$$Q = \begin{bmatrix} Q & & \\ & \ddots & \\ & & Q \end{bmatrix}, \quad R = \begin{bmatrix} R & & \\ & \ddots & \\ & & R \end{bmatrix}.$$

Thus, the definition of our OCP as a first order optimization problem with constraints on the input.

Additionally, the library has a Matlab function, named *generateCode* which allows the generation of C code files. The files in question are **_solver.c* and **_solver.h* where the fast Gradient Method and other functions are defined there respectively. Ultimately, we can also generate a Simulink s-function bloc that uses the **.mex* files to automate the computation.

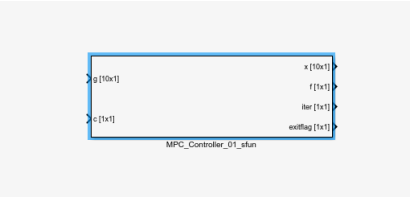


Figure 3.8: Bloc generated by FiOrdOs

3.4.2.2 Simulation results

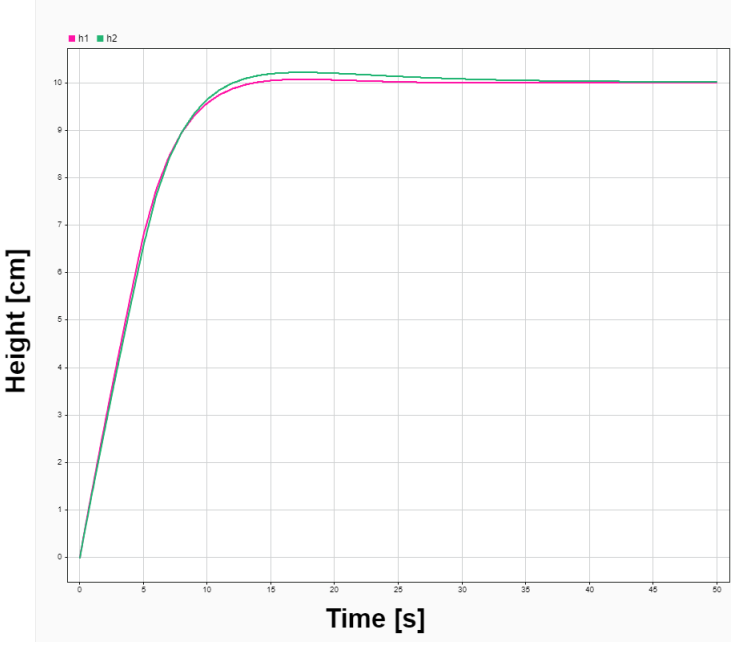


Figure 3.9: System's response h_1 (red) and h_2 (green) to two step reference signals of value 10 in the minimum phase setting using FiOrdOs LMPC

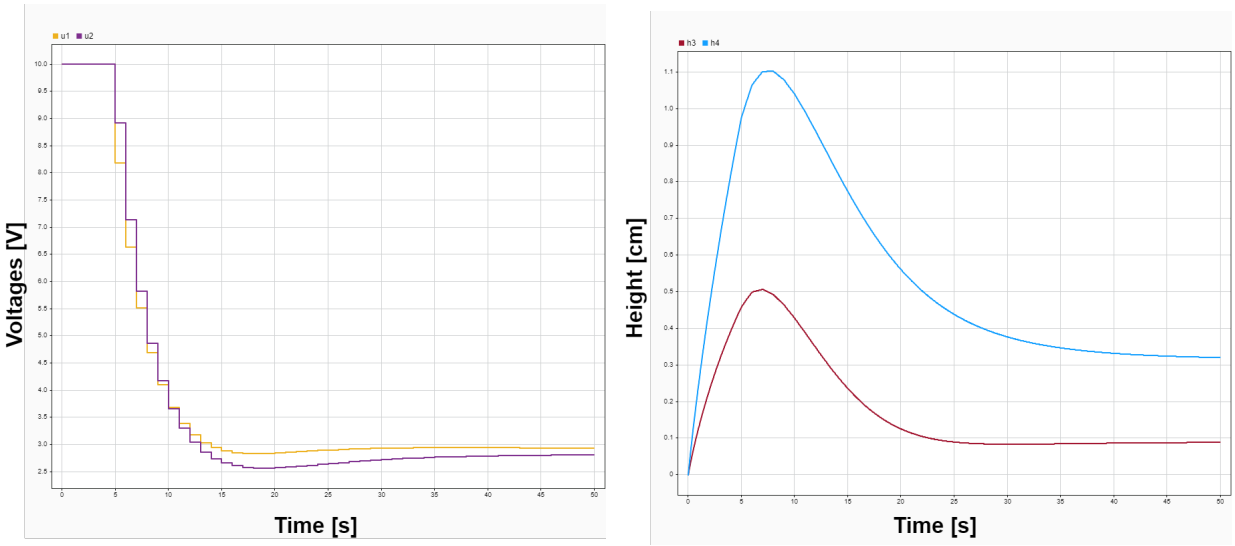


Figure 3.10: Corresponding Control signals (left) and the h_3 (brown) and h_4 (blue) (right) for the system's response to two step signals of value 10 using FiOrdOs LMPC

3.4.3 CasAdi

CasADi is an open-source software framework for numerical optimization, it offers a significant flexibility compared to your typical algebric modelling language. This flexibility is particularly beneficial for problems constrained by differential equations, such as optimal control problems [7]. CasADi is written in self-contained C++ but is most commonly used through its comprehensive interfaces for Python or Matlab. Since its inception in late 2009, it has been successfully applied in various fields, including process control, robotics, and aerospace, as well as in academic teaching [7].

Generally speaking, the types of problems that could be handled by CasADi are basic OCPs expressed in ordinary differential equations (ODE) such defined in the following [7]:

$$
\begin{aligned}
\min_{x(\cdot),u(\cdot),p} \quad & \int_0^T L(x(t),u(t),p)\,dt + E(x(T),p) \\
\text{subject to} \quad & \dot{x}(t) = f(x(t),u(t),p), \\
& u(t) \in U, \quad x(t) \in X, \quad t \in [0,T], \\
& x(0) \in X_0, \quad x(T) \in X_T, \quad p \in P,
\end{aligned}
\tag{3.16}
$$

where $x(t) \in \mathbb{R}^{N_x}$ represents the differential states, $u(t) \in \mathbb{R}^{N_u}$ denotes the control signals, and $p \in \mathbb{R}^{N_p}$ are the free parameters. The OCP includes a Lagrange term L, a Mayer term E, and an ODE with initial X_0 and terminal X_T conditions, along with admissible sets for states X, controls U, and parameters P [7].

These types of problems can be efficiently solved using direct methods, such as direct collocation and direct multiple shooting, which transcribe the OCP into a nonlinear program (NLP):

$$
\begin{aligned}
\min_{w} \quad & J(w) \\
\text{subject to} \quad & g(w) = 0, \\
& w \in W,
\end{aligned}
\tag{3.17}
$$

where $w \in \mathbb{R}^{N_w}$ is the decision variable, J is the objective function, and W represents the interval set.

3.4.3.1 NMPC design and Code generation

Seeing the same analogy between the Nonlinear program defined earlier and our problem in 3.14 we can transform fit the OCP for the NMPC case for this task. Multiple shooting setting is the main algorithm used to solve these type of problems [7].

Similarly to FiOrdOs, CasAdi is also able to perform c code generation beginning from it's unitary entity called a CasAdi function. The latter is similar to the classical Matlab symbolic function but with other functionalities that incorporates the solver alongside an optimizer to find the solution of a given NLP problem. The simulation results are displayed down below.

3.4.3.2 Simulation results

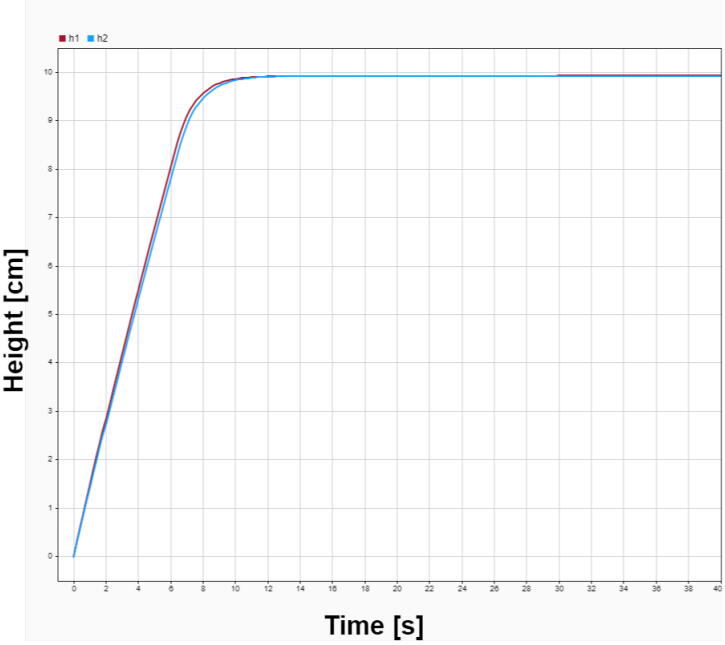


Figure 3.11: System's response h_1 (red) and h_2 (blue) to two step reference signals of value 10 in the minimum phase setting using CasAdi NMPC

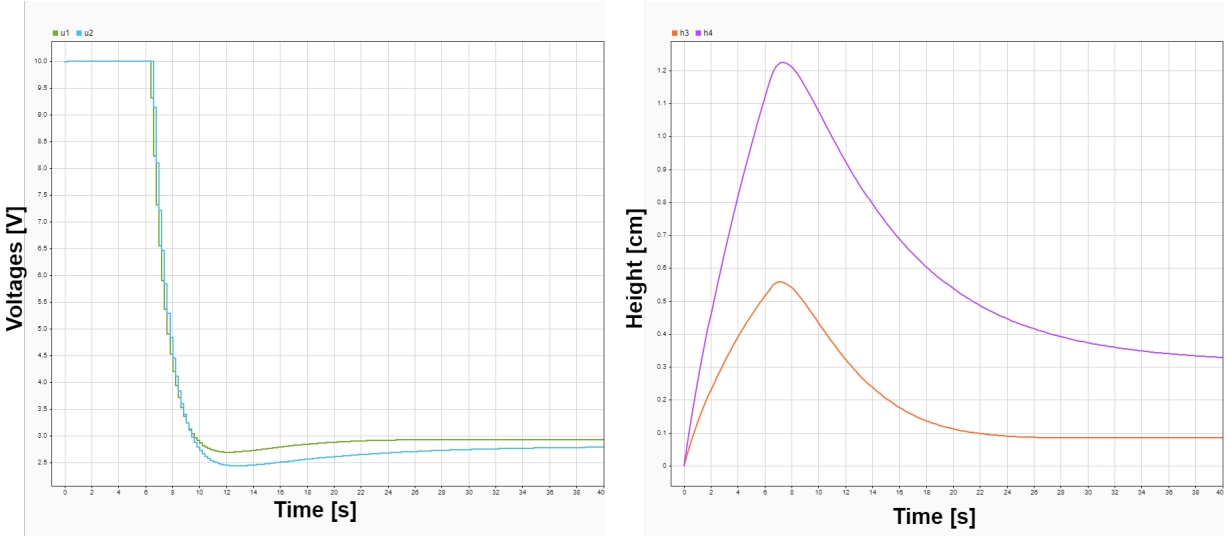


Figure 3.12: Corresponding Control signals (left) and h_3 (orange) and h_4 (purple) (right) for the system's response to two step signals of value 10 using CasAdi NMPC

3.4.4 Controllers assessment and Comparative Study

In the previous subsection, we introduced some tools to implement Model based predictive controllers in their linear and nonlinear aspects. Now as the simulation have been conducted with the same format (two step signals of a value of 10 with null initial conditions on the state vector), we give the following remarks regarding the general behavior of MPC controllers.

First of all, as the MPC strategy is based technically on a finite horizon LQR and a receding horizon strategy, when the error between the reference and the output is very large. The solution of the OCP tends to be the maximum value given by the controller, for our case the value 10[V] for each pump. This value is given for a certain period of time before the controller converges to a more reasonable value. The Latter being the equilibrium points equivalent to the desired output.

This being said, working for too much time under a saturation condition for a nonlinear system could cause a lot of troubles for the system. A deterioration of the performances and even the possibility of destroying the stability [58]. It is wise to remind ourselves that a saturation is inherently a nonlinear component in itself. connecting two of them could affect the phase plan and lead the dynamics towards an unstable limit cycle.

Now for the general behavior of the plant's outputs and states. the latter have a similar behavior in general with a little small distinctions between them. For instance, we can look at the LMPC generated by the Matlab MPC toolbox and see that it has a little overshoot in its response. The overshoot is little to none from the one generated by CasAdi.

For the designed NMPCs, the same observation could be made between the one designed classically and the one generated using CasAdi. As for the states h_3 and h_4, their behavior are quite similar; a rise into the level that corresponds to the maximum input signal generated by our pumps and then a trough is seen as soon as the value decreases a little. This is physically explained by the fact that we're conducting the simulation while being in the minimum phase setting. The latter corresponds to γ ratio that leads us to control the lower levels h_1 and h_2 using the two lower nozzles and not the water pouring down via the gravitational force.

In addition to these observations, we must mention the time consumed by each controller to compute a given control value at each sample. This is crucial for our case as we are planning to implement the

controller in real time on our experimental setup. All of these are detailed in the table down below.

Controller	Mean Computation Time per sampling [sec]	Sampling Time	Ratio	Implementable	System's Time Response (previous test)
FiOrdOs LMPC	0.0350	0.2	17.5%	yes	~ 8 [sec]
Matlab Toolbox LMPC	0.0110	0.2	5.5%	yes	~ 12 [sec]
CasAdi NMPC	0.0746	0.2	37.3%	yes	~ 10 [sec]
Matlab Toolbox NMPC	0.3422	0.5	68.4%	yes	~ 9 [sec]

Based on the previous values, All the designed controllers could be implemented in real time on our experimental plant. Even for the Matlab Toolbox NMPC where the sampling time is higher than the three others. For reference, the experiments done in [32] have used a sampling time of 1 second, so the fourth controller still fulfill the conditions for the implementation.

Saying all of this, the other computations and data operations done at each sampling time (like gathering the data from the sensors..etc) should be added to the time that we have mentioned to confirm whether or not these controllers could be implemented. In the end, the two best lightweight MPCs suited for implementation are the Matlab Toolbox LMPC and CasAdi NMPC.

Seeing the similarities between the controllers and taking into account the performances seen only, we will now conduct the standard experiments (the same ones that we did before in 2.5.2) on the Matlab Toolbox LMPC and NMPC and we obtain the results illustrated in the subsection down below.

3.4.4.1 LMPC in the Minimum Phase setting

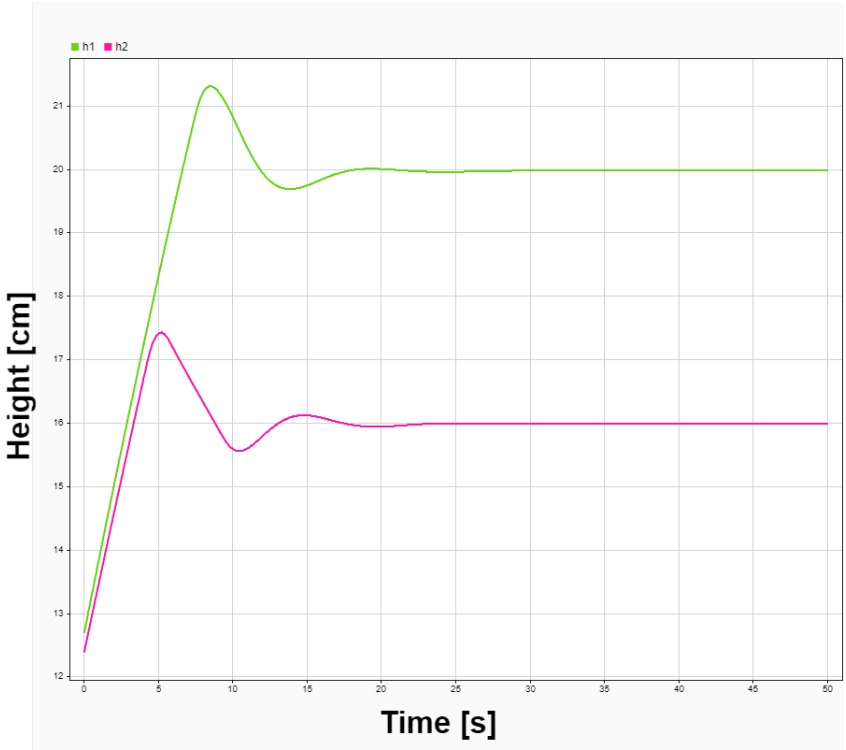


Figure 3.13: System's h_1 (green) and h_2 (rose) response in the standard conditions using the LMPC

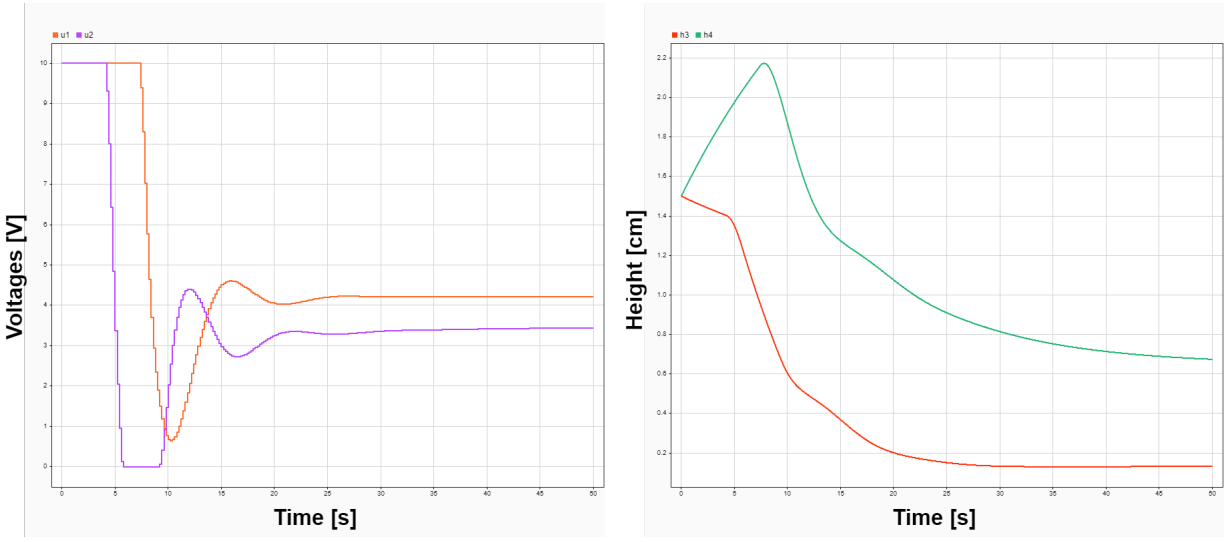


Figure 3.14: Corresponding Control signals (Left) and h_3 (orange) and h_4 (green) (Right) For the mentioned test

3.4.4.2 LMPC in the Non-minimum Phase setting

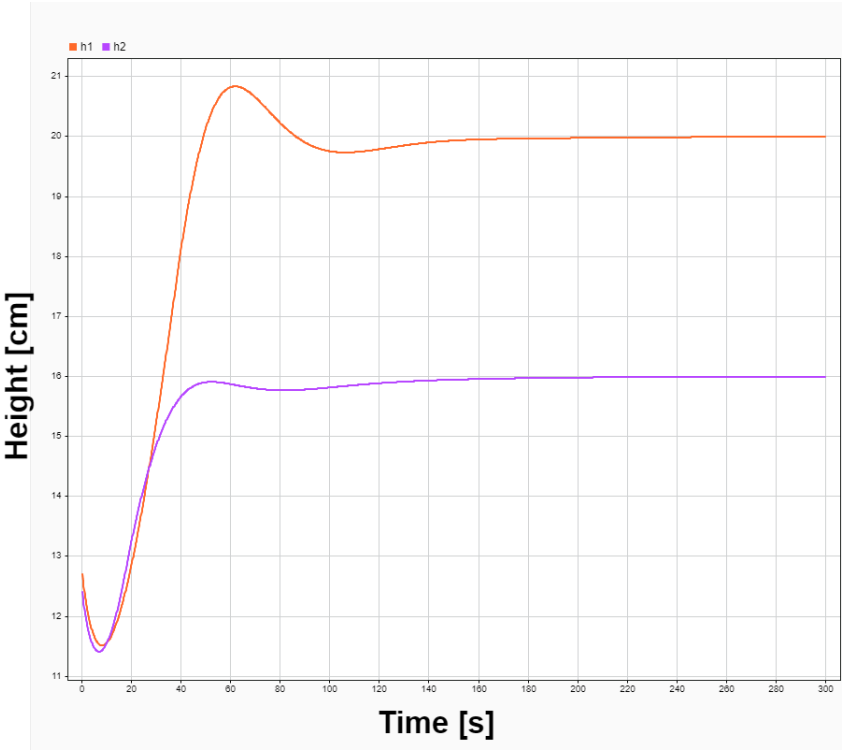


Figure 3.15: System's response h_1 (orange) and h_2 (purple) in the standard conditions using the LMPC

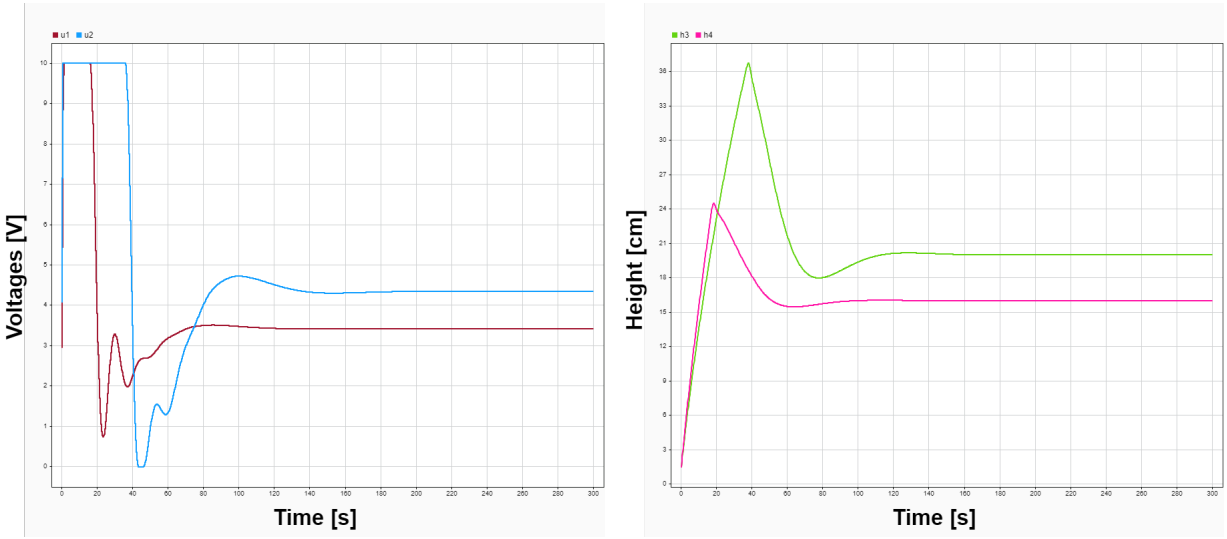


Figure 3.16: Corresponding Control signals (Left) and h_3 (green) and h_4 (rose) (Right) For the mentioned test

3.4.4.3 NMPC in the Minimum Phase setting

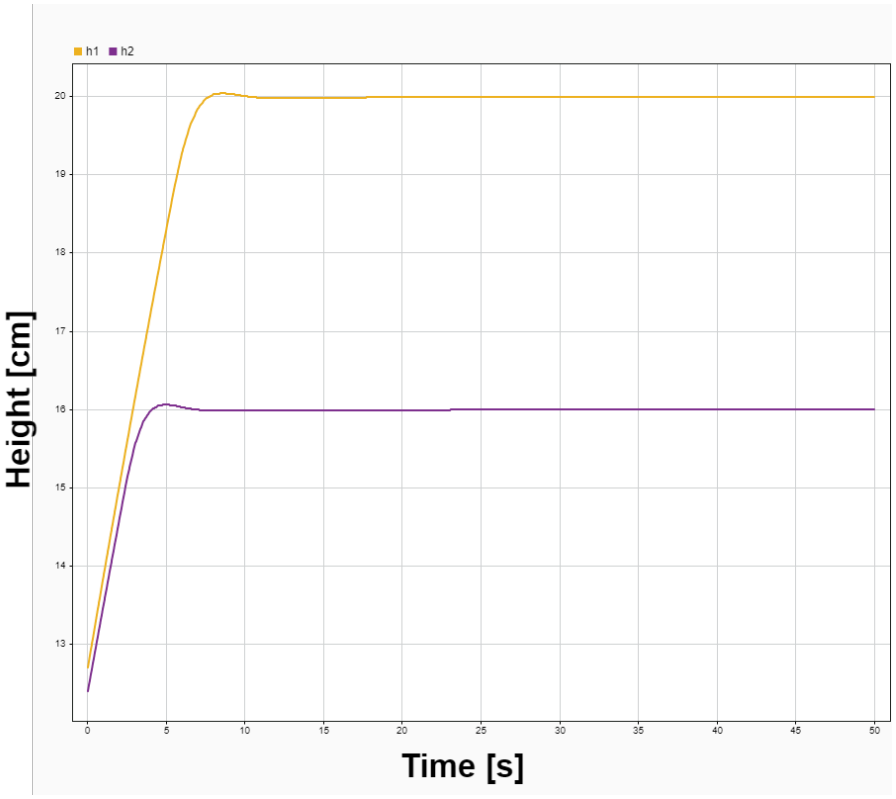


Figure 3.17: System's response h_1 (yellow) and h_2 (purple) in the standard conditions using the NMPC

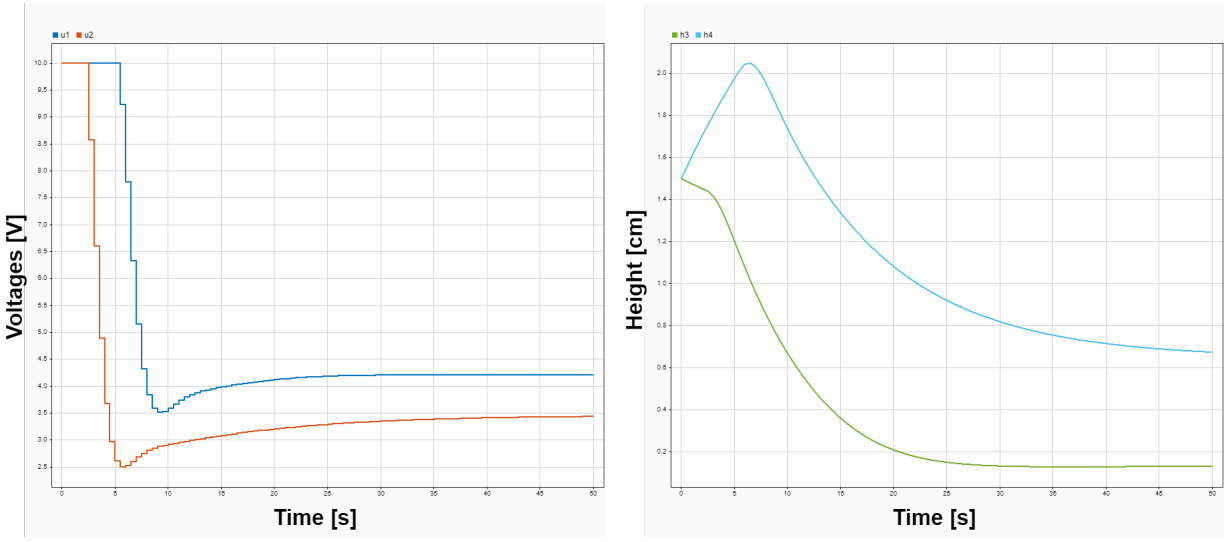


Figure 3.18: Corresponding Control signals (Left) and h_3 (green) and h_4 (blue) (Right) For the mentioned test

3.4.4.4 NMPC in the Non-minimum Phase setting

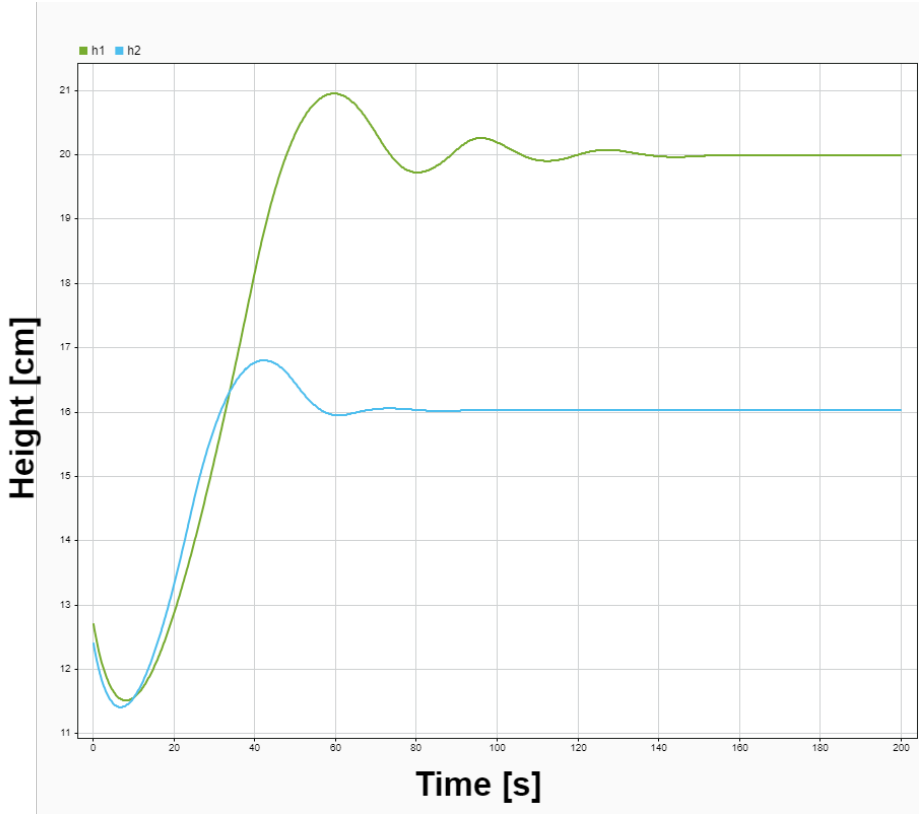


Figure 3.19: System's response h_1 (green) and h_2 (blue) in the standard conditions using the NMPC

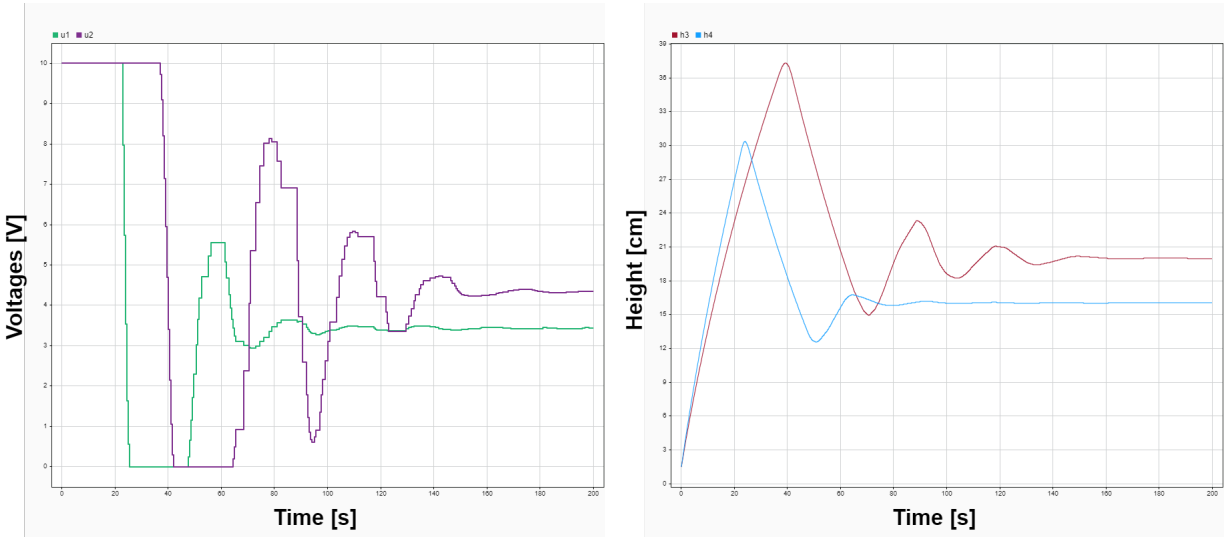


Figure 3.20: Corresponding Control signals (Left) and h_3 (brown) and h_2 (blue) (Right) For the mentioned test

3.4.4.5 Remarks and Observations

After conducting all these simulations, quite a few observations could be made.

