UNIVERSITÀ DEGLI STUDI DI PADOVA

UNIVERSITY OF PADOVA

DEPARTMENT OF ENGINEERING AND MANAGEMENT OF INDUSTRIAL SYSTEMS

UNIVERSITY OF AGDER

DEPARTMENT OF ENGINEERING SCIENCES

Robust design of a remote motion control system for hydraulic mechatronic drive with wireless operation

Author:
Riccardo Checchin

Supervisor:
Prof. Michael Ruderman
(University of Agder)

Co-Supervisor:
Prof. Roberto Oboe
(University of Padova)

A thesis submitted for the degree of

Master of Mechatronic Engineering

25 February 2022

Acknowledgements

This work had been a beautiful and challenging experience. I am grateful to all the many people that helped me in different ways to do this thesis, and if for conciseness I will not mention them all, I am sorry. First of all, I would like to thank my parents because they always support me emotionally and materially. In this case, also their economical help was essential. I am particularly thankful to Alice, who encourages me to go abroad for this project, even though managing the distance is not simple. I am grateful also to Benjamin. Fate made us do the thesis project in the same period and place. I had the pleasure to share with him the office and the laboratory. The discussions that we made together were great luck and particularly enriching. Talking together, we pointed out some of the considerations written in this report. He also helped me to establish TCP-IP communication. All the time spent together was a great gift that made him not only a colleague but rather a true friend.
Last but not least, I want to thank my supervisors: professor Michael Ruderman and professor Roberto Oboe. Both of them allowed me to work on this project in the beautiful country of Norway. In particular, I am grateful to professor Ruderman for his availability. He supervised me and the work step by step with professionalism.

Abstract

This Master thesis report presents the full design and practical implementation of a remote two degree of freedom PID control for a hydraulic actuator. The hydraulic cylinder is modelled as a linear system with a gain uncertainty that takes into account the change of the parameter used for the linearization and the variable time delay that affects the round time trip (RTT). The RTT is studied and modelled with a γ-distribution. The 2DOF PID controller is designed to verify the *Robust Stability Condition*, maximizing the integral action, thus the k_i parameter. Then, the optimization of load disturbance rejection with constraints on robustness to model uncertainties. The remote controller is implemented with a laptop that communicates with a TCP-IP protocol through an Ethernet or WIFI connection. The design is tested in the hydraulic system at UiA. Both the simulation and experimental results show the expected behaviour of the control system.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Nowadays, different actuators are in use in mechatronic applications. Although electric motors, linear drives, or pneumatic actuators are well suitable for fast system response, hydraulic actuators are still the first choice if compact form factor combined with high power density and reliability are demanded [28]. Furthermore, they have the ability to hold large forces constantly without overheating, as most other actuators would [26]. Nevertheless, hydraulic actuators contain several non-linearities, for instance, in the orifice equations, the mechanical friction, the flow leakage, the dead-zone and saturation of the control valve and others, which may require some in-depth investigation to come up with a proper modelling solution. Despite that solutions, either hydraulic actuators can not be modelled or the model becomes too complicated.

Moreover, there are uncertain model properties due to the huge variations of the parameters. For example, the oil density, the tightness of the gaskets, the mechanical deformations of the components, the usage, the load pressure and the discharge pressure can strongly vary due to many factors including the environmental temperature. Therefore, all these characteristics make the hydraulic systems not only more challenging to model but also to control. This is why we want to focus on the hydraulic system, in particular on a hydraulic cylinder controlled through a directional control valve.

Very often, operators manually control hydraulic systems to make several tasks. The result is a highly repetitive and tedious process. Fortunately, at least semi-automatic control has already entered these fields of application. In such cases, it is still the standard technique to use PID (proportional integral derivative) controllers for most of the closed-loop controls [26]. Despite the development of a vast array of advanced control strategies in the last years, the PID controller remains by far the most widely used within the industry [18]. Its popularity stems from its applicability and robust performance in a wide variety of operating scenarios. Furthermore, during the years several tuning methods were established that make the controller tuning fast and simple.

During the last few years, we see that modern automatic control systems are increasingly running via remote and wireless communication, while the process actuators, devices and whole machines are often decentralized. In some cases, the machine is not directly accessible for wired instrumentation and cabled signal processing by the control hubs and operators. Then, a wireless connection offers great benefits for remotely operating diverse industrial equipment due to the associated flexibility in instrumentation. Hence, using a large-long bus system or by the use of the internet (with TCP/IP or UDP as protocols), the data communication process is affected by a big variable transmission delay, for example see [30]. We want to highlight that with this approach the time delay is in the control loop, then, it strongly affects the controller design and consequently the reachable performances.

Hence, in this thesis, we want to design and implement a remote control for a hydraulic system with wireless communication. Moreover, seeing all the uncertainties in the system and the variable time delay, we want to study the robustness of the closed-loop system. The historical importance of a robust stability design approach is well-explained in [34].

In the literature several methods exist for a robust control design able to catch the optimal solution based on the performance criteria, for example, in [16], and [42]. *Quantitative Feedback*

Theory, *Linear Quadratic Control* and H_∞-*control* are some other methods commonly used in the literature to design for robust performance, see [10] and [44]. Unfortunately, the latter gives a controller with a high order of complexity. Instead, we choose to use a PID controller because, as we said, it is the most widely used in the industry [18] and it is quite reasonable to predict that PID control will continue to be used in the future [3]. Furthermore, PID controllers are the common solution in robotics, see [19], where the non-linearity makes the process design problem in a certain way similar to the hydraulic systems, and the robustness of the design plays an important role, see [13].
In the years, several techniques were invented for the robust PID controller design. Unfortunately, many of these can be applied to a restricted range of processes. For instance, in [37] and [36] the analysis is focused only on first-order processes with time delay. Even though it is possible to design a robust PID controller for time-delay systems based on the parameter space approach, see [15], we decide to use a design method provided by the works of Åström, see [43], [4], [24], [25] and [2]. The advantage of the proposed method is that the control system robustness is explicitly taken into account by means of the infinite norm of the sensitivity function. The procedure finds the optimal PID controller for the defined robustness level maximizing the integral coefficient k_i. Indeed, the maximization of the integral action means reducing the effect of a load disturbance on the system output. Furthermore, Åström in [4] also demonstrates that we reach the minimum integral error (IE) for the desired setpoint response if we take the highest k_i possible.
We will see that the presence of a large time delay in the control loop is the main limitation in the control design. Thus, we can not use an adaptive control strategy like in [18] because it may affect the stability of the system. Furthermore, because of its randomness variation, the implementation of a PI controller with the Smith Predictor would cause a loss of performance compared to a well designed PID controller, see [12].
Therefore, we can summarize the aims of this work in the following points:

1. to implement a remote controller that communicates through a WIFI connection to the system;
2. to evaluate and study the time delay in the control loop;
3. to find a linear model of the hydraulic system and analyze the parameter uncertainties;
4. to find a representation of the process with a perturbed model to study the robust stability;
5. to design optimally a two degree of freedom PID controller.

We see that it is not worthy to discover the best way to implement wireless communication. For that reason, we choose the TCP-IP communication protocol that guarantees a point to point connection and the correct data sequencing. If one is interested in improving the performance, it could be interesting to look at the UDP protocol. Indeed, the advantage is the fast streaming of the data, but the drawback is the loss of the sequencing. From a control point of view, the latter is a problem that has to be solved. For example, a time label could be assigned to the sample data so that it is possible with a routine to reorder them when received.
The thesis structure is mainly composed of six chapters, from the second to the eighth. We point out that we will go into details on applying the aforementioned control design to the experimental setup we use for the experiments only from chapter seven.

Chapter 2

Control oriented system modelling

This chapter shows how to obtain a model of the hydraulic actuator and how to linearise it. For the system analysis is very important to describe the hydraulic system with a set of equations. First of all, we need a model that is as close as possible to the real system. The so-called *full order model* achieves this purpose. Secondly, it is necessary to evaluate a linear model of the system for the application of the classical technique of the control system theory. In this chapter, the basic principles of the hydraulic actuators are marginally showed, because they are not the focus of the discussion. The chapter is composed of three parts, one for each model. The exposition follows the scheme to first present the theory in general, and then apply the result to the physical hydraulic system of University of Agder. Hence, we present a real practical example in the discussion.

2.1 Full order model

This section and the next one mainly follow the description of the hydraulic cylinder contained in the work of Ruderman, see [32], which describes theoretically the hydraulic cylinder. The scheme of a hydraulic cylinder controlled by a directional control valve (DCV) is shown in Figure 2.1.

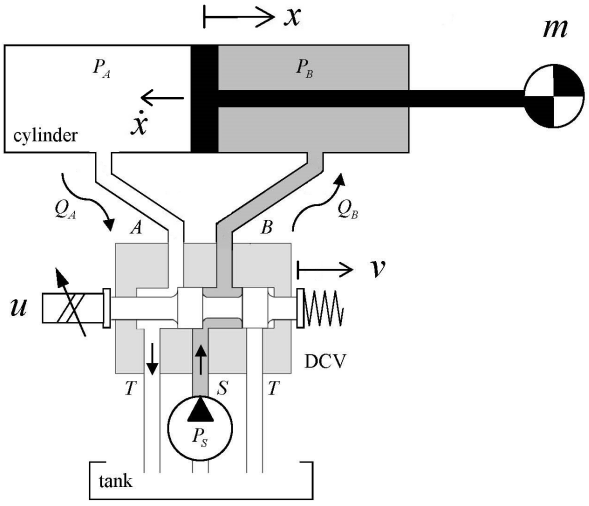


Figure 2.1: Principal structure of hydraulic cylinder controlled by directional control valve [32].

The hydraulic cylinder, the valve, the tank, the supply pump and the pipes that close the hydraulic circuit are easy to individuate in the Figure 2.1. The cylinder is made up of the piston and the left and right chambers. Note that we will refer to the left chamber with the letter A and to the right chamber with the letter B. The spool and the coil are important parts of the directional control valve. The first can move back and forward so that if it moves back it connects the B chamber with the supply pump and the A chamber with the tank, and if it moves forward the other way around. The coil is used to control the spool position: if it is

energized it attracts the spool forward, if not there is a spring (or another coil) that brings the spool back. Therefore, the control input u that actuates the valve strictly depends on the way in which the solenoid is low-level controlled. Basically, u can be the coil voltage applied through an amplifier to energize the electro-magnetic circuits of the solenoid. However, the more advanced and common case, which we also assume in the following, is that there is a direct proportionality between the control input u and the relative spool position. In this case, embedded electronics control the proportionality in the link between applied voltage and spool position through an internal coil current loop and external spool position loop. Note that a positive voltage u applied to the coil causes a positive displacement of the spool, denoted with ν, because of the convention adopted. Thus, in this case, chamber A is pressurized by the supply pump and chamber B is de-pressurized because connected to the tank. Hence, if the piston is free to move, it moves also in a positive direction x. For further details about the hydraulic components see [23]

Now we are going to analyze each part of the system from a control point of view so, from the control signal u to the position of the piston.

The system under study is equipped with the components listed in table 2.1.

Description	**Model number**
Cylinder	D633 R16KD1M0NSM2
Moog servo valve	CD25-40 25x200-SS-HC-SSN-NNN
Celesco linear-pot.	CLP-250

Table 2.1: Installed components of experimental system [27]

The input voltage signal u determines the spool position through the directional control valve with a dynamic that can be approximated by a second-order transfer function.

$$V(s) = \frac{\nu(s)}{u(s)} = \frac{\omega_0^2}{s^2 + 2\xi\omega_0 s + \omega_0^2} \tag{2.1}$$

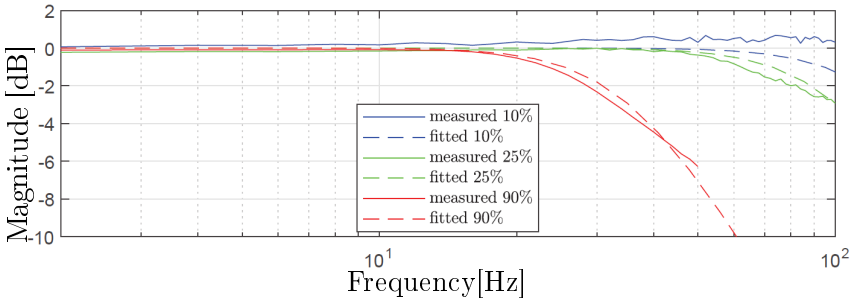
The valve dynamic is strongly correlated to the time constant of the mechanical movements of the spool, since the electro-magnetic dynamic is much faster. Thus, considering that the mechanical dynamics of the spool movement depends also on the forces that are applied to the spool, consequently on the pressure of the chambers, it is clear that the dynamics should vary to the change of the operative point. The latter affirmation is supported by experimental evidence, as it is pointed out in [27]. In the previous cited paper three different transfer functions correlated to three different values of the valve opening are obtained, of 10%, 25%, and 90% respectively, fitting the frequency response function. Figure 2.2 reports these results.

The parameters of the linearized transfer functions of figure 2.2 are summarized in table 2.2.

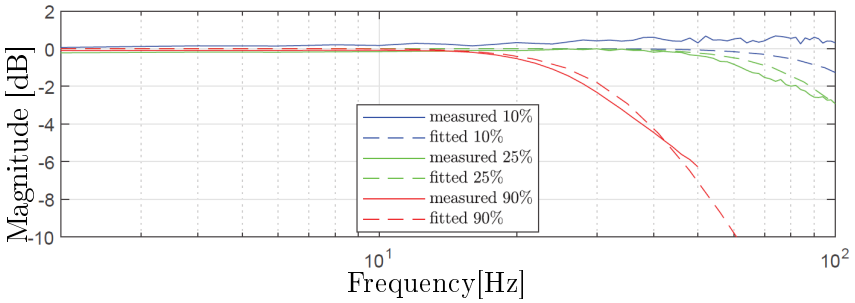
Valve opening [%]	$\omega_0[\frac{rad}{s}]$	ξ
10	816.8	0.7
25	628.3	0.7
90	220	0.7

Table 2.2: Servo valve second-order model parameters [27]

To reduce the valve leakage and sensitivity to the small input signals, the directional control valve is usually equipped with a closed center spool. Therefore, the spool-controlled flow characteristics are inherently subjected to dead-zone non-linearity. In fact, it is necessary to build a displacement in the spool position in both directions across the zero position, so the center. Furthermore, when the valve is completely open the actuator reaches the saturation point, that is another non-linearity. We obtain the following function combining these two non-linearities :



(a) Magnitude of the servo valve with different opening references.