First of all, All the control objectives have been reached for both controllers designed in different settings (Min and non-min settings). The response's shape has some pseudo-oscillations for the LMPC in both settings as well as the NMPC in the non-minimum phase setting. The latter has a response similar to the behavior of a first order system in it's minimum phase setting with just a little to none overshoot that's barely around 0.5% of the final value.

Regarding the control signals, we can see that the signal is smoother for the LMPC whilst the NMPC has more of a stair case signal type. This is mainly due to the sampling time that was chosen before the simulations where it was 0.2 seconds for the Linear predictive controller and 0.5 seconds for the Nonlinear one. For reference, [32] used a sampling time of 1 second and the results were very satisfactory for the conducted simulations and experimentation.

The significant observation that could be made comparing these results to the PID control scheme is the drastic improvement in the non-minimum phase setting step response time for both the Linear and the Nonlinear model predictive controllers. In fact, this time was around 340 seconds for the benchmark controller and it's now around 90 seconds for the NMPC and 100 seconds for the LMPC. This corresponds to reducing that time by around 70% of it's original configuration and shows the effectiveness of the predictive control strategies.

3.5 Robustness Tests and Validation of the Controller

For the next part, we will see how robust are the designed controllers with respect to the tests that we defined before in 2.5.3.1 and 2.5.3.2.

3.5.1 LMPC

3.5.1.1 Robustness with respect to High viscosity fluids

As we mentioned before, this test is based on the discharge coefficient C_d that could be considered at the output of each nozzle in our system.

The values taken for this test are the same as in 2.5.3.1 and the results are displayed down below for the minimum and the non-minimum phase settings. The latter being designed at $\gamma_1 = \gamma_2 = 0$ and will be used again in the next test.

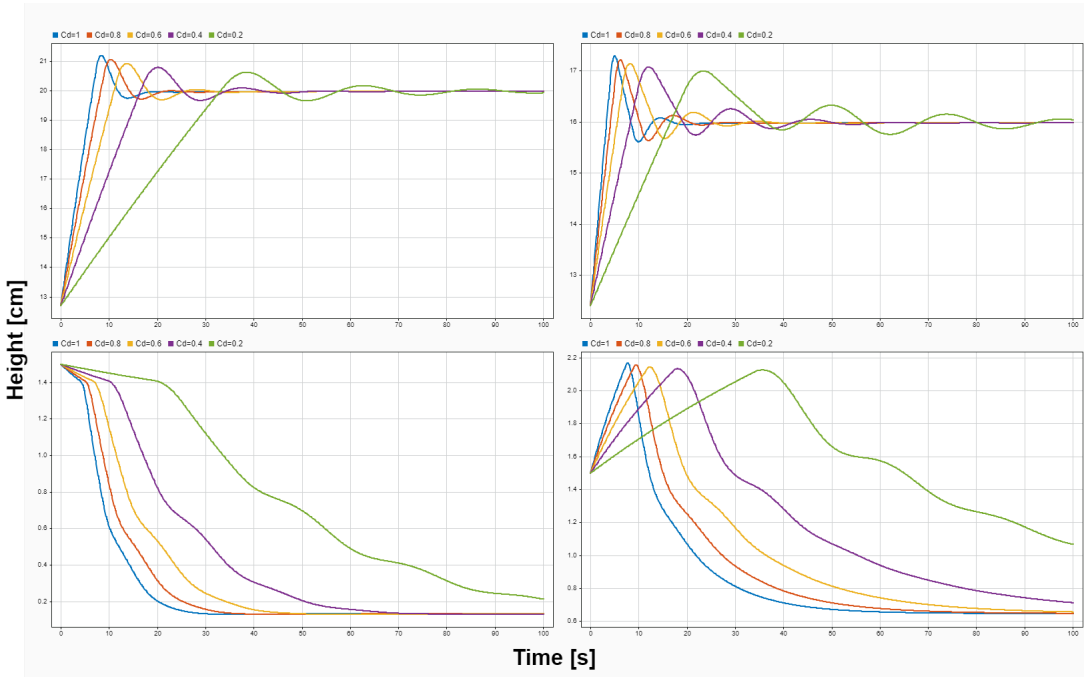


Figure 3.21: Evolution of the System's states under different values of C_d for the minimum phase setting, Upper left h1, Upper right h2, Lower left h3, Lower right h4

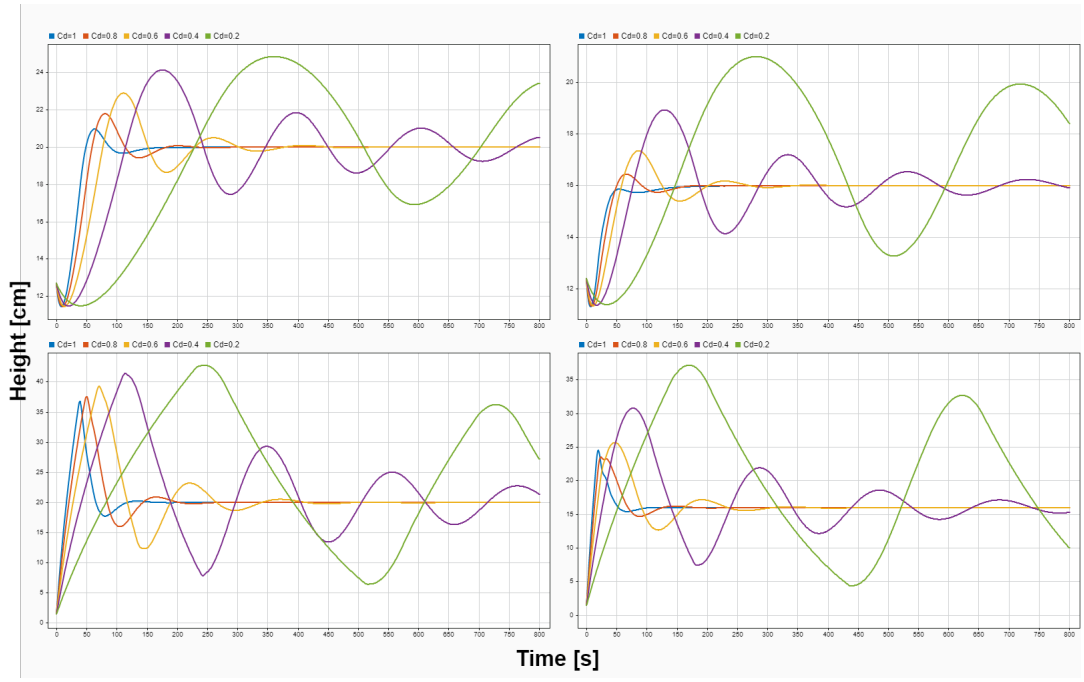


Figure 3.22: Evolution of the System's states under different values of C_d for the non-minimum phase setting, Upper left h1, Upper right h2, Lower left h3, Lower right h4

As we could explain early on in 2.5.3.1, high viscosity fluids like oil induce a latency into the system's dynamics. The flow is then changed and of course reduced compared to the original flow. This doesn't change from a controller to another and the LMPC doesn't escape the rules of physics.

In addition, the Linear MPC gain a pseudo-sinusoidal behavior when we arrive at the value $C_d = 0.2$. We can notice that the dynamics before this are the same but with an induced latency.

Another observation that could be made is that the overshoot isn't as much as it was during the tests conducted on the decentralized PI Controller. In fact, that value decreases as much as we decrease the value of the discharge coefficient. This is one of the main factors that differentiate MPC control than classical control strategies as it's based on the Receding Horizon Principle and not only on the errors between our output and the reference.

The non minimum phase setting is highly affected to variations in the viscosity than the minimum one. Even though that the results are looking somehow awful due to the increasing magnitude of the oscillations in our output yet it is definitely better than the results obtained via the decentralized PI controller in 2.5.3.1.

3.5.1.2 Robustness with respect to the interconnections

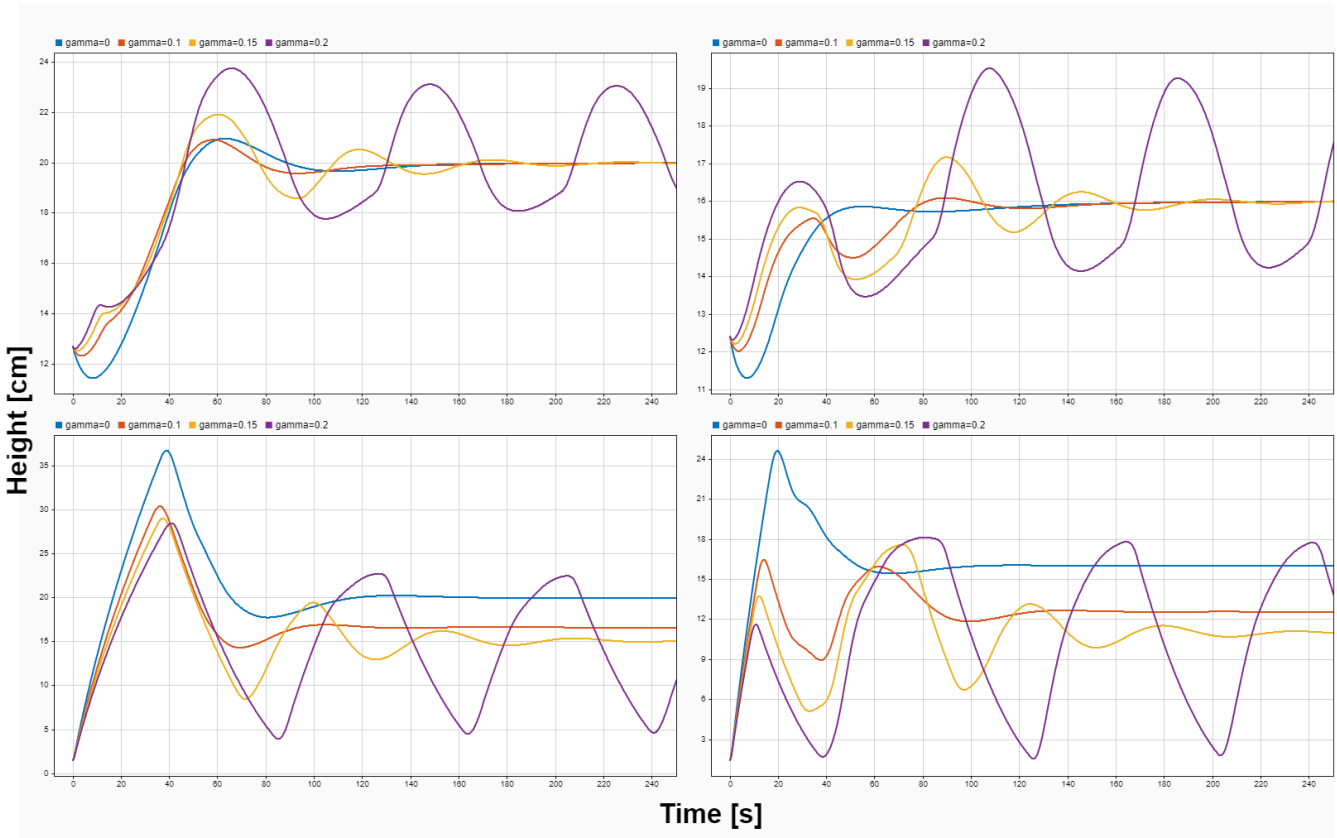


Figure 3.23: Evolution of the System's states under different values of γ for the non-minimum phase setting, Upper left h1, Upper right h2, Lower left h3, Lower right h4

In this part, we test the robustness of our controlled system by designing a controller without taking the interconnections into consideration and then introducing them as a noise signal with respect to our system.

As shown in the upper figure 3.5.2.2, the interconnections induces some oscillations in our system. These oscillations are somehow random and more chaotic let's say than the ones generated by the

benchmark controller. In addition, we can see that the final value for the two upper tanks (h_3 and h_4) is reduced each time we increase the ratio for the valve. This is due to the fact that the 02 lower tanks and our control objectives are being divided between what's been pulled down by the gravitation and what's been brought by the lower nozzle.

The major observation that we can see is that the system is robust for a value of $\gamma = 0.15$ which wasn't the case for our benchmark. The control objectives are reached for this value and the system, and so we can consider this another advantage of the LMPC Control law.

3.5.2 NMPC

3.5.2.1 Robustness with respect to High viscosity fluids

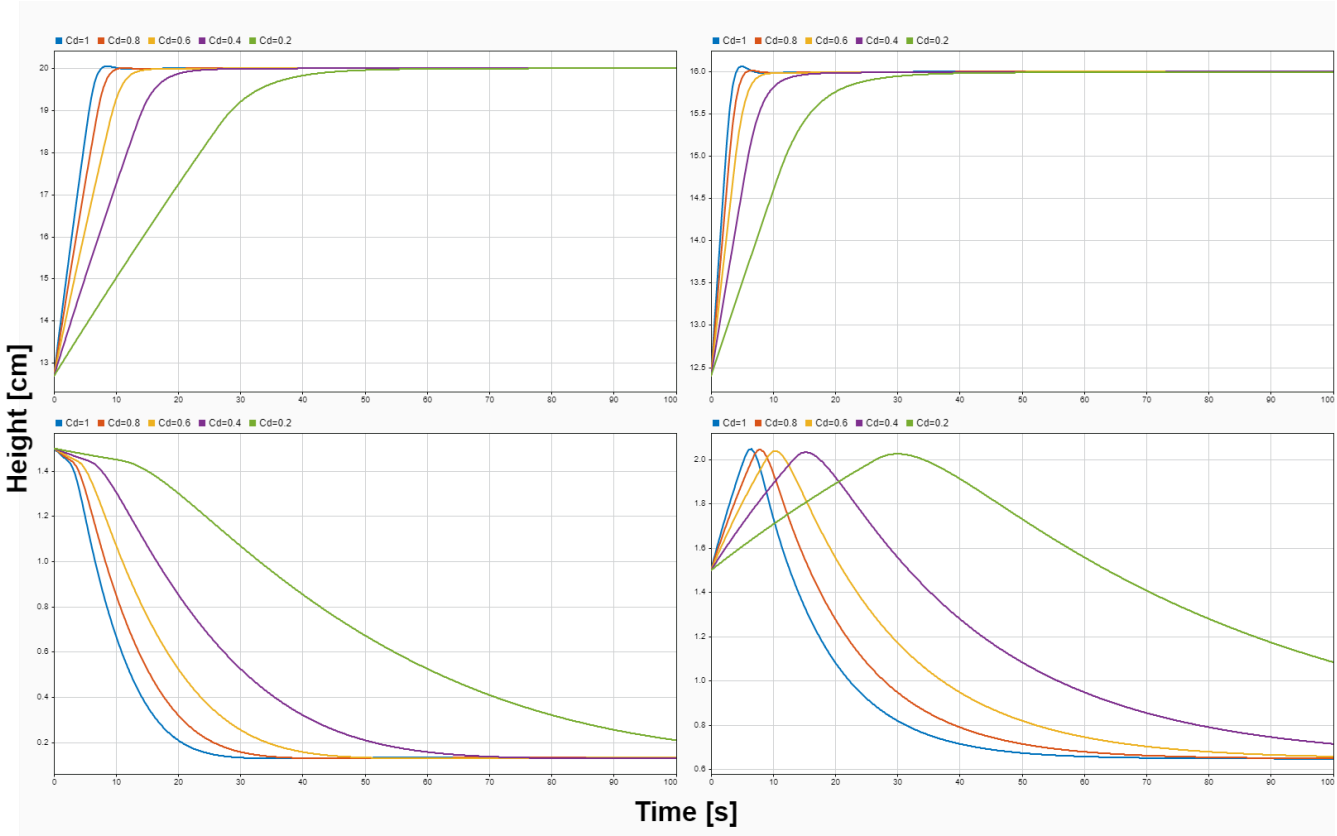


Figure 3.24: Evolution of the System's states under different values of C_d for the minimum phase setting, Upper left h1, Upper right h2, Lower left h3, Lower right h4

In the same way as we did before, we conduct the same high viscosity robustness test on the controlled system using the Nonlinear Model Predictive Controller. The results shows that the system is induced only with latency and no other behavioral deviation from it's original form. The overshoot didn't increase and stayed the same for h_3 and h_4 and it decreased barely for h_1 and h_2. And so to simplify it, the representation stayed practically the same but with an extension on the time axis for both the minimum and the non-minimum settings. We can clearly say that this is without a doubt the most robust controller with respect to high viscosity fluids.

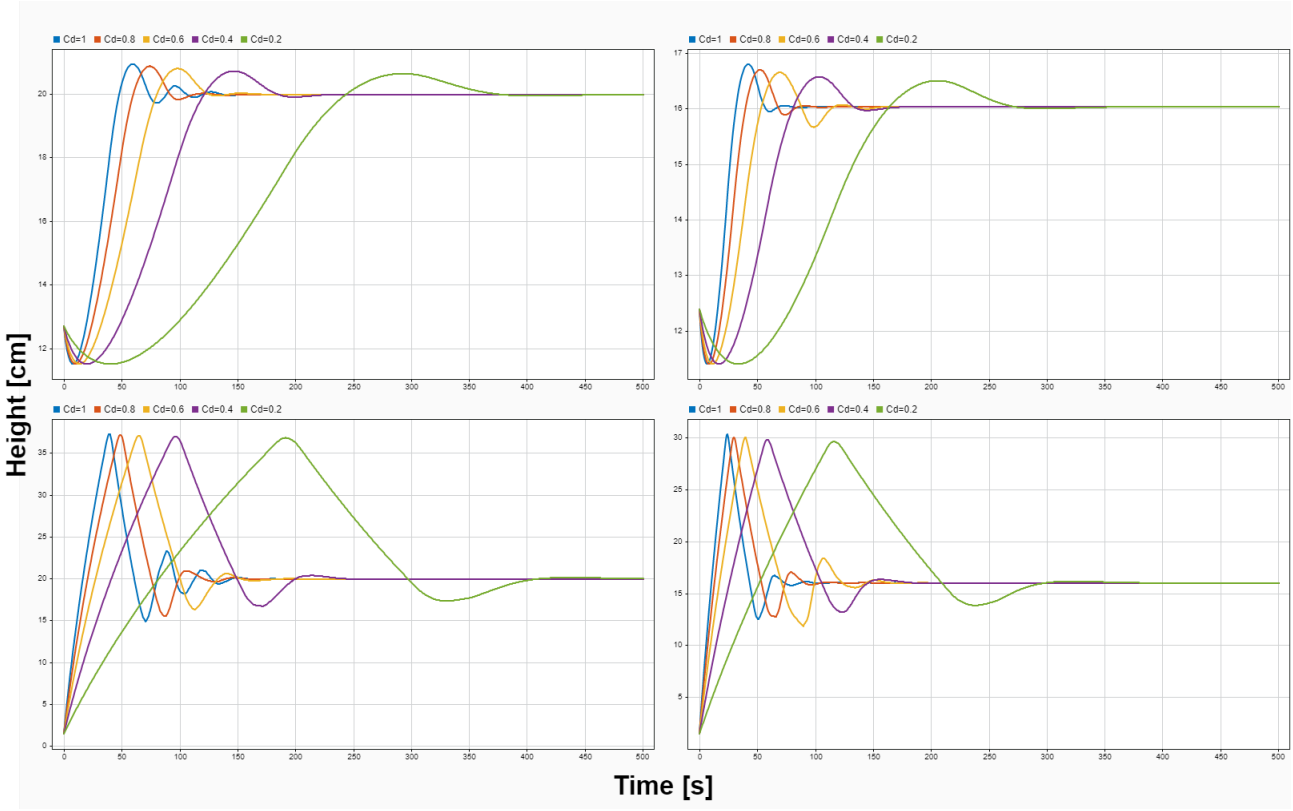


Figure 3.25: Evolution of the System's states under different values of C_d for the non-minimum phase setting, Upper left h1, Upper right h2, Lower left h3, Lower right h4

3.5.2.2 Robustness with respect to the interconnections

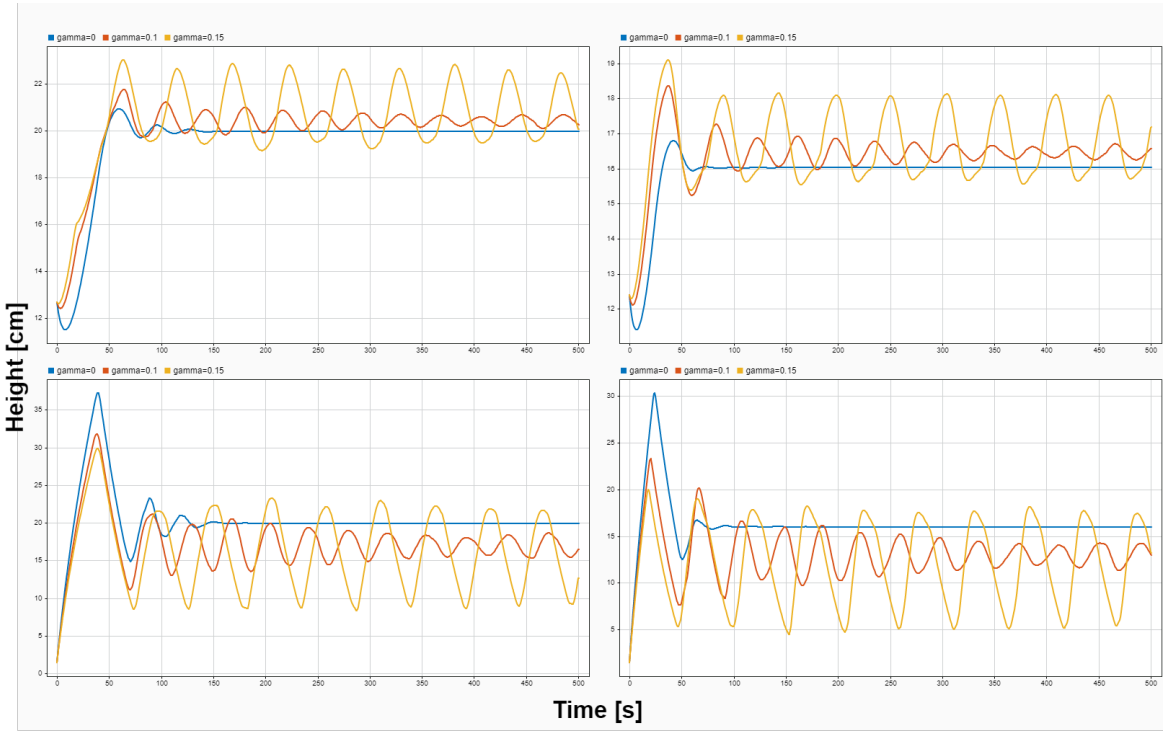


Figure 3.26: Evolution of the System's states under different values of γ for the non-minimum phase setting, Upper left h1, Upper right h2, Lower left h3, Lower right h4

On the other hand, after conducting the simulations on the system designed using the NMPC (figures down below) in the non-minimum phase setting and seeing the results of introducing the interconnec-

tions on that controller, we can see that the controlled plant can barely maintain the control objective for a value of $\gamma = 0.1$. The designed plant using the decentralized controller as well as the LMPC both succeeded in making the control objective and so the Nonlinear MPC falls short in this test compared to its peers.

3.6 Recurrent Neural Network based Predictive Control

As we could see before, we designed both linear and Nonlinear Model Predictive Controllers under different libraries available in the academia. We've seen one of the major problems concerning these optimization techniques where we had to increase sampling time of the NMPC designed with the Matlab Toolbox to 0.5[seconds] in 3.4.4.5. Approximately, 70% of this duration is spent on the computation of the solution of this optimization problem and will lead us eventually to reduce the number of tasks that could be implemented simultaneously on this system or completely change the controlling device that's embedded on. So the question now, how can we avoid this problem and reduce the effects of having such computational complexity on our controller?

In this section, we will propose a controller that uses a Recurrent Neural Network to predict the control input based on a training dataset that was generated given the designed NMPC from 3.4.4.5. This reduces tremendously the computational cost by not requiring to solve the online optimization problem at each sampling time [11].

3.6.1 Basic Idea

The main idea behind this method is to first design a Model Predictive Controller that satisfies the performance requirements for our system. Then we will apply a function approximation of that block using RNNs.

Function approximation techniques have been treated before for this problem, some of them used Neural Networks to approximate the dynamics of the system and thus a model which the MPC will be based on [11]. This method however requires an estimation of the hidden states at each sampling time which goes against our main computational objective. Other works also used vanilla Feed forward (FF) Neural Networks to approximate the MPC control law [11]. The latter was a mapping from the system's states to the optimal control inputs. The optimal law being dependant not only on the current state but also on the objective, the need to figure out another variable becomes crucial. Rather than this, Machine learning techniques have also been applied to find either the best control input but also the best prediction model that have the best closed loop performance [11].

The best solution was presented by [11] where, as we mentioned before, takes the best designed MPC that satisfies the process specifications and performances. Then data pairs are generated through simulation means and are made up of **control actions** and the **error between the reference and the output**. The temporal relationships between these variables are captured by the RNN and ensures that the new control value depends on its past values as well as the past errors between our output and the desired reference. This way the data contains how does the MPC solves the constrained optimization problem to generate the control policy.

Once the network is trained, there will be no need to perform an optimization step to return the optimal control law. Thus, the MPC is completely replaced by the RNN, the latter being employed for real-time control.

Other advantages of this method is that it doesn't need a state estimation algorithm as it is independent from the system's states. The next figure illustrates the training architecture of the new controller, known as the RNN-MPC Controller. We will discuss some fundamentals about Neural Networks in the next subsection.

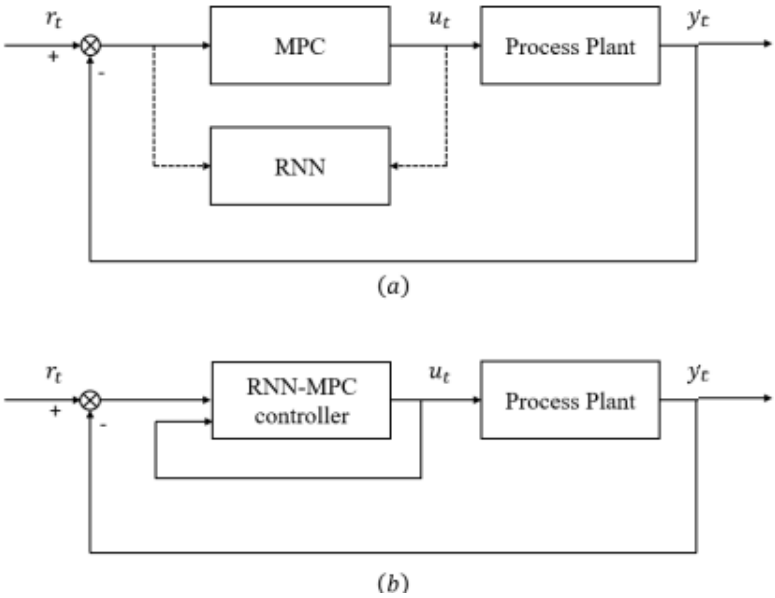


Figure 3.27: Training architecture for the RNN-MPC controller based on NMPC. (b) RNN-MPC controller implementation [11]

3.6.2 Artificial Neural Network

ANNs are computational models inspired by the behavior of the brain and its method of learning from mistakes. They consist of input, hidden, and output layers. The reasoning behind using ANNs to model complex dynamic processes lies in their ability to estimate unknown dynamic system outputs $y_t \in Y$ by forming a linear combination of neurons responses derived from inputs $x_t \in X$ and applying a nonlinear transformation [11].

ANNs are usually used either for classification problems where we want to estimate the probability that a certain set of inputs could belong to a class A a class B or either for making predictions and particularly regression problems. Our approach can be framed as a regression problem, which seeks to identify a function that most accurately characterizes the process dynamics through the mapping $f : X \to Y$, where $X \subseteq \mathbb{R}^{n_x}$ and $Y \subseteq \mathbb{R}^{n_y}$ represent the input and output spaces, respectively [11].

In an ANN, each neuron performs the following operation:

$$y_t = f_y(W_{xy}x_t + b) \tag{3.18}$$

where y_t is the neuron output, x_t is the input, $f(\cdot)$ is a nonlinear activation function, $W_{xy} \in \mathbb{R}^{n_y \times n_x}$ is the weight matrix from x_t to y_t, and b is the bias term. The ANN output, y, is easily computed by applying 3.18 to each neuron. Common activation functions include the sigmoid function, the hyperbolic tangent, and the rectified linear unit (ReLU). In this work, the hyperbolic tangent is used:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.19}$$

The learning process involves updating the network's weight matrices to minimize a chosen loss function. Typically for regression problems, the loss function is the Mean Squared Error (MSE) between the desired output data and the ANN's predictions, which is minimized using a gradient descent algorithm. The iterative application of the chain rule to compute the gradient leads to the back-propagation method.

3.6.3 Recurrent Neural Networks

A Recurrent Neural Network (RNN) introduces time dependence to conventional Feed-Forward (FF) Neural Networks. It can capture the system's dynamics using a hidden state $h_t \in H \subseteq \mathbb{R}^{n_h}$, generated from network inputs $x_t \in X$ up to time $t \in \mathbb{N}$ and past hidden states $h_{t-1} \in H$, which are updated recursively according to:

$$h_t = f_h(W_{xh}x_t + W_{hh}h_{t-1} + b) \tag{3.20}$$

where $W_{xh} \in \mathbb{R}^{n_h \times n_x}$ and $W_{hh} \in \mathbb{R}^{n_h \times n_h}$ are the weight matrices from the input layer to the hidden layer and from the hidden layer to itself, respectively. Typically, $h_0 = 0$.

Similarly, the output y_t is generated as follows:

$$y_t = f_y(W_{hy}h_t) \tag{3.21}$$

where $W_{hy} \in \mathbb{R}^{n_y \times n_h}$ is the weight matrix from the hidden layer to the output layer. The function f_y is typically an identity function mapping the hidden states to the RNN's output variable.

Training the RNN involves updating the parameters, which are primarily the weight matrices W_{xh}, W_{hh}, and W_{hy}. This update process is conducted using the back-propagation through time (BPTT) method.

3.6.3.1 RNN Architecture

The following illustration in 3.28 shows a basic architecture of a vanilla RNN. The main components of the scheme could be depicted as follows [18]:

- **Input Nodes (Blue)**: Denoted as $x^{(t)}$, these nodes receive input vectors at each timestep t.
- **Hidden Nodes (Green)**: Denoted as $h^{(t)}$, these nodes compute the recurrent relationships in the network, integrating current input with historical data.
- **Output Nodes (Purple)**: Denoted as $o^{(t)}$, these nodes produce the output based on the hidden state at each timestep.
- **Weight Matrices**:
  - U: Weight matrix connecting input nodes to hidden nodes.
  - W: Recurrent weight matrix connecting hidden nodes from the previous timestep to the current timestep.
  - V: Weight matrix connecting hidden nodes to output nodes.

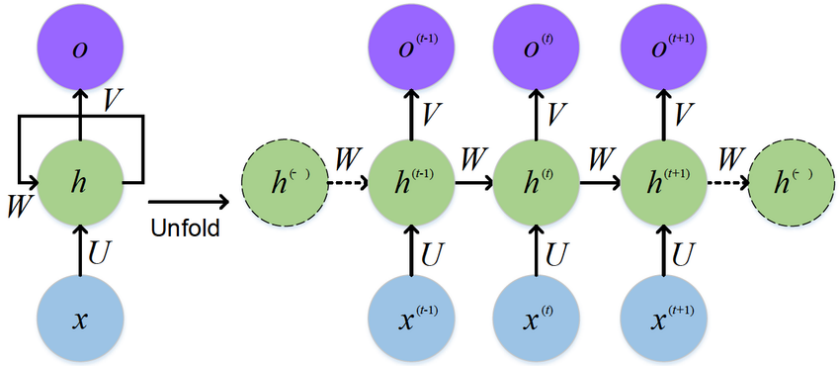


Figure 3.28: Folded and Unfolded RNN for a better visualization [18].

3.6.3.2 Backpropagation Through Time

BPTT is a method used to compute the gradient of the loss function of a recurrent neural network. It is essential for training RNNs to perform tasks involving sequential data [18].

The process of BPTT involves several key steps [18]:

1. Unfolding the RNN over time to create a deep network where each "layer" corresponds to a time step.
2. Performing a forward pass through this network to compute the outputs at each time step.
3. Calculating the loss at the final output or at each time step, depending on the application.
4. The gradients of the loss function L with respect to the network weights are computed by backpropagating errors from the output back to the inputs.
5. After calculating the gradients for each time step, they are accumulated across the sequence, and the network weights are updated accordingly. This approach ensures that learning captures both the immediate and more distant dependencies within the sequence.

3.6.4 Data Retrieving and Model Design

For the data generation, we will be using the designed NMPC using the Matlab Toolbox. The latter even though has achieved a some good performances, we still want to reduce its computational complexity.

The controlled plant (plant + controller) will be tracking first a pseudo random binary signal (PRBS). The MPC response is then saved and is determined to be very satisfactory for our test. This ensures a sufficient process excitation at all the important frequencies [11]. The stored data sequences are saved in a dataset that's divided into 80% for training the network and the remaining 20% will be used for the validation.

Of course, the data is being pre-processed first where a Min-Max normalization has been done first. This ensures the apparition of outliers as well as having a good gradient flow during the training phase. The input configuration for our Network will take the last five samples of the control variables as well as the five past errors between the desired and the current output. The formula is thus given as follows [11]:

$$u(k+1) = f(u(k), \ldots, u(k-5), e(k), \ldots, e(k-5)) \tag{3.22}$$

The hidden layer consist of 10 neurons, and the learning rate is set to 0.001. The Mean Square Error (MSE) was utilized as the loss function, and Stochastic Gradient Descent (SGD) served as the optimizer. The training process involves around 500 iterations for our system [11].

In addition to these characteristics, we added a Dropout layer with a 0.5 deactivation probability as well as an L_2 regularization term with a 0.001 weight decay to avoid having an over-fitting during the training phase

To assess the performance of the trained RNN, three distinct reference signal tests were employed: a Pseudo-Random Binary Sequence (PRBS), a sinusoidal signal, and a step signal.

In [11] they proved that the best network configuration is the one that uses classical RNN and not a similar configuration that happens to have the same hyper-parameters with an LSTM (Long Short Term Memory) neuron or a classical Feed Forward Neuron; Thus we will be training only the one that's proven to have better results. We will also have the Root Mean Squared Error as well as the

FIT criterion as metrics to see the model's performance.

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}} \tag{3.23}$$

$$\text{FIT} = 1 - \frac{\sum_{i=1}^{n}(\hat{y}i - y_i)^2}{\sum i = 1^n(|\hat{y}_i - \mu| + |y_i - \mu|)^2} \tag{3.24}$$

3.6.5 Results

Figure 3.29 illustrates the plots of the loss function for both the training and the validation set. The final values were 0.28 for the Test loss and 0.23 for the Train loss. This indicates that the training was well done and the over-fitting has been avoided (comparing the loss values to their original values). Some instability of the Train loss could also be seen after converging to a certain value, this shouldn't make us worry a lot because the validation loss is still stable.

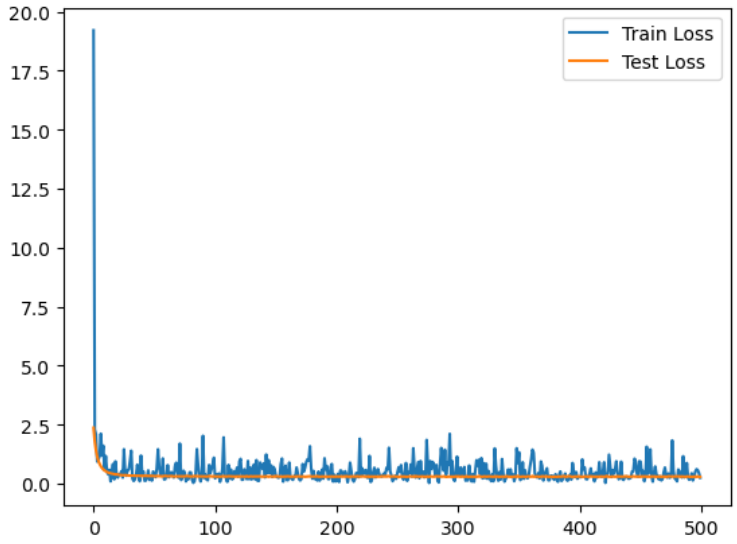


Figure 3.29: Train and Test Loss of our RNN during the training phase

We will now test the prediction of the control signal U on some validation data that we have. This could be seen in figure 3.30 where we can see the actual control signal vs the predicted one by our Recurrent Neural Network. As we can see the prediction is very reliable and we will confirm it even more in the next test.

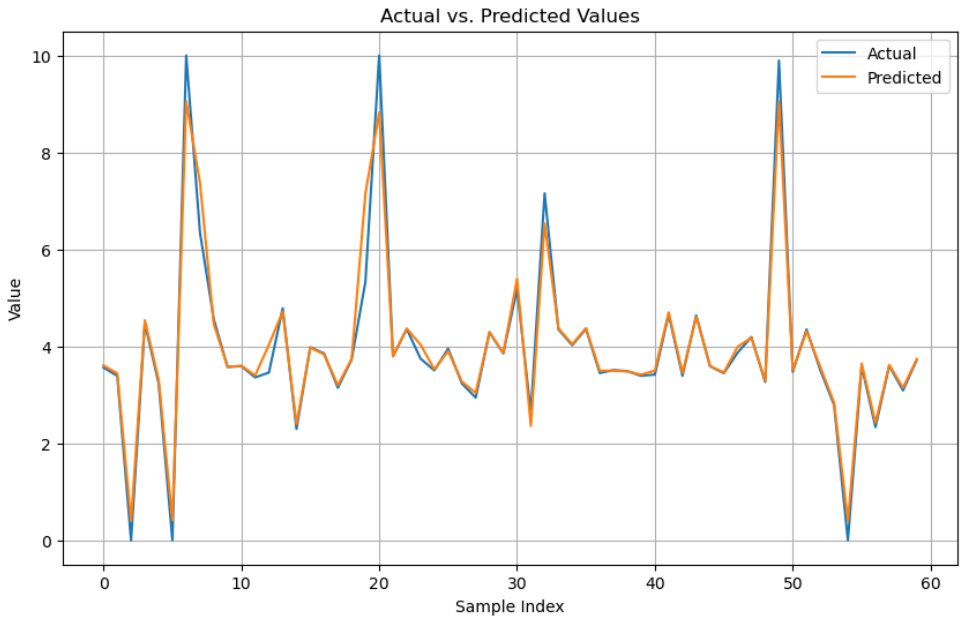
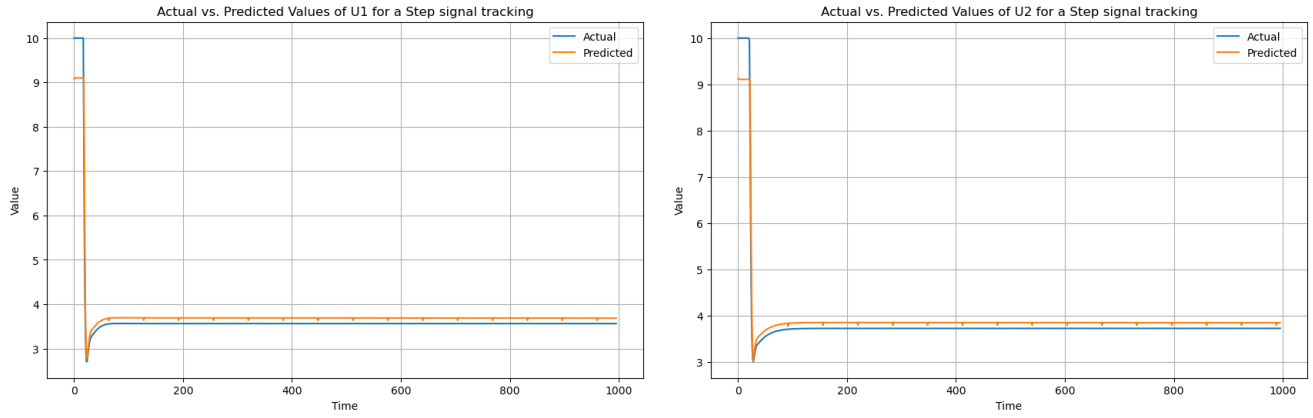


Figure 3.30: Prediction Results on PRBS from the shuffeled test data

The data was also shuffled before introducing it inside our network, making the signal looking like it has been randomized. In addition, we can notice that the signal is not reaching the peaks enough as it should be. Of course, these are some estimation errors made by our model.

Now we will test our prediction model on two test sets that weren't present in the validation set. These two sets are the data pairs collected from a step signal tracking as well as a sin signal tracking by the considered MPC for this application. The results are displayed in the figures down below.



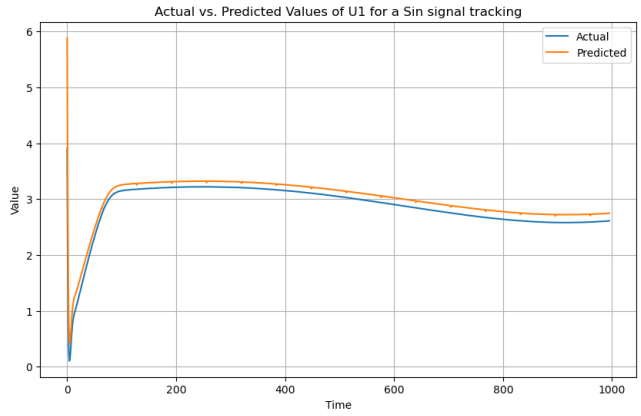
(a) Prediction Results of u_1 for a step signal tracking

(b) Prediction Results of u_2 for a step signal tracking

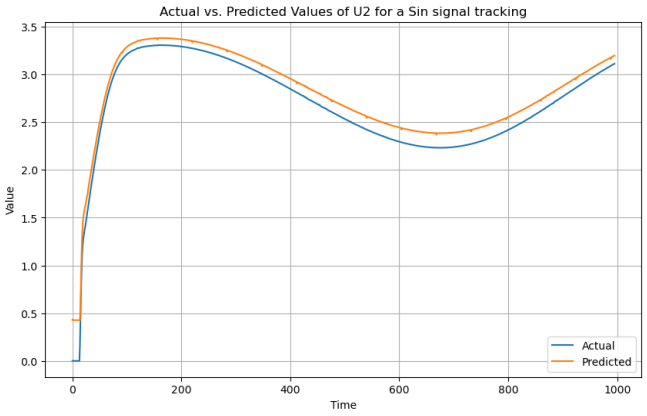
Figure 3.31: Results for a Step signal Tracking of value 10

As we can see from both figures, the prediction made is well rounded in comparison to the behavior observed in the predicted signal. It match well the real control signal with an error around 0.2 for the Sin signal tracking and 0.11 for the step signal tracking. Some peaks in the error could happen in some peak values like the beginning of the signal in 3.31. This however is solved automatically considering that we will begin the control with at least five time values that are initialized by the NMPC controller. Then we will switch to the Recurrent Network that provides a solution to the online optimization problem with a small static error as mentioned before.

The following table give some detailed information about the performance of the conducted tests and the values found by computing the metrics that we cited before.



(a) Prediction Results of u_1 for a sin signal tracking



(b) Prediction Results of u_2 for a sin signal tracking

Figure 3.32: Results for a Sin signal tracking

Signal Type	**Control Input**	**RMSE**	**FIT**
Sin Signal Tracking	U_1	0.5468226959821079	0.9697289628191685
	U_2	0.4572867623068304	0.9793170849221486
Step Signal Tracking	U_1	0.18406099622058472	0.9865796457382038
	U_2	0.1897116426874888	0.9882108518078806

Table 3.1: Performance Metrics for Sin and Step Signal Tracking

Seeing these values, we can say that we have achieved a similarity of at least 96.97% between the predicted control signal and it's actual value. Additionally, the computational time consumed on a single feed-forward over our Network is evaluated at around $\frac{2.79}{1992} = 1.4 \times 10^{-3}$[sec] Which is a drastic change to which it was estimated before in 3.4.4. We could say that we obtained our objective by making this application.

3.7 Conclusion

In this chapter, we dived thoroughly in Predictive Control techniques. We started it by giving a general historical perspective about this domain. It was then followed by the introduction of Model Predictive Control and its two variants, the Linear and Nonlinear MPCs and the practical implementation of these two controllers was rigorously assessed through simulations using standardized tests that we defined before during our design of our benchmark Decentralized PI Controller.

Then, we established some general observations about MPC controllers. A tendency to output the maximum control values when a large error between our output and the desired reference was a notable observation. Even though it reduces the errors pretty quickly, it still poses some risks, particularly for real nonlinear systems where stability and performance are being compromised. For this matter, reducing the values of the Weight matrices of the outputs could certainly help in dealing with that yet a static error will eventually appear in the mentioned output.

Additionally, we conducted the design of the LMPC and NMPC using different libraries. An assessment of these controllers and the feasibility of real-time implementation was also conducted. The controllers where then faced with some robustness tests that showed that the NMPC was the best controller till now to deal with high viscosity fluids. On the other hand, it gave terrible results in the robustness with respect to interconnections.

To address the computational challenges associated with the NMPC design, we proposed an approach based on RNNs to predict the control inputs without performing the online optimization at each sampling time. The objective was well attained as we reduced the time factor by hundreds of times.

In conclusion, this chapter has provided a comprehensive evaluation of MPC strategies, highlighting their strengths, limitations, and potential for real-time implementation. The integration of RNN based controller offers a promising method to overcome computational constraints and having way more efficient and robust predictive controllers.

Chapter 4

Fuzzy Control

4.1 Introduction

The word "fuzzy" is perhaps no longer fuzzy to many engineers today. Introduced in the earlier 1970s, fuzzy systems and fuzzy control theories as an emerging technology targeting industrial applications have added a promising new dimension to the existing domain of conventional control systems engineering. It is now a common belief that when a complex physical system does not provide a set of differential or difference equations as a precise or reasonably accurate mathematical model, particularly when the system description requires certain human experience in linguistic terms, fuzzy systems and fuzzy control theories have some salient features and distinguishing merits over many other approaches.

Fuzzy control methods and algorithms, including many specialized software and hardware available on the market today, may be classified as one type of intelligent control. This is because fuzzy systems modeling, analysis, and control incorporate a certain amount of human knowledge into its components (fuzzy sets, fuzzy logic, and fuzzy rule base). Using human expertise in system modeling and controller design is not only advantageous but often necessary. Classical controller design has already incorporated human skills and knowledge: for instance, what type of controller to use and how to determine the controller structure and parameters largely depend on the decision and preference of the designer, especially when multiple choices are possible. The relatively new fuzzy control technology provides one more choice for this consideration; it has the intention to be an alternative, rather than a simple replacement, of the existing control techniques such as classical control and other intelligent control methods (e.g., neural networks, expert systems, etc.). Together, they supply systems and control engineers with a more complete toolbox to deal with the complex, dynamic, and uncertain real world. Fuzzy control technology is one of the many tools in this toolbox that is developed not only for elegant mathematical theories but, more importantly, for many practical problems with various technical challenges.[13]

Almost all of the physical dynamical systems in real life cannot be represented by linear differential equations and have a nonlinear nature. At the same time, linear control methods rely on the key assumption of small range of operation for the linear model, acquired from linearizing the nonlinear system, to be valid. When the required operation range is large, a linear controller is prone to be unstable, because the nonlinearities in the plant cannot be properly dealt with. Another assumption of linear control is that the system model is indeed linearizable and the linear model is accurate enough for building up the controller. However, the highly nonlinear and discontinuous nature of many, for instance, mechanical and electrical systems does not allow linear approximation. It is also necessary, in the design process of controllers, that the system model is well achievable through a mathematical model and the parameters of the system model are reasonably well-known.

Nevertheless, for many nonlinear plants i.e. chemical processes, building a mathematical model is very difficult and only the input-output data yielded from running the process is accessible for an estimation. Many control problems involve uncertainties in the model parameters. A controller based on inaccurate or obsolete values of the model parameters may show significant performance degradation or even instability. There are some complicated approaches like auto-regressive model based on the input-output data to compensate model uncertainties, which usually use to design a process control. However due to the high nonlinearity of the process, the order of the model often becomes very high so that past effects are taken into account, even if that is physically unrealistic.

One way to cope with such difficulty is to develop a nonlinear model composing of a number of sub-models which are simple, understandable, and responsible for respective sub-domains. The idea of multi-model approach is not new, but the idea of fuzzy modeling using the concept of the fuzzy sets theory offers a new technique to build multi-models of the process based on the input-output data or the original mathematical model of the system. Facing complex and nonlinear systems, we have to recognize that modeling is an art and it is important to realize system modeling is generally an act to understand things directly rather than by computer. At most a linear combination like a fuzzy model is clearly understandable.[44]

Fuzzy logic provides a simple way to arrive at a definite conclusion based upon vague, imprecise, noisy, or missing input information. Hence, unlike classical logic, that requires a deep understanding of a system, exact equations and precise numeric values, fuzzy logic incorporates an alternative way of thinking, which allows modeling complex systems using a higher level of abstraction originating from our knowledge and experience. The principal advantage of fuzzy logic systems is their aptitude to approximate any nonlinear function

In this chapter, a recall is given on modeling by two particular structures of fuzzy models that are Mamdani and T-S fuzzy models. however, the construction procedure of a T-S fuzzy model is detailed and will be applied for the coupled tanks system.

We will then use the so approved fuzzy model to design controllers based on the Parallel Distributed Compensation (PDC) control law for stabilizing the system first. Then, a reference tracking when adding an integral structure to the latter. The controllers will be divided into two: A complete model that will be based on 16 fuzzy rules and a simplified model that will be based on 04 fuzzy rules only. We will then validate them using the robustness tests defined earlier in state of the art chapter. A thorough comparison will also be made with the benchmark decentralized PI controller throughout the whole observations written during this chapter.

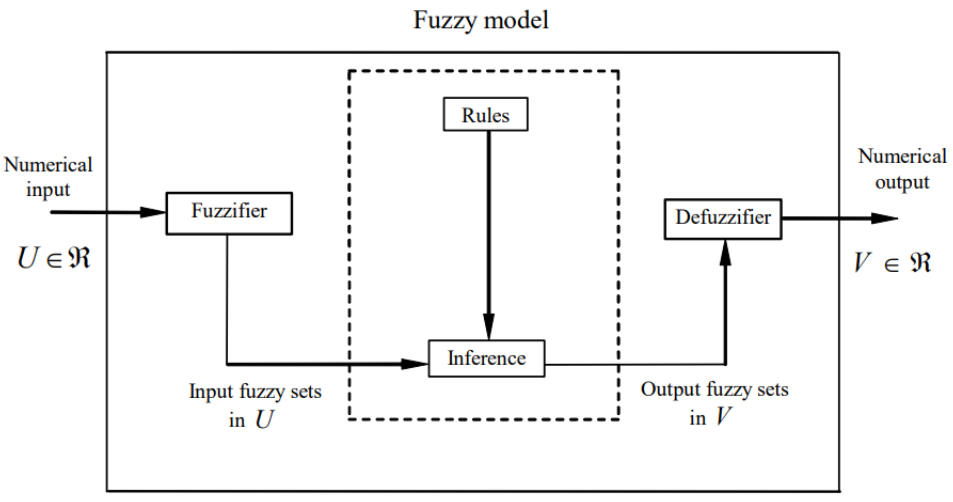


Figure 4.1: Fuzzy Model Structure [44]

4.2 Takagi-Sugeno Fuzzy Models

A fuzzy controller is a model that uses fuzzy rules, which are linguistic if-then statements involving fuzzy sets, fuzzy logic, and fuzzy inference. Fuzzy rules play a key role in representing expert control/modeling knowledge and experience and in linking the input variables of fuzzy controllers/models to output variable (or variables). Two major types of fuzzy rules exist, namely, Mamdani fuzzy rules and Takagi-Sugeno (TS, for short) fuzzy rules. [44]

Mamdani fuzzy systems

Let's first start with the familiar Mamdani fuzzy systems. A simple but representative Mamdani fuzzy rule describing the movement of a car is:

IF *Speed* is *High* AND *Acceleration* is *Small* THEN *Braking* is (should be) *Modest*

where Speed and Acceleration are input variables and Braking is an output variable. "High," "Small," and "Modest" are fuzzy sets, and the first two are called input fuzzy sets while the last one is named the output fuzzy set.

The variables as well as linguistic terms, such as "High", can be represented by mathematical symbols. Thus, a Mamdani fuzzy rule for a fuzzy controller involving three input variables and two output variables can be described as follows:

IF x_1 is M_1 AND x_2 is M_2 AND x_3 is M_3 THEN u_1 is $M_4; u_2$ is M_5;

where ,

$x_1, x_2, \text{and} x_3$ are input variables (e.g., error, its first derivative and its second derivative), such that

u_1, u_2 are output variables (e.g., valve openness)

y^i is the output of the i-th fuzzy IF-THEN rule.

In theory, these variables can be either continuous or discrete; practically speaking, however, they should be discrete because virtually all fuzzy controllers and models are implemented using digital computers. M_1, M_2, M_3, M_4, and M_5 are fuzzy sets, and AND / OR are fuzzy logic "&" and "+" operators respectively. "IF x_1 is M_1 AND x_2 is M_2 AND x_3 is M_3" is called the rule antecedent, whereas the remaining part is named the rule consequent.

Takagi and Sugeno fuzzy systems

Takagi and Sugeno (Takagi & Sugeno, 1985) came up with the alternative rule format in order to make automated tuning possible and to reduce the number of fuzzy rules. A T-S fuzzy model is described by fuzzy IF-THEN rules defined by the following:

Rule i : IF $z_1(t)$ is M_{i1} and $\dots z_p(t)$ is M_{ip} THEN $y^i = a_0^i + a_1^i z_1(t) + a_2^i z_2(t) + \cdots + a_p^i z_p(t)$

Where,

$M_{i1}, \dots M_{ip}$ are fuzzy sets,

$a_0^i, \dots a_p^i$ are the coefficients of the i-th linear consequent,and

y^i is the output of the i-th fuzzy IF-THEN rule.

T-S model represents a dynamical system whose IF-THEN rules represent local linear input-output relations of the nonlinear dynamical system. The main feature of a T-S fuzzy model is to express the local dynamics of each fuzzy rule by a linear sub-model, and then the overall fuzzy system is obtained by fuzzy "blending" of the linear sub-models;

4.2.1 Principle of the T-S fuzzy multimodel approach

The T-S approach is founded on decomposing the dynamic behavior of the nonlinear system into a set of r operating regions, with each region characterized by a linear sub-model The Figure down below illustrates this principle in a two-dimensional case, where the set of operating points of the coordinate system $x(t) = (x_1(t), x_2(t))$ has been decomposed into four operating domains denoted as $D1$, $D2$, $D3$, and $D4$. The overall operating domain is then defined by the union of these local domains, $D = D1 \cup D2 \cup D3 \cup D4$. On each of these local domains, or sub-domains, a local model can be constructed;

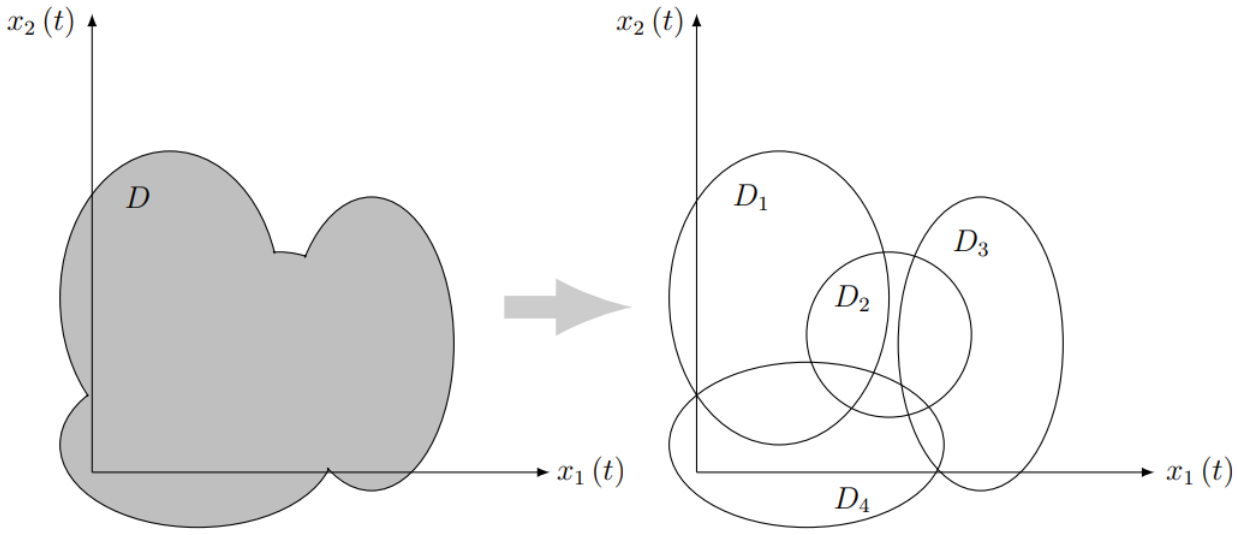


Figure 4.2: T-S fuzzy multimodel approach[52]

4.3 Construction of Takagi Sugeno Fuzzy Model

T-S fuzzy system is used to describe the nonlinear plant by fuzzy IF-THEN rules that represent locally linear input-output relations of a system[31].

The Takagi–Sugeno (TS) fuzzy model (Takagi and Sugeno, 1985), on the other hand, uses crisp functions in the consequents. Hence, it can be seen as a combination of linguistic and mathematical regression modeling in the sense that the antecedents describe fuzzy regions in the input space in which consequent functions are valid[8].

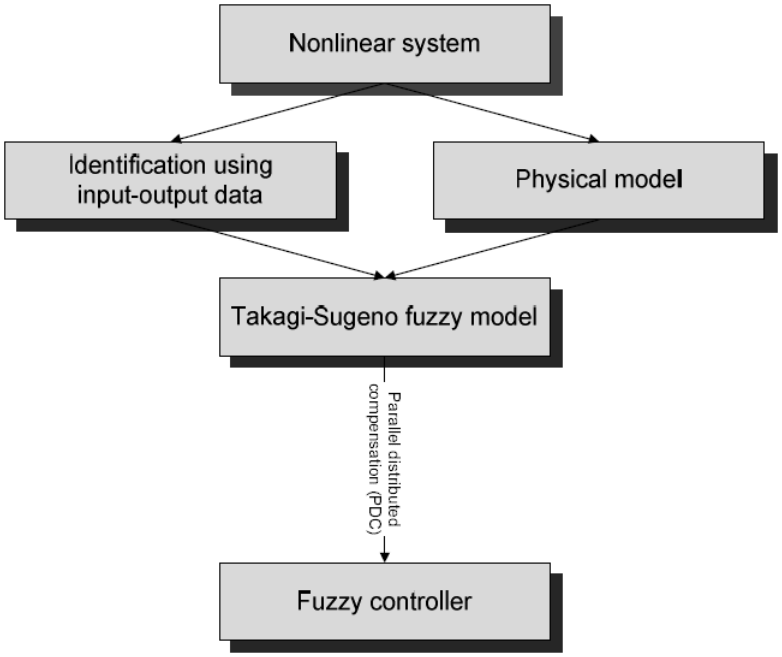


Figure 4.3: Model-based fuzzy control design[44]

To design a T-S fuzzy controller, we need a T-S fuzzy model for a nonlinear system. Therefore, the construction of a fuzzy model represent an important and basic procedure in this approach. In general there are two approaches for constructing fuzzy models:

1. Identification (fuzzy modeling) using input-output data
2. Derivation from given nonlinear system equations.

The identification approach is suitable for modelling of plants that are difficult to represent using analytical models, whereas when the nonlinear dynamical models are available, the second approach could be seen as more appropriated.

In both cases, we obtain a T-S fuzzy model whose i-th rule form is:

Rule i : IF $z_1(t)$ is M_{i1} and $\dots z_p(t)$ is M_{ip} THEN:

$$\begin{cases} \dot{x}(t) = A_i x(t) + B_i u(t) \\ y(t) = C_i x(t) \end{cases} \tag{4.1}$$

The final outputs of the fuzzy model are inferred as follows:

$$\begin{cases} \dot{x}(t) = \sum_{i=1}^{r} h_i(z(t))(A_i x(t) + B_i u(t)) \\ y(t) = \sum_{i=1}^{r} h_i(z(t)) C_i x(t) \end{cases} \tag{4.2}$$

where $\mathbf{Z}(t) = [z_1(t), z_2(t), \ldots, z_p(t)]$ is the premise variable vector that may be functions of the state variables, measurable external disturbances and/or time. $A_i \in \mathbb{R}^{n\times n}$, $B_i \in \mathbb{R}^{n\times m}$, $C_i \in \mathbb{R}^{q\times n}$, $x(t) \in \mathbb{R}^n$ is the state vector, $u(t) \in \mathbb{R}^m$ is the input vector, $y(t) \in \mathbb{R}^q$ is the output vector; r is the number of IF-THEN rules and M_{ij} is a fuzzy set. $h_i(z(t))$ is the normalized weight for each rule (or the activation function) and $w_i(t)$ is the membership function for rule i, that could be written as:

$$\forall i \in \{1, 2, \ldots, r\}, \quad h_i(z(t)) \geq 0, \quad \sum_{i=1}^{r} h_i(z(t)) = 1 \tag{4.3}$$

$$h_i(z(t)) = \frac{w_i(t)}{\sum_{i=1}^{r} w_i(t)} \tag{4.4}$$

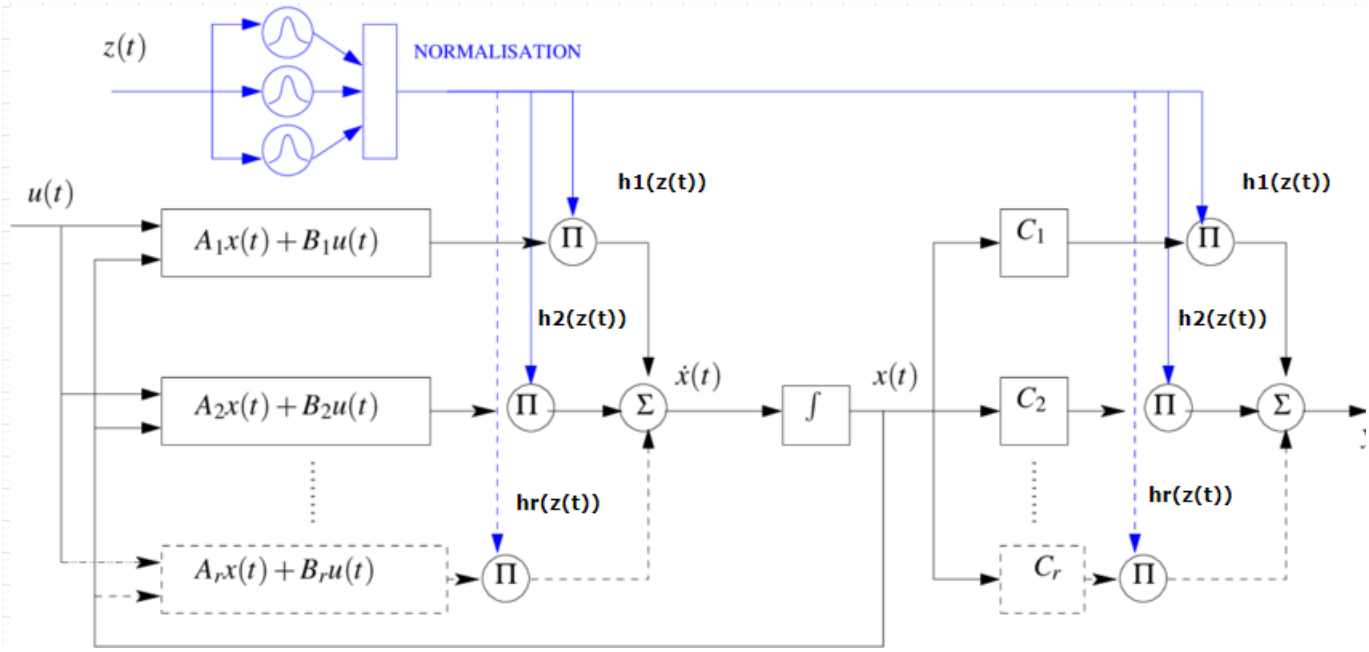


Figure 4.4: Takagi-Sugeno multiple model architecture[46]

4.3.1 Identification approach

From input-output data, we obtain linear sub-models around the different operational points. The local linear sub-models, are fuzzy IF-THEN rules, whose consequent parts are linear models. This identification allows us to find an optimal model after estimating the parameters and validating, the final model. However, a state representation is used in the consequent part in order to extend the state feedback control principle to the nonlinear case.

4.3.2 Nonlinear dynamical model

When nonlinear dynamical models are easy to obtain, the linearization, the principle of sector nonlinearity or local approximation are more appropriated for constructing the fuzzy model [52].