(b) Phase of the servo valve with different opening references.

Figure 2.2: Measured FRFs versus fit obtained with linear model in Equation: 2.1 [27].

$$z = \begin{cases} \alpha \operatorname{sign}(\nu), & \text{if } |\nu| \geq \alpha + \beta, \\ 0, & \text{if } |\nu| < \beta, \\ \nu - \beta \operatorname{sign}(\nu), & \text{otherwise}, \end{cases} \tag{2.2}$$

Where α is the saturation level, and β is the dead-zone size in one direction. For simplicity, we assume that the parameters are the same in both directions. z is an internal state that represents the effective spool displacement that controls the opening of the valve. From equation 2.2 we can see that, if the absolute value of the spool position is less than β, then z is equal to 0. This is a theoretical approximation because the valve has some fluid leakage that causes a very slow motion of the cylinder, even with the presence of the spool displacement. This is experimentally found by Pasolli, see [27]. In Pasolli's work, the dead-zone is measured by applying a constant input signal u and processing the data of position x. Indeed, as will be more evident at the end of this chapter, when the piston reaches a constant speed, the speed value is almost proportional to the effective input signal z. It is, therefore, possible to experimentally evaluate the relationship between u and z. In figure 2.3 is shown the linearization of the data acquired in [27] and used in this thesis. The linear dead-zone and saturation characteristic of figure 2.3 is reported in equation 2.3.

$$z = \begin{cases} -1, & \text{for } u \leq -1.1 \\ u + 0.1, & \text{for } -1.1 < u \leq -0.1041\bar{6} \\ 0.04u, & \text{for } -0.1041\bar{6} < u \leq 0.1041\bar{6} \\ u + 0.1, & \text{for } 0.1041\bar{6} < u \leq 1.1 \\ 1, & \text{for } u > 1.1 \end{cases} \tag{2.3}$$

The effective opening position of spool z determines the volumetric flow rate. This is also associated with the pressure of both chambers in the following relations (2.4 and 2.5) that are known as *orifice equations*:

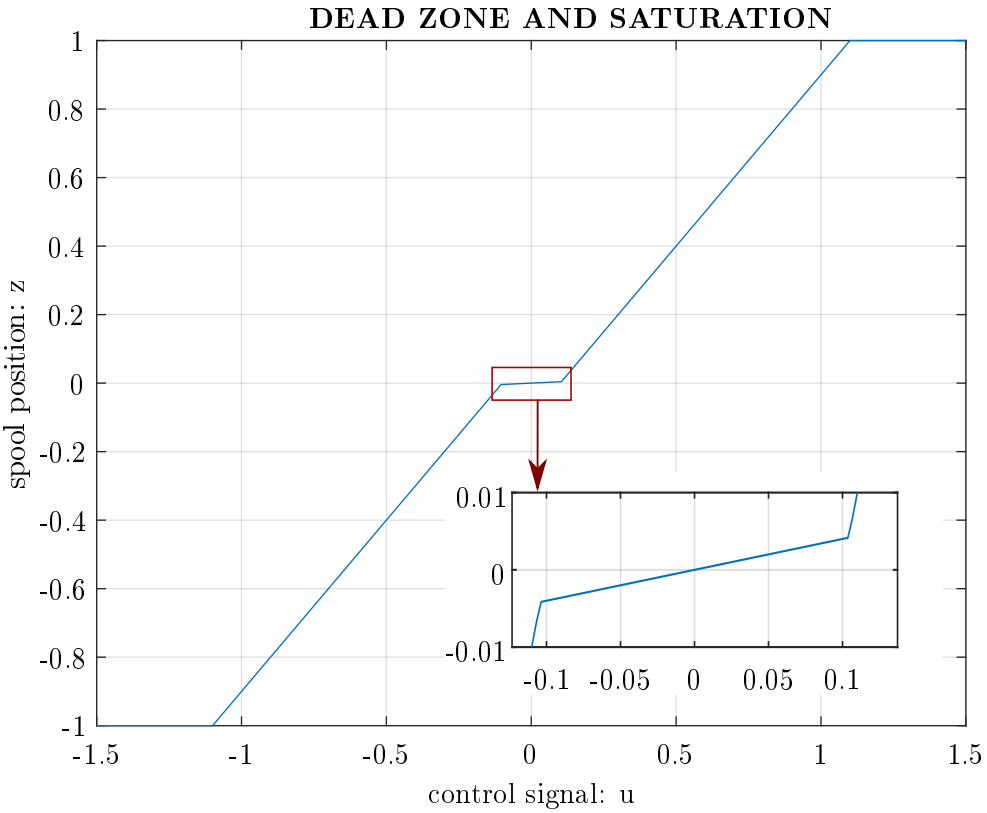


Figure 2.3: Valve dead-zone and saturation of the spool position.

$$Q_A = \begin{cases} zK\sqrt{P_S - P_A}, & \text{for } z > 0, \\ zK\sqrt{P_A - P_T}, & \text{for } z < 0, \\ 0, & \text{otherwise}, \end{cases} \tag{2.4}$$

$$Q_B = \begin{cases} -zK\sqrt{P_B - P_T}, & \text{for } z > 0, \\ -zK\sqrt{P_S - P_B}, & \text{for } z < 0, \\ 0, & \text{otherwise}, \end{cases} \tag{2.5}$$

In the Equations 2.4, and 2.5, P_S is the supply pressure, P_T is the pressure of the tank, P_A, and P_B are the pressures in chamber A, and B respectively. In the *orifice equations* K is the *valve flow coefficient*, that is defined in 2.6:

$$K = C_d \omega \sqrt{\frac{2}{\rho}} \tag{2.6}$$

The coefficient K is proportional to the width of the orifice area ω through the parameter C_d that is called *discharge orifice coefficient*. Furthermore, K increases if the oil density ρ decreases. From the continuity of the fluid pressure, we can find the following equations:

$$\dot{P}_A = \frac{E}{V_A}(Q_A - A_A\dot{x} - C_L(P_A - P_B)) \tag{2.7}$$

$$\dot{P}_B = \frac{E}{V_B}(Q_B + A_B\dot{x} - C_L(P_B - P_A)) \tag{2.8}$$

where Q_A, and Q_B, are the flow rate of chamber A and B, respectively. E is the *bulk modulus*, and it represents the incompressibility of the fluid in the hydraulic circuit. For simplicity, we assume the E constant, even though is generally pressure-dependent. A_A and A_B are the side A and side B effective piston areas of the cylinder, respectively. The C_L is the internal leakage coefficient,

which takes into account the pressure drop. Indeed, the fluid penetrates from the chamber with the highest pressure to the other chamber, producing the pressure drop. That happens because the piston is not perfectly isolating the cylinder chambers. Finally, in equations 2.7, 2.8, V_A and V_B are the total effective volumes of the pipes that connect the chambers in the hydraulic circuits and the chambers volumes of A and B respectively. Because of this definition, we obtain the total effective volumes from the equations below:

$$V_A = V_A^0 + A_A x \tag{2.9}$$

$$V_B = V_B^0 - A_B x \tag{2.10}$$

In equations 2.9, and 2.10, with V_A^0, and V_B^0 it is indicated the total volume in the hydraulic circuits including pipes, when the rod drive is in the initial zero position: $x = 0$.
The mechanical dynamic gives the last equation that we include in the model. Thank the second Newton's law and the definition of the pressure, for the all moving mass m of the hydraulic cylinder, the following relation exists:

$$m\ddot{x} = P_A A_A - P_B A_B - f - F_L \tag{2.11}$$

In the above equation, the sum of forces at the right side cause the acceleration $\ddot{x}$. In particular, the force, which depends on the difference between the pressure in both chambers, is the controlled parameter. F_L indicates all the external forces that oppose the defined positive rod motion. f represents all the friction forces. The Stribeck friction model well describes f:

$$f(\dot{x}) = \text{sign}(\dot{\text{x}})(F_c + (F_s - F_c)e^{-\left(\frac{|\dot{x}|}{\chi}\right)^{\delta}}) + \sigma\dot{x} \tag{2.12}$$

Indeed, first of all, it considers the constant Coulomb friction F_c separately from the stiction friction force F_s. Secondly, equation 2.12 considers the velocity-weakening effects in the low-velocity range around zero and the viscous friction that is considered proportional to the speed with the constant σ. In the Stribeck characteristic curve δ and χ are Stribeck shape factors that have to satisfy the constraints: $\chi > 0$, $\delta \neq 0$.
In table 2.3 the model parameters of the system are summarized, and in figure 2.4 a scheme of the *full order model* obtained is shown.

Parameter	**Value**	**Unit**
m	1.7026	Kg
A_A	1.310^{-3}	m^3
A_B	0.7610^{-3}	m^3
K	0.25210^{-6}	$\frac{m^2}{s\sqrt{Pa}}$
E	10^{-9}	Pa
P_T	0	Pa
P_S	10^7	Pa
V_A	0.0007	m^3
V_B	0.0007	m^3
F_L	0	N
C_L	0	$\frac{1}{s}$
l	0.2	m

Table 2.3: System parameters from [27] and [28]

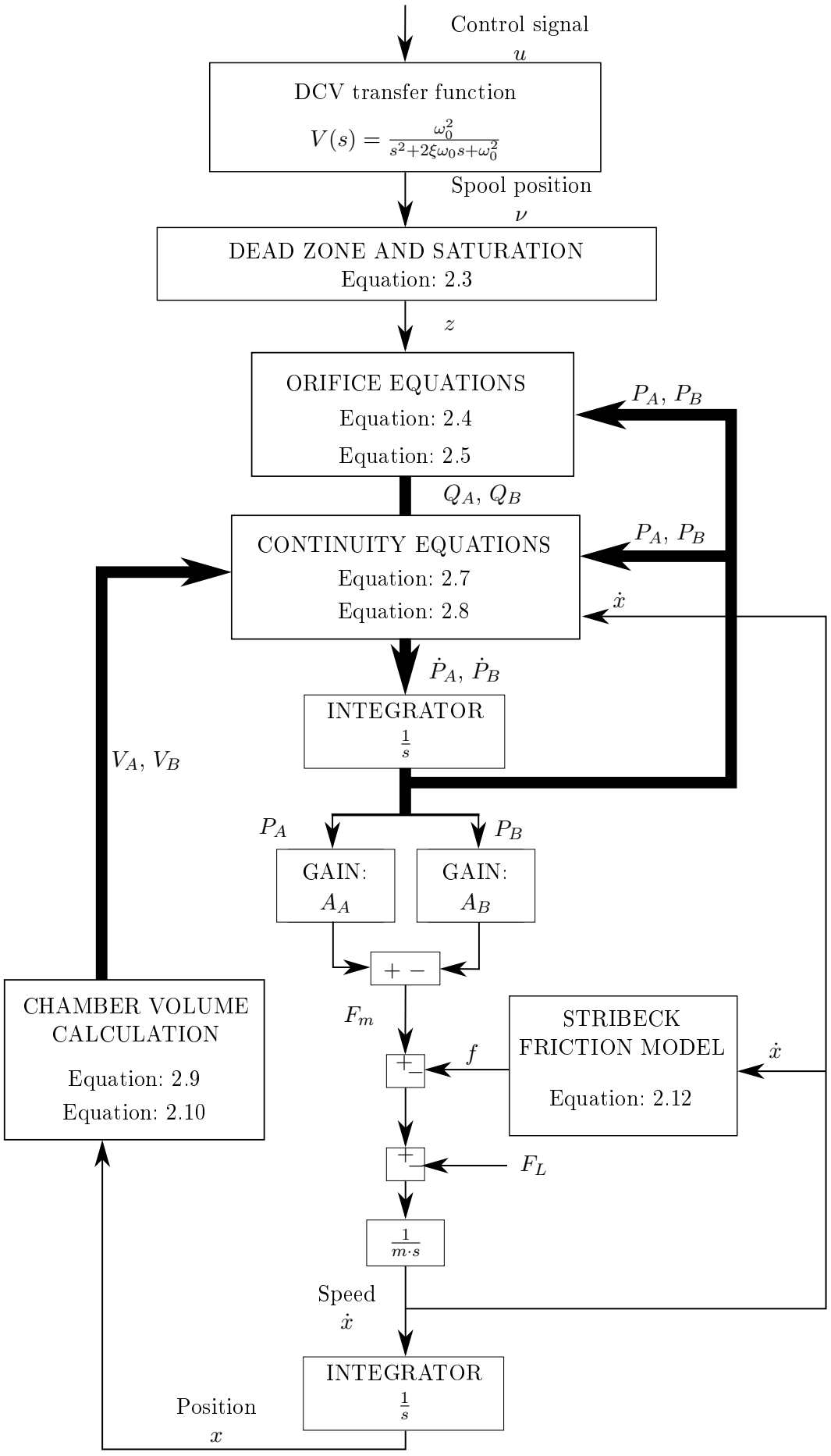


Figure 2.4: Scheme of the *full order model* of the hydraulic cylinder with the DCV.