4.3.2.1 Linearization

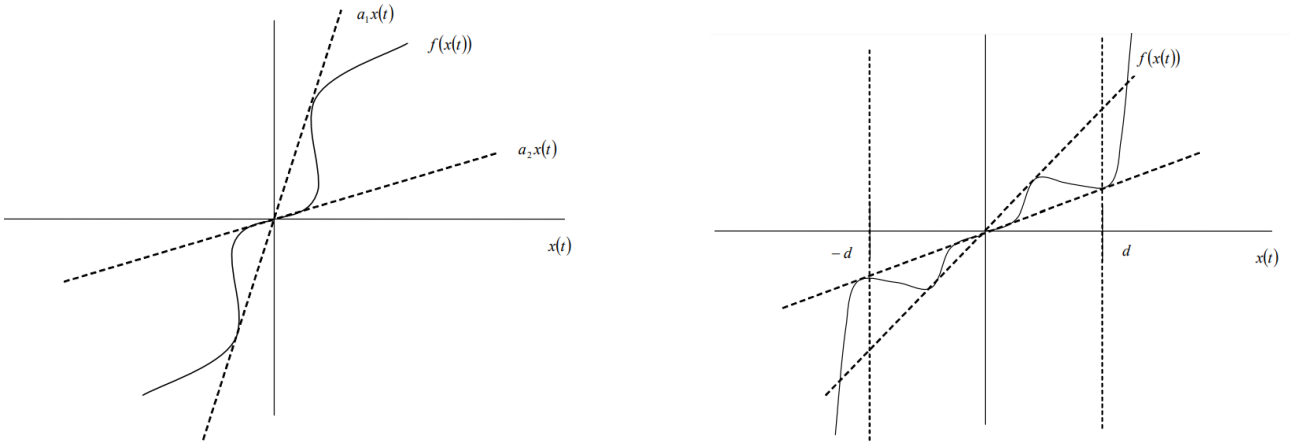
The basic idea is to linearize the nonlinear analytical model of the process at various operating points, which are selected intelligently and carefully to obtain affine linear models connected by activation functions [52].

4.3.2.2 Sector nonlinearity

The first apparition of sector non linearity in fuzzy model construction was back in 1992, it is based on considering a simple nonlinear system . $x(t) = f(x(t))$ where $f(0) = 0$. The objective is to find the global sector such that .

$$x(t) = f(x(t)) \in [-a; a]$$

An exact fuzzy model construction is guaranteed with this method. However, it is sometimes difficult to find global sectors, then local sector nonlinearity is considered. The nonlinear system is represented exactly by the fuzzy model in the "local" region $d < x(t) < d$. But, it is often desirable to simplify the original nonlinear system as much as possible in order to reduce the number of rules. the following two figures illustrates this concept.[52]



(a) Global sector nonlinearity[2]

(b) Local sector nonlinearity[2]

Figure 4.5: Comparison between global and local non-linearities

To obtain the TS-type model, we will follow the following approach : We consider the nonlinear mathematical model, assuming the existence of a compact set of premise variables ($z \in \mathcal{C} \subset \mathbb{R}^z$) in which the nonlinearities are bounded ($z_i \in [\min z_i, \max z_i]$), where $\min z_i$ and $\max z_i$ are the lower and upper bounds of the nonlinear terms for $i \in \{1, \ldots, k\}$.

then we can write :

$$(z_i(t)) = (\max z_i) \times M_i^0(z_i(t)) + (\min z_i) \times M_i^1(z_i(t)) \tag{4.5}$$

where:

$$\begin{cases} M_i^0(z_i(t)) = \dfrac{z_i(t) - (\min z_i)}{(\max z_i) - (\min z_i)} \\ M_i^1(z_i(t)) = \dfrac{(\max z_i) - z_i(t)}{(\max z_i) - (\min z_i)} \end{cases} \tag{4.6}$$

Example

Consider the following nonlinear system:

$$\begin{pmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{pmatrix} = \begin{pmatrix} -x_1(t) + x_1(t)x_2^2(t) \\ -x_2(t) + (3 + x_2(t))x_1^3(t) \end{pmatrix}$$

For simplicity, we assume that $x_1 \in [-1, 1]$ and $x_2 \in [-1, 1]$. Of course, we can assume any range for $x_1(t)$ and $x_2(t)$ to construct a fuzzy model. Equation can be written as

$$\dot{x}(t) = \begin{pmatrix} -1 & x_1(t)x_2^2(t) \\ (3 + x_2(t))x_1^2(t) & -1 \end{pmatrix} x(t);$$

where $x(t) = [x_1(t) \quad x_2(t)]^T$ and $x_1(t)x_2^2(t)$ and $(3 + x_2(t))x_1^2(t)$ are nonlinear terms.

For the nonlinear terms,

$$z_1(t) = x_1(t)x_2^2(t), \quad \text{non linearity } :nl_1$$
$$z_2(t) = (3 + x_2(t))x_1^2(t) \quad \text{non linearity } :nl_2$$

Then, we have :

$$\dot{x}(t) = \begin{pmatrix} -1 & z_1(t) \\ z_2(t) & -1 \end{pmatrix} x(t);$$

Next, we should calculate the minimum and maximum values of $z_1(t)$ and $z_2(t)$ under $x_1(t) \in [-1, 1]$ and $x_2(t) \in [-1, 1]$. They are obtained as follows:

$$\max_{x_1(t), x_2(t)} z_1(t) = 1; \qquad \min_{x_1(t), x_2(t)} z_1(t) = -1;$$
$$\max_{x_1(t), x_2(t)} z_2(t) = 4; \qquad \min_{x_1(t), x_2(t)} z_2(t) = 0.$$

From the maximum and minimum values, $z_1(t)$ and $z_2(t)$ can be represented by

$$z_1(t) = x_1(t)x_2^2(t) = M_1(z_1(t)) \cdot 1 + M_2(z_1(t)) \cdot (-1);$$
$$z_2(t) = (3 + x_2(t))x_1^2(t) = N_1(z_2(t)) \cdot 4 + N_2(z_2(t)) \cdot 0;$$

where

$$M_1(z_1(t)) + M_2(z_1(t)) = 1;$$
$$N_1(z_2(t)) + N_2(z_2(t)) = 1.$$

Therefore the membership functions can be calculated as

$$M_1(z_1(t)) = \frac{z_1(t) + 1}{2}; \quad M_2(z_1(t)) = \frac{1 - z_1(t)}{2};$$
$$N_1(z_2(t)) = \frac{z_2(t)}{4}; \quad N_2(z_2(t)) = \frac{4 - z_2(t)}{4};$$

We name the membership functions "Positive," "Negative," "Big," and "Small," respectively. Then, the nonlinear system is represented by the following fuzzy model.

Model Rule 1: IF $z_1(t)$ is "Positive" and $z_2(t)$ is "Big," THEN $\dot{x}(t) = A_1 x(t)$.

Model Rule 2: IF $z_1(t)$ is "Positive" and $z_2(t)$ is "Small," THEN $\dot{x}(t) = A_2 x(t)$.

Model Rule 3: IF $z_1(t)$ is "Negative" and $z_2(t)$ is "Big," THEN $\dot{x}(t) = A_3 x(t)$.

Model Rule 4: IF $z_1(t)$ is "Negative" and $z_2(t)$ is "Small," THEN $\dot{x}(t) = A_4 x(t)$.

Here,

$$A_1 = \begin{pmatrix} -1 & 1 \\ 4 & -1 \end{pmatrix}; \quad A_2 = \begin{pmatrix} -1 & 1 \\ 0 & -1 \end{pmatrix};$$

$$A_3 = \begin{pmatrix} -1 & -1 \\ 4 & -1 \end{pmatrix}; \quad A_4 = \begin{pmatrix} -1 & -1 \\ 0 & -1 \end{pmatrix};$$

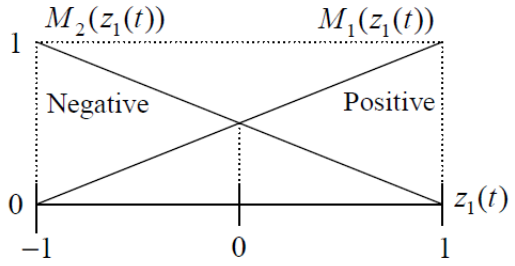


Figure 4.6: Membership functions $M_1(z_1(t))$ and $M_2(z_1(t))$[44]

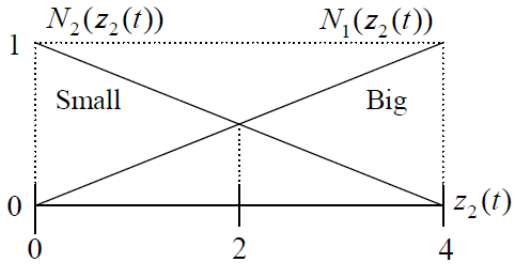


Figure 4.7: Membership functions $N_1(z_2(t))$and $N_2(z_2(t))$[44]

After the defuzzification, we have:

$$\dot{x}(t) = \sum_{i=1}^{4} h_i(z(t)) A_i x(t)$$

where

$$h_1(z(t)) = M_1(z_1(t)) \times N_1(z_2(t))$$
$$h_2(z(t)) = M_1(z_1(t)) \times N_2(z_2(t))$$
$$h_3(z(t)) = M_2(z_1(t)) \times N_1(z_2(t))$$
$$h_4(z(t)) = M_2(z_1(t)) \times N_2(z_2(t))$$

Finally, this model represents the nonlinear system in the region $[-1, 1] \times [-1, 1]$ of the $x_1 - x_2$ space.

4.4 TS fuzzy modeling of Coupled Tanks

The dynamic model of the coupled tanks system is the result of applying mass balances and Bernoulli's law. Defined earlier in the state of the art chapter, it is represented by the following continuous-time differential equations:

$$\frac{dh_1(t)}{dt} = -\frac{a_1}{A_1}\sqrt{2gh_1(t)} + \frac{a_3}{A_1}\sqrt{2gh_3(t)} + \frac{\gamma_1 K_1}{A_1}v_1(t) \tag{4.7}$$

$$\frac{dh_2(t)}{dt} = -\frac{a_2}{A_2}\sqrt{2gh_2(t)} + \frac{a_4}{A_2}\sqrt{2gh_4(t)} + \frac{\gamma_2 K_2}{A_2}v_2(t) \tag{4.8}$$

$$\frac{dh_3(t)}{dt} = -\frac{a_3}{A_3}\sqrt{2gh_3(t)} + \frac{(1-\gamma_2)K_2}{A_3}v_2(t) \tag{4.9}$$

$$\frac{dh_4(t)}{dt} = -\frac{a_4}{A_4}\sqrt{2gh_4(t)} + \frac{(1-\gamma_1)K_1}{A_4}v_1(t) \tag{4.10}$$

4.4.1 fuzzy model with 16 rules

From the nonlinear system of the Tank system, one can see that the non-linearity is related to the states $\sqrt{h_i(t)}$ for $i = 1, 2, 3, 4$[62]

Thus, by considering the following variable change

$$Z_i(t) = \frac{1}{\sqrt{h_i}} \text{ for } i = 1, 2, 3, 4 \text{ and assuming that } Z_{i\min} \leq Z_i(t) \leq Z_{i\max} \tag{4.11}$$

the following approximation can be obtained

$$\begin{cases} Z_1(t) &= S_1(Z_1(t)) \times Z_{1\min} + S_2(Z_1(t)) \times Z_{1\max} \\ Z_2(t) &= R_1(Z_2(t)) \times Z_{2\min} + R_2(Z_2(t)) \times Z_{2\max} \\ Z_3(t) &= M_1(Z_3(t)) \times Z_{3\min} + M_2(Z_3(t)) \times Z_{3\max} \\ Z_4(t) &= N_1(Z_4(t)) \times Z_{4\min} + N_2(Z_4(t)) \times Z_{4\max} \end{cases} \tag{4.12}$$

$$\begin{cases} S_1(Z_1(t)) + S_2(Z_1(t)) = 1 \\ R_1(Z_2(t)) + R_2(Z_2(t)) = 1 \\ M_1(Z_3(t)) + M_2(Z_3(t)) = 1 \\ N_1(Z_4(t)) + N_2(Z_4(t)) = 1 \end{cases} \tag{4.13}$$

assuming that the height of each tank as the system output, one can get

$$\begin{cases} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{cases} \tag{4.14}$$

where:

$$X = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix},\ U = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix},\ C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} B = \begin{bmatrix} \frac{\gamma_1 K_1}{A_1} & 0 \\ 0 & \frac{\gamma_2 K_2}{A_2} \\ 0 & \frac{(1-\gamma_2)K_2}{A_3} \\ \frac{(1-\gamma_1)K_1}{A_4} & 0 \end{bmatrix}$$

$$A = \begin{bmatrix} \frac{-a_1\sqrt{2g}}{A_1\sqrt{h_1}} & 0 & \frac{a_3\sqrt{2g}}{A_1\sqrt{h_3}} & 0 \\ 0 & \frac{-a_2\sqrt{2g}}{A_2\sqrt{h_2}} & 0 & \frac{a_4\sqrt{2g}}{A_2\sqrt{h_4}} \\ 0 & 0 & -\frac{a_3\sqrt{2g}}{A_3\sqrt{h_3}} & 0 \\ 0 & 0 & 0 & -\frac{a_4\sqrt{2g}}{A_4\sqrt{h_4}} \end{bmatrix}$$

the matrix A can be rewritten as follows $A = \sum_{i=1}^{r} h_i(z(t))A_i,$ for $i = 1, 2, 3, \ldots, r$. for r=16
The fuzzy model has the following 16 rules as depicted in the next table.

Rule	$z_1(t)$	$z_2(t)$	$z_3(t)$	$z_4(t)$	$\dot{x}(t)$
1	$S_1(Z_1(t))$	$R_1(Z_2(t))$	$M_1(Z_3(t))$	$N_1(Z_4(t))$	$A_1x(t) + Bu(t)$
2	$S_2(Z_1(t))$	$R_1(Z_2(t))$	$M_1(Z_3(t))$	$N_1(Z_4(t))$	$A_2x(t) + Bu(t)$
3	$S_1(Z_1(t))$	$R_2(Z_2(t))$	$M_1(Z_3(t))$	$N_1(Z_4(t))$	$A_3x(t) + Bu(t)$
4	$S_2(Z_1(t))$	$R_2(Z_2(t))$	$M_1(Z_3(t))$	$N_1(Z_4(t))$	$A_4x(t) + Bu(t)$
5	$S_1(Z_1(t))$	$R_1(Z_2(t))$	$M_2(Z_3(t))$	$N_1(Z_4(t))$	$A_5x(t) + Bu(t)$
6	$S_2(Z_1(t))$	$R_1(Z_2(t))$	$M_2(Z_3(t))$	$N_1(Z_4(t))$	$A_6x(t) + Bu(t)$
7	$S_1(Z_1(t))$	$R_2(Z_2(t))$	$M_2(Z_3(t))$	$N_1(Z_4(t))$	$A_7x(t) + Bu(t)$
8	$S_2(Z_1(t))$	$R_2(Z_2(t))$	$M_2(Z_3(t))$	$N_1(Z_4(t))$	$A_8x(t) + Bu(t)$
9	$S_1(Z_1(t))$	$R_1(Z_2(t))$	$M_1(Z_3(t))$	$N_2(Z_4(t))$	$A_9x(t) + Bu(t)$
10	$S_2(Z_1(t))$	$R_1(Z_2(t))$	$M_1(Z_3(t))$	$N_2(Z_4(t))$	$A_{10}x(t)+Bu(t)$
11	$S_1(Z_1(t))$	$R_2(Z_2(t))$	$M_1(Z_3(t))$	$N_2(Z_4(t))$	$A_{11}x(t)+Bu(t)$
12	$S_2(Z_1(t))$	$R_2(Z_2(t))$	$M_1(Z_3(t))$	$N_2(Z_4(t))$	$A_{12}x(t)+Bu(t)$
13	$S_1(Z_1(t))$	$R_1(Z_2(t))$	$M_2(Z_3(t))$	$N_2(Z_4(t))$	$A_{13}x(t)+Bu(t)$
14	$S_2(Z_1(t))$	$R_1(Z_2(t))$	$M_2(Z_3(t))$	$N_2(Z_4(t))$	$A_{14}x(t)+Bu(t)$
15	$S_1(Z_1(t))$	$R_2(Z_2(t))$	$M_2(Z_3(t))$	$N_2(Z_4(t))$	$A_{15}x(t)+Bu(t)$
16	$S_2(Z_1(t))$	$R_2(Z_2(t))$	$M_2(Z_3(t))$	$N_2(Z_4(t))$	$A_{16}x(t)+Bu(t)$

Table 4.1: Fuzzy Model Rules

The weights are then computed by multiplying the four values from each column for every single rule, such that:

$$w_i(t) = S_i(Z_1(t)) \times R_j(Z_2(t)) \times M_k(Z_3(t)) \times N_l(Z_4(t)) \qquad \text{for :} \quad i, j, k, l = 1, 2 \tag{4.15}$$

The general form of the matrices A_i can be represented as $A^{(i)}$, where i is the index of the model rule.

$$A^{(i)} = \begin{bmatrix} -\frac{a_1\sqrt{2g}Z^{(i)}_{1,\text{range}}}{A_1} & 0 & \frac{a_3\sqrt{2g}Z^{(i)}_{3,\text{range}}}{A_1} & 0 \\ 0 & -\frac{a_2\sqrt{2g}Z^{(i)}_{2,\text{range}}}{A_2} & 0 & \frac{a_4\sqrt{2g}Z^{(i)}_{4,\text{range}}}{A_2} \\ 0 & 0 & -\frac{a_3\sqrt{2g}Z^{(i)}_{3,\text{range}}}{A_3} & 0 \\ 0 & 0 & 0 & -\frac{a_4\sqrt{2g}Z^{(i)}_{4,\text{range}}}{A_4} \end{bmatrix}$$

Where $Z^{(i)}_{m,\text{range}}$ is defined by the following rules:

Model Rule i	Non-zero elements in $A^{(i)}$
1	$Z_{1,\min}, Z_{2,\min}, Z_{3,\min}, Z_{4,\min}$
2	$Z_{1,\max}, Z_{2,\min}, Z_{3,\min}, Z_{4,\min}$
3	$Z_{1,\min}, Z_{2,\min}, Z_{3,\max}, Z_{4,\min}$
4	$Z_{1,\max}, Z_{2,\max}, Z_{3,\min}, Z_{4,\min}$
5	$Z_{1,\min}, Z_{2,\min}, Z_{3,\max}, Z_{4,\min}$
6	$Z_{1,\max}, Z_{2,\min}, Z_{3,\max}, Z_{4,\min}$
7	$Z_{1,\min}, Z_{2,\max}, Z_{3,\max}, Z_{4,\min}$
8	$Z_{1,\max}, Z_{2,\max}, Z_{3,\max}, Z_{4,\min}$
9	$Z_{1,\min}, Z_{2,\min}, Z_{3,\min}, Z_{4,\max}$
10	$Z_{1,\max}, Z_{2,\min}, Z_{3,\min}, Z_{4,\max}$
11	$Z_{1,\min}, Z_{2,\max}, Z_{3,\min}, Z_{4,\max}$
12	$Z_{1,\max}, Z_{2,\max}, Z_{3,\min}, Z_{4,\max}$
13	$Z_{1,\min}, Z_{2,\min}, Z_{3,\max}, Z_{4,\max}$
14	$Z_{1,\max}, Z_{2,\min}, Z_{3,\max}, Z_{4,\max}$
15	$Z_{1,\min}, Z_{2,\max}, Z_{3,\max}, Z_{4,\max}$
16	$Z_{1,\max}, Z_{2,\max}, Z_{3,\max}, Z_{4,\max}$

A could then be rewritten as follows:

$$\begin{aligned} A &= \sum_{i=1}^{r} h_i(z(t)) \times A_i \\ \text{where } \forall i \in \{1,2,\ldots,r\},\ h_i(z(t)) &= \frac{w_i(t)}{\sum_{i=1}^{r} w_i(t)} \end{aligned} \tag{4.16}$$

where the membership functions, are as follows:

$$\begin{cases} S_1(Z_1(t)) = \frac{Z_1(t) - Z_{1\max}}{Z_{1\min} - Z_{1\max}}, S_2(Z_1(t)) = 1 - S_1(Z_1(t)) \\ R_1(Z_2(t)) = \frac{Z_2(t) - Z_{2\max}}{Z_{2\min} - Z_{2\max}}, R_2(Z_2(t)) = 1 - R_1(Z_2(t)) \\ M_1(Z_3(t)) = \frac{Z_3(t) - Z_{3\max}}{Z_{3\min} - Z_{3\max}}, M_2(Z_3(t)) = 1 - M_1(Z_3(t)) \\ N_1(Z_4(t)) = \frac{Z_4(t) - Z_{4\max}}{Z_{4\min} - Z_{4\max}}, N_2(Z_4(t)) = 1 - N_1(Z_4(t)) \end{cases} \tag{4.17}$$

4.4.1.1 Validation of the fuzzy model of 16 rules

for the simulation of the fuzzy model on SIMULINK , we will take the values

```
% limits on the Tank's heights
h_min = 0.1;
h_max = 20;
Z_min = 1/sqrt(h_max);
Z_max = 1/sqrt(h_min);

X0 = [10;10;7;8] ; % initial conditions
```

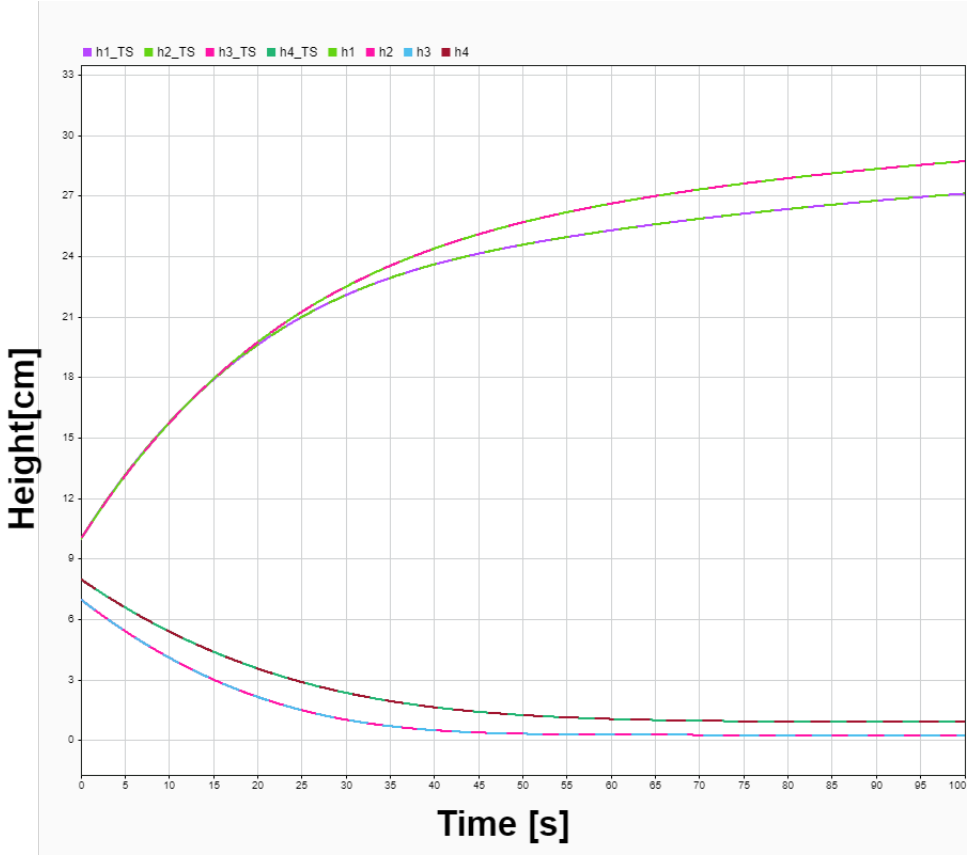


Figure 4.8: Time responses of the original states and its fuzzy approximations.

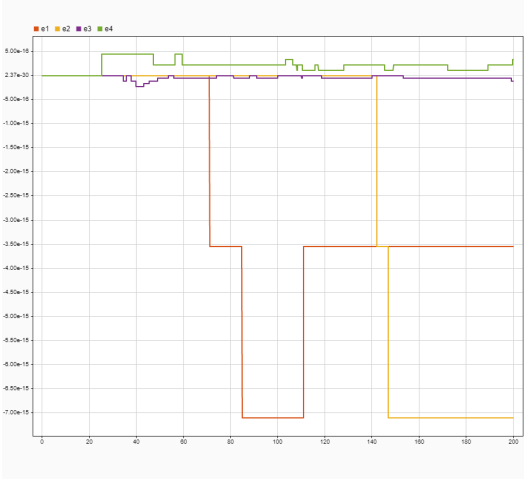


Figure 4.9: Error between the original states and its fuzzy approximations.

4.4.2 fuzzy model with 04 rules

in the same manner used to construct the fuzzy model with 16 rules, we have :

From the nonlinear system of the Tank system, one can see that the non-linearity is related to the states $\sqrt{h_i(t)}$ for $i = 1, 2, 3, 4$[62] Since equation of the tank system clearly shows a relationship between $\sqrt{h_1(t)}, \sqrt{h_2(t)}$ and $\sqrt{h_3(t)}, \sqrt{h_4(t)}$ we can directly obtain the variables $h_1(t), h_2(t)$ by approximating the variables $h_3(t), h_4(t)$.

Note that, the operating point for the states $h_1$and h_2 must be fulfilled the nonlinear relationships.

In the steady state we have:

$$\begin{aligned} \dot{h}_3(t) = 0 \implies v_2(t) = \frac{a_3\sqrt{2gh_3(t)}}{(1-\gamma_2)K_2} \\ \dot{h}_4(t) = 0 \implies v_1(t) = \frac{a_4\sqrt{2gh_4(t)}}{(1-\gamma_1)K_1} \end{aligned} \tag{4.18}$$

Substituting into the equations of $\dot{h}_1(t)$ and $\dot{h}_2(t)$ we will have:

$$\begin{aligned} h_1(t) = \left(\frac{a_3\sqrt{h_3(t)}}{a_1} + \frac{\gamma_1 a_4\sqrt{h_4(t)}}{(1-\gamma_1)a_1}\right)^2 \\ h_2(t) = \left(\frac{a_4\sqrt{h_4(t)}}{a_2} + \frac{\gamma_2 a_3\sqrt{h_3(t)}}{(1-\gamma_2)a_2}\right)^2 \end{aligned} \tag{4.19}$$

Note that,

$$\forall i \in \{1,2,3,4\}, \quad \sqrt{h_i(t)} = \frac{h_i(t)}{\sqrt{h_i(t)}} \tag{4.20}$$

thus, by considering the following variable change $Z_i(t) = \frac{1}{\sqrt{h_i}}$ for $i = 3,4$ and assuming That $Z_{i\min} \leq Z_i(t) \leq Z_{i\max}$

the following approximation can be obtained

$$\begin{cases} Z_3(t) & = M_1(Z_3(t)) \times Z_{3\min} + M_2(Z_3(t)) \times Z_{3\max} \\ Z_4(t) & = N_1(Z_4(t)) \times Z_{4\min} + N_2(Z_4(t)) \times Z_{4\max} \end{cases} \tag{4.21}$$

with $M_1(Z_3(t)) + M_2(Z_3(t)) = 1$ and $N_1(Z_4(t)) + N_2(Z_4(t)) = 1$

assuming that the height of each tank as the system output, one can get

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

where:

$$X = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix}, \; U = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}, \; C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$A = \begin{bmatrix} \frac{-a_1\sqrt{2g}}{A_1\sqrt{h_1}} & 0 & \frac{a_3\sqrt{2g}}{A_1\sqrt{h_3}} & 0 \\ 0 & \frac{-a_2\sqrt{2g}}{A_2\sqrt{h_2}} & 0 & \frac{a_4\sqrt{2g}}{A_2\sqrt{h_4}} \\ 0 & 0 & -\frac{a_3\sqrt{2g}}{A_3\sqrt{h_3}} & 0 \\ 0 & 0 & 0 & -\frac{a_4\sqrt{2g}}{A_4\sqrt{h_4}} \end{bmatrix}$$

$$B = \begin{bmatrix} \frac{\gamma_1 K_1}{A_1} & 0 \\ 0 & \frac{\gamma_2 K_2}{A_2} \\ 0 & \frac{(1-\gamma_2)K_2}{A_3} \\ \frac{(1-\gamma_1)K_1}{A_4} & 0 \end{bmatrix}$$

the matrix A can be rewritten as follows $A = \sum_{i=1}^{r} h_i(z(t))A_i,$ for $i = 1, 2, 3, \ldots, r$

$$A_1 = \begin{bmatrix} -\delta_{11}^1 & 0 & \frac{a_3\sqrt{2g}Z_{3\min}}{A_1} & 0 \\ 0 & -\delta_{22}^1 & 0 & \frac{a_4\sqrt{2g}Z_{4\min}}{A_2} \\ 0 & 0 & -\frac{a_3\sqrt{2g}Z_{3\min}}{A_3} & 0 \\ 0 & 0 & 0 & -\frac{a_4\sqrt{2g}Z_{4\min}}{A_4} \end{bmatrix} \qquad A_2 = \begin{bmatrix} -\delta_{11}^2 & 0 & \frac{a_3\sqrt{2g}Z_{3\min}}{A_1} & 0 \\ 0 & -\delta_{22}^2 & 0 & \frac{a_4\sqrt{2g}Z_{4\max}}{A_2} \\ 0 & 0 & -\frac{a_3\sqrt{2g}Z_{3\min}}{A_3} & 0 \\ 0 & 0 & 0 & -\frac{a_4\sqrt{2g}Z_{4\max}}{A_4} \end{bmatrix}$$

$$A_3 = \begin{bmatrix} -\delta_{11}^3 & 0 & \frac{a_3\sqrt{2g}Z_{3\max}}{A_1} & 0 \\ 0 & -\delta_{22}^3 & 0 & \frac{a_4\sqrt{2g}Z_{4\min}}{A_2} \\ 0 & 0 & -\frac{a_3\sqrt{2g}Z_{3\max}}{A_3} & 0 \\ 0 & 0 & 0 & -\frac{a_4\sqrt{2g}Z_{4\min}}{A_4} \end{bmatrix} \qquad A_4 = \begin{bmatrix} -\delta_{11}^4 & 0 & \frac{a_3\sqrt{2g}Z_{3\max}}{A_1} & 0 \\ 0 & -\delta_{22}^4 & 0 & \frac{a_4\sqrt{2g}Z_{4\max}}{A_2} \\ 0 & 0 & -\frac{a_3\sqrt{2g}Z_{3\max}}{A_3} & 0 \\ 0 & 0 & 0 & -\frac{a_4\sqrt{2g}Z_{4\max}}{A_4} \end{bmatrix}$$

and we have : the matrix A can be rewritten as follows $A = \sum_{i=1}^{r} h_i(z(t)) \times A_i$

$$\forall i \in \{1, 2, \ldots, r\}, h_i(z(t)) = \frac{w_i(t)}{\sum_{i=1}^{r} w_i(t)}$$

Where:

$$\begin{aligned} w_1(z(t)) &= M_1(Z_3(t)) \times N_1(Z_4(t)) \\ w_2(z(t)) &= M_1(Z_3(t)) \times N_2(Z_4(t)) \\ w_3(z(t)) &= M_2(Z_3(t)) \times N_1(Z_4(t)) \\ w_4(z(t)) &= M_2(Z_3(t)) \times N_2(Z_4(t)) \end{aligned} \tag{4.22}$$

$$\begin{aligned} M_1(Z_3(t)) &= \frac{Z_3(t) - Z_{3\max}}{Z_{3\min} - Z_{3\max}}, \qquad M_2(Z_3(t)) = 1 - M_1(Z_3(t)) \\ N_1(Z_4(t)) &= \frac{Z_4(t) - Z_{4\max}}{Z_{4\min} - Z_{4\max}}, \qquad N_2(Z_4(t)) = 1 - N_1(Z_4(t)) \end{aligned} \tag{4.23}$$

$$\delta_{11}^{1} = \frac{a_1\sqrt{2g}}{A_1} \times \frac{1}{\frac{a_3}{a_1 Z_{3\,\min}} + \frac{\gamma_1 a_4}{(1-\gamma_1)a_1 Z_{4\,\min}}}$$

$$\delta_{22}^{1} = \frac{a_2\sqrt{2g}}{A_2} \times \frac{1}{\frac{a_4}{a_2 Z_{4\,\min}} + \frac{\gamma_2 a_3}{(1-\gamma_2)a_2 Z_{3\,\min}}}$$

$$\delta_{11}^{2} = \frac{a_1\sqrt{2g}}{A_1} \times \frac{1}{\frac{a_3}{a_1 Z_{3\,\min}} + \frac{\gamma_1 a_4}{(1-\gamma_1)a_1 Z_{4\,\max}}}$$

$$\delta_{22}^{2} = \frac{a_2\sqrt{2g}}{A_2} \times \frac{1}{\frac{a_4}{a_2 Z_{4\,\max}} + \frac{\gamma_2 a_3}{(1-\gamma_2)a_2 Z_{3\,\min}}} \quad (4.24)$$

$$\delta_{11}^{3} = \frac{a_1\sqrt{2g}}{A_1} \times \frac{1}{\frac{a_3}{a_1 Z_{3\,\max}} + \frac{\gamma_1 a_4}{(1-\gamma_1)a_1 Z_{4\,\min}}}$$

$$\delta_{22}^{3} = \frac{a_2\sqrt{2g}}{A_2} \times \frac{1}{\frac{a_4}{a_2 Z_{4\,\min}} + \frac{\gamma_2 a_3}{(1-\gamma_2)a_2 Z_{3\,\max}}}$$

$$\delta_{11}^{4} = \frac{a_1\sqrt{2g}}{A_1} \times \frac{1}{\frac{a_3}{a_1 Z_{3\,\max}} + \frac{\gamma_1 a_4}{(1-\gamma_1)a_1 Z_{4\,\max}}}$$

$$\delta_{22}^{4} = \frac{a_2\sqrt{2g}}{A_2} \times \frac{1}{\frac{a_4}{a_2 Z_{4\,\max}} + \frac{\gamma_2 a_3}{(1-\gamma_2)a_2 Z_{3\,\max}}} \quad (4.25)$$

4.4.2.1 Validation of the fuzzy model

for the simulation of the fuzzy model on SIMULINK , we will take these values

```
%% Parameters Definition
A1=138.9;
A3=138.9;
A2=138.9;
A4=138.9;
a1=0.5027;
a3=0.5027;
a2=0.5027;
a4=0.5027;
gamma_1=0.836;
gamma_2=0.897;
K1=26.00;
K2=22.94;
g = 981;

% limits on the Tank's heights
h_min = 0.1;
h_max = 20;
Z_min = 1/sqrt(h_max);
Z_max = 1/sqrt(h_min);

X0 = [7;7;3;3] ; % initial conditions
```

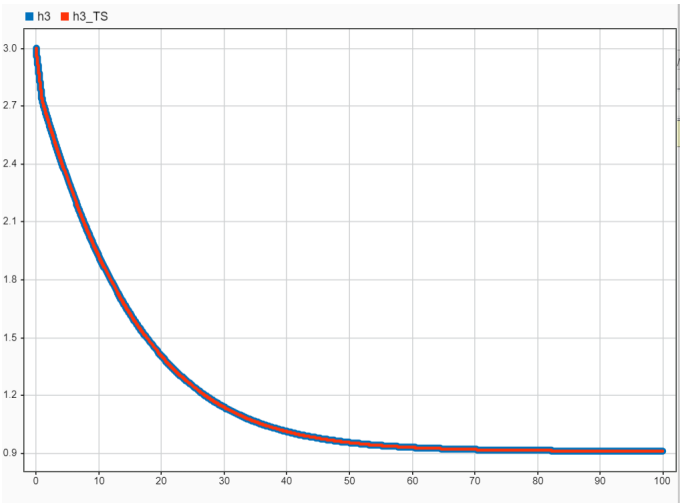


Figure 4.10: Time responses of the original state h_3 and its fuzzy approximation.

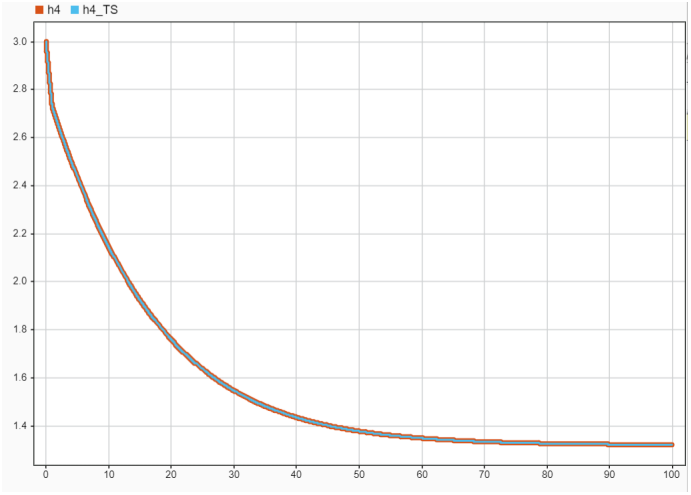


Figure 4.11: Time responses of the original state h_4 and its fuzzy approximation.

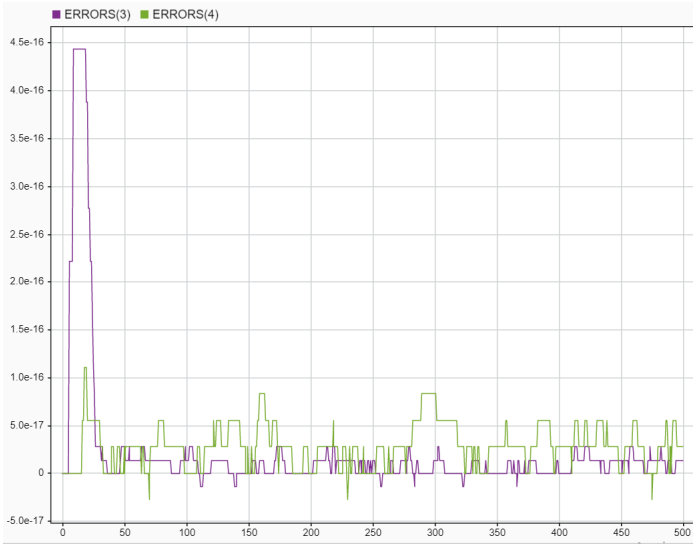


Figure 4.12: Error between original state and its fuzzy approximation.

In [62], they proven that you don't need to have the right estimation on the states h_1 and h_2 to actually control our system. In fact, an error could be seen on these two states due to the made up approximation. This will be proven in the next section by demonstrating the simulations results.

4.5 Stability and Stabilization of T-S Fuzzy Systems

The stability of systems in closed loop is one of the most significant problems in control theory. The T-S systems stability analysis was the objective of several works, In the majority, the goal was the obtention of a global asymptotic stability by applying Lyapunov's direct method based on Lyapunov functions, which measure the system's energy[2]

Whether for stability analysis or for the calculation of stabilizing control laws, the approach based on the second method of Lyapunov remains by far the most used. Indeed, the structure of a TS model lends itself to the application of classical automatic control concepts and approaches because this type of model is an interconnection of a set of local linear models.[51]

4.5.1 Lyapunov functions

In general, there is no a systematic method to find candidate Lyapunov functions. The degree of conservatism of the obtained stability conditions depends on the Lyapunov function form and the system structure. Different Lyapunov functions forms are used by different authors in the literature

4.5.1.1 Quadratic Lyapunov function

This one is the classical form, it is given by:

$$V(x(t)) = x^T(t)Px(t), \quad P > 0, \quad P^T = P \tag{4.26}$$

used initially to stability study of linear systems and then for MIMO nonlinear systems, The principle of the method is to search a positive definite matrix P, by the way of convex formulation of the problem. The drawback of this quadratic approach is the conservative stability conditions, but it remains from a practical point of view easy to implement.[2]

4.5.1.2 Non-Quadratic Lyapunov function

This function is of the form:

$$V(x(t)) = \sum_{i=1}^{r} h_i(z(t))x^T(t)P_ix(t) \tag{4.27}$$

where P_i is a positive definite matrix and $h_i(z(t)) \geq 0$,

$$\sum_{i=1}^{r} h_i(z(t)) = 1. \tag{4.28}$$

An interesting advantage is that, the non-quadratic form of Lyapunov function takes into account the speed variation of the decision variables, what allows the conservatism reduction and more relaxed stability conditions

4.5.2 Stability Analysis of TS Models

To find a Lyapunov function amounts to finding a positive definite matrix P, we speak about quadratic stability. The following stability theorem that is based on quadratic Lyapunov functions gives sufficient

conditions to assure stability of the open loop T-S fuzzy system given by:

$$\dot{x}(t) = \sum_{i=1}^{r} h_i(z(t)) A_i x(t), \tag{4.29}$$

Theorem 1 (Tanaka & Sugeno, 1992)

The equilibrium of a fuzzy system is globally asymptotically stable if there exists a common positive definite matrix P such that

$$A_i^T P + P A_i < 0, \quad i = 1, \ldots, r \tag{4.30}$$

Where, a common P has to exist for all sub-models [52].

This theorem presents sufficient conditions for quadratic stability. However, they are conservative since the $h_i(z(t))$ are not taken into account. The common P problem can be solved efficiently via convex optimization techniques and LMIs (Linear Matrix Inequalities); we call this an LMI feasibility problem. Therefore, recasting a control problem (such as stabilization via a PDC controller) as an LMI problem is equivalent to finding a "solution" to the original problem. The existence of P depends on two conditions: the first one is related to the stability of all sub-models, where each matrix A_i must be Hurwitz. The second condition relates to the existence of a common Lyapunov function for the r sub-models. It requires that $\sum_{i=1}^{r} A_i$ must also be Hurwitz. However, if r, that is, the number of IF-THEN rules, is large, it might be difficult to find a common P [52].

The proof of Theorem 1 is given : In the following polytopic system [52]

$$\dot{x}(t) = \sum_{i=1}^{r} h_i(z(t)) A_i x(t)$$

the derivative of V along the nonzero trajectory $x(t)$ is given by

$$\dot{V}(x(t)) = \dot{x}^T(t) P x(t) + x^T(t) P \dot{x}(t)$$

$$= \left(\sum_{i=1}^{r} h_i(z(t)) A_i x(t) \right)^T P x(t) + x^T(t) P \left(\sum_{i=1}^{r} h_i(z(t)) A_i x(t) \right)$$

$$= \sum_{i=1}^{r} h_i(z(t)) \left[x^T(t)(A_i^T P) x(t) + x^T(t)(P A_i) x(t) \right]$$

$$= \sum_{i=1}^{r} h_i(z(t)) x^T(t) \left(A_i^T P + P A_i \right) x(t) < 0$$

since $A_i^T P + P A_i$ is negative when P is positive definite, then the polytopic system is quadratically stable if there exists a symmetric matrix P satisfying the following inequalities:

$$P > 0, \tag{4.31}$$

$$A_i^T P + P A_i < 0, \quad i = 1, \ldots, r \tag{4.32}$$

[52]

4.5.3 Stabilitsation of TS Models

In the literature, different control laws were proposed to stabilize fuzzy models. These, are based on stability constraints transformable into LMIs to obtain the gains matrices.[2]

4.5.3.1 Parallel distributed compensation

The main idea of the PDC controller design is to derive each control rule from the corresponding rule of T-S fuzzy model so as to compensate it. The resulting overall fuzzy controller, which is nonlinear in general, is a fuzzy blending of each individual linear controller, knowing that the fuzzy controller shares the same fuzzy sets with the fuzzy model; Wang et al, [52] used this concept to design fuzzy controllers to stabilize fuzzy systems.

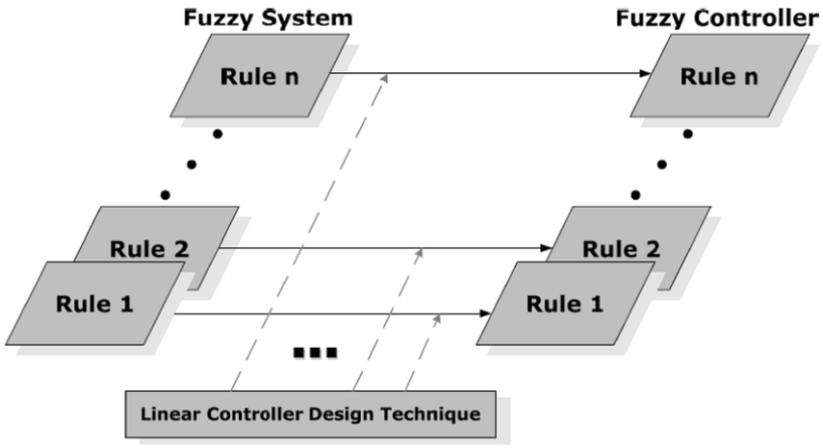


Figure 4.13: PDC controller design[14]

For the fuzzy system, the following fuzzy controller via PDC is obtained [52]:

$$\textbf{Rule i}: \textit{IF } z_1(t) \text{ is } M_{i1} \text{ and } \dots \text{ and } z_p(t) \text{ is } M_{ip} \textit{ THEN } u(t) = -F_i x(t), \quad i = 1, 2, \dots, r$$

which has a state feedback controller in the consequent parts. The overall fuzzy controller is represented by

$$u(t) = -\sum_{i=1}^{r} h_i(z(t)) F_i x(t) \tag{4.33}$$

The PDC scheme that stabilizes the T-S fuzzy model was proposed by Wang et al., as a design framework comprising a control algorithm and a stability test using optimization involving LMI constraints. The goal is to find appropriate F_i gains that ensure the closed-loop stability [56].

Stability conditions in closed loop

The overall T-S fuzzy system is given by:

$$\dot{x}(t) = \sum_{i=1}^{r} h_i(z(t))(A_i x(t) + B_i u(t)),$$

$$y(t) = \sum_{i=1}^{r} h_i(z(t)) C_i x(t)$$

We note that this equation is a polytopic form of the fuzzy system. Hence, by substituting expression of the PDC control u in the TS fuzzy system, we obtain the T-S closed-loop fuzzy system as follows:

$$\dot{x}(t) = \sum_{i=1}^{r} \sum_{j=1}^{r} h_i(z(t)) h_j(z(t)) [A_i - B_i F_j] x(t), \tag{4.34}$$

which can be rewritten as

$$\dot{x}(t) = \sum_{i=1}^{r} h_i(z(t)) h_i(z(t)) G_{ii} x(t) + 2 \sum_{i=1}^{r} \sum_{i<j} h_i(z(t)) h_j(z(t)) \left\{ \frac{G_{ij} + G_{ji}}{2} \right\} x(t), \tag{4.35}$$

where $G_{ij} = A_i - B_i F_j$ and $G_{ii} = A_i - B_i F_i$.[52]

we obtain the closed loop stability conditions given by the following theorem.

Theorem 2 (Tanaka & Wang [52]): The equilibrium of the continuous fuzzy control system is globally asymptotically stable if there exists a common positive definite matrix P such that the following two conditions are satisfied:

$$G_{ii}^T P + P G_{ii} < 0, \qquad i = 1, \ldots, r, \tag{4.36}$$

$$\frac{1}{2}\left(G_{ij} + G_{ji}\right)^T P + P \frac{1}{2}\left(G_{ij} + G_{ji}\right) \leq 0, \qquad i = 1, \ldots, r, \quad i < j \text{ s.t. } h_i \cap h_j \neq \emptyset \tag{4.37}$$

The proof of this theorem follows directly from theorem 1. We notice that these conditions contributes to the conservatism reduction since, it is not necessary that all the sub-models are stable.

The common B matrix case:

Assume that $B_1 = B_2 = \ldots = B_r$. The equilibrium of the fuzzy control system is globally asymptotically stable if there exists a common positive matrix P satisfying :

$$G_{ii}^T P + P G_{ii} < 0, \quad i = 1, \ldots, r, \tag{4.38}$$

The objective is to select F_i to stabilize the closed-loop system. The stability conditions corresponding to a quadratic Lyapunov function were derived by Tanaka and Sugeno in [52]. They give sufficient conditions for quadratic stabilization by the following theorem:

Theorem 3 (Tanaka & Wang [52]): The fuzzy system can be stabilized via the PDC controller if there exists a common positive definite matrix X and M_i $(i = 1, \ldots, r)$ such that

$$-X A_i^T - A_i X + M_i^T B_i^T + B_i M_i > 0, \tag{4.39}$$

$$-X A_i^T - A_i X - X A_j^T - A_j X + M_j^T B_i^T + B_i M_j + M_i^T B_j^T + B_j M_i \geq 0, \qquad \forall i < j \text{ s.t. } h_i \cap h_j \neq \emptyset, \tag{4.40}$$

where

$$X = P^{-1}, \quad M_i = F_i X.$$

The feedback gains F_i and the common P are given by

$$P = X^{-1}, \quad F_i = M_i X^{-1}.$$

Meanwhile, the single quadratic Lyapunov function is given by

$$V(x(t)) = x(t) X^{-1} x(t).$$

This approach requires finding a common positive definite matrix P for r sub-models, which makes it very conservative. An attempt to reduce the conservatism using the same Lyapunov function was given by Tanaka et al. [51], who proposed relaxed stability conditions given by this theorem.

Theorem 4 [52]: Assume that the number of rules that fire for all t is less than or equal to s, where $1 < s \leq r$. The equilibrium of the continuous fuzzy control system described before is globally

asymptotically stable if there exists a common positive definite matrix P and a common positive semidefinite matrix Q such that

$$A_i^T P + P A_i - (s-1)Q < 0, \qquad i = 1, \ldots, r, \tag{4.41}$$

$$\frac{1}{2}\left(G_{ij} + G_{ji}\right)^T P + P\frac{1}{2}\left(G_{ij} + G_{ji}\right) - Q \leq 0, \qquad i = 1, \ldots, r,$$

$$i < j \text{ s.t. } h_i \cap h_j \neq \emptyset. \tag{4.42}$$

where s > 1.

4.5.3.2 State stabilization for the fuzzy model of 16 rules via PDC control

we present the simulation results of the proposed control strategy for stabilisation of the states using PDC control on our TS fuzzy system of 16 rules

```
% limits on the Tank's heights
h_min = 0.1;
h_max = 20;
Z_min = 1/sqrt(h_max);
Z_max = 1/sqrt(h_min);

X0 = [7;7;3;3] ; % initial conditions
```

To stabilize the system, we use a state feedback of the form:

$$u(t) = -\sum_{i=1}^{r} h_i(z(t)) F_i x(t) \quad for i = 1, 2, \ldots, 16$$

The matrices P and F_i are obtained by solving LMIs of Theorem 2 using Yalmip. Thus, from the T-S formalization of the nonlinear model of the coupled tanks, P and F_i are given in C.

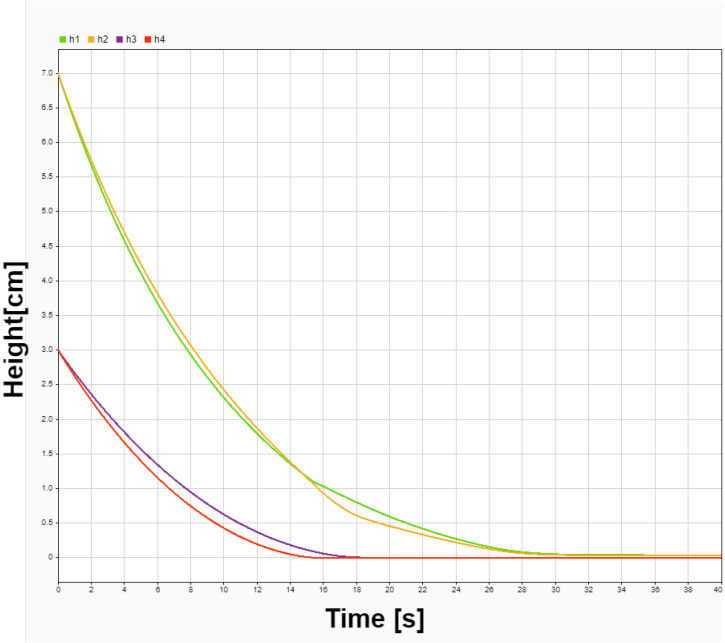


Figure 4.14: The states of the system in closed-loop (PDC control law)

4.5.3.3 State stabilization for the fuzzy model of 4 rules via PDC control

we present the simulation results of the proposed control strategy for stabilisation of the states using PDC control

```
% limits on the Tank's heights
h_min = 0.1;
h_max = 20;
Z_min = 1/sqrt(h_max);
Z_max = 1/sqrt(h_min);

X0 = [7;6;3;2] ; % initial conditions
```

To stabilize the system, we use a state feedback of the form:

$$u(t) = -\sum_{i=1}^{r} h_i(z(t))F_i x(t) = -(h_1 \times F_1 + h_2 \times F_2 + h_3 \times F_3 + h_4 \times F_4)x(t)$$

The matrices P and F_i are obtained by solving LMIs of Theorem 2 using Yalmip. Thus, from A1, A2, A3, A4, B, deduced from the T-S formalization of the nonlinear model of the coupled tanks, P and F_i are given in C.

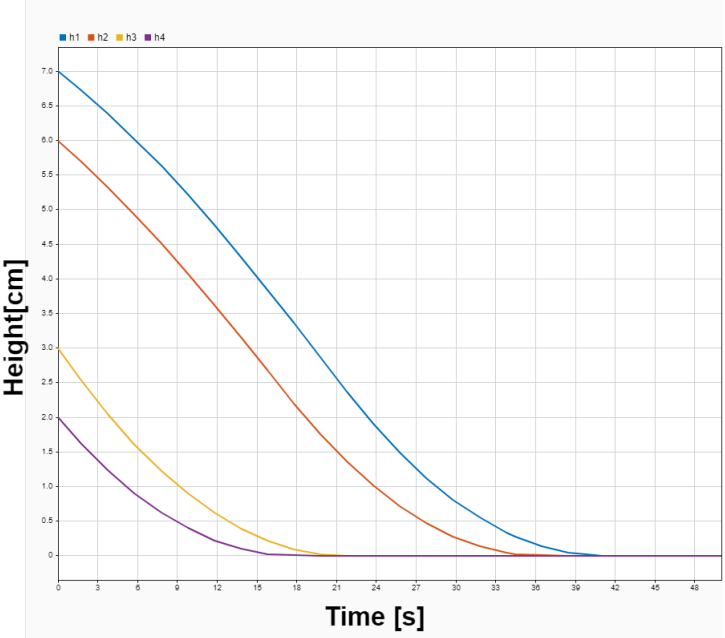


Figure 4.15: The states of the system in closed-loop (PDC control law)

It is observed that the PDC method allows for the stabilization of the nonlinear system using a controller derived from a TS fuzzy model with 04 and 16 rules respectively, With each controller having its own stabilization time.

4.6 Trajectory tracking (PDC with integral action)

As in the case of a linear system model, adding an integral action allows for zero steady-state error. This is already used with a PDC controller to synthesize a stabilizing control law and ensure setpoint tracking at the same time.

In order to achieve the natural equilibrium without static error, an integral action must be added. Theoretically, the integral action consists in augmenting the initial system state $x(t)$[59]

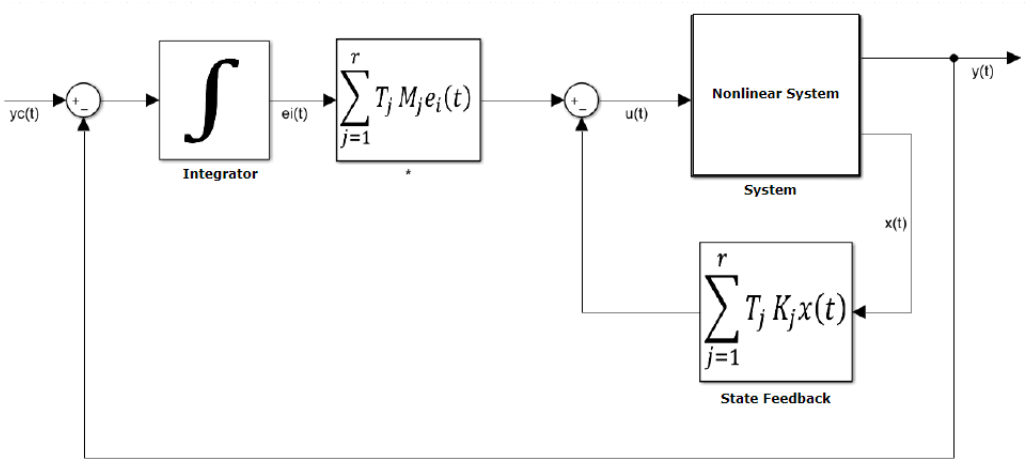


Figure 4.16: PDC with Integral structure[6]

This control structure ensures the convergence of the error $e(t)$ to the origin, which allows for setpoint tracking. We will use the same method to stabilize the TS fuzzy system model; it is sufficient to rewrite the model by augmenting the state. From the structure, we can write:

$$\dot{E} = y_c - y \tag{4.43}$$

The T-S fuzzy system is given by:

$$\dot{x}(t) = \sum_{i=1}^{r} h_i(z(t))(A_i x(t) + B_i u(t)),$$

$$y(t) = \sum_{i=1}^{r} h_i(z(t)) C_i x(t)$$

We consider the augmented state

$$X(t) = \begin{bmatrix} x(t) \\ E(t) \end{bmatrix} \tag{4.44}$$

which gives the following model:

$$\dot{X}(t) = \begin{bmatrix} \dot{x} \\ \dot{E} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{r} h_i(z(t))(A_i x(t) + B_i u(t)) \\ y_c - y \end{bmatrix} \tag{4.45}$$

By replacing the output $y(t)$ with its expression:

$$\dot{X}(t) = \begin{bmatrix} \sum_{i=1}^{r} h_i(z(t))(A_i x(t) + B_i u(t)) \\ y_c - \sum_{i=1}^{r} h_i(z(t)) C_i x(t) \end{bmatrix} \tag{4.46}$$

Since $\sum_{i=1}^{r} h_i(z(t)) = 1$ and y_c does not depend on i:

$$\dot{X}(t) = \begin{bmatrix} \sum_{i=1}^{r} h_i(z(t))(A_i x(t) + B_i u(t)) \\ \sum_{i=1}^{r} h_i(z(t))(y_c - C_i x(t)) \end{bmatrix} \tag{4.47}$$

$$\dot{X}(t) = \sum_{i=1}^{r} h_i(z(t)) \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} A_i x(t) + B_i u(t) \\ y_c - C_i x(t) \end{bmatrix} \tag{4.48}$$

$$\dot{X}(t) = \sum_{i=1}^{r} h_i(z(t)) \left(\begin{bmatrix} A_i & 0 \\ -C_i & 0 \end{bmatrix} x(t) + \begin{bmatrix} B_i \\ 0 \end{bmatrix} u(t) + \begin{bmatrix} 0 \\ I \end{bmatrix} y_c \right) \tag{4.49}$$

We define:

$$\bar{A}_i = \begin{bmatrix} A_i & 0 \\ -C_i & 0 \end{bmatrix}, \quad \bar{B}_i = \begin{bmatrix} B_i \\ 0 \end{bmatrix}, \quad B_c = \begin{bmatrix} 0 \\ I \end{bmatrix} \tag{4.50}$$

We apply the PDC control law to the augmented model:

$$u(t) = -\sum_{i=1}^{r} h_i(z(t)) \bar{F}_j X(t) \tag{4.51}$$

With:

$$\bar{F}_j = \begin{bmatrix} F_j & -M \end{bmatrix} \tag{4.52}$$

To calculate the $\bar{F}_j$, it is sufficient to solve the following inequalities:

$$X A_i^T + A_i X - N_i^T B_i^T - B_i N_i + (r-1)M + (r-1)M < 0, \quad i \in \{1, \ldots, r\} \tag{4.53}$$

$$X A_i^T + A_i X + X A_j^T + A_j X - N_j^T B_i^T - B_i N_j - N_i^T B_j^T - B_j N_i - M \leq 0, \quad i < j \tag{4.54}$$

With:

$$A_i = \bar{A}_i \quad \text{and} \quad B_i = \bar{B}_i \tag{4.55}$$

Theorem 5 [52]: *Assume that the initial condition $x(0)$ is known. The constraint $\|u(t)\|_2 \leq \mu$ is enforced at all times $t \geq 0$ if the LMIs*

$$\begin{bmatrix} 1 & x(0)^T \\ x(0) & X \end{bmatrix} \geq 0, \tag{4.56}$$

$$\begin{bmatrix} X & M_i^T \\ M_i & \mu^2 I \end{bmatrix} \geq 0 \tag{4.57}$$

hold, where $X = P^{-1}$ and $M_i = F_i X$[52]. The proof related to this theorem is given in [52].

Important Note : This condition gives us an upper limit on the norm of our control signal u, still we don't have an LMI that express a constraint for the lower value of u [52]. This will imply that our control signal will always start (and for majority of its applications) with a negative value and our control input doesn't tolerate that. Thus, we must put a saturation that limits the control signal to a value of 0 for all the values smaller than 0, which happens generally when we launch the controller from a given set point. The effects will be shown directly through the next section.

4.6.1 Trajectory tracking for the TS fuzzy system with 16 rules

4.6.1.1 Minimum Phase setting

This phase corresponds to

$$1 < \gamma_1 + \gamma_2 < 2$$

The matrices F_i and M_i are obtained by solving LMIs using YALMIP. Thus, from the matrices deduced from the T-S formalization of the nonlinear model of the coupled tanks system,F_i and M_i are given in C.

We Assume that the initial condition $x(0) = [12.7; 12.4; 1.5; 1.5]$, and we simulate the same standard test that we conducted on our decentralized PI controller in 2.5.2. μ is tuned manually such that we have $u(t) \leq 10$. Thus, after some tests we put $\mu = 27$
the corresponding Simulation results are the following :

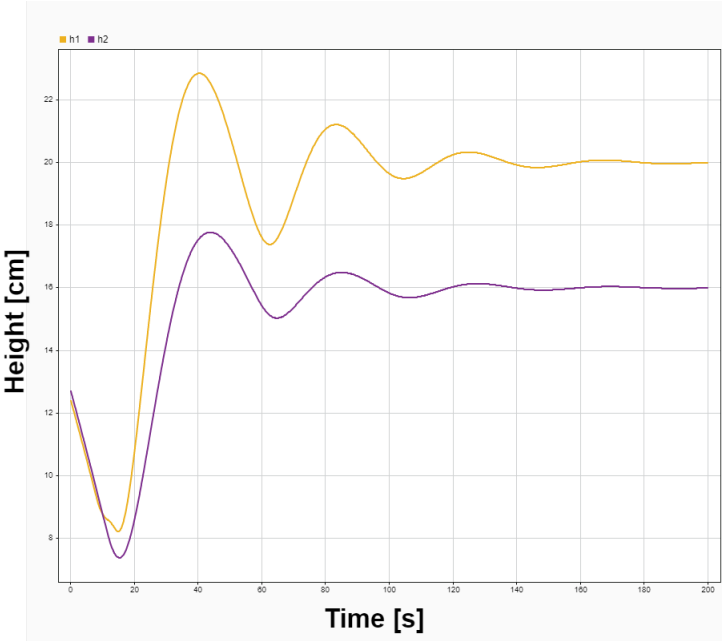


Figure 4.17: System's Output Response h_1 (yellow) and h_2 (purple) with the mentioned conditions

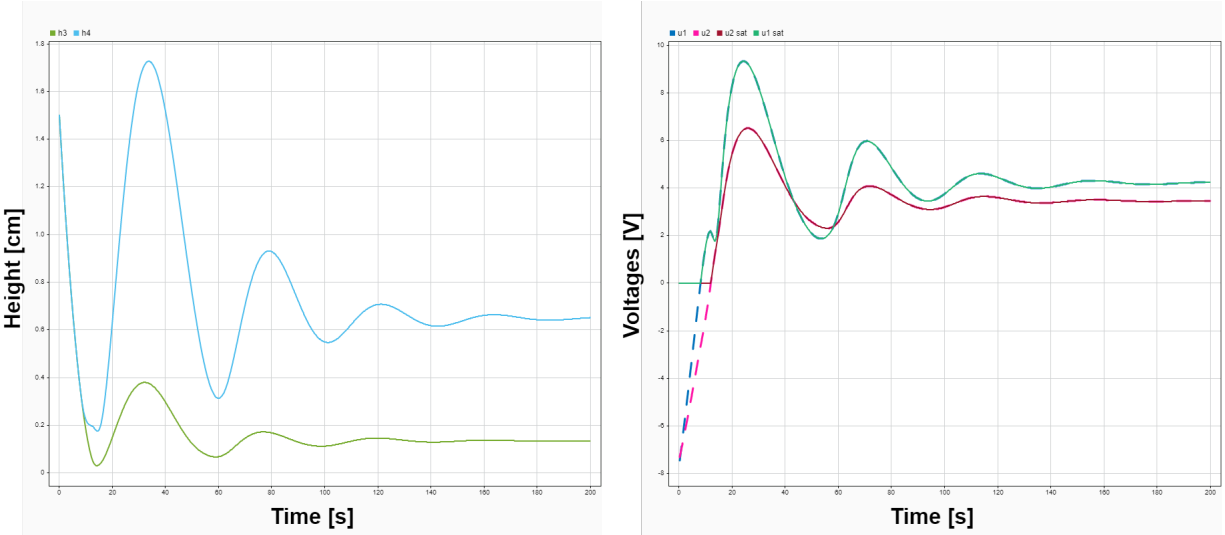


Figure 4.18: Corresponding State Response of h_3 (green) and h_4 (blue) (left) and the Control Signals (right)

4.6.1.2 Non-minimum Phase setting

This phase corresponds to

$$0 < \gamma_1 + \gamma_2 < 1$$

.

We will take $\gamma_1 = 0$ and $\gamma_2 = 0$. This would be equivalent to no coupling at all, seeming like it would be easier to do this task. Yet it is purely to see the non-minimum phase behavior as well as seeing the robustness of this particular controller in face of interconnections.