2.2 Reduced order model

Now we want to reduce the complexity of the *full order model*, hence create a *reduced order model*. The objective is to find a system from which it is easy to compute a linear model. As it is shown in figure 2.4 on page 8, the *full order model* has a structure that is based on a parallel calculation of P_A and P_B from Q_A and Q_B because of the two pairs of equations: 2.4, 2.5, 2.7, and 2.8. Therefore, reducing the two branches into one is the first aim. To do that, for the *orifice equations*, 2.4 and 2.5, introducing the load-related pressure is sufficient:

$$P_L = P_A - P_B \tag{2.13}$$

For simplicity, from this point, we consider that the tank pressure is equal to zero $P_T = 0$ because it is significant the relative difference with the supply pressure P_S and not its absolute value. We also assume a closed hydraulic circuit, so that $|Q_A| = |Q_B|$. In particular, it is easy to see that if $Q_L = Q_A = -Q_B$, because of the just assumed hypothesis, then:

$$Q_L = zK\sqrt{\frac{1}{2}(P_S - \text{sign(z)}\, P_L)} \tag{2.14}$$

In the same way, also for the *continuity equations* 2.7 and 2.8 finding a collapsed representation for the load-related pressure is possible. This is made using the following definitions:

$$\bar{A} = \frac{A_A + A_B}{2} \tag{2.15}$$

$$V_t = V_A^0 + V_B^0 \tag{2.16}$$

so we can express the aggregated *continuity equation* in terms of mean area and total volume:

$$\dot{P}_L = \frac{4E}{V_t}(Q_L - \bar{A}\dot{x} - C_L P_L) \tag{2.17}$$

In this case, we have also assumed negligible the difference in the piston areas of the two surfaces, hence $A_A = A_B$. Otherwise, the latter equation produces an average error of half of the rod cross-section area. Finally, from the same assumptions, we can express the mechanical dynamic equation 2.11 using the new defined state variable P_L with the following:

$$m\ddot{x} = P_L\bar{A} - f(\dot{x}) - F_L \tag{2.18}$$

In summary, considering these following hypothesis:

1. the hydraulic circuit is closed, so there are no losses and $|Q_A| = |Q_B|$;
2. the tank pressure is assumed to be zero $P_T = 0$;
3. the difference on the piston areas of the two surfaces is negligible, hence $A_A = A_B$;

new state variables Q_L and P_L are calculated to collapse the two parallel branches of the system into one.
Furthermore, we can neglect the second-order closed-loop behaviour of the directional control valve. Indeed, usually, the dynamic of the DCV spool is much faster than that of the hydraulic and mechanical sub-systems. In other words, the time constants of the last two sub-systems are much greater than the time constant of the DCV dynamic. Simulation results show the goodness of this last assumption using the parameters of the particular system considered in this thesis. The overall scheme of the *reduced order model* is shown in figure 2.5 on page 10.

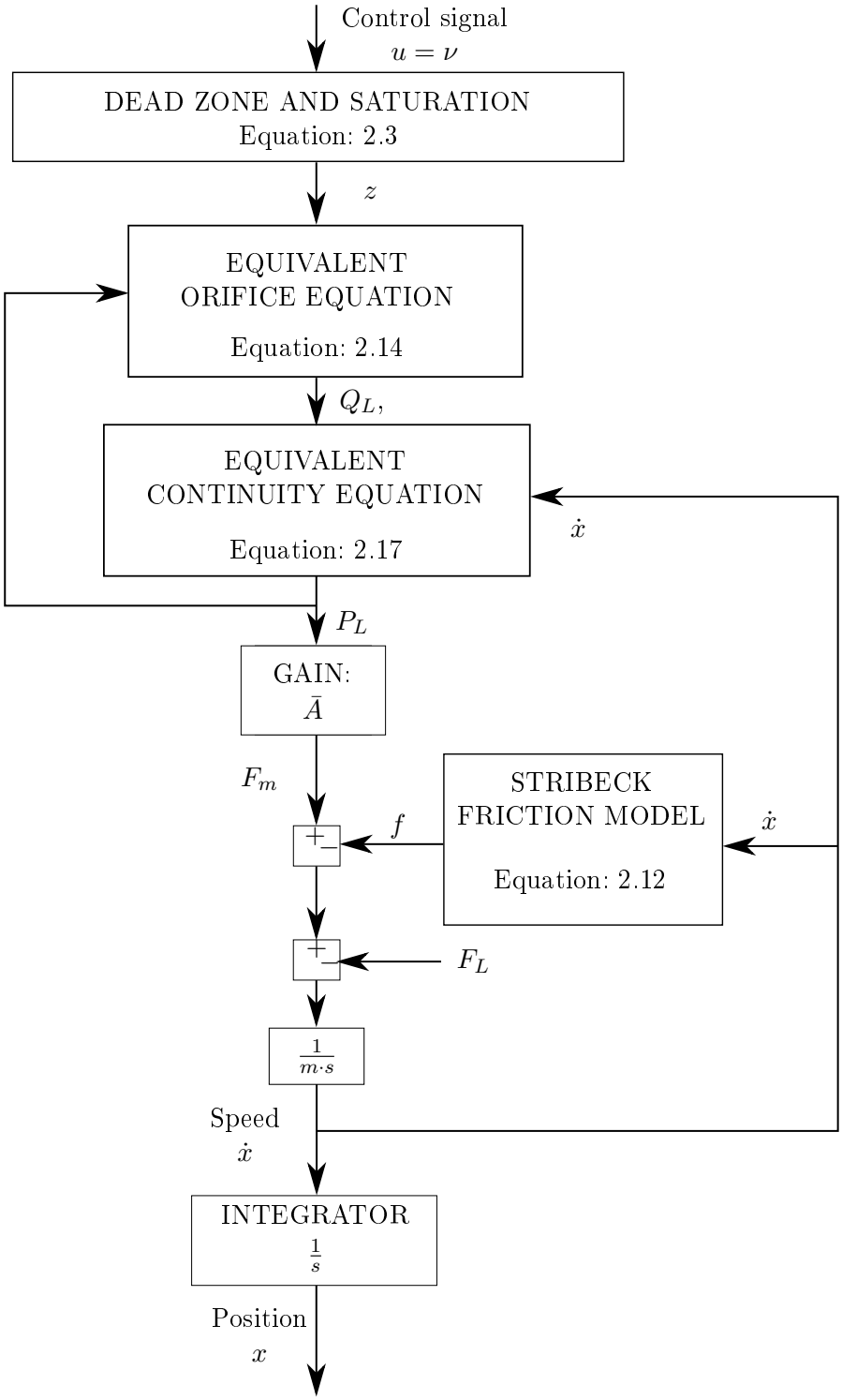


Figure 2.5: Scheme of the *reduced order model* of the hydraulic cylinder.

2.3 System linearisation

The two models evaluated in the previous sections, the *full order model* and the *reduced order model*, have non-linearities. This implies that we can not use both for the design of the control system based on a PID controller. Therefore, it is necessary to linearise the system over an operative point. We want to highlight that we use the *reduced order model* for the linearization, see figure 2.5 for clearness.
At the beginning of this section, we discuss how the non-linearities are linearised and, then we explain how we choose the linear viscous coefficients for the friction models.

2.3.1 Linearised model

In the *reduced order model* the main non-linear parts are given by the *orifice equation* 2.14 and the Stribeck friction model 2.12. The first is a function of two inputs and one output $Q_L = f(P_L, z)$, so it is possible to linearise Q_L at the operative point $\hat{Q}_L \simeq (\hat{P}_L, \hat{z})$ using the chain rule. For the first-order approximation, we get:

$$\hat{Q}_L = \left.\frac{\partial Q_L}{\partial z}\right|_{\hat{P}_L} z + \left.\frac{\partial Q_L}{\partial P_L}\right|_{\hat{z}} P_L \tag{2.19}$$

Hence, using the following definitions:

$$\hat{C}_q \coloneqq \left.\frac{\partial Q_L}{\partial z}\right|_{\hat{P}_L} = K\sqrt{\frac{1}{2}(P_S - \text{sign(z)}\,\hat{P}_L)} \tag{2.20}$$

$$\hat{C}_{qp} \coloneqq -\left.\frac{\partial Q_L}{\partial P_L}\right|_{\hat{z}} = \frac{\hat{z}K\,\text{sign}(\hat{z})}{4\sqrt{\frac{1}{2}(P_S - \text{sign}(\hat{z})\,P_L)}} \tag{2.21}$$

We obtain the equation 2.19 write in this form:

$$\hat{Q}_L = \hat{C}_q z - \hat{C}_{qp} P_L \tag{2.22}$$

C_q is called *flow-gain coefficient* and C_{qp} is called *flow-pressure coefficient*.
Regarding the Stribeck friction model, being it not worthy to linearise the curve at an operative point $\hat{\dot{x}}$, the linear model that we consider is the one below:

$$f_{LIN}(\dot{x}) = \sigma_{LIN}\dot{x} \tag{2.23}$$

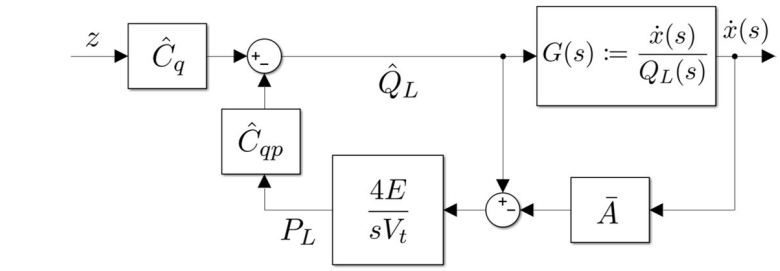
It is important to highlight that in the equation 2.23 σ_{LIN} is not the same parameter as σ in equation 2.12. We explained the choice of these two parameters in section 2.3.2. We neglected the dead-zone and the saturation of the directional control valve because in section 4.4.3 we will introduce a Feed-Forward action in the control that compensates the dead-zone nonlinearity.
Now that there are no more non-linearities, it is possible to find the overall transfer function from the input control signal u to the output position x. Using the linear friction model 2.23, evaluating the transfer function for the all sub-system from Q_L to the speed $\dot{x}$ is possible, simply combining equations 2.18 and 2.17. Indeed, considering $F_L = 0$ and $C_L = 0$, if there are no internal leakage and external forces, we obtain:

$$G(s) \coloneqq \frac{\dot{x}(s)}{Q_L(s)} = \frac{1}{(ms + \sigma_{LIN})V_t s + \bar{A}} = \frac{\bar{A}^{-1}}{\frac{s^2}{\omega_c^2} + 2\zeta\frac{s}{\omega_c} + 1} \tag{2.24}$$

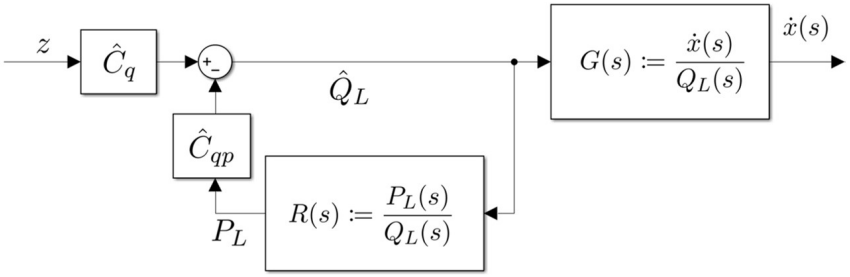
where we have used the following definitions:

$$\omega_c = 2\bar{A}\sqrt{\frac{E}{V_t m}} \tag{2.25}$$

$$\zeta = \frac{\sigma_{LIN}}{4\bar{A}}\sqrt{\frac{V_t}{Em}} \tag{2.26}$$



(a) first block reduction



(b) second block reduction

Figure 2.6: Block diagram of the linear model of the process from z to $\dot{x}$.

In figure 2.6a the linearised block diagram obtained combining the equations (2.24), (2.22) and (2.17) is shown. From that diagram we can compute the transfer function from Q_L to P_L:

$$R(s) := \frac{P_L(s)}{Q_L(s)} = \frac{4E(1 - G(s)\bar{A})}{sV_t} = (sm + \sigma_{LIN})G(s)\bar{A}^{-1} \tag{2.27}$$

In this way the scheme in figure 2.6a becomes the block diagram in figure 2.6b. From this last scheme evaluating the transfer function from z to the speed $\dot{x}$ is very simple:

$$\begin{aligned} \hat{G}(s) := \frac{\dot{x}(s)}{z(s)} &= \frac{\hat{C}_q G(s)}{1 + \hat{C}_{qp} R(s)} = \frac{\hat{C}_q G(s)}{1 + \hat{C}_{qp}(sm + \sigma_{LIN})G(s)\bar{A}^{-1}} \\ &= \frac{\hat{C}_q \bar{A}}{s^2 \frac{mV_t}{4E} + s(\hat{C}_{qp} m + \frac{\sigma_{LIN} V_t}{4E}) + \sigma_{LIN}\hat{C}_{qp} + \bar{A}^2} \end{aligned} \tag{2.28}$$

Hence, directly from the equation (2.28) we obtain the overall process transfer function, from the effective control signal z to the measured position x:

$$P(s) := \frac{x(s)}{z(s)} = \frac{1}{s}\hat{G}(s) = \frac{\frac{\hat{C}_q \bar{A}}{\sigma_{LIN}\hat{C}_{qp} + \bar{A}^2}}{s\left[s^2 \frac{mV_t}{4E(\sigma_{LIN}\hat{C}_{qp} + \bar{A}^2)} + s\frac{\hat{C}_{qp} m + \frac{\sigma_{LIN} V_t}{4E}}{\sigma_{LIN}\hat{C}_{qp} + \bar{A}^2} + 1\right]} \tag{2.29}$$

2.3.2 Viscous friction

During the linearisation of the process, to find the overall transfer function $P(s)$ from control signal to position, it is highlighted that the linear viscous coefficient σ in the Stribeck friction model (see (2.12)) differs from the linear viscous coefficient σ_{LIN} in the linear friction model, see (2.23). This is because the objective of the Stribeck curve is to be as realistic as possible. Instead, the previous aim is not the same as for the linear model because of its simplicity. Thus this parameter has to be chosen with proper arguments.
Previous made work on the same system, reported on [27], and data of the position vs force were collected and here shown in figure 2.7. The author obtained these data by giving a constant control signal to the valve and measuring both position x and pressure in both chambers P_A and P_B. Then, the position is fitted with a line with constant slope using the minimum square error method, and from the pressure evaluating the effective force produced by the cylinder is possible. The friction force is the same as cylinder force under the zero acceleration condition. The friction model that we use for fitting data is not exactly the one in equation (2.12) because computational problems in the simulation are caused by a function that is neither continuous nor differentiable, see [33]. Therefore, we replace the sign($\dot{x}$) operator with the continuous tanh($K\dot{x}$):

$$f(\dot{x}) = \tanh(2000\dot{x})(F_c + (F_s - F_c)e^{-(\frac{|\dot{x}|}{\chi})^{\delta}}) + \sigma\dot{x} \tag{2.30}$$

K value determines the shape of the curve: the higher the K value, the better the approximation of tanh to the sign. We chose $K = 2000$ after having seen that it is high enough. The result is not satisfying using the Matlab fitting application. Instead, we use the optimization toolbox of Matlab, see [8]. In this way, the fitting problem is reduced by minimizing the integral square error of the data and the model as is shown in the extracted Matlab code reported below. Adding constraint that avoids solutions with $F_s < F_c$ for physical interpret-ability is necessary. Furthermore, it is important to chose the proper initial condition and the correct interval range for the model-free variables: σ, χ, δ, F_c, and F_s.