The matrices F_i and M_i are obtained by solving LMIs using YALMIP. Thus, from the matrices deduced from the T-S formalization of the nonlinear model of the coupled tanks system. F_i and M_i are given in C

We Assume that the initial condition $x(0) = [12.7; 12.4; 10; 10]$, and we simulate the same standard test defined before 2.5.2. μ is tuned manually such that we have $u(t) \leq 10$. Thus, after some tests we put $\mu = 20$
the corresponding Simulation results are the following :

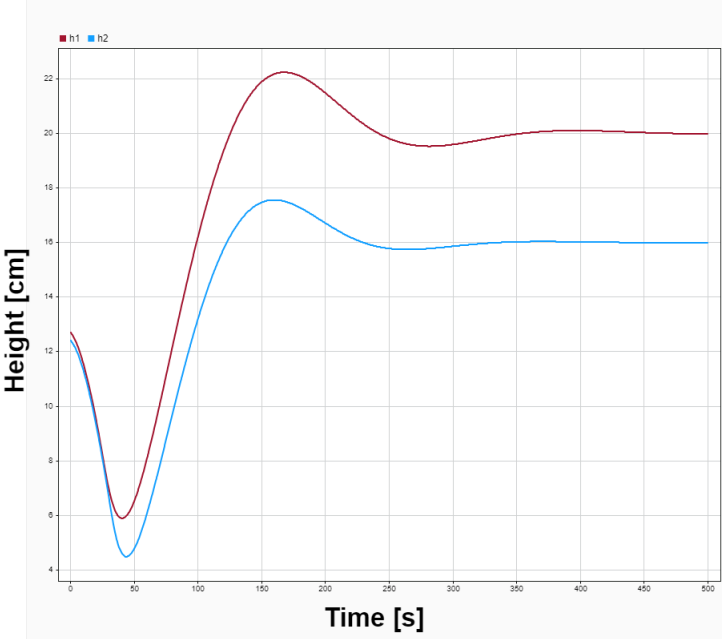


Figure 4.19: Output System Response h_1 (red) and h_2 (blue) with the mentioned conditions

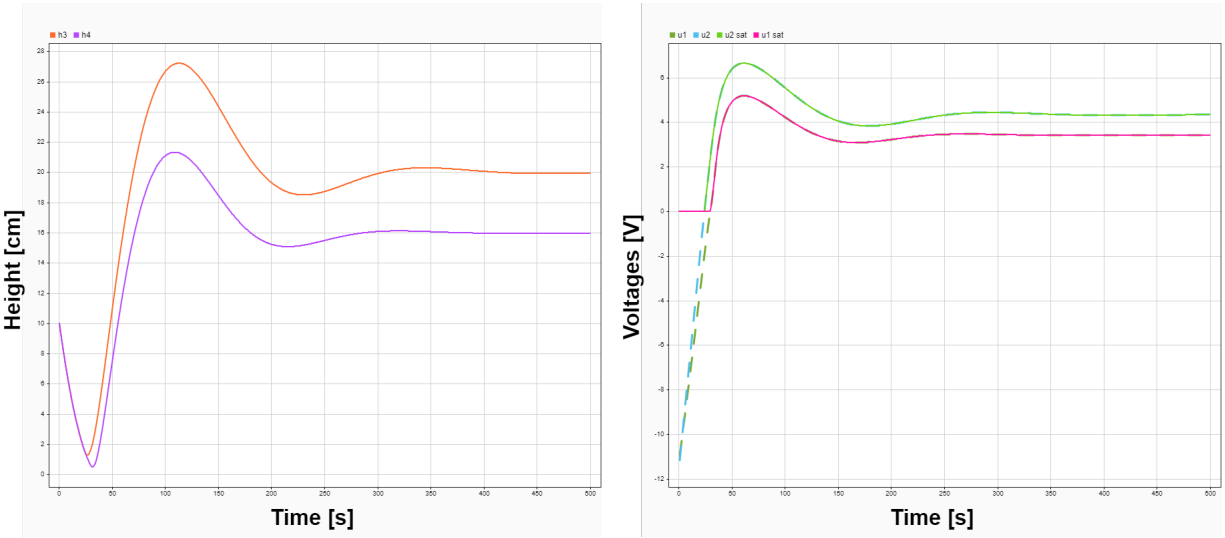


Figure 4.20: Corresponding State Response of h_3 (orange) and h_4 (purple) (left) and the Control Signals (right)

Observations

In the minimum phase setup, the system reaches stability more quickly, having fewer oscillations after the initial peak. Both outputs, h_1 and h_2, settle close to their target values and stabilize around their respective set points, with oscillations decreasing rapidly. On the other hand, the non-minimum phase setup takes a longer time to stabilize, with much more noticeable and prolonged oscillations. The initial overshoot is prominent in both configurations, with h_1 peaking at approximately 22.5s and h_2 peaking at 18.5s in the minimum phase and around 16.5 in the non-minimum phase. Additionally, the behaviors of h_3 and h_4 are similar, initially rising to levels driven by maximum input from the

pumps, then dipping slightly as values decrease.

Compared to our benchmark decentralized PI controller, the minimum phase response at 95% of the final value is much more slower (105 seconds to 14 seconds). A small to none overshoot was seen in the benchmark yet it is the case for the 16 rules based TS controller. The biggest difference between the two is that the response shown with the TS controller exhibits a non-minimum phase behavior that corresponds to the saturation on the lower bound (value 0) during the launch of the system.

On the other hand, the non-minimum phase response shows that the TS controller is a little faster with a time response of 225 seconds compared to the 330 seconds of the benchmark controller. The trough gets also bigger naturally due to the lower bound on our control signal.

4.6.2 Trajectory tracking for the TS fuzzy system with 4 rules

4.6.2.1 Minimum Phase setting

Previously in this chapter, we successfully derived the following matrices when establishing the fuzzy modeling with four rules.

$$A_1 = \begin{bmatrix} -0.0059 & 0 & 0.0059 & 0 \\ 0 & -0.0037 & 0 & 0.0037 \\ 0 & 0 & -0.0358 & 0 \\ 0 & 0 & 0 & -0.0358 \end{bmatrix} \qquad A_2 = \begin{bmatrix} -0.0263 & 0 & 0.0263 & 0 \\ 0 & -0.0222 & 0 & 0.0222 \\ 0 & 0 & -0.0358 & 0 \\ 0 & 0 & 0 & -0.5069 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} -0.0069 & 0 & 0.0069 & 0 \\ 0 & -0.0041 & 0 & 0.0041 \\ 0 & 0 & -0.5069 & 0 \\ 0 & 0 & 0 & -0.0358 \end{bmatrix} \qquad A_4 = \begin{bmatrix} -0.0831 & 0 & 0.0831 & 0 \\ 0 & -0.0522 & 0 & 0.0522 \\ 0 & 0 & -0.5069 & 0 \\ 0 & 0 & 0 & -0.5069 \end{bmatrix}$$

$$B = \begin{bmatrix} 0.1565 & 0 \\ 0 & 0.1481 \\ 0 & 0.0170 \\ 0.0307 & 0 \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

The matrices K_i and M_i are obtained by solving LMIs using MATLAB Yalmip Library. Thus, from the matrices deduced from the T-S formalization of the nonlinear model of the coupled tanks system,K_i and M_i are given in C.

We Assume that the initial condition $x(0) = [12.7; 12.4; 1.5; 1.5]$, and we simulate the same standard test defined before 2.5.2. μ is tuned manually such that we have $u(t) \leq 10$. Thus, after some tests we put $\mu = 27$

The corresponding Simulation results are the following :

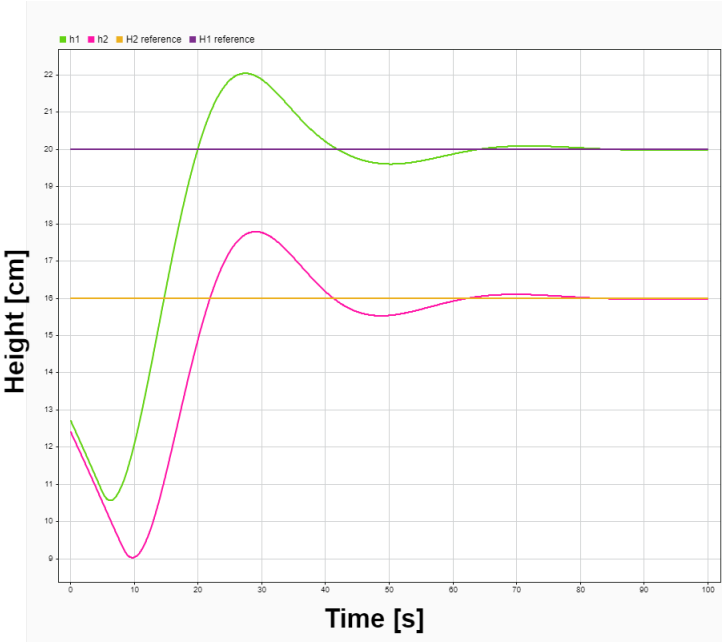


Figure 4.21: Output System Response h_1 (green) and h_2 (rose) with the mentioned conditions

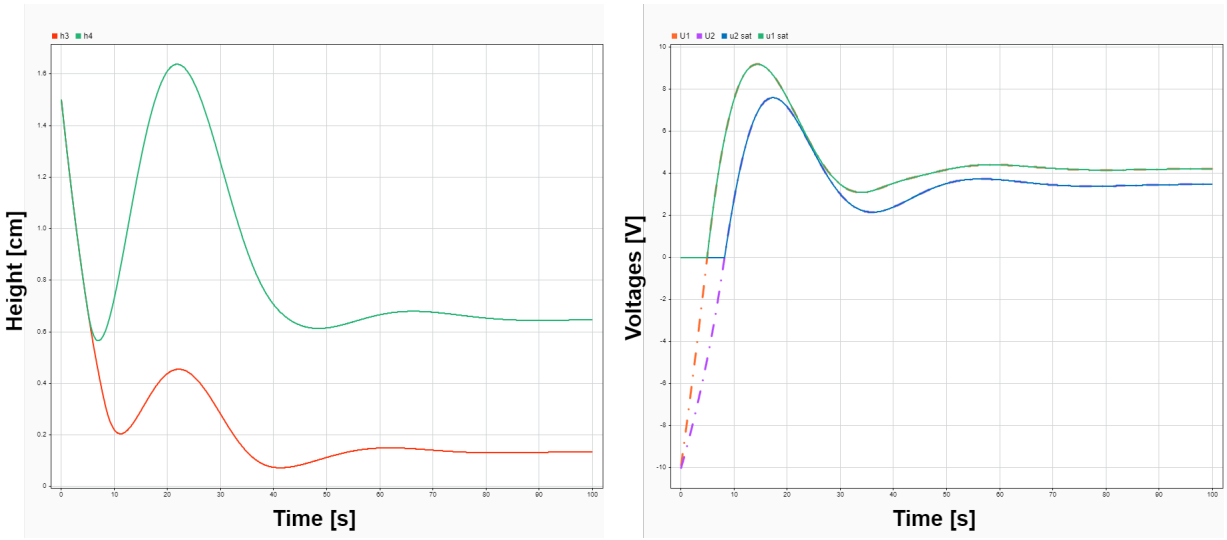


Figure 4.22: Corresponding State Response of h_3 (orange) and h_4 (green) (left) and the Control Signals (right)

4.6.2.2 Non-Minimum Phase setting

This phase corresponds to

$$0 < \gamma_1 + \gamma_2 < 1$$

. We will take $\gamma_1 = 0$ and $\gamma_2 = 0$. This would be equivalent to no coupling at all.

The matrices F_i and M_i are obtained by solving LMIs using YALMIP. Thus, from the matrices deduced from the T-S formalization of the nonlinear model of the coupled tanks system,F_i and M_i are given in C.

We Assume that the initial condition $x(0) = [12.7; 12.4; 1.5; 1.5]$, and we simulate the same standard test defined before 2.5.2. μ is tuned manually such that we have $u(t) \leq 10$. Thus, after some tests we put $\mu = 27$
The corresponding Simulation results are the following :

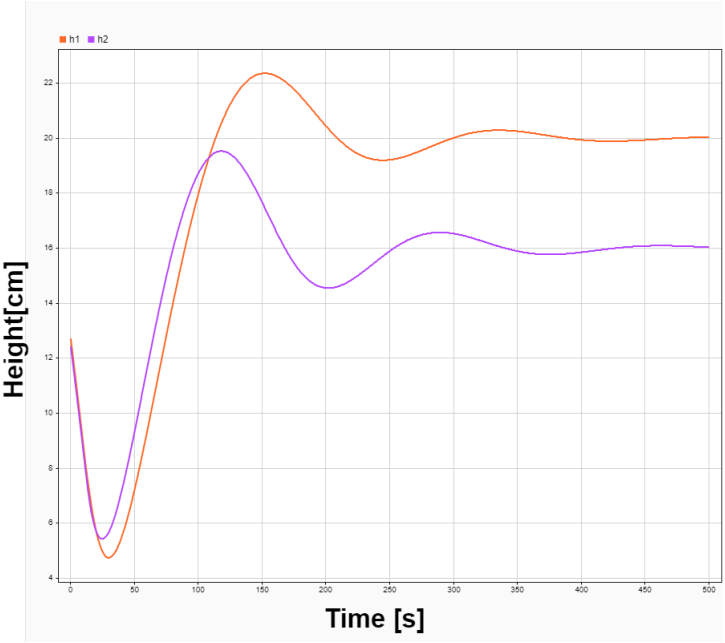


Figure 4.23: Output System Response h_1 (orange) and h_2 (purple) with the mentioned conditions

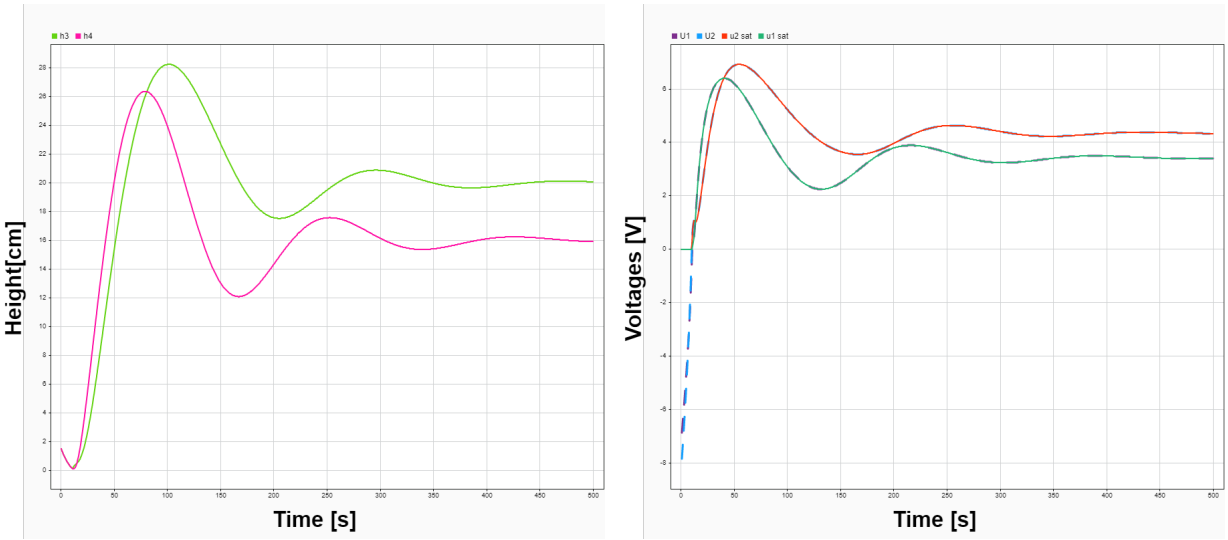


Figure 4.24: Corresponding State Response of h_3 (green) and h_4 (rose) (left) and the Control Signals (right)

Observations

The minimum phase configuration achieves stability relatively quickly with fewer residual oscillations post the initial overshoot. Both outputs h_1 and h_2 stabilize around their respective set points.The oscillations in the minimum phase configuration are less frequent and dissipate faster. The two states h_3 and h_4 will get to an eventual steady stable state for both cases.

Compared to our benchmark, the system alongside a 4 rule based TS PDC controller has a response time of approximately 38 seconds. This value is already better than the one based on 16 rules yet it's still smaller than the PI where the value was around 4 seconds for the minimum phase setting. However, for the nonminmum phase setting, the response time will be around 260 seconds which is better than the 330 seconds made by the benchmark.

Some oscillations appears also in the response and of course a trough that is a characteristic of a nonminimal phase setting, yet, it's still appearing in the minimum phase setting. We could explain this physically by the lower bound and saturation made on the control signal that shouldn't be a negative value at all during the working phase.

Comparison between TS Fuzzy Controllers with 16 Fuzzy Rules and 4 Fuzzy Rules in the Minimum Phase Configuration

When comparing the two figures, which represent the output responses $h1$ and $h2$ in the minimum phase configuration for two different TS fuzzy controllers (one with 16 rules and the other with 4 rules), we observe that under the same initial conditions and control constraints (u), the controller with 4 fuzzy rules achieves trajectory tracking and convergence to the reference value more rapidly. In contrast, the controller with 16 fuzzy rules takes about twice to thrice as long, with a response time of approximately 105 seconds.

From the illustrations, we can also see that The 16-rule controller shows more pronounced sinusoidal behavior compared to the 4-rule controller . The overshoot values are nearly identical for both controllers, with both reaching approximately 22 cm for $h1$ and 18 cm for $h2$, given reference signals of 20 cm and 16 cm,

The major similarity between the two stays in the fact that the designed control value always starts with a negative value that is replaced then with a zero as a lower bound (or a lower saturation on our system). This will lead physically to a response that's looking like a non minimal phase behavior for a minimal phase setup.

4.7 Robustness tests and Validation of the TS Fuzzy controller with 16 rules

4.7.1 Robustness with respect to High viscosity fluids

High viscosity fluids have a different dynamics than lower ones like water. Meanwhile, if we look at our model we can clearly see that there isn't a single parameter that represents the characteristics of the used fluid in the QTP.

In this part, we take care to recall the discharge coefficient C_d already used previously to verify the robustness tests [16], a parameter that will model the ratio between the actual flow and the theoretical flow i.e:

$$C_d = \frac{Q_{actual}}{Q_{Th}} \tag{4.58}$$

We will choose adequate values of C_d according to the literature [16] and we will see its effects on the step response of our system . The chosen values were $1, 0.8, 0.6, 0.4, 0.2$ and the distinction is clear seeing the representing figures.

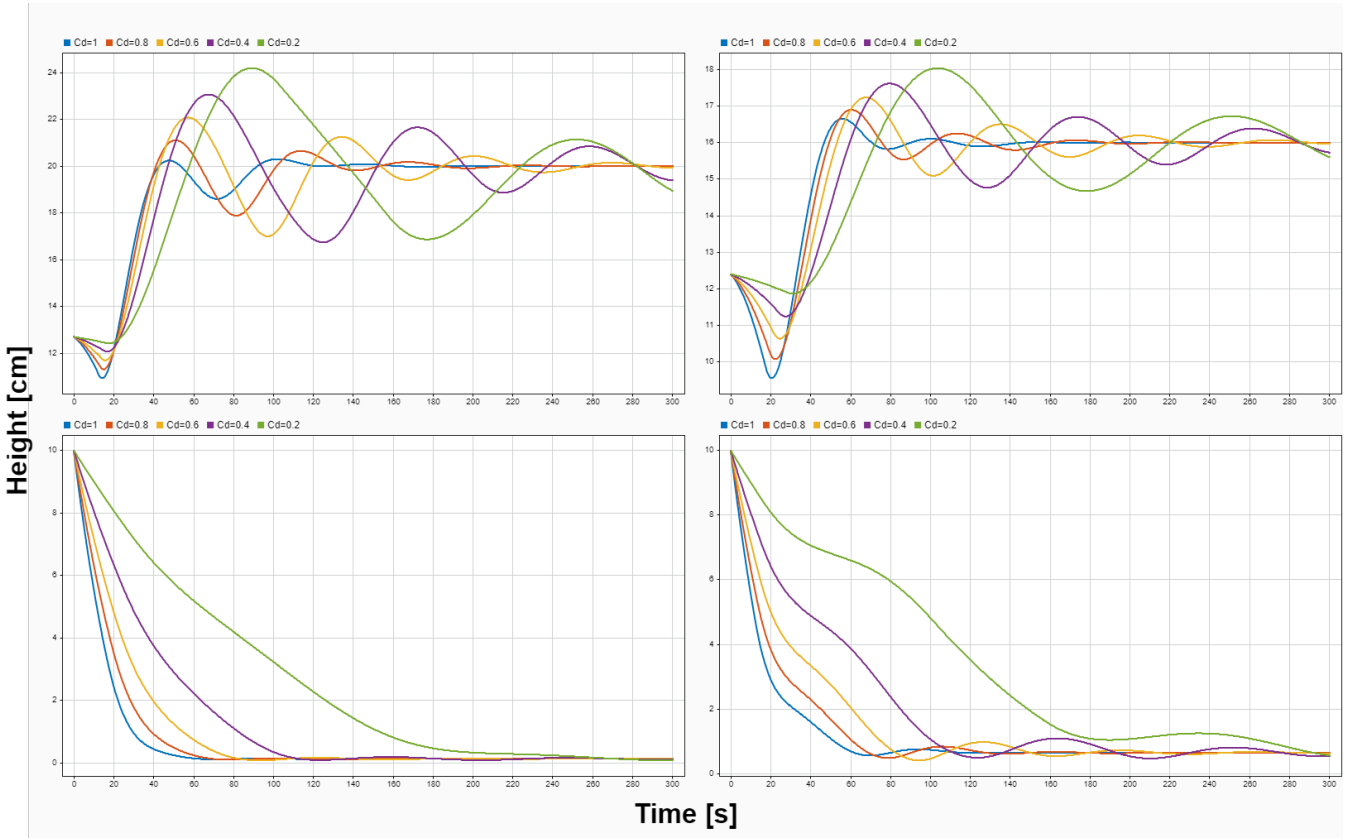


Figure 4.25: Evolution of the System's states under different values of C_d for the minimum phase setting for the 16 rules controller, Upper left h_1, Upper right h_2, Lower left h_3, Lower right h_4

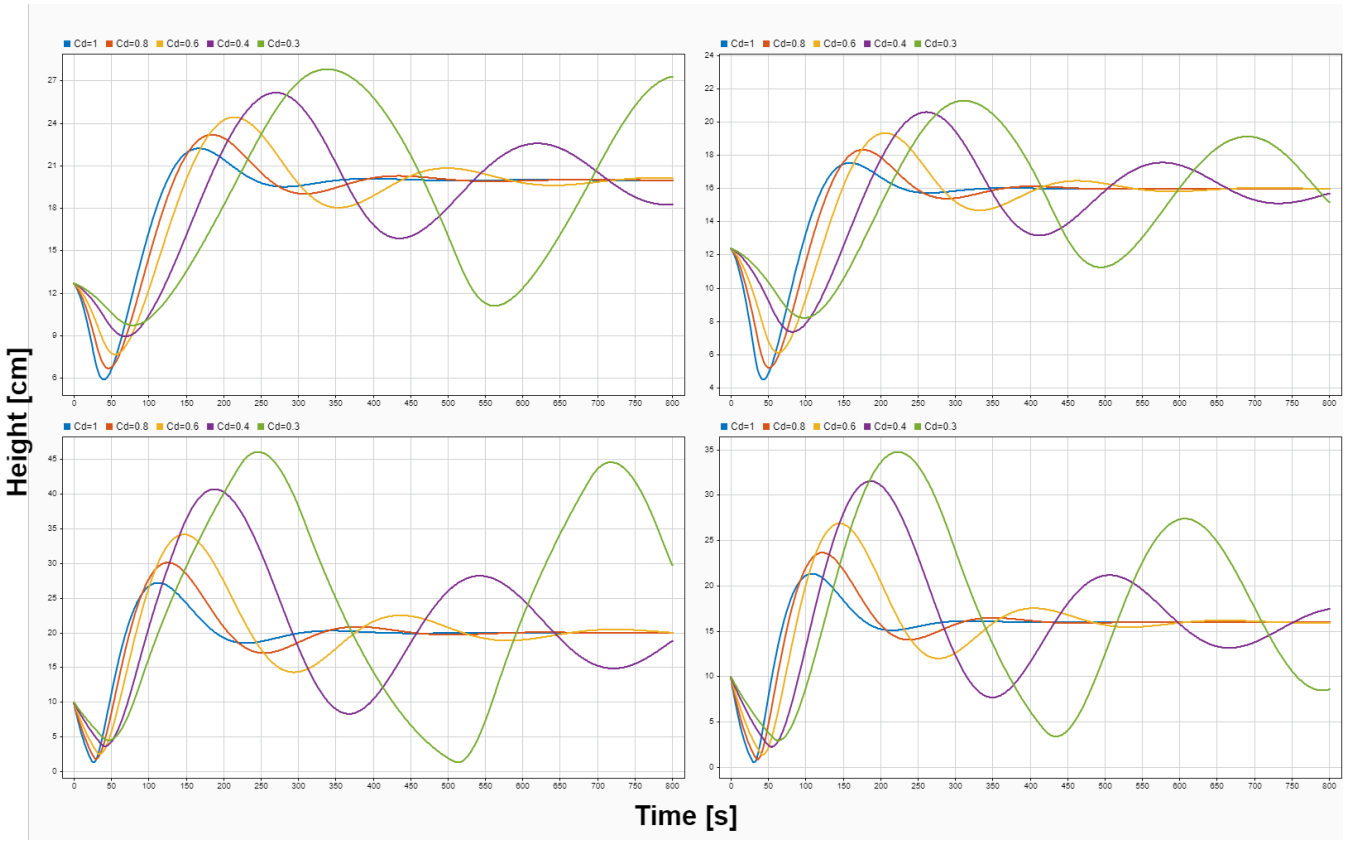


Figure 4.26: Evolution of the System's states under different values of C_d for the non-minimum phase setting for the 16 rules controller, Upper left h_1, Upper right h_2, Lower left h_3, Lower right h_4

Observations:
Initially, the system responses exhibit oscillations before stabilizing. For $C_d = 1$, the system reaches its peak response faster and with fewer oscillations compared to lower C_d values. As C_d decreases towards 0.2, the oscillations become more pronounced and the time to reach steady-state increases. Higher C_d values result in more damped responses and quicker settling times. However, as C_d decreases, the system experiences increased oscillations and slower responses, highlighting the significant impact of C_d on stability and transient behavior. Controlling the non-minimum phase system is inherently more challenging due to the initial opposite direction of response, and the impact of C_d is more severe in this case.

In addition to the latency, a pseudo-sinusoidal behavior could also be seen. The overshoot increases every time we reduce the value of C_d . It is well logically following the physical behavior of higher viscosity fluids as the gravitational force takes a lot of time to change the value of the desired level.

Compared to the benchmark performance, we can see that the 16 rules based TS PDC controller is still much slower in the minimum phase setting. It does also have a more oscillatory behavior. The trough (nonminimal phase behavior is also observed during the beginning of the minimal phase setting and it's probably the major difference between the two of them.

On the other hand, in the non-minimal phase setting and particularly for the value $C_d = 0.2$ The system stays stable for the closed loop system behavior but this wasn't the case for the performance of the decentralize PI controller.

4.7.2 Robustness with respect to the interconnections

For this test, we will only consider the controller in the non-minimum phase setting. We designed that controller early on whilst considering $\gamma_1 = \gamma_2 = 0$. These values give us a setting where we will not consider the interconnections towards the two lower tanks. Making these two fill up completely using gravitational force and liquid descending from the two upper tanks. But how about we induce the designed controller with an interconnection i.e. $\gamma_1 = \gamma_2 = \gamma \neq 0$ and see their impact on our controller. This is shown through figure 3.27.

The first thing we notice is that the interconnections induces a sinusoidal behavior over the system's state response. In fact, it does increase the step response time at 95% of the final value and the response is completely periodic for values $\gamma \geq 0.25$. The increased time is added up to the already long response and we get a convergence time of over 1000[sec] for $\gamma = 0.3$.

In addition, we can see that Each time gamma increases, the oscillations of the responses increase as well as the response time. the final value for the two upper tanks (h_3 and h_4) is reduced each time we increase the ratio for the valve, This is due to the fact that the 02 lower tanks and our control objectives are being divided between what's been pulled down by the gravitation and what's been brought by the lower nozzles.

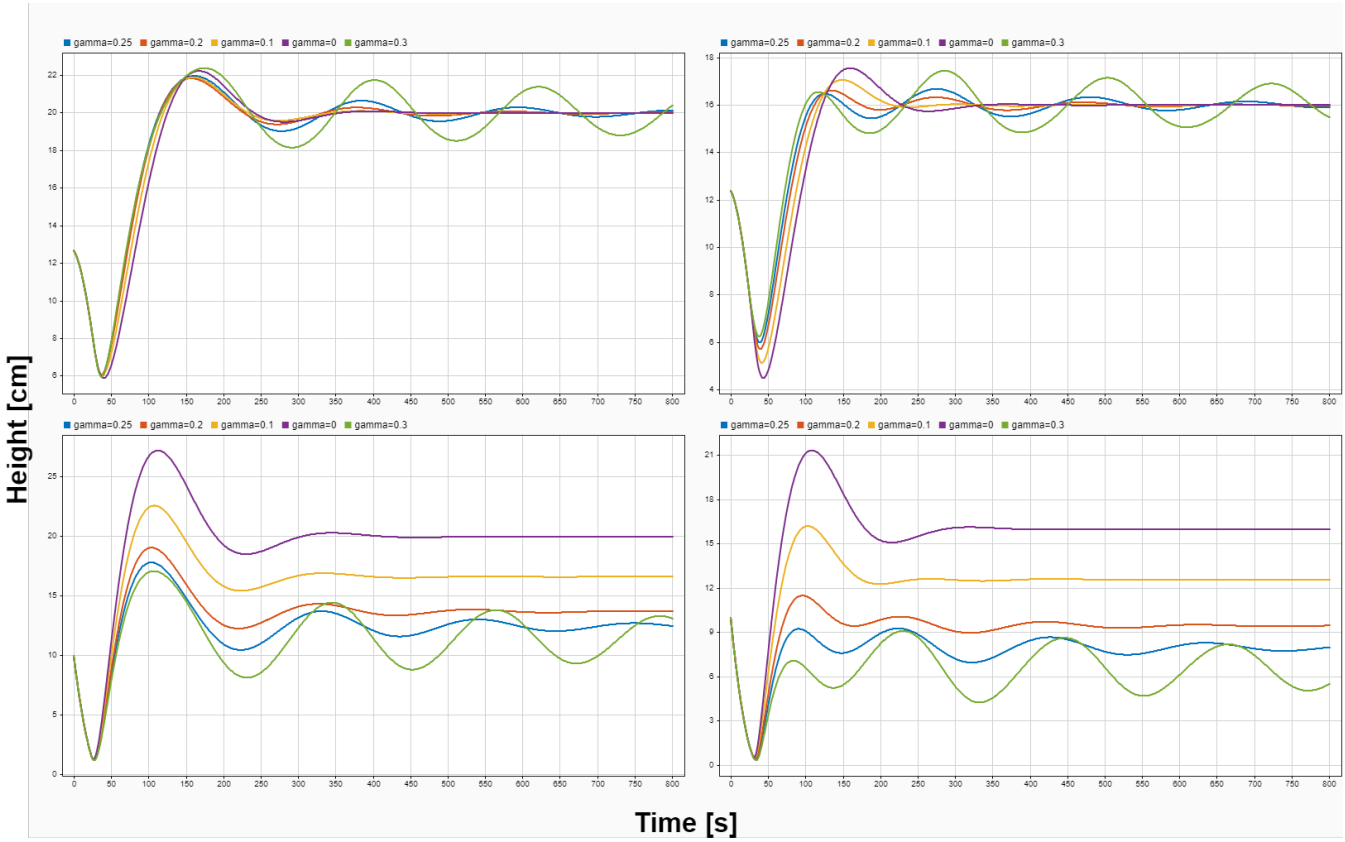


Figure 4.27: Evolution of the System's states under different values of γ for the non-minimum phase setting for the 16 rules controller under a robustness of interconnections, Upper left h_1, Upper right h_2, Lower left h_3, Lower right h_4

Comparing this to the performance of the decentralized PI controller, we can see that the Fuzzy PDC controller based on 16 rules is way more robust. First, the response still converges till for even a value of $\gamma = 0.3$ but for a longer response time. This wasn't the case for the performance made by the PI, where the behavior was completely oscillatory for a value of $\gamma = 0.2$

4.8 Robustness tests and Validation of the TS Fuzzy controller with 4 rules

4.8.1 Robustness with respect to High viscosity fluids

Again, we take care to recall the discharge coefficient C_d already used previously to verify the robustness tests [16], a parameter that will model the ratio between the actual flow and the theoretical flow i.e:

$$C_d = \frac{Q_{actual}}{Q_{Th}} \tag{4.59}$$

We will choose adequate values of C_d according to the literature [16] and we will see its effects on the step response of our system . The chosen values were $1, 0.8, 0.6, 0.4, 0.2$ and the distinction is clear seeing the representing figures.