Listing 2.1: Extracted Matlab code to fit the Stribeck friction model to the experimental data

```
%% LOAD FRICTION DATA MEASURED:
% 'FR'=force[N]; 'xd'=speed[m/s]
prob = optimproblem('ObjectiveSense','min');
SIGMA = optimvar('SIGMA','LowerBound',800,'UpperBound',1100);
CHI = optimvar('CHI','LowerBound',0.0001,'UpperBound',1);
DELTA = optimvar('DELTA','LowerBound',0.001,'UpperBound',2);
FC = ...
    optimvar('FC','LowerBound',min(abs(FR))*0,'UpperBound',min(abs(FR))+50);
FS = ...
    optimvar('FS','LowerBound',max(abs(FR))*0,'UpperBound',max(abs(FR))+100);

IAE_cost = optimexpr(length(xd));
syms Fc Fs Chi Delta Sigma real positive
for i=1:length(xd)
    IAE_fnc = matlabFunction(abs(FR(i) - tanh(2000*xd(i))*...
        (Fc+(Fs-Fc)*exp(-(abs(xd(i)/Chi))^Delta))-Sigma*xd(i))); % ...
           @(Chi,Delta,Fc,Fs,Sigma)
    IAE_cost(i) = fcn2optimexpr(IAE_fnc, CHI,DELTA,FC,FS,SIGMA);
end
prob.Constraints.Grater = FS>=FC;
prob.Objective = IAE_cost'*IAE_cost;
clear x0
x0.SIGMA = 980;
x0.CHI = 0.014;
x0.DELTA = 0.9;
x0.FC = min(abs(FR));
x0.FS = max(abs(FR));
[x, cost] = solve(prob, x0)
```

The code produces this output:

$$\text{Fitting solution} = \begin{cases} \chi = 0.0039 \\ \delta = 0.5687 \\ F_c = 54.5604 \\ F_s = 278.0013 \\ \sigma = 849.0919 \end{cases} \tag{2.31}$$

The fitted result is reported in figure 2.7 and is summarized in the following equation:

$$f(\dot{x}) = \tanh(2000\dot{x})(54.5604 + (278.0013 - 54.5604)e^{-(\frac{|\dot{x}|}{0.0039})^{0.5687}}) + 849.0919\dot{x} \tag{2.32}$$

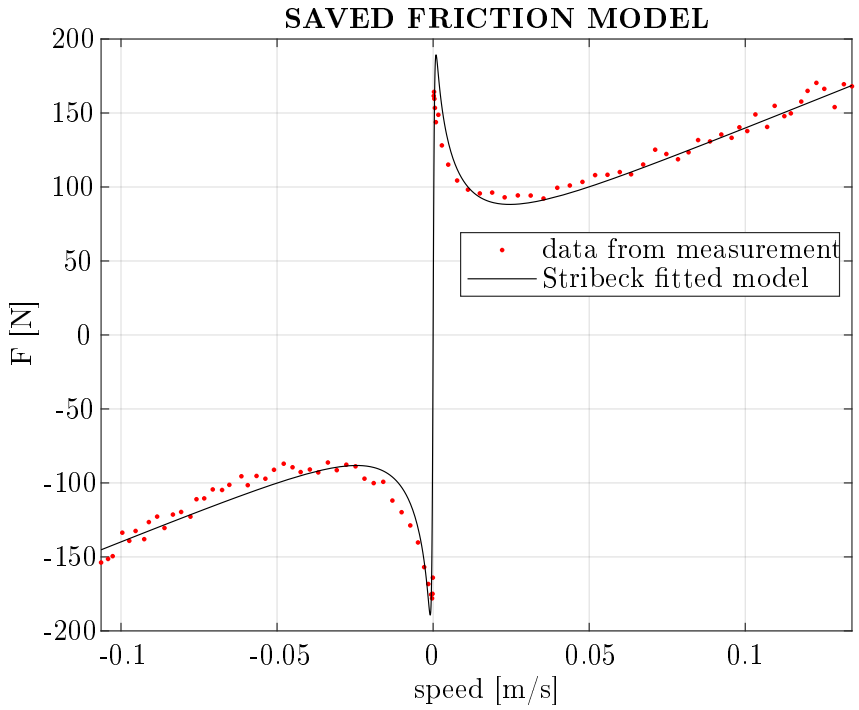


Figure 2.7: Measurement data and fitted Stribeck friction model

We treat now the choice of the linear viscous coefficient for the linear model: σ_{LIN}. From figure 2.5 we can see that it is not easy to compensate constant friction torque with a feed-forward action. Indeed, due to the non-linearities of the model and due to the internal feedback loops of pressure and speed, the linearisation can be easily made only with the simple model chosen in equation (2.23). Hence, σ_{LIN} has to represent the friction torque in the all speed range, both for positive and negative values. So, it is important to know approximately what is the operative speed range. To do this we made several experimental tests. The idea is to give the maximum allowed control signal to the valve to get the highest steady-state speed. Therefore, the control signal is a square wave that amplitude is the one that reaches the actuator saturation, and the time width has to be less enough to avoid arriving at the end of the piston stroke. Practically we compute the inverse of Laplace transform of the process transfer function $P(s)$ multiplied by the step transfer function of amplitude A and time length $\hat{t}$. The result is superimposed equally to the maximum drive position that we can obtain l_{MAX}:

$$x(\hat{t}) = \mathscr{L}_s^{-1}\left[P(s)\frac{A}{s}\right](\hat{t}) \stackrel{!}{=} l_{MAX} \tag{2.33}$$

If we neglect from the calculation result of the inverse Laplace transform in equation (2.33) the oscillatory terms, and introducing into $P(s)$ (see equation (2.29)) the system parameters of

table 2.3 we obtain the following relation:

$$x(\hat{t}) \simeq A(-0.00014 + 0.37264\hat{t}) \stackrel{!}{=} l_{MAX} \Longrightarrow \hat{t} \leq \frac{l_{MAX}}{A0.37264} \tag{2.34}$$

In the right side of equation 2.34 we see that the relation between step time and amplitude is managed by the unit speed of $\nu = 0.37264[\frac{m}{s}]$. For safety we introduce a multiplicative factor of 1.4, so $\nu' = 1.4\nu = 0.53[\frac{m}{s}]$. It is now possible design the set of experiments, that are shown in table 2.5.

Experiment	[Exp.1]	[Exp.2]	[Exp.3]	[Exp.4]	[Exp.5]	[Exp.6]	[Exp.7]	[Exp.8]
A [up to 1]	0.2	0.5	−0.5	0.7	−0.7	1	−1	1
$\mathbf{l_{MAX}}$**[m]**	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
$\hat{\mathbf{t}}$**[s]**	0.9434	0.3774	−0.5	0.7	−0.7	1	−1	1

Table 2.4: Set of experiments for speed range evaluation

From the experiments, we acquire the position measurements. These are affected by a high noise level. Thus it is necessary to evaluate the speed using a filtered derivative, for example, with a second-order filter:

$$F_\nu = \frac{s}{(\tau_f s + 1)^2} \tag{2.35}$$

The cut off frequency of the filter has to be grater than the sampling frequency of the control, that is $f_s = 2[Khz]$, so we expect to use a time constant $\tau_f \geq 0.0005[s]$. From experimental evaluation of the data with a set of derivative filters we see in figure 2.8 that a proper value for τ_f is 10 time grater than $\frac{1}{f_s}$, so $\tau_f = 0.005[s]$. The results of the steady state speed calculations are shown in table 2.5, and for [Exp.8] in figure 2.9.

Experiment	[Exp.1]	[Exp.2]	[Exp.3]	[Exp.4]	[Exp.5]	[Exp.6]	[Exp.7]	[Exp.8]
Max S. S. $\dot{\mathbf{x}}[\frac{\mathbf{m}}{\mathbf{s}}]$	0.07	0.27	−0.23	0.26	−0.33	0.27	−0.43	0.26

Table 2.5: Steady state speed of the driven mass in different experiments

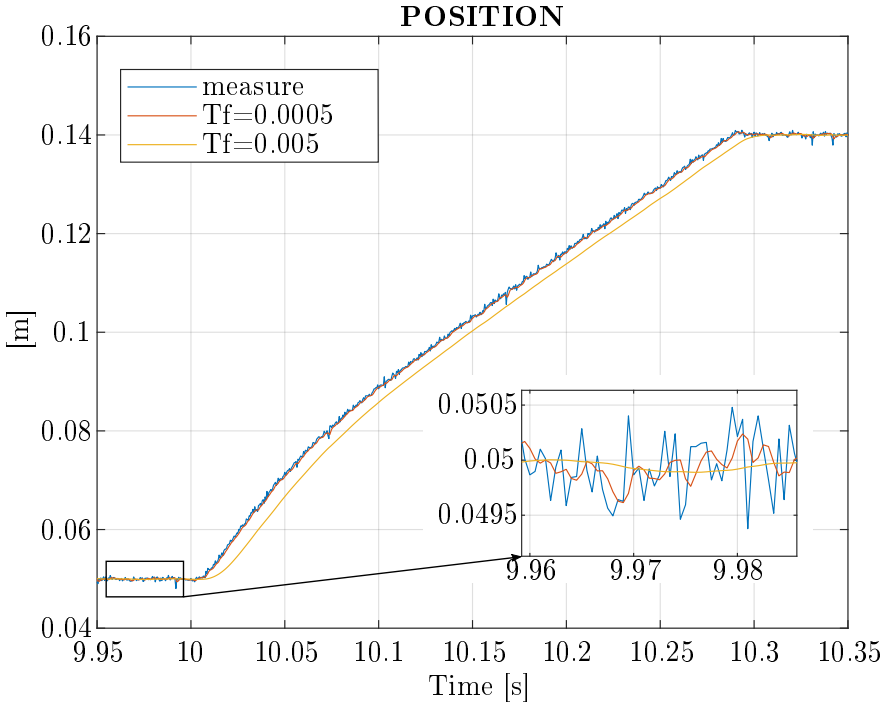
From the results on table 2.5 we can notice that generally with the same amplitude of the control signal in the back direction (so with a negative value of A), we reach higher speed values. That is due to the asymmetry of the system, in particular of the piston areas in chambers A and B. Furthermore, we can assume for the speed range $|\dot{x}| \leq 0.25[\frac{m}{s}]$ because it is big enough for most of the experiments, and also during the control if we avoid the actuator saturation, it is difficult to reach the maximum physical speed value.
Now that we have the operative speed range, we choose σ_{LIM} with four different approaches:

1. σ_{LIM} allows the linear model of equation (2.23) to reach the same value of the friction data at the extreme of the speed interval: $\dot{x} = 0.1342[\frac{m}{s}]$.

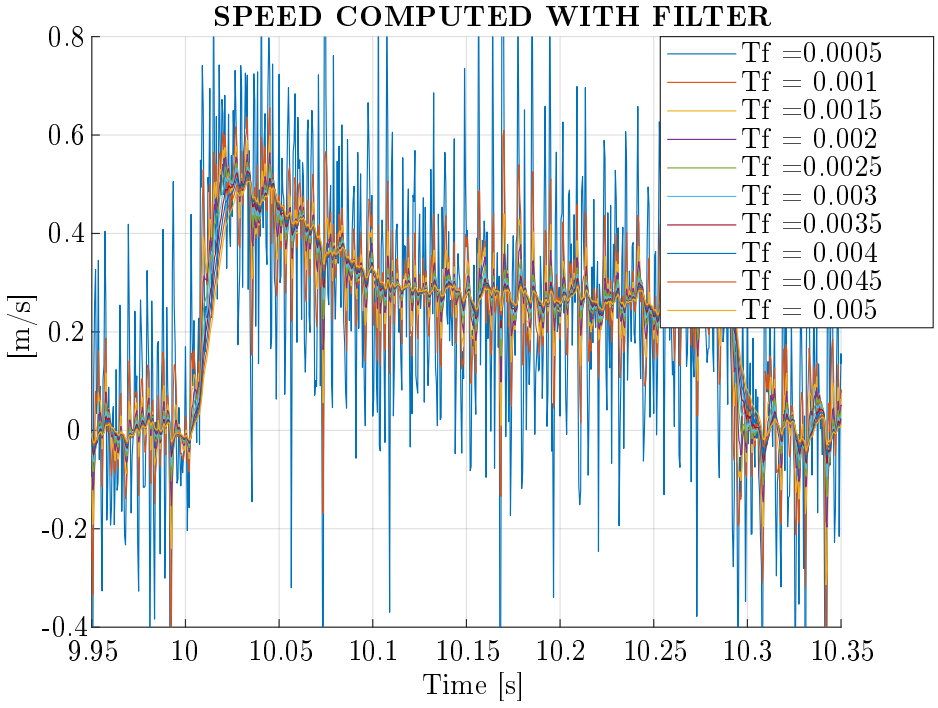
2. σ_{LIM} minimizes the integral error between the linear model of equation (2.23) and the fitted Stribeck friction model of equation 2.32 in the speed interval covered by the experimental data, so without extrapolation: $\dot{x}_{MAX} = 0.1342[\frac{m}{s}]$.

3. σ_{LIM} allows the linear model of equation (2.23) to reach the same value of the fitted Stribeck friction model of equation 2.32 at the extreme of the speed interval: $\dot{x} = 0.25[\frac{m}{s}]$.

4. σ_{LIM} minimizes the integral error between the linear model of equation (2.23) and the fitted Stribeck friction model of equation 2.32 in the operative speed interval: $\dot{x}_{MAX} = 0.25[\frac{m}{s}]$.



(a) The rod end position measured by an experimental test and the position filtered by two LPF with a cut off frequency of $\frac{1}{\tau_f} = 1/0.0005[Hz]$ and $\frac{1}{\tau_f} = 1/0.005[Hz]$.



(b) speed computed with $\frac{s}{1+\tau_f s}$ with ten time constant values form $\tau_f = 0.0005[s]$ to $\tau_f = 0.005[s]$

Figure 2.8: Rod end speed evaluation with a set of derivative filters with different cut off frequencies $\frac{1}{\tau_f}$

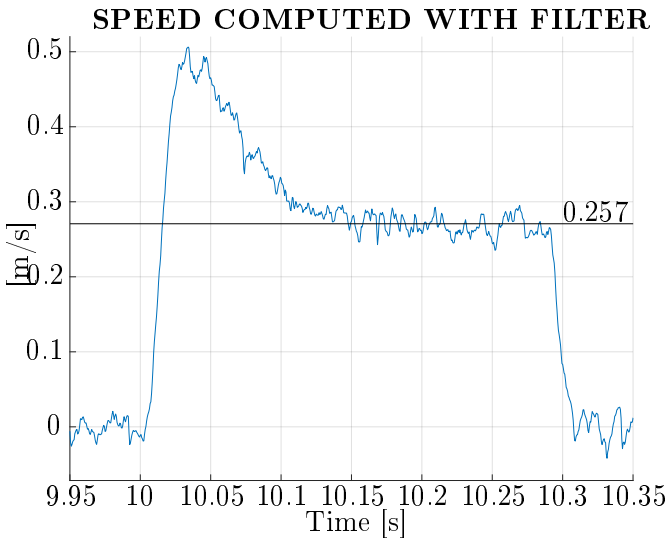


Figure 2.9: Position measurement and speed evaluation of [Exp.8] with $\tau_f = 0.005[s]$

	[case1]	[case2]	[case4]
$\sigma_{LIN}[\frac{Ns}{m}]$	1252.3	1808.3	1327.8

Table 2.6: Evaluation of σ_{LIN} using three different method

For the first two cases, the results of the linear friction model that comes out are shown in figure 2.10 over the experimental data and the fitted Stribeck friction model. For the combination of the first case and the last one, see figure 2.10. The first and the third approaches use the fact that inside the speed range, the friction force, that is a stabilizer, is always less than the expected one. Hence, the model for designing the control is the worst case. Unfortunately, we do not have the experimental data for the operative speed range, and it is dangerous to extrapolate any data because friction is a strongly non-linear phenomenon. For the latter reason, we exclude the third strategy. At the same time, these two approaches could be too conservative. Therefore, the idea behind the second and the fourth points is to compensate for the error of the estimated friction force in the speed range. In this case, we can not say anything about the conservativeness of this criterion because we do not know apriori how much time the driven mass will spend at which speed. Looking at table 2.6 and at figure 2.10 we can see that the first and the fourth cases have a very close value of σ_{LIN}. Therefore, a reasonable choice is to take one of these values for the linear model of friction because is the trade-off between the main two approaches described before. Finally, we use the smallest one between the two to be more conservative.

$$\sigma_{LIN} = 1252.3 \left[\frac{Ns}{m}\right] \tag{2.36}$$

The evaluation of σ_{LIN} with the second and the fourth approach is obtained using again the optimisation toolbox of Matlab. Here is shown the source code for clarity.

Listing 2.2: Extracted Matlab code to calculate σ_{LIN}

```
%% FIT VISCOUS FRICTION COEFFICIENT FOR LINEAR MODEL:

% case 1: friction(@ max_speed)=friction linear model
sigma_linearModel_1 = FR(end)/xd(end);

% case 2: integral_mean(friction - linear_friction_model)(@speed_range)=0
prob = optimproblem('ObjectiveSense','min');
SIGMA = optimvar('SIGMA','LowerBound',100,'UpperBound',200000);
```

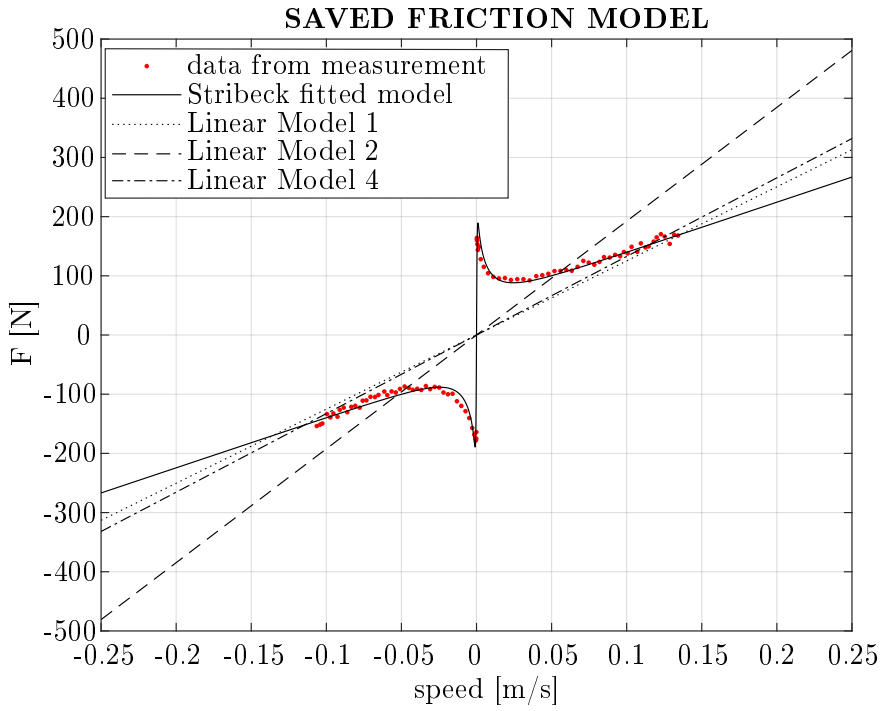


Figure 2.10: Comparison between experimental data, fitted Stribeck model and linear friction model. The latter is computed in three different ways.

```
syms Sigma real positive
IAE_cost_fnc = matlabFunction(int( sign(x1)*(Sigma*x1-f_model), x1, ...
    [-xd_lim xd_lim]));
IAE_cost = fcn2optimexpr( IAE_cost_fnc, SIGMA);
prob.Objective = (IAE_cost)^2;
clear x0
x0.SIGMA = 1000;
[x, cost] = solve(prob, x0);
sigma_linearModel_2 = x.SIGMA;
```

Chapter 3

Uncertainty Analysis

In the previous chapter, we evaluate the process transfer function $P(s)$ with the equation (2.29) thanks to the linearisation of the *reduced order model*. In particular, we made the linearisation of the *orifice equation* (2.14) around an operative point that gives specific values of the parameters $\hat{C}_q$, and $\hat{C}_{qp}$. Hence, the process transfer-function $P(s)$ is correct only in a close interval around this operative point. If the system works in a different state point, then, to be precise, we have to compute the value of $\hat{C}_q$, and $\hat{C}_{qp}$ again to have the correct transfer function. Because this thesis work aims to design a controller with only one robust PID parameter configuration, it is necessary to fix the value of $\hat{C}_q$, and $\hat{C}_{qp}$. Thus, the advantage is working with only one transfer function, but the price is that we have to consider the uncertainty of these parameters. As we can see, the analysis of the process uncertainty and the choice of the nominal model plays an important role in the controller design. The choice of $\hat{C}_q$, and $\hat{C}_{qp}$ is not obvious, several considerations have to be made. Therefore in this chapter, we explore first of all the variation of these parameters of $P(s)$. Then, in the second section, we introduce the control time delay presence τ in the model of the system. In this case, the analysis of the statistical distribution of τ has primary importance for the simulation model. Finally, in the third section, we combine all these three variational parameters to define the nominal model and the entity of his uncertainty.

3.1 Parameter uncertainty of transfer function

As briefly anticipated, in the process transfer function $P(s)$ the uncertain parameters are C_q, and C_{qp}. Notice that we refer to a specific value that these parameters assume in a particular operational point when they are signed with the hat: $\hat{C}_q$, $\hat{C}_{qp}$. Otherwise, without the hat we consider the parameters as uncertainties. Hence, basically from equations (2.20), and (2.21), we can express the uncertainties functions respectively with the following equations:

$$C_q(P_L) = K\sqrt{\frac{1}{2}(P_S - \text{sign(z)}\, P_L)} \tag{3.1}$$

$$C_{qp}(P_L, z) = \frac{zK\,\text{sign(z)}}{4\sqrt{\frac{1}{2}(P_S - \text{sign(z)}\, P_L)}} \tag{3.2}$$

We highlight that $C_q(P_L)$ is a function of only P_L even though in the equation (3.1) appears the term sign(z), because we consider that the P_L is positive when z is positive, and vice versa. Therefore sign(z) P_L is the same as $|P_L|$. This assumption is valid if the sum of the counteracting forces is less than the force produced by the hydraulic cylinder. In this condition, it is a direct consequence of equations (2.14), and (2.17), because from the last one $\dot{x}$ has always the same sign of P_L. It is clear that this assumption works perfectly for a simple step response, but could be violated if the system has to follow a sawtooth trajectory when the speed change direction. Another critical case for this hypothesis could be when the system is oscillating, always because of the fast-changing in speed direction. We directly face this problem simulating the *full order model*, and *reduced order model*, open-loop response to a sin wave of a certain amplitude and frequency, and closed-loop response with a high amount of oscillations. The solution consists to

manage the solver type and reduce the maximum step size. Furthermore it can be added some constraint to the sign of P_A, and P_B, for the *full order model*, or P_L for the *reduced order model*. From equation (3.2) we can notice that the value of C_{qp} is not fully defined in the all the variational range of the dependent variables $-1 \leq z \leq 1$, $-P_S \leq P_L \leq P_S$. Indeed, for $P_L = \pm P_S$, if $z \neq 0$, we get an improper value for C_{qp}:

$$C_{qp}(P_S, z \neq 0) = \lim_{P_L \to \pm P_S} \frac{zK\,\mathrm{sign(z)}}{4\sqrt{\frac{1}{2}(P_S - \mathrm{sign(z)}\,P_L)}} = \lim_{h \to 0^+} \frac{zK\,\mathrm{sign(z)}}{4\sqrt{h}} = +\infty \tag{3.3}$$

That is a problem because the arithmetical mean of the variation interval of C_{qp} is also improper. Furthermore, in the experimental system without external counteracting forces is impossible to reach the maximum value of P_S due to the pressure losses on the pipes and the directional control valve. Hence, we can limit the maximum value of the load pressure to $P_{L,\mathrm{MAX}} < P_S$, and a good choice is to take $P_{L,\mathrm{MAX}} = 95\% \cdot P_S$.
After this clarification, we can analyse the parameter distributions in the new ranges. In figure 3.1 is shown the variation of C_q due to P_L, while in figure 3.2 is shown the variation of C_{qp} due to both variables z, and P_L.

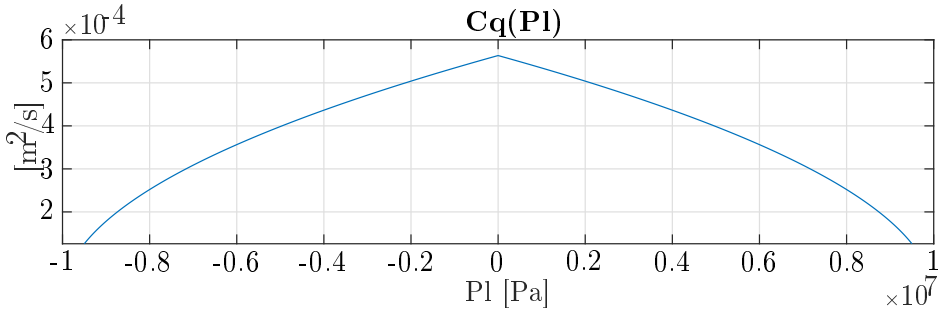


Figure 3.1: Variation of the transfer function parameter $C_q(P_L)$. The maximum of P_L is 95%of P_S

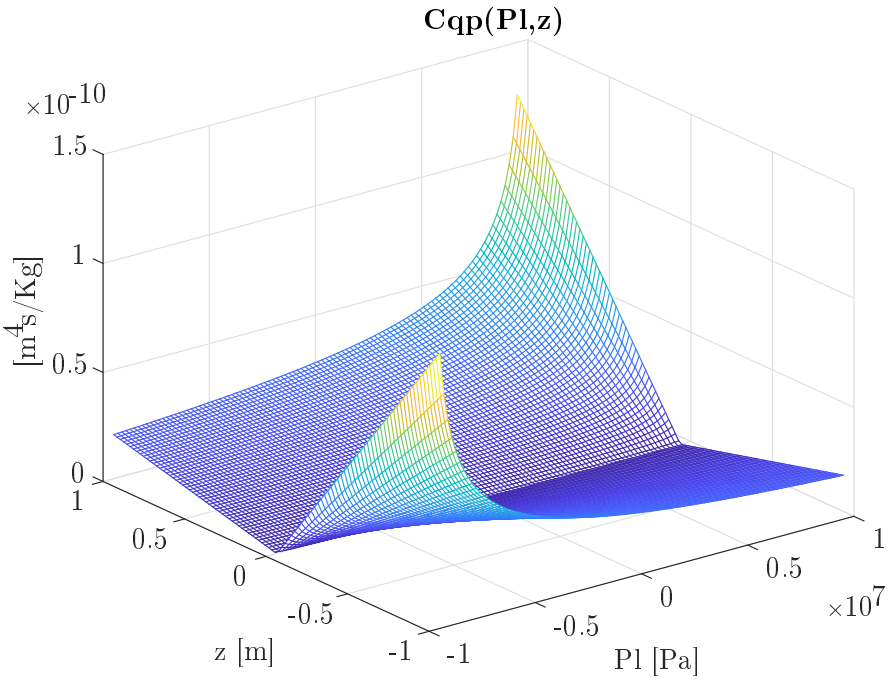


Figure 3.2: Variation of the transfer function parameter $C_{qp}(P_L, z)$. The maximum of P_L is 95% of P_S

From the pictures 3.1 it is clear that statistically, C_q assume the lowest values with less frequency than the highest ones. We expect the same for C_{qp}, because looking at the figure 3.2 we can notice that the lowest values are statistically more numerous than the highest ones. Thus, an algebraical mean of the variational range does not take into account this fact, so it is useful to evaluate the integral mean for both the parameter functions:

$$\bar{C}_q \coloneqq \frac{1}{2P_{L,\mathrm{MAX}}} \int_{-P_{L,\mathrm{MAX}}}^{P_{L,\mathrm{MAX}}} C_q(P_L)\, dP_L \tag{3.4}$$

$$\bar{C}_{qp} \coloneqq \frac{1}{z_{\mathrm{MAX}} - z_{\mathrm{min}}} \int_{z_{\mathrm{min}}}^{z_{\mathrm{MAX}}} \left[\frac{1}{2P_{L,\mathrm{MAX}}} \int_{-P_{L,\mathrm{MAX}}}^{P_{L,\mathrm{MAX}}} C_{qp}(P_L)\, dP_L \right] dz \tag{3.5}$$

Therefore, evaluating the equations (3.4), and (3.5) in the discussed parameter ranges we obtain

$$\bar{C}_q = \frac{1}{2 \cdot 0.95 P_S} \int_{-0.95P_S}^{0.95P_S} C_q(P_L)\, dP_L = 3.9101 \cdot 10^{-4} \left[\frac{m^2}{s} \right] \tag{3.6}$$

$$\bar{C}_{qp} = \frac{1}{2} \int_{-1}^{1} \left[\frac{1}{2 \cdot 0.95 P_S} \int_{-0.95P_S}^{0.95P_S} C_{qp}(P_L)\, dP_L \right] dz = 1.7391 \cdot 10^{-11} \left[\frac{m^4 s}{Kg} \right] \tag{3.7}$$

To conclude the analysis of C_q, and C_{qp}, we add the table 3.1 that show the maximum and minimum values reached.