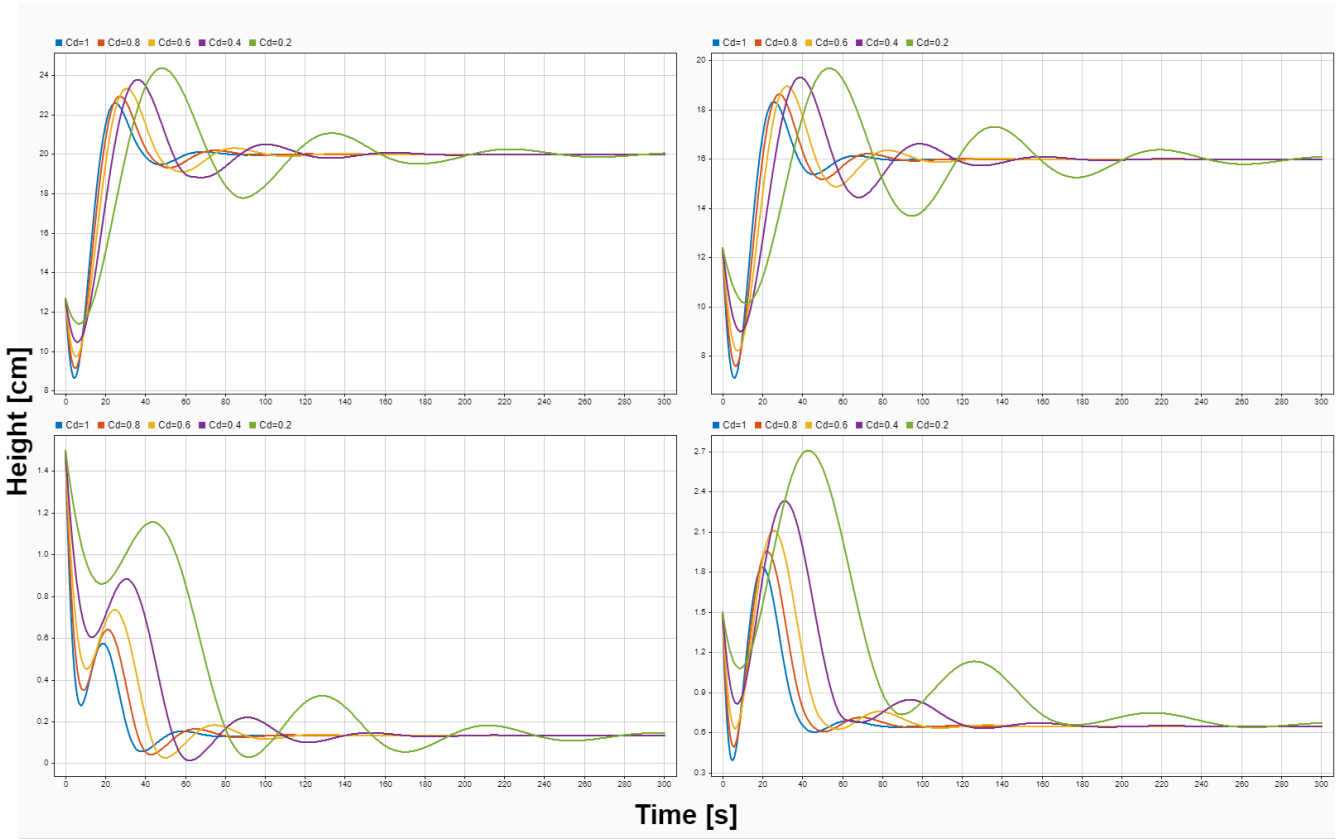


Figure 4.28: Evolution of the System's states under different values of C_d for the minimum phase setting for the 4 rules controller, Upper left h_1, Upper right h_2, Lower left h_3, Lower right h_4

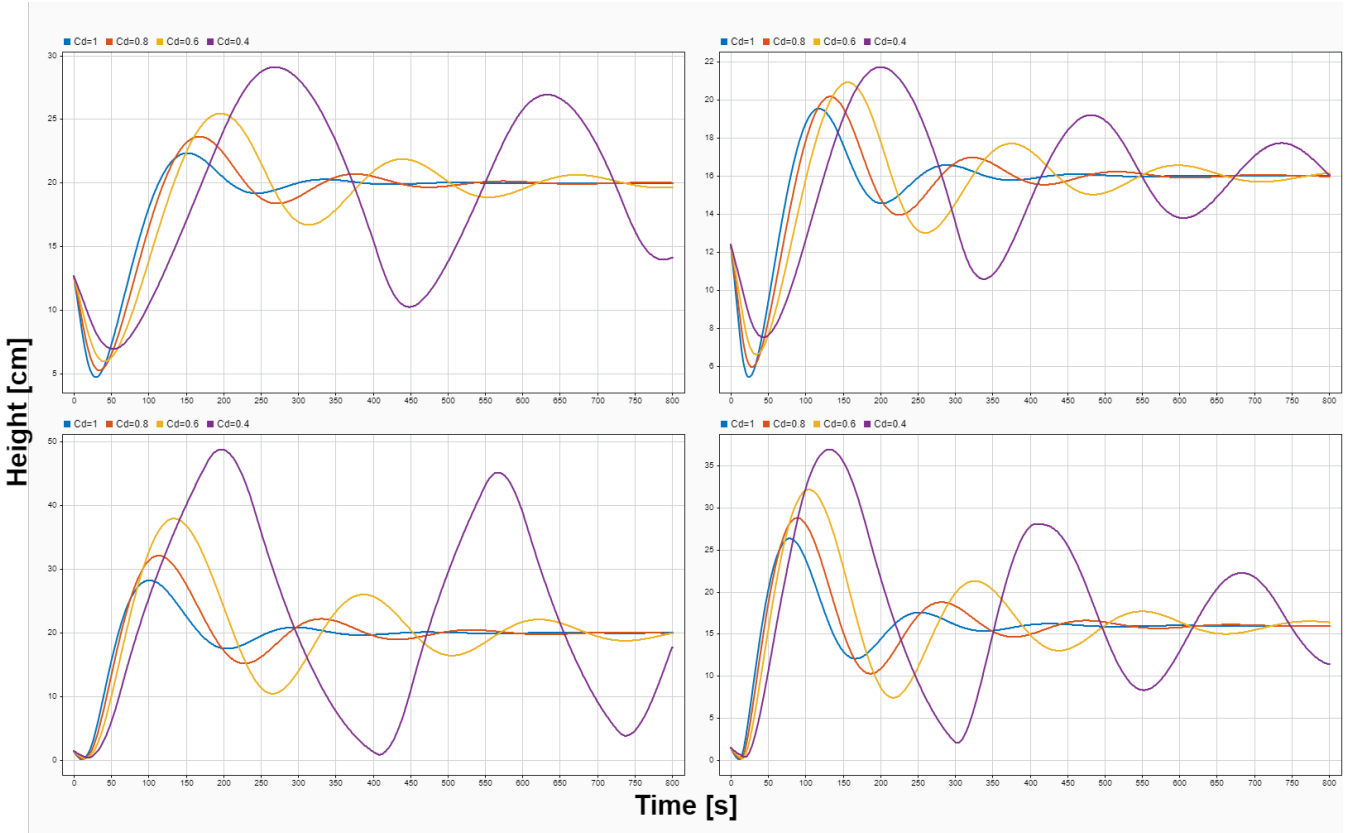


Figure 4.29: Evolution of the System's states under different values of C_d for the non-minimum phase setting for the 4 rules controller, Upper left h_1, Upper right h_2, Lower left h_3, Lower right h_4

As always, a latency is observed in the responses as much as we decrease the value of C_d as well as an pseudo-oscillatory behavior for both the minimal and the nonminimal phase setting.

Compared to the benchmark and the 16 rules based PDC, the PDC based on 4 rules has a better time response as much as we decrease the value of C_d For the minimal phase setting. On the other hand for the nonminimal phase setting, the response is still faster yet it is not robust for a value of $C_d = 0.2$ Thus, our reason for not representing it in the illustration.

4.8.2 Robustness with respect to the interconnections

For this test, we will only consider the controller in the non-minimum phase setting. We designed that controller early on whilst considering $\gamma_1 = \gamma_2 = 0$. These values give us a setting where we will not consider the interconnections towards the two lower tanks.

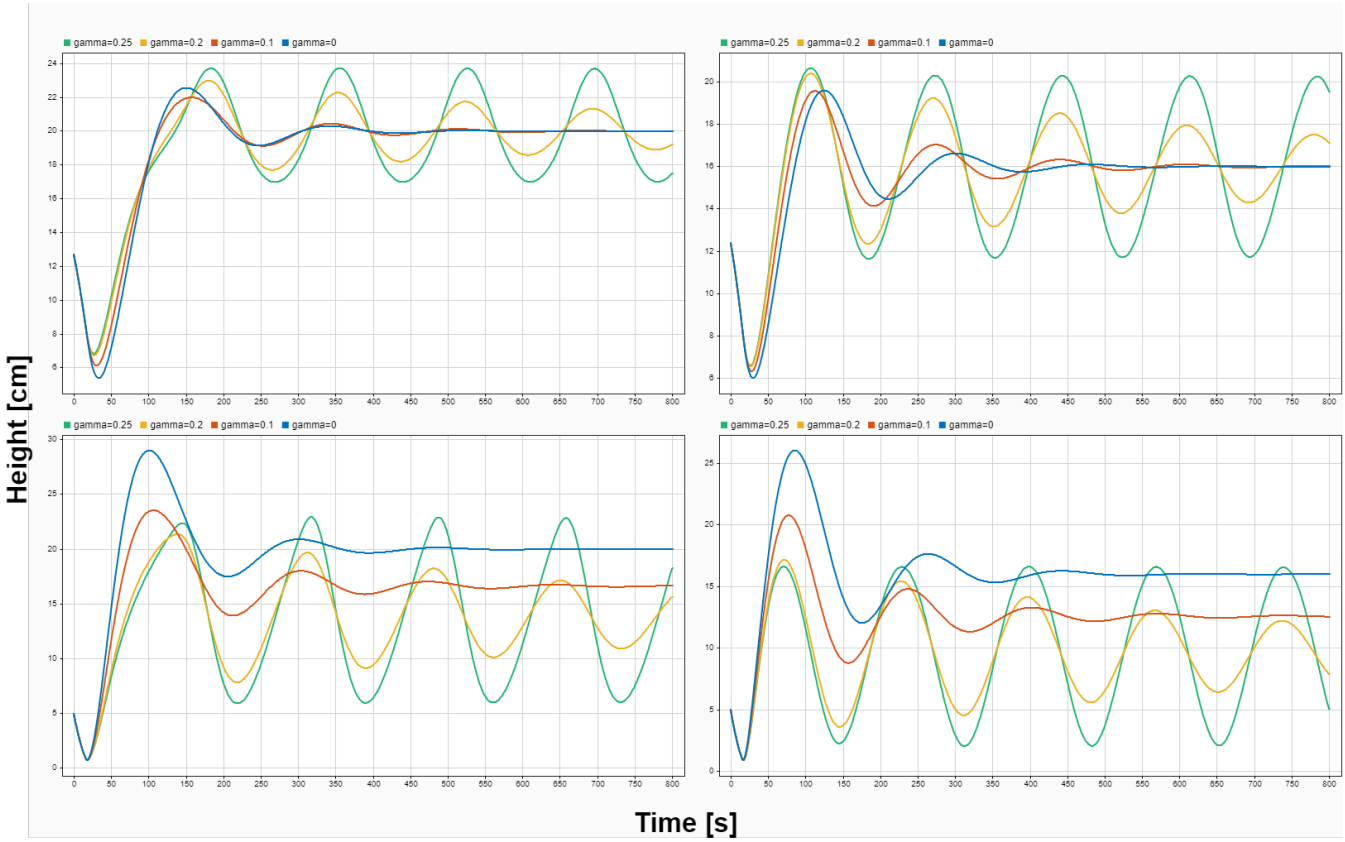


Figure 4.30: Evolution of the System's states under different values of γ for the non-minimum phase setting under a robustness of interconnections, Upper left h_1, Upper right h_2, Lower left h_3, Lower right h_4

the best performance in terms of response time and minimal overshoot is achieved with $\gamma = 0$. It is evident that as the value of the coefficient γ increases, both the overshoot and the response time increase The first thing we notice is that the interconnections induces a sinusoidal behavior over the system's state response.

Compared to before, the TS PDC controller based on 4 rules could be considered more robust than the PI because the intensity of the sinusoidal behavior is much smaller than the performances obtained by

our benchmark. It is definitely more robust in that sense yet it isn't as robust as the 16 rules based TS PDC controller.

4.9 How can we fix the trough happening at the launch of our system?

As we could've seen before, our system has an inherent negative value on the control signal when we launch it from a given set point directly. This will make us choose a saturation for the lower bound that must be set to zero. This will have no effects on the pumps as it corresponds to the absence of the Voltage in the input. Yet, it will lead the water level in the tanks to decrease due to the gravitational force. This, will imply us to see a behavior similar to the nonminimal phase while the system is in its minimum phase setting.

To prove this, we will do a test where we will give two step signals as references. The 2nd one being delayed with respect to the first one to a point where the control input is already positive and stable. The results are illustrated directly in figure 3.31 and 3.32.

We can see clearly that for the second step signal that was ahead of time, the response didn't have the trough that's seen when we launch our system. This is due to the fact that the control input has already reached a positive steady value and it just needs to adjust to fit the new trajectory. Additionally, The major improvement that could be seen is in the time response of our adjusted system which is much smaller than before.

To suggest a proper solution for this problem, we could consider first and at each starting of the system to regulate it first till a given set point using the benchmark decentralized PI controller and when we are sure that the control signal generated by the Fuzzy PDC controller is above zero, then we switch the controllers mid-way alongside a given operational point.

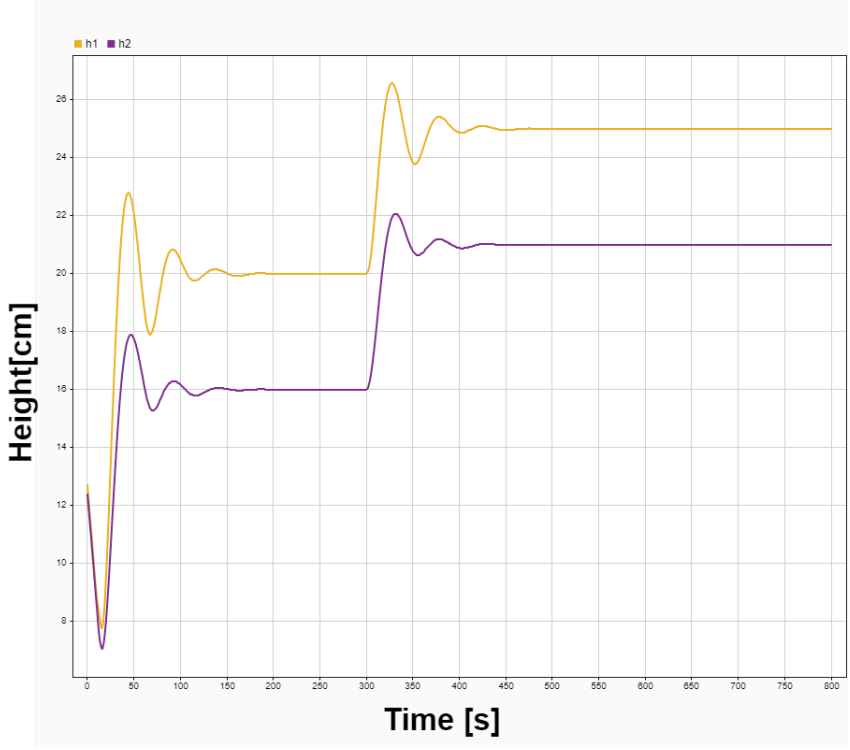


Figure 4.31: Output System Response with the mentioned conditions

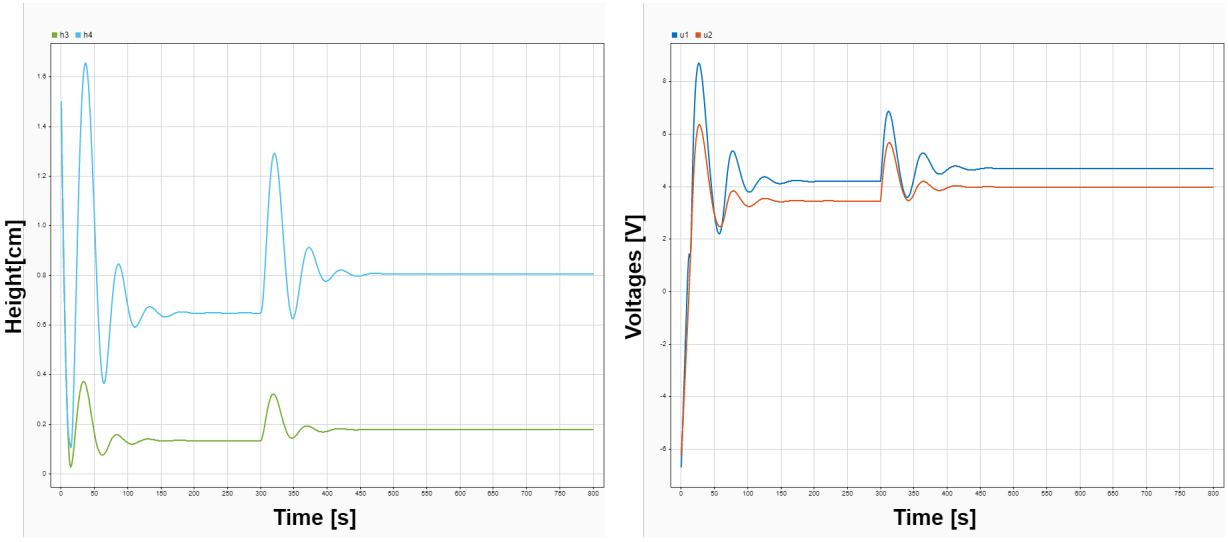


Figure 4.32: Corresponding State Response of h_3 and h_4 (left) and the Control Signals (right)

A better response could be obtained if we make a pole placement in LMI regions that will eventually reduce the overshoot in the system's outputs and Making it more smoother. Additionally, the trough could maybe solved directly with this technique where we would ensure that the initial value of u is positive.

4.10 Conclusion

In this chapter, we have detailed the method for modeling a nonlinear system using a multi-model approach (T-S model) and the sector nonlinearity approach. The problem of Stability and Stabilization have been dealt with a quadratic lyapunov function approach to design the PDC controller.

Following this, Two different TS models for the QTP have been presented: one with 16 rules and the other with 4 rules only. After that, we applied PDC stabilization and trajectory tracking for both minimum and Non-minimum phase settings based on the derived models. We then validated the controllers by applying two different robustness tests that deals mainly with parametric variations.

The tests have shown that the TS fuzzy controller with 4 rules achieved quicker stabilization and less oscillatory behavior compared to the 16-rule controller. In contrast, the 16-rule controller exhibited more oscillations and slower response times, especially under the minimum phase setting. Both setups showed initial overshoot, with the minimum phase configuration stabilizing faster and with fewer oscillations, while the nonminimum phase setting had a much slower response time. Additionally, High viscosity fluids resulted in increased oscillations and slower responses, with the effect being more severe in the non-minimum phase setting. The benchmark PI controller generally performed better in terms of response time and overshoot under minimum phase settings.

One of the main deductions that we saw was that the TS fuzzy controllers, particularly the one with 16 rules, demonstrated superior robustness in the presence of interconnections, outperforming the 4-rule controller and the benchmark PI controller in this regard. The 4-rule TS fuzzy controller offers quicker response times and less oscillatory behavior under standard conditions. On the other hand, the 16-rule controller provides better robustness under more challenging scenarios, making it a viable alternative depending on specific system requirements and operational conditions.

Chapter 5

Conclusion & Perspectives

The thesis began with an exploration of the benchmark equipment provided by Feedback instruments available at the LCP laboratory in ENP, specifically the Coupled Tanks process, followed by a detailed modeling of the plant and its characteristics.

We then dived into the implementation and evaluation of a Particle Swarm Optimization (PSO) algorithm for optimizing a decentralized PI controller as a benchmark control method for the four tanks coupled system, used to compare its performance with the performances of other controllers designed in the thesis. Robustness tests were conducted under varied conditions, including the introduction of high viscosity fluids and interconnections through variations of the discharge coefficient C_d and the valve opening ratio γ respectively.

Chapter 2 extensively covered Predictive Control techniques, including the implementation of Linear and Nonlinear Model Predictive Controllers. We designed these controllers and assessed their feasibility for real-time implementation using different libraries such as CasAdi. The designed controllers have been tested based on the defined robustness tests. Additionally, we proposed an approach using Recurrent Neural Networks to predict control inputs to avoid the online optimization problem given in the classical MPC approach.

Chapter 3 detailed the modelling of nonlinear systems using a multi-model approach, specifically the T-S model. We derived two different fuzzy models using the nonlinear sector method for the Quadruple Tank Process: one based on 16 rules and the other based on 4 rules. These models were then used to apply a PDC stabilization and trajectory tracking for both minimum and Non-minimum phase settings. The constraint on the input was incorporated in the control law. The resulting controllers have been approved through the robustness tests, showing a good efficiency.

In summary, our thesis has contributed to the advancement of control techniques for the classical Quadruple Tank Process exploring mainly classical, Predictive and Fuzzy control design. This being said, some perspectives could be explored for future project:

- First, the implementation and testing all the designed algorithms must be done on the real plant. Unfortunately, the material needed a PCI to PCI adapter to link between the laboratory PC and the Advantech PCI1711 Interface card and it hasn't been provided during our stay in the lab. The real time implementation will give additional insight about the performances of the designed algorithms.
- Explore other techniques to implement MPCs, notably the ones that are based on C++ as they could be more efficient and faster than classical libraries.
- On the other hand, we could enhance the performance obtained by the Fuzzy TS PDC controllers by performing a pole placement in LMI regions.
- The work is based on a benchmark equipment that hasn't been used during the control en-

gineering course. We could eventually consider using this thesis as a base to create a set of practical work that must be done by the students to enhance their experiences with real-life control problems.

Appendix A

PSO Algorithm

Meta-heuristic algorithms are advanced optimization techniques that seek near-optimal solutions using sophisticated heuristic methods. These algorithms employ stochastic search and exploration strategies to intelligently navigate possible solutions, often outperforming traditional optimization algorithms and iterative methods. The increasing computational power and advancements in hardware interfaces have made meta-heuristics widely popular for solving optimization problems in recent years.

Meta-heuristics are categorized into several types:

- **Search Strategy:** Some algorithms focus on enhancing candidate solutions within a local neighborhood, while others explore broader search spaces to find solutions.
- **Nature of the Solution:** Certain algorithms work with a single solution, continually improving it, while others use a particle-based approach to simultaneously refine multiple candidates.
- **Inspiration and Origin:** Many effective algorithms draw inspiration from nature, mimicking the behaviors of species or natural phenomena that seem to solve problems optimally.

In 1995, Kennedy and Eberhart introduced the Particle Swarm Optimization (PSO) algorithm, a particle-based meta-heuristic inspired by swarm intelligence and the social interactions seen in bird flocking or fish schooling.

PSO can be used independently to find global solutions or in conjunction with other algorithms. It is based on the principles of social behavior:

- **Intrinsic Experience:** Each particle's actions are influenced by its own previous results.
- **Social Influence:** The actions of each particle are also affected by the results of its neighbors within the swarm.

A.1 Algorithm structure

In PSO, candidate solutions are represented as particles, denoted by i, where $i \in \{1, \ldots, n\}$. At a given time step T, each particle has the following attributes:

- **Position** $x_i(T)$**:** This is a vector in $\mathbb{X} \in \mathbb{R}^m$, representing the solution space for each particle. It constitutes a single solution to the problem, with m variables.
- **Velocity** $v_i(T)$**:** This vector, of the same dimension as x_i, describes the "movement" of the particle.

- **Personal Best** $P_i(T)$**:** This is the best solution $x_i(t)$, for $t \in \{1, \ldots, T\}$, according to a given cost function for each particle.

- **Global Best** $G(T)$**:** This is the best solution among all particles up to time T.

Given these values, the mathematical model of PSO for each particle's variable x_{ij}, where $j \in \{1, \ldots, m\}$, is defined as follows:

$$\begin{cases} v_{ij}(t+1) = w \cdot v_{ij}(t) + c_1 \cdot r_1 \cdot (P_{ij}(t) - x_{ij}(t)) + c_2 \cdot r_2 \cdot (G_j(t) - x_{ij}(t)), \\ x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \end{cases} \tag{A.1}$$

where r_1 and r_2 are random numbers between 0 and 1, and w, c_1, and c_2 are real numbers.

The coefficients w, c_1, and c_2 are dynamic; as the algorithm progresses, c_2 increases and w decreases. This adjustment aims to give more weight to the global best term, promoting convergence towards it and reducing exploration of other regions.

Having saying all of this, here is the detailed algorithm:

Algorithm 1 Particle Swarm Optimization (PSO)

for each particle $i = 1, \ldots, S$ **do**
  Initialize the particle's position with a uniformly distributed random vector: $x_i \sim U(b_{lo}, b_{up})$
  Initialize the particle's best known position to its initial position: $p_i \leftarrow x_i$
if $f(p_i) < f(g)$ **then**
    Update the swarm's best known position: $g \leftarrow p_i$
end if
  Initialize the particle's velocity: $v_i \sim U(-|b_{up} - b_{lo}|, |b_{up} - b_{lo}|)$

while a termination criterion is not met **do**
  for each particle $i = 1, \ldots, S$ **do**
    for each dimension $d = 1, \ldots, n$ **do**
      Pick random numbers: $r_p, r_g \sim U(0, 1)$
      Update the particle's velocity: $v_{i,d} \leftarrow wv_{i,d} + \phi_p r_p(p_{i,d} - x_{i,d}) + \phi_g r_g(g_d - x_{i,d})$
    end for
    Update the particle's position: $x_i \leftarrow x_i + v_i$
    if $f(x_i) < f(p_i)$ **then**
      Update the particle's best known position: $p_i \leftarrow x_i$
      if $f(p_i) < f(g)$ **then**
        Update the swarm's best known position: $g \leftarrow p_i$
      end if
    end if
  end for
end while

Appendix B

Controllers Results

In the following table, we give a resume about all the designed controllers during this thesis.

Controller	**Phase**	**Response Time (sec)**	**Pseudo Oscillations**	**Remarks**
PI	Min	14	No	Zero/small overshoot, U1 is saturated for around 1 second, robust to high viscosity fluids
	Nonmin	330	Yes	Overshoot, robust till a value of $C_d = 0.4$, not robust enough to interconnections
Fuzzy PDC (16 rules)	Min	105	Yes	A trough at the start, robust to high viscosity fluids and highly robust against interconnections
	Nonmin	225	Small	Big nonmin phase behavior, robust till a value of $C_d = 0.4$, robust to interconnections till a value of ($\gamma = 0.3$)
Fuzzy PDC (4 rules)	Min	38	small oscillations	10% overshoot, robust to high viscosity fluids
	Nonmin	260	Yes	Huge trough initially, overshoot, robust till $C_d = 0.4$, better interconnections robustness than PI ($\gamma = 0.25$)
LMPC	Min	13	Small pseudo	5% overshoot, saturated control initially, robust to high viscosity fluids

Controller	Phase	Response Time (sec)	Oscillations	Remarks
	Nonmin	100	Small	5% overshoot, saturated control initially, robust to high viscosity fluids, robust to $\gamma = 0.15$ interconnections
NMPC	Min	8	No	Very small overshoot, saturated control for set time, robust to high viscosity fluids
	Nonmin	90	Pseudo (5%)	Saturated control for set time, robust to high viscosity fluids, not robust to $\gamma = 0.1$ interconnections (ongoing oscillations)

Appendix C

Linear Matrix Inequalities (LMIs)

C.1 Linear Matrix Inequalities (LMIs)

Linear Matrix Inequalities are the control version of the semidefinite programs (SDP) that are convex problems, allowing the resolution of a great number of problems in relation to uncertain systems. A powerful and efficient polynomial-time interior-point algorithm was developed for linear programming by Karmakar in 1984 [38], then extended in 1988 by Nesterov and Nemirovskii, who developed interior-point methods that apply directly to linear matrix inequalities [48]. It was then recognized that LMIs can be solved with convex optimization on a computer, and in 1995 Gahinet and Nemirovskii [21] wrote a commercial Matlab package called the LMI Toolbox for Matlab. The advantage of SDP is the polynomial time of global minimum computation using the interior point methods [48].

C.1.1 Definition of a Linear Matrix Inequality

A linear matrix inequality is a matrix inequality of the form:

$$F(x) = F_0 + \sum_{i=1}^{m} x_i F_i > 0 \tag{C.1}$$

where $x(t) = [x_1(t), \ldots, x_m(t)]^T$ is the variable vector to find and $F_i = F_i^T \in \mathbb{R}^{n \times n}, i = 0, \ldots, m$ are given matrices. The inequality implies that $F(x)$ must be positive definite, i.e., all its eigenvalues are positive. The LMI is a convex constraint on x, i.e., the set $\{x \mid F(x) > 0\}$ is convex. It can also gather several convex constraints $F_1(x) > 0, F_2(x) > 0, \ldots, F_m(x) > 0$ in a block diagonal matrix:

$$\begin{bmatrix} F_1(x) & 0 & \cdots & 0 \\ 0 & F_2(x) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & F_m(x) \end{bmatrix} > 0$$

C.1.2 Some Standard LMI Problems

Among the most encountered convex optimization LMI problems, we cite:

C.1.2.1 LMI Problems

Given an LMI $F(x) > 0$, the LMI problem is to find x_{feas} such that $F(x_{\text{feas}}) > 0$ or determine that the LMI is infeasible. This is a convex feasibility problem that can be solved by convex optimization algorithms such as interior-point methods. For example, the Lyapunov stability conditions given in section 3.4 can be expressed as an LMI problem where P is the variable [10], and this is available for all the stability conditions encountered in this work.

C.1.2.2 Eigenvalue Problem

The eigenvalue problem (EVP) is to minimize the maximum eigenvalue of a matrix that depends affinely on a variable, subject to an LMI constraint (or determine that the constraint is infeasible), in other terms:

$$\min \lambda \quad \text{subject to} \quad \lambda I - A(x) > 0, \quad B(x) > 0 \tag{C.2}$$

C.1.2.3 Generalized Eigenvalue Problem

The generalized eigenvalue problem (GEVP) is to minimize the maximum eigenvalue of a pair of matrices that depend affinely on a variable, subject to an LMI constraint. The general form of GEVP is:

$$\min \lambda \quad \text{subject to} \quad \lambda B(x) - A(x) > 0, \quad B(x) > 0, \quad C(x) > 0 \tag{C.3}$$

All these problems can be solved by different tools such as ellipsoid algorithms, simplex methods, and interior-point methods. However, there exist some tools that facilitate the passage from a non-convex formulation to an LMI, that is convex, among them:

C.1.2.4 Schur Complement

Nonlinear (convex) inequalities are converted to LMI form using Schur complements. For the following LMI:

$$\begin{bmatrix} Q(x) & S(x) \\ S(x)^T & R(x) \end{bmatrix} > 0$$

where $Q(x) = Q(x)^T$, $R(x) = R(x)^T$, and $S(x)$ depend affinely on x, is equivalent to:

$$R(x) > 0, \quad Q(x) - S(x)R(x)^{-1}S(x)^T > 0 \tag{C.4}$$

The lemma is also valid when changing the sign of the inequalities.

C.1.2.5 Polytopic Form

A polytopic form is defined as follows: A set of matrices $\{A_1, A_2, \ldots, A_n\}$ is said to be polytopic if there exists a set of positive parameters such that [61]:

$$\forall\, 0 \leq \lambda_i \leq 1, \quad \sum_{i=1}^{n} \lambda_i = 1, \quad A = \sum_{i=1}^{n} \lambda_i A_i > 0$$

Hence the matrices form a polytopic $\Lambda = \text{Co}\{A_1, A_2, \ldots, A_n\}$, where Co denotes the convex hull. The notion of convexity plays an important role since the stability analysis problems are represented in terms of convex optimization problems, which allows a reasonable computing time and finding a global minimum.

C.2 Resolution of LMIs with YALMIP Interface

YALMIP is a modeling language compatible with MATLAB syntax designed to model and solve optimization problems, invented by Dr. Johan Löfberg in 2004 [43]. Its main motivation is that it implements a large number of modeling tricks, allowing the user to focus on the high-level model. YALMIP also implements external algorithms to solve optimization problems (LMIs). In this thesis, all LMIs have been solved using the YALMIP interface with the SeDuMi method.

C.2.1 YALMIP Overview

YALMIP was initially developed to model SDPs and solve these by interfacing external solvers. The toolbox makes the development of optimization problems in general, and control-oriented SDP problems in particular, extremely simple.

Rapid prototyping of an algorithm based on SDP can be done in a matter of minutes using standard MATLAB commands. In fact, learning three YALMIP-specific commands will be enough for most users to model and solve their optimization problem.Due to a flexible solver interface and internal format, adding new solvers, and even new problem classes, can often be done with modest effort.

YALMIP automatically detects what kind of problem the user has defined and selects a suitable solver based on this analysis. If no suitable solver is available, YALMIP tries to convert the problem to be able to solve it. As an example, if the user defines second-order cone constraints, but no second-order cone programming solver is available, YALMIP converts the constraints to LMIs and solves the problem using any installed SDP solver.