Parameter	**Max**	**Min**	**Mean**	**$\int$-mean**
$C_q \left[\frac{m^2}{s}\right]$	$5.6322 \cdot 10^{-4}$	$1.2600 \cdot 10^{-4}$	$3.4461 \cdot 10^{-4}$	$3.9101 \cdot 10^{-4}$
$C_{qp} \left[\frac{m^4 s}{Kg}\right]$	$1.2600 \cdot 10^{-10}$	$2.0380 \cdot 10^{-13}$	$6.3102 \cdot 10^{-11}$	$1.7391 \cdot 10^{-11}$

Table 3.1: Principal statistical parameters of the transfer function uncertainties C_q and C_{qp}

3.2 $e^{-s\tau}$ experimental identification and analysis

In this section, we introduce in the control loop the time delay due to the controller distance from the system. The objective is to measure the round trip time delay τ, study the entity of this uncertainty, and obtain a model that we can use in the simulations. As we have already explained, this thesis project aims to control the hydraulic cylinder system and the directional control valve with a controller wirelessly connected to the system interface. In particular, we obtain wireless communication using a *wifi* bridge that uses a TC-IP communication protocol. This protocol type guarantees that the data packets the client (or server) send go from the latter and arrive at the server (or client) without losses and in the correct order. Because of these guarantees, the client (or server) has to ask with another data packet if the server (or client) is listening when the first has to send a data packet to the second. Then, the client sends the data and after, it has to check if the data is arrived before repeating the process for a new data packet. This process makes the communication slower than other communication protocols, for example, UDP-IP communication. Also, in this case, the communication time delay is variable due to the external disturbances that affect the communication channel. We highlight we choose the TCP-IP communication protocol because this thesis aims to design a robust control system under the condition of variable time delay, and not to achieve the best performance in the communication. Hence, with TCP-IP protocol, we avoid all the practical problems caused by a non-point to point communication. For example, using the UDP-IP communication protocol, if the server has to send some data to the client when the communication channel is established, it sends the data continuously so the client could receive the information in the wrong order, or could happen that some packets are lost. These are undesired problems from a control point of view. Therefore, it requires the implementation of corrective strategies.
After this clarification about the choice of the communication protocol, it is also clear that the *wifi* communication introduce a not negligible time delay in the control loop. To better understand what the time delay is made of, we look at the figure 3.3. The figure shows four grey blocks, the main parts of the control system. From right to left we have the *system with the encoder*, the *interface of the system*, the *wifi communication*, and the *controller*. In this case, the figure shows the typically closed-loop configuration and highlights the principal contribution to the round trip time delay. We can see that the position measure out from the linear encoder $x(t)$, through the system interface and the *wifi* bridge, goes to the controller and, it arrives with a time delay of τ_1 because of the communication channel. Then the controller from this signal and the reference computes the control signal $u(t)$. In this case, the delay due to the calculation is at least the solver step size τ_2. For arriving at the process, the control signal is also delayed by the communication time delay τ_3. To sum up, the round trip time delay τ is given by the following:

$$e^{-s\tau} = e^{-s\tau_1} \cdot e^{-s\tau_2} \cdot e^{-s\tau_3} = e^{-s(\tau_1+\tau_2+\tau_3)} \tag{3.8}$$

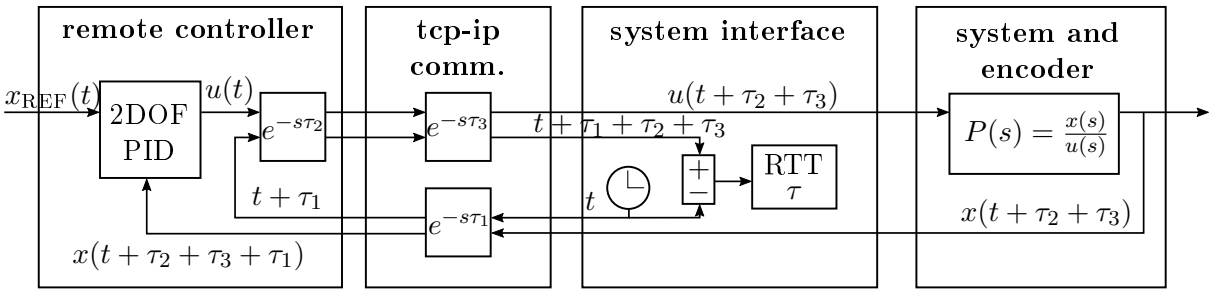
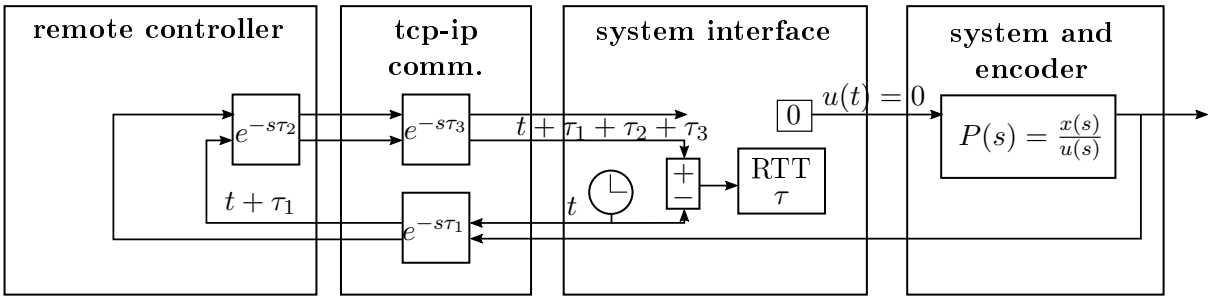
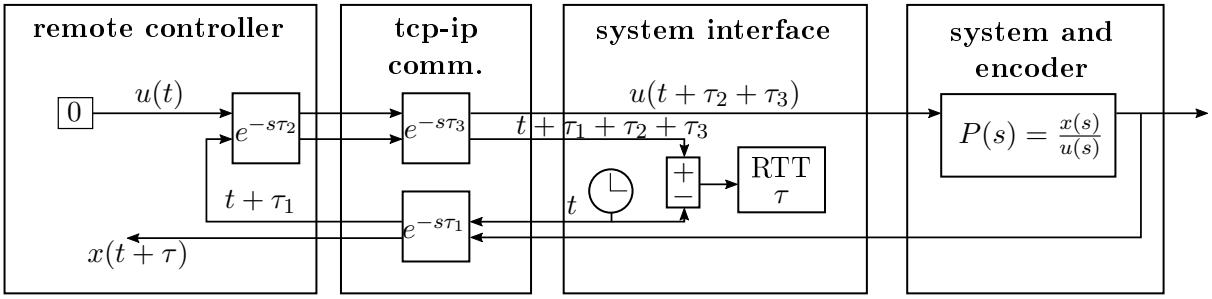


Figure 3.3: Block diagram of the round trip time delay in the control system

Now the question is: how to measure the time delay of the round trip? If we compare the measured position $x(t)$ and the control signal $u(t)$, it is almost impossible to find out the delay based on the correlation of the two signals. Indeed, the position is affected by the noise of the encoder. Therefore, we are forced to create another signal that follows the same round trip of the control loop. In this way, the comparison between the phase shift is very simple. Figure 3.3



(a) signal in the loop without control



(b) signal in half loop without control

Figure 3.4: Block diagrams of the system configurations to test the TCP-IP communication and to measure the round trip time delay.

shows the additional signal used to measure the time delay of the round trip. We see that is generated in the system interface t and arrives back to the interface shifted t_3 after the control loop travel. We designed two tests types for the preliminary measure of τ. In the first one, see figure 3.4 (a), it is sent to the hydraulic cylinder a 0 control signal (practically the power unit is switched off), and the acquired position measure (only noise) makes the hall control loop travel. We made this position travel choice to have approximately the same load in the communication channel. Furthermore, we choose to test first the system with the power unit switched off for safety. For the second test type, see figure 3.4 (b), we make an open-loop control, sending from the remote controller a pretty low control signal. The aim is to make the piston move for a long time to be able to collect enough data for making a statistical analysis.
To clarify the experiment label see table 3.2. With the test type *signal in the loop without control* we intend the first type of test discussed above, the configuration shown in figure 3.4 (a). Instead, with the test type:*signal in half-loop without control* we mean the second experiment typology discussed above and shown in figure 3.4 (b). Always from table 3.2, we see that for each test type, we design several experiments varying the physical distance in the wireless communication or simply repeating the same test more than one time. Furthermore, in the case of the *signal in half-loop without control* test, also the amplitude of the control signal is changed.
We want to focus on the analysis and interpretation of the results of the designed tests. Table 3.3 shows the main statistical parameters obtained or computed. First of all, we see that the minimum time delay possible is $\tau_{\min} = 0.01[s]$ that is the *communication sample rate*: $\tau_{\text{COMM}} = 0.01[s]$. This happens because, when the sampling rate in the communication is greater than the sampling rate in the control system, so if $\tau_{\text{COMM}} > \tau_S$, then the transmission of the signal requires a sampling process made by a holder. In this way, for the first trip, the holder adds a delay in the signal of half the sampling rate, and the same happens for the way back. Hence:

$$e^{-s\tau_{\min}} = e^{-s\tau_{1,\min}} \cdot e^{-s\tau_{2,\min}} \cdot e^{-s\tau_{3,\min}} = e^{-s\frac{1}{2}\tau_{\text{COMM}}} \cdot e^{-s\frac{1}{2}\tau_{\text{COMM}}} \cdot e^{-s\cdot 0} = e^{-s\tau_{\text{COMM}}} \tag{3.9}$$

In the relation above, we see that $\tau_{3,\min} = 0$. That means that the delay for the computational process in the controller is much little than $\frac{1}{2}\tau_{\text{COMM}}$. Thus, the computational time resides inside the step time of the older, resulting in no impact on the communication delay. Furthermore,

Table 3.2: List of the experiments made to evaluate the round trip time delay τ

NAME	TEST TYPE	WIFI DISTANCE	CONTROL SIGNAL
Test1	signal in the loop without control	$0.2[m]$	0
Test2	signal in the loop without control	$0.2[m]$	0
Test3	signal in the loop without control	$10[m]$	0
Test4	signal in the loop without control	$10[m]$	0
Test5	signal in half loop without control	$0.2[m]$	0.1
Test6	signal in half loop without control	$0.2[m]$	-0.1
Test7	signal in half loop without control	$0.2[m]$	-0.1
Test8	signal in half loop without control	$10[m]$	0.1
Test9	signal in half loop without control	$10[m]$	-0.1
Test10	signal in half loop without control	$10[m]$	0.12

from the table (3.3), we see that the variance is around one order of magnitude higher in the case of *signal in the loop without control* than in the one of the type: *signal in half-loop without control*. The difference between these two cases is the load of the communication channel: higher in the first case than in the second. Therefore, this fact reflects that a busier channel increases the probability that starvation or queue could happen. In the *signal in the loop without control* case we also see that the mean of the communication time delay increase with the distance of the wireless communication. We know that the time for the propagation of the electromagnetic waves is negligible. Hence, one reason that can explain why there is an increment of the delay correlated to the distance is the following: the data packets have to be sent again if in the communication we lose more information.

Table 3.3: Experimental results to evaluate the round trip time delay

TEST NAME	MAX $[s]$	MIN $[s]$	MEAN $[s]$	MODE $[s]$	VAR $[s]$
Test1	0.3	0.01	0.024	0.01	0.0015
Test2	0.32	0.01	0.022	0.01	0.0011
Test3	0.34	0.01	0.044	0.03	0.002
Test4	0.57	0.01	0.041	0.03	0.0016
Test5	0.23	0.01	0.039	0.03	0.000235
Test6	0.1	0.01	0.039	0.03	0.000203
Test7	0.14	0.02	0.04	0.03	0.000187
Test8	0.31	0.01	0.036	0.03	0.000479
Test9	0.2	0.01	0.029	0.03	0.000442
Test10	0.25	0.01	0.04	0.03	0.000706

We want now to analyse the statistical distribution of τ. Indeed, it is necessary to find a model that can describe from a general point of view the data collected. In the figures 3.6a, 3.6b, 3.6c, and 3.6d, are shown the distribution histograms of the data collected respectively from Tset1, Test6, Test3, and Test10.
We can see that the distributions change significantly from experiment to experiment. Hence, the choice of the appropriate distribution model to use for fitting the data is not immediate and easy. For instance, if we look at figure 3.5 we see that several probability models fit quite well the data collected in the experiment Test8. In particular, we have fitted the data with the *Weibull distribution*, *Log-normal distribution*, *Exponential distribution*, and γ *distribution*. Therefore, we have to take a step back for understanding the reasons behind the time delay that we have measured.

The choice of the probability model that best fits and represents the distribution of the round trip time (RTT) over internet communication is an open problem in the literature. As it is expose in [31], the *Round Trip Time* (RTT) depends on several factors. Some of them are not related to network performance but are determined by physical constraints. First, in *wifi* communication digital information travels in the air as an electromagnetic wave with the speed of light. Second, there is a minimum processing delay introduced by each router along the way, of the order of $50 - 250\mu s$ per-hop on average, summing up to a few milliseconds for a typical path, [31]. Third, on top of this, the presence of cross traffic along the route can cause data packets to be queued in the routers. When the traffic reaches congestion level, the queueing time becomes a very significant part of the RTT and packet loss also sets in. Therefore, in other words, network delay is mainly composed of propagation, serialization, switching/routing and queuing delay [17]. Serialization delay is the time it takes to serialize the digital data onto the physical links of the interconnecting equipment. That is how long it takes to put the bits on the wire. Propagation delay is the time it takes the signal to travel the physical distance from end to end. The propagation delay is determined by the travel time of an electromagnetic wave through the physical channel of the communication path and is independent of actual traffic on the link. Switching/routing delay is the time the router takes to switch the packet. This time is needed to analyse the packet header, check the routing table, and route the packet to the output port. This delay depends on the architecture of the route engine and the size of the routing table. Queuing delay, which is a large source of latency, is the amount of time that a packet remains buffered in a network element while it waits for transmission. Network traffic loads result in variable queuing delays. For an unloaded network, this delay is negligible. For a heavily congested network, it is usually the main delay component [17]. The observed result is a huge statistical fluctuation of the RTT [31]. Several authors attempted to find a statistical model for describing the RTT variability. The most common utilised models are: γ-*distribution*, *Weibull distribution*, *Log-normal distribution*, *Exponential distribution*. For instance, in [17], they found that the model that best fit the end-to-end delay, which comes directly from the measure of the RTT, is in most cases described by a γ-*distribution*. Nevertheless, these measures for some specific paths are better fitted with a *Weibull distribution*, and a *Log-normal distribution*. In [9] the main finding of the research is that the RTT can be well approximated by a *truncated normal distribution*. In this case, the study is made specifically for TCP/IP Networks, but the result differs from the literature trend, in fact, in [39] we can read that the *Exponential distribution* describes the data on network delay better, than *truncated normal distribution*. Also in [41] the *Exponential distribution* seems to be the one that describes the RTT and equivalently the end-to-end delay. Nevertheless, also from the previously cited book: "Evolution and Structure of the Internet" [31], it is clear that exist a long tail in the statistical distribution of the RTT. Possible reasons for the large tail include a First-Come-First-Served (FCFS) scheduling and a systematic batching of traffic [22]. Because of that, practically γ-*distribution* could fit well a curve with this shape. We found many works that supports γ-*distribution* model for fitting the RTT over internet communication, see [20], [1], [22], [6], [21], and [30]. For instance, in [20] the authors say that although the Gamma fit worked well for the low-frequency delay component, this observation cannot be easily generalized. distributions. In [1] was found that the best fits for almost 90% of the empirical distributions are two standard distributions:γ-*distribution* and *Logistic distribution*. Also in [6] it is found that classification of numerous histograms of the end-to-end delay of a fixed path demonstrates that about 84% are *typical* histograms possessing a γ-*like shape* with a sub-exponential (or heavy) tail. In [30] it is discussed the design approach of a mechatronic system over a communication network, and they adopted for the end-to-end delay model a γ-*distribution*. The same is done in [21], where the authors pointed out that γ-*distributed* delays with a gap can also be encountered in the problem of controlling objects over communication networks. After these considerations, we guess that also in our case γ-*distribution* could be the appropriate model for fitting the data acquired. If we look again at figure 3.5 we see that γ-*distribution* actually fits well the histogram. In the figures 3.6a, 3.6c is reported the gamma-fit of the data obtained respectively in experiments Test1, and Test6. While, in figures 3.6b, and 3.6d, is reported the gamma-fit for the experiments Test3, and Test10.

To sum up the results of the fitting in figure 3.7 are shown all the γ-*distribution* obtained before together. We see that in the case of the first test type *signal in the loop without control*, so in the case of Test1, and Test6, we have a greater mean value than with the other test type *signal in half-loop without control* with the same distance condition. Instead, generally, we can see that at the same distance we have almost the same shape. For instance, Test6, and Test10 have a similar shape but the first one has a greater mean value. Research work from Amarnath Mukherjee, see [22] can clarify these observations. Indeed, for all the network paths he has studied, he says that the distribution of delay is approximately a shifted Gamma with a large tail even for networks with low congestion levels. Furthermore, he says that the shape and scale parameters of the empirical γ-*distribution* vary with load and network segment. So as expected, because from *signal in half-loop without control* to *signal in the loop without control* test type we think that the communication load increase, due to the difference in the signal that the remote controller sends to the system interface, the mean value of the γ-*distribution* increase. Regarding the difference in the shape that appears changing the distance, we suppose that with a greater distance, the probability that other *wifi* signals disturb the communication increase significantly. Hence, the *Central limit theorem* suggests that the distribution shape should be more like a shifted Gaussian. The remains data collected with the experiments: *Test2, Test4, Test5, Test7*, and *Test9* are reported in the appendix B. In these last figures, the models used to fit the distributions highlight that the γ-*distribution* best fits all the cases and, therefore, that supports the considerations made before.

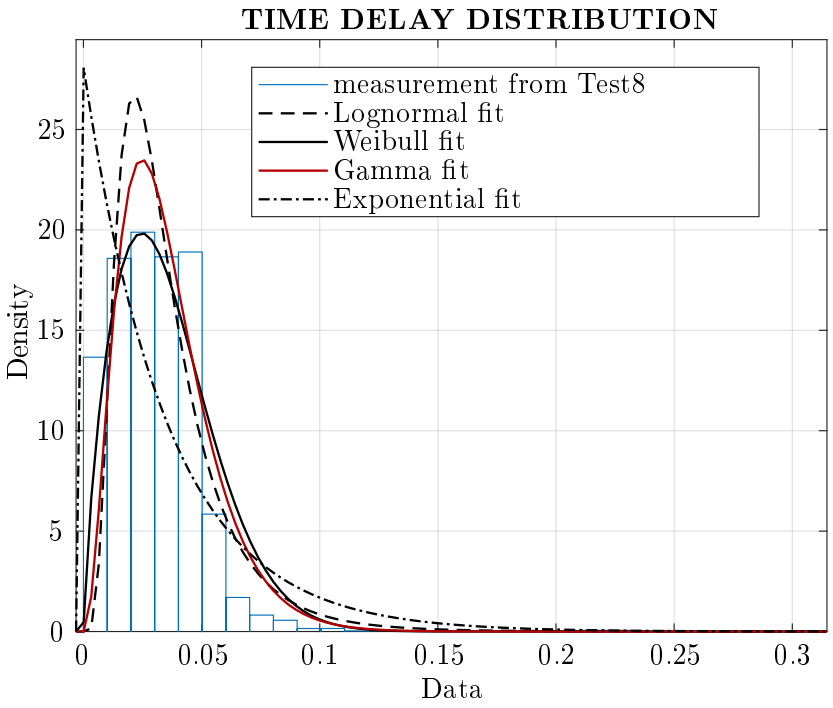
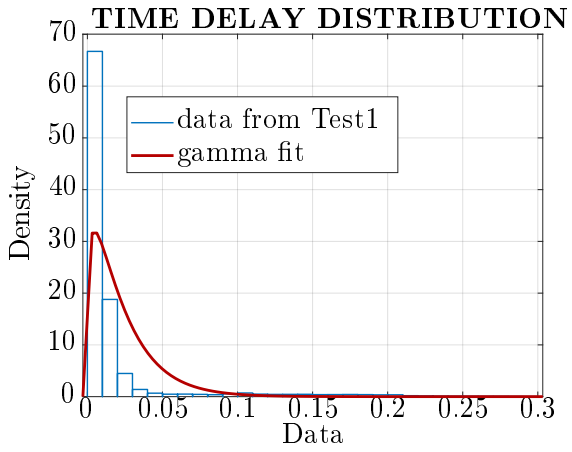
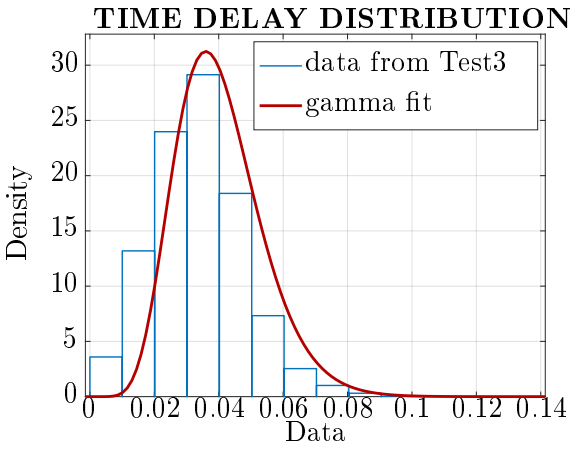


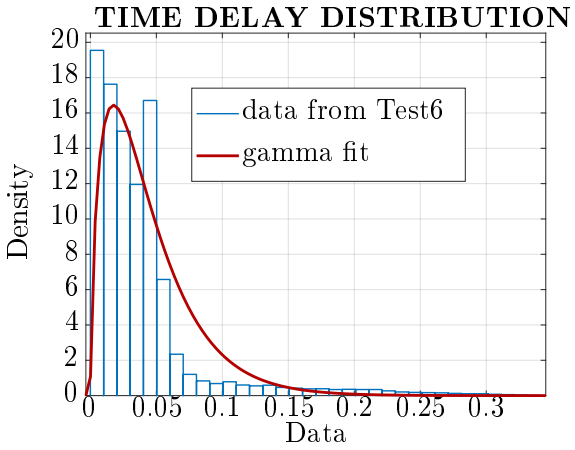
Figure 3.5: Various distribution models used to fit the data collected in Test8



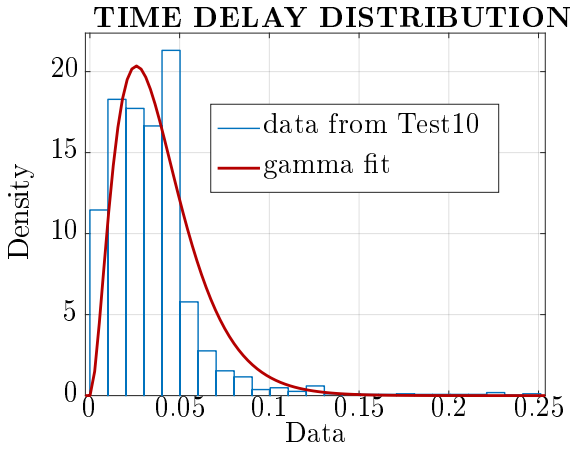
(a) γ-*distribution* fit of τ distribution in Test1



(b) γ-*distribution* fit of τ distribution in Test3



(c) γ-*distribution* fit of τ distribution in Test6



(d) γ-*distribution* fit of τ distribution in Test10

Figure 3.6: γ-*distributed* fits of τ for some experimental data. Data and fits of the other tests are show in appendix B.

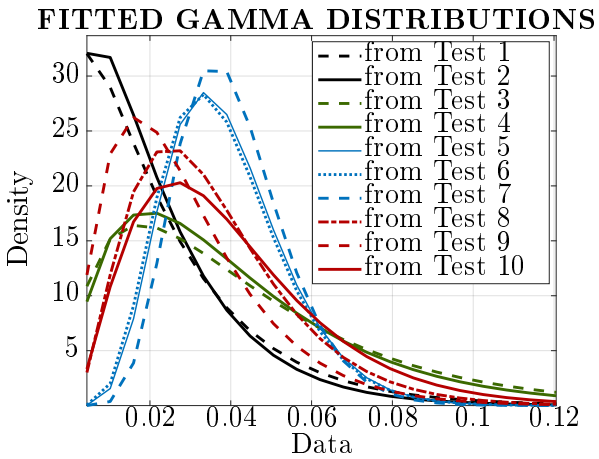


Figure 3.7: γ-*distributions* of the time delay τ computed in the ten tests

Table 3.4: γ-*distributions* fit parameters

TEST NAME	**a**	**b**
Test1	1.23797	0.019232
Test2	1.43424	0.015358
Test3	1.65788	0.026531
Test4	1.87734	0.022056
Test5	6.87971	0.005693
Test6	6.54022	0.005891
Test7	9.10988	0.004438
Test8	3.27649	0.010844
Test9	2.34026	0.012349
Test10	2.91554	0.013567

3.3 Gain uncertainty and nominal model

In section 2.3 we have seen that, after the linearization of the *reduced order model*, the process can be expressed with the transfer function of equation (2.29). Furthermore, in the previous section 3.2 the time delay τ of the *Round Trip Time* is introduced in the analysis of the uncertainty. Hence, the overall process seen by the controller is given by the following equation:

$$P_\tau(s) \coloneqq \frac{x(s)}{u(s)} = e^{-s\tau}P(s) = \frac{e^{-s\tau}\frac{\hat{C}_q\bar{A}}{\sigma_{LIN}\hat{C}_{qp}+\bar{A}^2}}{s\left[s^2\frac{mV_t}{4E(\sigma_{LIN}\hat{C}_{qp}+\bar{A}^2)} + s\frac{\hat{C}_{qp}m+\frac{\sigma_{LIN}V_t}{4E}}{\sigma_{LIN}\hat{C}_{qp}+\bar{A}^2} + 1\right]} \tag{3.10}$$

Possible choices for the nominal model could be the followings:

1. $\hat{C}_q$, $\hat{C}_{qp}$ are the arithmetical mean values, so they are the middle of the respectively interval variation. $\tau_{\text{nom}} = 0$ The time delay is not considered.
2. $\hat{C}_q$, $\hat{C}_{qp}$ are the arithmetical mean values, so they are the middle of the respectively interval variation. $\tau_{\text{nom}} = \frac{\tau_{\text{MAX}}-\tau_{\text{min}}}{2}$, hence, also the time delay is the arithmetical mean.
3. $\hat{C}_q$, $\hat{C}_{qp}$ are the arithmetical mean values, so they are the middle of the respectively interval variation. $\tau_{\text{nom}} = \bar{\tau}_\gamma$, thus, the time delay is the statistical mean obtained from the γ-*distribution* of the acquired data.
4. $\hat{C}_q$, $\hat{C}_{qp}$ are the integral mean values, so $C_{q,\text{nom}} = \bar{C}_q, C_{qp,\text{nom}} = \bar{C}_{qp}$. $\tau_{\text{nom}} = \bar{\tau}_\gamma$, thus, also the time delay is the statistical mean obtained from the γ-*distribution* of the acquired data.