C.2.2 Using YALMIP to Solve LMIs: An Example

The system

$$\dot{x} = Ax$$

is stable (eigenvalues have negative real part) if and only if there exists a $P > 0$ such that

$$A^T P + PA < 0$$

YALMIP Code for Stability Analysis:

```
A = [-1 2 0; -3 -4 1; 0 0 -2];
P = sdpvar(3,3);
F = [P >= eye(3)];
F = [F, A'*P+P*A <= 0];
optimize(F);
```

If Feasible, YALMIP Code to Retrieve the Solution:

```
Pfeasible = value(P);
```

C.3 Gains obtained in PDC Stabilisation for the TS fuzzy system with 16 rules

The matrices F_i and X are obtained by solving LMIs using YALMIP. Thus, from the matrices deduced from the T-S formalization of the nonlinear model of the coupled tanks system,F_i and X are given by:

X matrix:

$$\begin{bmatrix} 42.5222 & -5.4684 & -7.4939 & -1.0240 \\ -5.4684 & 41.9596 & -0.6015 & -7.5939 \\ -7.4939 & -0.6015 & 29.1960 & 2.4573 \\ -1.0240 & -7.5939 & 2.4573 & 28.6882 \end{bmatrix}$$

F matrix for rule 1:

$$\begin{bmatrix} 0.1638 & 0.1626 & 0.1878 & 0.1868 \\ 0.1574 & 0.1628 & 0.1746 & 0.1921 \end{bmatrix}$$

F matrix for rule 2:

$$\begin{bmatrix} 0.1646 & 0.1695 & 0.1841 & 0.2160 \\ 0.1574 & 0.1645 & 0.1728 & 0.2034 \end{bmatrix}$$

F matrix for rule 3:

$$\begin{bmatrix} 0.1655 & 0.1624 & 0.2001 & 0.1848 \\ 0.1612 & 0.1631 & 0.1917 & 0.1899 \end{bmatrix}$$

F matrix for rule 4:

$$\begin{bmatrix} 0.1649 & 0.1688 & 0.1907 & 0.2131 \\ 0.1606 & 0.1640 & 0.1877 & 0.1987 \end{bmatrix}$$

F matrix for rule 5:

$$\begin{bmatrix} 0.1661 & 0.1777 & 0.1871 & 0.2012 \\ 0.1694 & 0.2319 & 0.1717 & 0.2664 \end{bmatrix}$$

F matrix for rule 6:

$$\begin{bmatrix} 0.1646 & 0.1709 & 0.1844 & 0.2129 \\ 0.1622 & 0.1959 & 0.1726 & 0.2220 \end{bmatrix}$$

F matrix for rule 7:

$$\begin{bmatrix} 0.1669 & 0.1763 & 0.1960 & 0.1985 \\ 0.1660 & 0.2329 & 0.1566 & 0.2698 \end{bmatrix}$$

F matrix for rule 8:

$$\begin{bmatrix} 0.1649 & 0.1701 & 0.1911 & 0.2099 \\ 0.1622 & 0.1964 & 0.1723 & 0.2230 \end{bmatrix}$$

F matrix for rule 9:

$$\begin{bmatrix} 0.2345 & 0.1742 & 0.2638 & 0.1844 \\ 0.1661 & 0.1640 & 0.1827 & 0.1918 \end{bmatrix}$$

F matrix for rule 10:

$$\begin{bmatrix} 0.2368 & 0.1706 & 0.2697 & 0.1663 \\ 0.1645 & 0.1652 & 0.1797 & 0.2015 \end{bmatrix}$$

F matrix for rule 11:

$$\begin{bmatrix} 0.1976 & 0.1671 & 0.2188 & 0.1848 \\ 0.1618 & 0.1630 & 0.1890 & 0.1901 \end{bmatrix}$$

F matrix for rule 12:

$$\begin{bmatrix} 0.1978 & 0.1695 & 0.2173 & 0.1948 \\ 0.1607 & 0.1641 & 0.1848 & 0.1993 \end{bmatrix}$$

F matrix for rule 13:

$$\begin{bmatrix} 0.2397 & 0.1007 & 0.2900 & 0.1004 \\ 0.0981 & 0.2410 & 0.0896 & 0.2961 \end{bmatrix}$$

F matrix for rule 14:

$$\begin{bmatrix} 0.2402 & 0.1335 & 0.2870 & 0.1095 \\ 0.1352 & 0.2010 & 0.1401 & 0.2455 \end{bmatrix}$$

F matrix for rule 15:

$$\begin{bmatrix} 0.2011 & 0.1434 & 0.2358 & 0.1565 \\ 0.1290 & 0.2403 & 0.1031 & 0.2915 \end{bmatrix}$$

F matrix for rule 16:

$$\begin{bmatrix} 0.2000 & 0.1576 & 0.2276 & 0.1761 \\ 0.1495 & 0.1994 & 0.1528 & 0.2353 \end{bmatrix}$$

C.4 Gains obtained in PDC Stabilisation for the TS fuzzy system with 4 rules

$$P = \begin{pmatrix} 0.0078 & -0.0010 & -0.0008 & -0.0005 \\ -0.0010 & 0.0077 & -0.0005 & -0.0007 \\ -0.0008 & -0.0005 & 0.0137 & -0.0007 \\ -0.0005 & -0.0007 & -0.0007 & 0.0136 \end{pmatrix}$$

$$F_1 = \begin{bmatrix} 0.2552 & -0.0242 & -0.0086 & 0.0082 \\ -0.0205 & 0.2646 & -0.0057 & -0.0130 \end{bmatrix}$$

$$F_2 = \begin{bmatrix} 0.2211 & -0.0253 & 0.0272 & -0.0402 \\ -0.0180 & 0.2310 & 0.0038 & -0.0150 \end{bmatrix}$$

$$F_3 = \begin{bmatrix} 0.2558 & -0.0204 & -0.0222 & 0.0130 \\ -0.0193 & 0.2665 & -0.0305 & -0.0089 \end{bmatrix}$$

$$F_4 = \begin{bmatrix} 0.1295 & -0.0232 & 0.0025 & -0.0375 \\ -0.0201 & 0.1802 & -0.0301 & -0.0040 \end{bmatrix}$$

C.5 Gains obtained in Trajectory tracking for the TS fuzzy system with 16 rules

The matrices F_i and M_i are obtained by solving LMIs using YALMIP. Thus, from the matrices deduced from the T-S formalization of the nonlinear model of the coupled tanks system,F_i and M_i are given by:

C.5.1 Minimum Phase settings

$$F1 = \begin{bmatrix} 0.3127 & 0.0367 & 0.3101 & 0.0364 \\ 0.0085 & 0.3048 & 0.0025 & 0.3092 \end{bmatrix}$$

$$M1 = \begin{bmatrix} 0.0894 & 0.0062 \\ 0.0016 & 0.0878 \end{bmatrix}$$

$$F2 = \begin{bmatrix} 0.3054 & 0.0507 & 0.2988 & 0.1472 \\ -0.0058 & 0.2965 & -0.0157 & 0.4064 \end{bmatrix}$$

$$M2 = \begin{bmatrix} 0.0867 & 0.0050 \\ -0.0042 & 0.0843 \end{bmatrix}$$

$$F3 = \begin{bmatrix} 0.3015 & 0.0276 & 0.3977 & 0.0231 \\ 0.0168 & 0.3041 & 0.0449 & 0.3070 \end{bmatrix}$$

$$M3 = \begin{bmatrix} 0.0864 & 0.0022 \\ 0.0026 & 0.0875 \end{bmatrix}$$

$$F4 = \begin{bmatrix} 0.2988 & 0.0404 & 0.4084 & 0.0748 \\ -0.0049 & 0.2971 & -0.0518 & 0.4463 \end{bmatrix}$$

$$F4 = \begin{bmatrix} 0.0853 & 0.0049 \\ -0.0007 & 0.0826 \end{bmatrix}$$

$$F5 = \begin{bmatrix} 0.3080 & 0.1195 & 0.3028 & 0.0908 \\ -0.0354 & 0.6247 & -0.0501 & 0.4587 \end{bmatrix}$$

$$M5 = \begin{bmatrix} 0.0888 & 0.0104 \\ -0.0110 & 0.0657 \end{bmatrix}$$

$$F6 = \begin{bmatrix} 0.3057 & 0.0871 & 0.2995 & 0.1441 \\ -0.0035 & 0.5237 & -0.0128 & 0.2955 \end{bmatrix}$$

$$M6 = \begin{bmatrix} 0.0868 & 0.0052 \\ -0.0017 & 0.0828 \end{bmatrix}$$

$$F7 = \begin{bmatrix} 0.2980 & 0.1207 & 0.3884 & 0.0872 \\ -0.0418 & 0.6245 & -0.0943 & 0.4610 \end{bmatrix}$$

$$M7 = \begin{bmatrix} 0.0860 & 0.0097 \\ -0.0093 & 0.0659 \end{bmatrix}$$

$$F8 = \begin{bmatrix} 0.2964 & 0.0782 & 0.3897 & 0.1091 \\ -0.0059 & 0.5266 & -0.0387 & 0.3038 \end{bmatrix}$$

$$M8 = \begin{bmatrix} 0.0851 & 0.0047 \\ -0.0001 & 0.0829 \end{bmatrix}$$

$$F9 = \begin{bmatrix} 0.6263 & -0.0111 & 0.4544 & -0.0272 \\ 0.0610 & 0.3041 & 0.0402 & 0.3072 \end{bmatrix}$$

$$M9 = \begin{bmatrix} 0.0673 & -0.0050 \\ 0.0075 & 0.0879 \end{bmatrix}$$

$$F10 = \begin{bmatrix} 0.6258 & -0.0174 & 0.4567 & -0.0865 \\ 0.0657 & 0.2999 & 0.0387 & 0.4338 \end{bmatrix}$$

$$M10 = \begin{bmatrix} 0.0674 & -0.0031 \\ 0.0072 & 0.0833 \end{bmatrix}$$

$$F11 = \begin{bmatrix} 0.5261 & 0.0137 & 0.2936 & 0.0056 \\ 0.0399 & 0.3038 & 0.0608 & 0.3062 \end{bmatrix}$$

$$M11 = \begin{bmatrix} 0.0843 & 0.0014 \\ 0.0039 & 0.0873 \end{bmatrix}$$

$$F12 = \begin{bmatrix} 0.5282 & 0.0146 & 0.2973 & -0.0127 \\ 0.0292 & 0.2986 & 0.0333 & 0.4300 \end{bmatrix}$$

$$M12 = \begin{bmatrix} 0.0844 & 0.0033 \\ 0.0025 & 0.0831 \end{bmatrix}$$

$$F13 = \begin{bmatrix} 0.6280 & 0.0355 & 0.4579 & 0.0082 \\ -0.0152 & 0.6330 & -0.0318 & 0.4697 \end{bmatrix}$$

$$M13 = \begin{bmatrix} 0.0691 & 0.0009 \\ -0.0031 & 0.0674 \end{bmatrix}$$

$$F14 = \begin{bmatrix} 0.6291 & -0.0156 & 0.4584 & -0.0527 \\ 0.0364 & 0.5305 & 0.0166 & 0.3073 \end{bmatrix}$$

$$M14 = \begin{bmatrix} 0.0679 & -0.0009 \\ 0.0040 & 0.0828 \end{bmatrix}$$

$$F15 = \begin{bmatrix} 0.5257 & 0.0838 & 0.2946 & 0.0537 \\ -0.0494 & 0.6314 & -0.0498 & 0.4658 \end{bmatrix}$$

$$M15 = \begin{bmatrix} 0.0842 & 0.0072 \\ -0.0059 & 0.0663 \end{bmatrix}$$

$$F16 = \begin{bmatrix} 0.5279 & 0.0341 & 0.2930 & 0.0054 \\ 0.0069 & 0.5287 & -0.0117 & 0.3021 \end{bmatrix}$$

$$M16 = \begin{bmatrix} 0.0844 & 0.0041 \\ 0.0017 & 0.0829 \end{bmatrix}$$

C.5.2 Non-Minimum Phase settings

$$F1 = \begin{bmatrix} -0.0457 & 0.2001 & -0.0359 & 0.2223 \\ 0.1833 & -0.0341 & 0.2108 & -0.0260 \end{bmatrix}$$

$$M1 = \begin{bmatrix} -0.0020 & 0.0359 \\ 0.0339 & -0.0011 \end{bmatrix}$$

$$F2 = \begin{bmatrix} -0.0338 & 0.2591 & -0.0246 & 0.4198 \\ 0.1687 & -0.0104 & 0.1961 & 0.0106 \end{bmatrix}$$

$$M2 = \begin{bmatrix} -0.0019 & 0.0338 \\ 0.0335 & -0.0004 \end{bmatrix}$$

$$F3 = \begin{bmatrix} -0.0144 & 0.1728 & 0.0007 & 0.2104 \\ 0.2558 & -0.0107 & 0.4408 & -0.0095 \end{bmatrix}$$

$$M3 = \begin{bmatrix} -0.0009 & 0.0355 \\ 0.0318 & -0.0013 \end{bmatrix}$$

$$F4 = \begin{bmatrix} -0.0252 & 0.2477 & -0.0144 & 0.4204 \\ 0.2568 & -0.0090 & 0.4449 & -0.0023 \end{bmatrix}$$

$$M4 = \begin{bmatrix} -0.0014 & 0.0337 \\ 0.0319 & -0.0009 \end{bmatrix}$$

$$F5 = \begin{bmatrix} 0.0111 & 0.5674 & 0.0129 & 0.4186 \\ 0.1736 & -0.0188 & 0.1991 & -0.0111 \end{bmatrix}$$

$$M5 = \begin{bmatrix} 0.0001 & 0.0118 \\ 0.0336 & -0.0024 \end{bmatrix}$$

$$F6 = \begin{bmatrix} -0.0281 & 0.5043 & -0.0167 & 0.4726 \\ 0.1660 & 0.0003 & 0.1943 & 0.0033 \end{bmatrix}$$

$$M6 = \begin{bmatrix} -0.0013 & 0.0293 \\ 0.0335 & -0.0008 \end{bmatrix}$$

$$F7 = \begin{bmatrix} -0.0083 & 0.5716 & -0.0050 & 0.4160 \\ 0.2580 & -0.0073 & 0.4404 & -0.0044 \end{bmatrix}$$

$$M7 = \begin{bmatrix} -0.0004 & 0.0119 \\ 0.0318 & -0.0018 \end{bmatrix}$$

$$F8 = \begin{bmatrix} -0.0202 & 0.5102 & -0.0020 & 0.4751 \\ 0.2575 & 0.0029 & 0.4391 & -0.0026 \end{bmatrix}$$

$$F9 = \begin{bmatrix} -0.0065 & 0.1810 & -0.0040 & 0.2048 \\ 0.5878 & 0.0042 & 0.4345 & 0.0085 \end{bmatrix}$$

$$F10 = \begin{bmatrix} -0.0052 & 0.2590 & -0.0013 & 0.4214 \\ 0.5921 & -0.0135 & 0.4317 & -0.0090 \end{bmatrix}$$

$$F11 = \begin{bmatrix} -0.0085 & 0.1742 & -0.0025 & 0.1994 \\ 0.5028 & -0.0251 & 0.4901 & -0.0149 \end{bmatrix}$$

$$F12 = \begin{bmatrix} 0.0051 & 0.2556 & -0.0048 & 0.4191 \\ 0.5071 & -0.0155 & 0.4914 & 0.0018 \end{bmatrix}$$

$$F13 = \begin{bmatrix} 0.0078 & 0.5751 & -0.0097 & 0.4066 \\ 0.5965 & 0.0005 & 0.4212 & -0.0146 \end{bmatrix}$$

$$F14 = \begin{bmatrix} -0.0041 & 0.5074 & 0.0002 & 0.4757 \\ 0.5898 & -0.0011 & 0.4321 & -0.0025 \end{bmatrix}$$

$$F15 = \begin{bmatrix} -0.0022 & 0.5704 & -0.0044 & 0.4156 \\ 0.5013 & -0.0039 & 0.4907 & -0.0019 \end{bmatrix}$$

$$F16 = \begin{bmatrix} 0.0082 & 0.5172 & -0.0012 & 0.4737 \\ 0.5086 & 0.0120 & 0.4895 & 0.0026 \end{bmatrix}$$

$$M8 = \begin{bmatrix} -0.0010 & 0.0296 \\ 0.0318 & -0.0013 \end{bmatrix}$$

$$M9 = \begin{bmatrix} -0.0016 & 0.0355 \\ 0.0107 & -0.0003 \end{bmatrix}$$

$$M10 = \begin{bmatrix} -0.0015 & 0.0339 \\ 0.0109 & -0.0006 \end{bmatrix}$$

$$M11 = \begin{bmatrix} -0.0008 & 0.0353 \\ 0.0273 & -0.0012 \end{bmatrix}$$

$$M12 = \begin{bmatrix} -0.0014 & 0.0339 \\ 0.0276 & -0.0009 \end{bmatrix}$$

$$M13 = \begin{bmatrix} -0.0003 & 0.0118 \\ 0.0107 & -0.0008 \end{bmatrix}$$

$$M14 = \begin{bmatrix} -0.0011 & 0.0295 \\ 0.0107 & -0.0007 \end{bmatrix}$$

$$M15 = \begin{bmatrix} -0.0007 & 0.0118 \\ 0.0275 & -0.0014 \end{bmatrix}$$

$$M16 = \begin{bmatrix} -0.0012 & 0.0295 \\ 0.0275 & -0.0011 \end{bmatrix}$$

C.6 gains obtained in Trajectory tracking for the TS fuzzy system with 4 rules

The matrices F_i and M_i are obtained by solving LMIs using YALMIP. Thus, from the matrices deduced from the T-S formalization of the nonlinear model of the coupled tanks system,F_i and M_i are given by:

C.6.1 Minimum Phase settings

$$F1 = \begin{bmatrix} 1.0766 & -0.2951 & 0.0334 & -0.0110 \\ -0.2876 & 1.1093 & -0.0139 & 0.0260 \end{bmatrix} \qquad M1 = \begin{bmatrix} 0.2221 & -0.0328 \\ -0.0318 & 0.2203 \end{bmatrix}$$

$$F2 = \begin{bmatrix} 1.0244 & -0.2904 & 0.1297 & -0.0146 \\ -0.2893 & 1.0722 & -0.0202 & 0.0367 \end{bmatrix} \qquad M2 = \begin{bmatrix} 0.2199 & -0.0312 \\ -0.0323 & 0.2219 \end{bmatrix}$$

$$F3 = \begin{bmatrix} 1.0736 & -0.2962 & 0.0270 & -0.0110 \\ -0.2872 & 1.1084 & -0.0111 & 0.0281 \end{bmatrix} \qquad M3 = \begin{bmatrix} 0.2225 & -0.0328 \\ -0.0318 & 0.2203 \end{bmatrix}$$

$$F4 = \begin{bmatrix} 0.9666 & -0.3131 & 0.0824 & -0.0225 \\ -0.2683 & 1.0249 & -0.0251 & 0.0585 \end{bmatrix} \qquad M4 = \begin{bmatrix} 0.2276 & -0.0358 \\ -0.0312 & 0.2235 \end{bmatrix}$$

C.6.2 Non-Minimum Phase settings

$$F1 = \begin{bmatrix} -0.0629 & 0.4586 & -0.0673 & 0.8378 \\ 0.8045 & -0.1310 & 0.3469 & -0.0373 \end{bmatrix} \qquad M1 = \begin{bmatrix} -0.0056 & 0.0302 \\ 0.0696 & -0.0091 \end{bmatrix}$$

$$F2 = \begin{bmatrix} 0.0098 & 0.9072 & -0.0137 & 0.1123 \\ 0.8014 & -0.1567 & 0.3533 & -0.1808 \end{bmatrix} \qquad M2 = \begin{bmatrix} 0.0011 & 0.0496 \\ 0.0707 & -0.0154 \end{bmatrix}$$

$$F3 = \begin{bmatrix} -0.0799 & 0.6427 & -0.0741 & 1.0342 \\ 0.4211 & -0.0309 & 0.3723 & 0.0159 \end{bmatrix} \qquad M3 = \begin{bmatrix} -0.0153 & 0.0298 \\ 0.0805 & 0.0024 \end{bmatrix}$$

$$F4 = \begin{bmatrix} -0.1128 & 0.2778 & -0.1029 & 0.3495 \\ 0.4220 & -0.1073 & 0.3836 & -0.1082 \end{bmatrix} \qquad M4 = \begin{bmatrix} -0.0180 & 0.0505 \\ 0.0816 & -0.0030 \end{bmatrix}$$

Bibliography

[1] *Coupled Tanks Control Experiments*. Park Road, Crowborough, East Sussex, TN6 2QR, UK. For use with MATLAB R2007a or later.

[2] Ibtissem Abdelmalek. *Non Linear Systems Control: LMI Fuzzy Approach*. PhD thesis, Université Hadj Lakhdar de Batna, Faculté des Sciences de l'ingénieur, Département d'Electronique, 2009.

[3] Nurhuda Zainal Abidin, Shafishuhaza Sahlan, and Norhaliza Abdul Wahab. Optimization tuning of pi controller of quadruple tank process. In *2013 Australian Control Conference*, pages 331–335, 2013.

[4] Sevil Ahmed, Ivan Ganchev, Albena Taneva, and Michail Petrov. Decoupling neuro-fuzzy model predictive controllers applied to quadruple tanks. In *2016 IEEE 8th International Conference on Intelligent Systems (IS)*, pages 610–615, 2016.

[5] R.J. Albin Raj and S.N. Deepa. Modeling and implementation of various controllers used for quadruple-tank. In *2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, pages 1–5, 2016.

[6] Moncef Allel and Islam Sellah. Application des techniques de commande par logique floue pour la régulation de la distance inter-véhicules en circulation. Mémoire de projet de fin d'études, École Nationale Polytechnique (ENP), Alger, Algérie, juillet 2021. Pour l'obtention du diplôme d'ingénieur d'état en Automatique.

[7] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019.

[8] Robert Babuska. Fuzzy systems, modeling and identification. *Electr. Eng.*, 01 2001.

[9] H. W. Bode. *Network Analysis and Feedback Amplifier Design*. Van Nostrand, New York, 1945.

[10] Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*. Society for Industrial and Applied Mathematics, 1994.

[11] Elmer Calle and José Oliden. Recurrent neural network based predictive control applied to 4 coupled-tank system. In *2021 IEEE International Conference on Automation/XXIV Congress of the Chilean Association of Automatic Control (ICA-ACCA)*, pages 1–6, 2021.

[12] E.F. Camacho and C.B. Alba. *Model Predictive Control*. Advanced Textbooks in Control and Signal Processing. Springer London, 2013.

[13] Guanrong Chen and Trung Tat Pham. *Introduction to Fuzzy Sets, Fuzzy Logic, and Fuzzy Control Systems*. CRC Press LLC, Texas, 2001.

[14] Tim Chen, D. Kuo, and C. Chen. Fuzzy c-means robust algorithm for nonlinear systems. *Soft Computing*, 25, 04 2021.

[15] T. Deepa, P. Lakshmi, and S. Vidya. Level control of quadruple tank process using discrete time model predictive control. In *2011 3rd International Conference on Electronics Computer Technology*, volume 1, pages 162–166, 2011.

[16] S. Essien, A. Archibong-Eso, and L. Lao. Discharge coefficient of high viscosity liquids through nozzles. *Experimental and Thermal Fluid Science*, 2023. Oil and Gas Engineering Centre, School of Water, Energy & Environment, Cranfield University, UK; Department of Agricultural and Food Engineering, University of Uyo, Nigeria; Department of Mechanical Engineering, Cross River University of Technology, Calabar, Nigeria.

[17] Feedback Instruments Ltd. *Coupled Tanks. Installation and Commissioning*, 2013.

[18] Weijiang Feng, Naiyang Guan, Yuan Li, Xiang Zhang, and Zhigang Luo. Audio visual speech recognition with multimodal recurrent neural networks. pages 681–688, 05 2017.

[19] Frank Ferrese, Qing Dong, Saroj Biswas, and Jason Batcho. Decentralized control of coupled nonlinear dynamic systems with application to quadruple-tank process. In *IECON 2014 - 40th Annual Conference of the IEEE Industrial Electronics Society*, pages 3657–3661, 2014.

[20] Abderraouf Gaaloul and Faouzi M'Sahli. High gain output feedback control of a quadruple tank process. In *MELECON 2008 - The 14th IEEE Mediterranean Electrotechnical Conference*, pages 23–28, 2008.

[21] Pascal Gahinet, Arkadii Nemirovskii, Alan Laub, and Mohammad Chilali. *LMI Control Toolbox*. The MathWorks Inc., 1995.

[22] A. Gattami and R. Murray. A frequency domain condition for stability of interconnected mimo systems. In *Proceedings of the 2004 American Control Conference*, volume 4, pages 3723–3728 vol.4, 2004.

[23] Mohammad Sajjad Ghasemi, Ali A Afzalian, and M H Ramezani. Stabilized hybrid model predictive control for the quadruple-tank process. In *2015 23rd Iranian Conference on Electrical Engineering*, pages 829–833, 2015.

[24] Aref Ghoreishee, Mohammad Shahrokhi, and Mohammaderfan Mohit. Fuzzy observer-based control of mimo interconnected systems subject to state delay, input nonlinearities, quantized input and output and sensor and actuator faults. *European Journal of Control*, 77:100964, 2024.

[25] Brais González García. Modelado y automatización de una planta piloto de nivel para la mejora en el diseño de controladores. Master's thesis, Universitat Politècnica de Catalunya · Barcelona Tech - UPC, 2018.

[26] Houssemeddine Gouta, Salim Hadj Said, and Faouzi M'Sahli. Observer-based backstepping liquid level controller for a quadruple tank process. In *2015 16th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, pages 351–356, 2015.

[27] Yu Haifen and Tan Wen. Partially decentralized control for a quadruple tank process. In *2013 25th Chinese Control and Decision Conference (CCDC)*, pages 2294–2299, 2013.

[28] Katalin M. Hangos, József Bokor, and Gábor Szederkényi. *Analysis and Control of Nonlinear Process Systems*. Advanced Textbooks in Control and Signal Processing. Springer, 2004.

[29] Yukio Prof. (auth.) Hasegawa and Prof. Shimon Y. (eds.) Nof. *Large-Scale Complex Systems*, pages 619–638. Springer-Verlag Berlin Heidelberg, 2009.

[30] Mária Hypiusová and Danica Rosinová. Robust control of quadruple-tank process via lmi. In *2016 Cybernetics Informatics (KI)*, pages 1–6, 2016.

[31] Abdelmalek Ibtissem and Noureddine Golea. A non-quadratic fuzzy stabilization and tracking approach to a two-link robot manipulator control. volume 3, pages 109 – 114, 11 2006.

[32] Carlos Ibáñez López. Implementation of optimization-based controllers for industrial processes. Master's thesis, Escola Tècnica Superior d'Enginyeria Industrial de Barcelona, Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Barcelona, Spain, 2017. Treball de Fi de Màster, Màster Universitari en Enginyeria Industrial, Director: Carlos Ocampo-Martínez, Convocatòria: Juny 2017.

[33] Damjan Ivetić, Miloš Milašinović, Milan Stojković, Aleksandar Šotić, Nicolas Charbonnier, and Nikola Milivojević. Framework for dynamic modelling of the dam and reservoir system reduced functionality in adverse operating conditions. *Water*, 14(10), 2022.

[34] Jayaprakash J. Comparison of controller performance for mimo process. *International Journal of Emerging Technology and Advanced Engineering*, 3:51, 08 2013.

[35] Jayaprakash J. State variable analysis of four tank system. 03 2014.

[36] Jayaprakash J, SenthilRajan T, and T Harish Babu. Analysis of modelling methods of quadruple tank system. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, 03:11552–11565, 08 2014.

[37] K.H. Johansson. The quadruple-tank process: a multivariable laboratory process with an adjustable zero. *IEEE Transactions on Control Systems Technology*, 8(3):456–465, 2000.

[38] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.

[39] P Kranthi Kumar and Ketan P Detroja. Design of reinforcement learning based pi controller for nonlinear multivariable system. In *2023 European Control Conference (ECC)*, pages 1–6, 2023.

[40] S. Larguech, S. Aloui, A. Chaari, A. El Hajjaji, and Y. Koubaa. Improved sliding mode control of a class of nonlinear systems: Application to quadruple tanks system. In *2013 European Control Conference (ECC)*, pages 3203–3208, 2013.

[41] Zhijun Li, Changbing Zheng, and Fumin Guo. H[infinity] loop shaping control for quadruple tank system. In *2014 Sixth International Conference on Intelligent Human-Machine Systems and Cybernetics*, volume 2, pages 117–120, 2014.

[42] G.P. Liu and S. Daley. Optimal-tuning nonlinear pid control of hydraulic systems. *Control Engineering Practice*, 8(9):1045–1053, 2000.

[43] Johan Löfberg. Yalmip : A toolbox for modeling and optimization in matlab. `http://users.isy.liu.se/johanl/yalmip/`, 2004. Available from http://users.isy.liu.se/johanl/yalmip/.

[44] Kamyar Mehran. *Takagi-Sugeno Fuzzy Modeling for Process Control*. PhD thesis, Newcastle University, 2008. Industrial Automation, Robotics and Artificial Intelligence (EEE8005).

[45] Chatti Venkata Nageswara Rao, M S N Murty, and Devendra Potnuru. Control of four tank system using grasshopper algorithm. In *2020 IEEE India Council International Subsections Conference (INDISCON)*, pages 200–203, 2020.

[46] Anca Nagy-Kiss. Analyse et synthèse de multimodèles pour le diagnostic. application à une station d'épuration. *PhD Institut National Polytechnique de Lorraine*, 11 2010.

[47] Yurii Nesterov. *Introductory Lectures on Convex Optimization*. Springer, 2004.

[48] Yurii Nesterov and Arkadii Nemirovskii. *Interior-point polynomial algorithms in convex programming*. Society for Industrial and Applied Mathematics, 1994.

[49] A. Numsomran, V. Tipsuwanporn, and K. Tirasesth. Modeling of the modified quadruple-tank process. In *2008 SICE Annual Conference*, pages 818–823, 2008.

[50] U. Sabura Banu and S.K. Lakshmanaprabu. Adaptive multi-loop fractional order pid controller tuning using bat colony optimization for quadruple tank process. In *2015 International Conference on Robotics, Automation, Control and Embedded Systems (RACE)*, pages 1–8, 2015.

[51] Tomohiro Takagi and Michio Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-15(1):116–132, 1985.

[52] Kazuo Tanaka and Hua Wang. *Fuzzy Control Systems Design and Analysis: A Linear Matrix Inequality Approach*. 01 2001.

[53] Amani Turki, Salim Hadj Said, and Faouzi M'Sahli. Backstepping control for a quadruple tank process based on adaptive observer. In *2015 IEEE 12th International Multi-Conference on Systems, Signals Devices (SSD15)*, pages 1–5, 2015.

[54] Fabian Ullmann. FiOrdOs: A Matlab Toolbox for C-Code Generation for First Order Methods. Master's thesis, ETH Zurich, 2011.

[55] Kemal Uçak and Gülay Öke. Modeling of quadruple tank system using support vector regression. In *2011 International Symposium on Innovations in Intelligent Systems and Applications*, pages 234–240, 2011.

[56] Qian Wang and Robert F. Stengel. Robust control of nonlinear systems with parametric uncertainty. *Automatica*, 38(9):1591–1599, 2002.

[57] Suk-Un Yoon, Andrew Markham, Niki Trigoni, Traian E. Abrudan, Orfeas Kypris, and Christian Wietfeld. Chapter 7 - advances and challenges in underground sensing. In Sibel Pamukcu and Liang Cheng, editors, *Underground Sensing*, pages 357–415. Academic Press, 2018.

[58] Miaomiao Yu, Shuchen Wu, and Xiaodi Li. Exponential stabilization of nonlinear systems under saturated control involving impulse correction. *Nonlinear Analysis: Hybrid Systems*, 48:101335, 2023.

[59] Chaima Zammali, Lahoucine Idkhajine, Sami Hlioui, Thach Ngoc Dinh, Rostaing Gilles, and H. Ben Ahmed. Pdc control strategy for a synchronous machine subject to magnetic saturations. pages 1–6, 10 2022.

[60] Wuji Zhang and Shaoyuan Li. Nonlinear model predictive control based on local linearized gaussian process and its application to a quadruple-tank system. In *2020 39th Chinese Control Conference (CCC)*, pages 2434–2439, 2020.

[61] Y. Zhao and N. Sadikovic. A simple necessary and sufficient condition for quadratic stability of time-varying polytopic systems. *IEEE Transactions on Automatic Control*, 40(9):1684–1687, 1995.

[62] Jorge A. Zolorza, Hicham El Aiss, Karina A. Barbosa, and Jonathan M. Palma. gain-scheduled controller design for multi coupled tank represented by takagi–sugeno fuzzy systems. In *2021 IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*, pages 1–6, 2021.

[63] Xuejun Zong, Zhongjun Yang, and Decheng Yuan. Nonlinear modeling and predictive control of the four-tank system. In *The 2nd International Conference on Software Engineering and Data Mining*, pages 268–271, 2010.