The reasons of those possible choices are reported below respectively in the same order:

1. The arithmetical mean of C_q, and C_{qp} allow minimizing the variational range of these parameters. With $\tau_{\text{nom}} = 0$ we are simplifying significantly the process transfer function, so the design process of the controller will be easier, see next chapter.
2. The reason of point 1, plus the fact that also for $\tau_{\text{nom}} = \frac{\tau_{\text{MAX}}-\tau_{\text{min}}}{2}$ we are minimizing the variational range of the time delay uncertainty.
3. The reason of point 1, but considering $\tau_{\text{nom}} = \bar{\tau}_\gamma$ the nominal model has statistically the least error for the time delay, that, as we will see in section 5.3, is the most problematic parameter.
4. If all the parameters has the statistical mean value, so if $C_{q,\text{nom}} = \bar{C}_q, C_{qp,\text{nom}} = \bar{C}_{qp}$, and $\tau_{\text{nom}} = \bar{\tau}_\gamma$, then the nominal model will represent more faithfully the real system.

Choosing the first option, we will indirectly consider a system that is very close to the base case scenario in terms of stability. This is due to the zero time delay in fact, as it is known and as we will see in section 5.3, it is a crucial factor. Designing the controller with the nominal values of the parameters that are the arithmetical mean we will minimize the uncertainty size, so, seeing section 5.3 it will be easier to respect the robust stability conditions. On the other hand, the system is more often operating in a different condition than the nominal one. Furthermore, using a higher time delay as the nominal value the controller design will be more conservative. The result is a control system with less performance. All these observations lead us to choose the fourth option. Indeed, using the statistical mean values in the nominal model guarantee to achieve both the best representation of the real system and good performance statistically more often. Hence we have:

$$C_{q,\text{nom}} = \bar{C}_q = 3.9110 \cdot 10^{-4}\left[\frac{m^2}{s}\right] \tag{3.11}$$

$$C_{qp,\text{nom}} = \bar{C}_{qp} = 1.7391 \cdot 10^{-11}\left[\frac{m^4 s}{Kg}\right] \tag{3.12}$$

$$\tau_{\text{nom}} = \bar{\tau}_\gamma = 0.03[s] \tag{3.13}$$

So the equation (3.10) becomes:

$$P_{\text{nom}}(s) = \frac{e^{-s\tau_{\text{nom}}} \frac{C_{q,\text{nom}}\bar{A}}{\sigma_{LIN}C_{qp,\text{nom}}+\bar{A}^2}}{s\left[s^2\frac{mV_t}{4E(\sigma_{LIN}C_{qp,\text{nom}}+\bar{A}^2)} + s\frac{C_{qp,\text{nom}}m+\frac{\sigma_{LIN}V_t}{4E}}{\sigma_{LIN}C_{qp,\text{nom}}+\bar{A}^2} + 1\right]} = e^{-0.03s}\frac{8.255\cdot 10^5}{s\left[s^2+948s+2.219\cdot 10^6\right]} \tag{3.14}$$

Now that the nominal model is defined, we want analyse the variations of the system parameters in term of gain k, angular natural frequency ω_n, and damping factor ξ. Thus, from equation (3.10) we see the following relations:

$$k(\hat{C}_q, \hat{C}_{qp}) = \frac{\hat{C}_q\bar{A}}{\sigma_{LIN}\hat{C}_{qp}+\bar{A}^2} \tag{3.15}$$

$$\omega_n(\hat{C}_{qp}) = \sqrt{\frac{4E(\sigma_{LIN}\hat{C}_{qp}+\bar{A}^2)}{mV_t}} \tag{3.16}$$

$$\xi(\hat{C}_{qp}) = \frac{\hat{C}_{qp}m+\frac{\sigma_{LIN}V_t}{4E}}{\sigma_{LIN}\hat{C}_{qp}+\bar{A}^2}\cdot\frac{\omega_n}{2} = \left(\hat{C}_{qp}m+\frac{\sigma_{LIN}V_t}{4E}\right)\sqrt{\frac{E}{mV_t(\sigma_{LIN}\hat{C}_{qp}+\bar{A}^2)}} \tag{3.17}$$

Table 3.5: Values of $k(C_q, C_{qp})$ gain of the process transfer function related to different values of C_{qp} and C_q.

$(\hat{\mathbf{C}}_{\mathbf{qp}}, \hat{\mathbf{C}}_{\mathbf{q}})$	$\mathbf{k}(\hat{\mathbf{C}}_{\mathbf{qp}}, \hat{\mathbf{C}}_{\mathbf{q}})$
$\bar{\mathbf{C}}_{\mathbf{qp}}, \bar{\mathbf{C}}_{\mathbf{q}}$	0.372
$\mathbf{C}_{\mathbf{qp,MAX}}, \mathbf{C}_{\mathbf{q,MAX}}$	0.476
$\mathbf{C}_{\mathbf{qp,MAX}}, \mathbf{C}_{\mathbf{q,min}}$	0.1065
$\mathbf{C}_{\mathbf{qp,min}}, \mathbf{C}_{\mathbf{q,MAX}}$	0.5467
$\mathbf{C}_{\mathbf{qp,min}}, \mathbf{C}_{\mathbf{q,min}}$	0.1223

From equation (3.15) we see that k is a function of C_{qp}, and C_q. Hence, the extreme values of the variational range of k could happens for any pairs of the possible combination of $C_{qp,\text{MAX}}$, $C_{q,\text{MAX}}$, $C_{qp,\text{min}}$, $C_{q,\text{min}}$. Therefore, we can evaluate the values of all combinations to be sure to catch the maximum variation possible from the nominal value, that is the integral mean value. The results are shown in table 3.5. Thus, now it is possible to evaluate the maximum relative error that occurs if we consider the nominal value: $\hat{C}_{qp} = \bar{C}_{qp} = C_{qp,\text{nom}}$. The following equation gives the calculation:

$$\text{MAX}\Big[\text{relative error}\big[k(C_{qp}, C_q)\big]\Big] = \frac{k(C_{qp,\text{MAX}}, C_{q,\text{min}})}{k(\bar{C}_{qp}, \bar{C}_q)} = 0.7137 = 71.37\% \tag{3.18}$$

Furthermore, from the equations (3.16), (3.17), we see that ω_n, and ξ are functions of C_{qp}. Thus, we can easily compute the range of variation, simply substituting the minimum and maximum value of C_{qp} reported in table 3.1. The results are shown in table 3.6. In the same table is also pointed out the maximum relative error computed in the same way as equation (3.18) for $C_{qp,\text{MAX}}$.

Comparing the result of equation (3.18), and last right column of table 3.6, we see that k has the highest variation from the nominal value. Indeed, k relative error is almost three times bigger than ξ relative error and almost twelve times bigger than ω_n relative error. This observation is really important because we are allowed to consider only the gain uncertainty $k - k_{\text{nom}}$ as the first approximation. Hence, in this way, we can drastically simplify the uncertainty model and the

Table 3.6: Variation of ξ and ω_n process transfer function parameters related to C_{qp} different values.

$\hat{\mathbf{C}}_{\mathbf{qp}} =$	$\bar{\mathbf{C}}_{\mathbf{qp}}$	$\mathbf{C}_{\mathbf{qp,MAX}}$	$\mathbf{C}_{\mathbf{qp,MIN}}$	**MAX(relative error)**
$\xi(\hat{C}_{qp})$	0.3182	0.3981	0.3048	0.2511
$\omega_n(\hat{C}_{qp})$	$1.49 \cdot 10^3 \left[\frac{rad}{s}\right]$	$1.58 \cdot 10^3 \left[\frac{rad}{s}\right]$	$1.47 \cdot 10^3 \left[\frac{rad}{s}\right]$	0.061

robust analysis, as we will see in chapter 5. We can use *Matlab Robust Control Toolbox* software, see [5], to model the equation (3.10) with all the system uncertainty. The aim is to obtain a bode plot of the uncertain system with a random variation of the uncertainties. Because it is not possible to represent the pure delay uncertainty $e^{-s\tau}$ in such a model, we have to approximate this with the so-called *Padè approximation*, see [14]. We choose a tenth Padè order to guarantee that the approximation is good, at least for a complete phase shift of $2\pi = 360$ for all possible τ. Therefore, in figure 3.8 is shown the *Bode* plot of the uncertainty system obtained as just explained, and in red, the nominal model pointed out in equation (3.14).

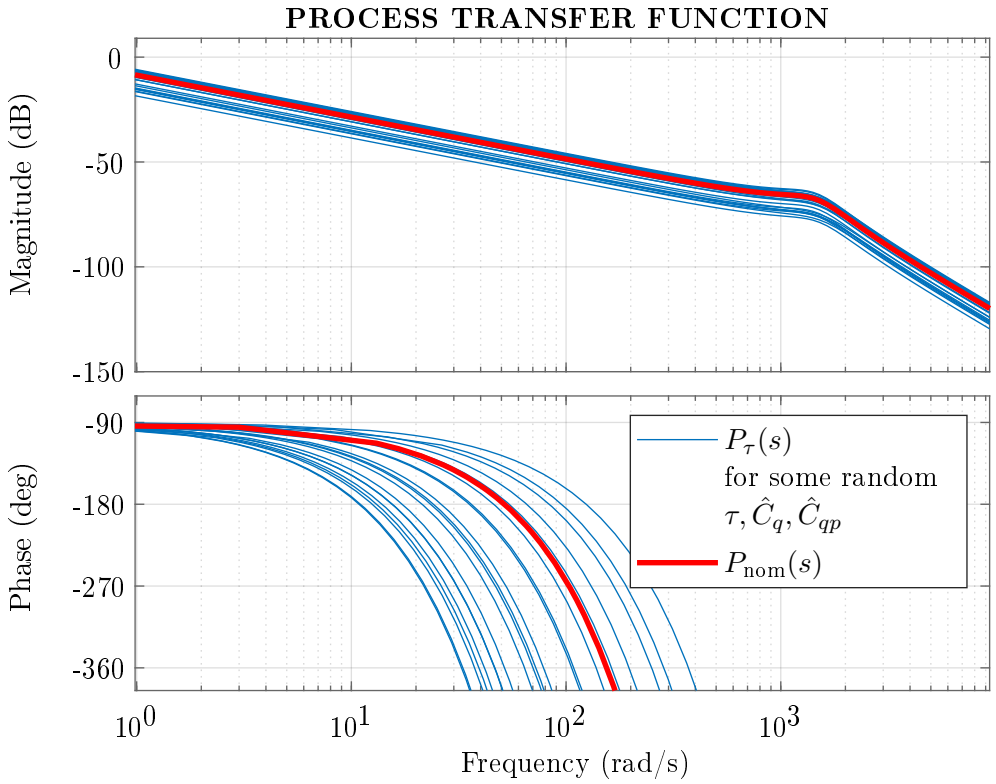


Figure 3.8: Bode plot of the uncertainty model and in *red* the nominal model of the process transfer function.

From figure 3.8 it is clear that the natural angular frequency ω_n variation, and the damping coefficient ξ variation, are not impacting significantly the uncertainty model. Furthermore, ξ applies changes around ω_n, that is much higher than the critical frequency of this system, see chapter 4. This observation confirms the discussion based on the results of tables 3.5, 3.6, and equation (3.18).

Chapter 4

Optimization procedure for PID control design

The previous chapters finally pointed out the nominal model of the process transfer function see equation (3.14). Therefore, this is the model that we use for the controller design. We have already anticipated in the introduction 1 that designing the controller for only the nominal system has to be robust enough to guarantee at least the stability for all the possible variations of the parameters. Hence, the closed-loop control system has to be *Robust stable*, see chapter 5 for further information. At the same time, we want to achieve the best performance possible. In other words, we want to design a controller that has a high level of robustness and maximize performance. Furthermore, we want to keep the control structure simple, so we choose to use a PID controller with two degrees of freedom (2DOF).
To solve this problem, first of all, we define a way to measure the robustness of the controller. Secondly, we implement an optimization algorithm that catches the solution that gives the best performance. The study follows the work of Åström and authors in [44], [43], [4], [24], [25], and [2].
Thus, in this chapter, we will see as first the mathematical formulation of the optimization problem considering the stability constraint. Secondly, the algorithms proposed in the literature are presented showing the reasons for the final algorithm used in this work. Thirdly, we will analyze the results using the classical method of *root locus diagram*. Finally, we will discuss and design the full-control structure.

4.1 Optimisation problem

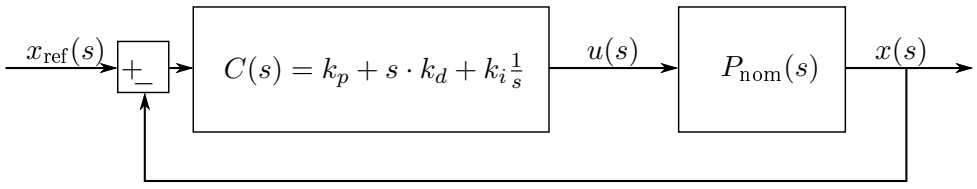


Figure 4.1: Simplified block diagram of the control system structure

To be clear, we want to remark some well-known definitions in control theory before mathematically formulating the optimization problem. We use the basic structure of the control system shown in figure 4.1, which represents the typical feedback structure. We use the transfer function $C(s)$ for referring to the ideal PID controller:

$$C(s) = k_p + \frac{k_i}{s} + sk_d \tag{4.1}$$

Furthermore, we use the notation $L(s)$ for the *Open Loop Transfer Function*, $S(s)$ sor the *Sensitivity Function*, and $T(s)$ for the *Complementary Sensitivity Function*. These transfer functions

are defined as follows:

$$L(s) = C(s) \cdot P_{\text{nom}}(s) \tag{4.2}$$

$$S(s) = \frac{1}{1 + L(s)} = \frac{1}{1 + CP_{\text{nom}}(s)} \tag{4.3}$$

$$T(s) = \frac{CP_{\text{nom}}(s)}{1 + CP_{\text{nom}}(s)} = 1 - S(s) \tag{4.4}$$

Another important definition is the *Infinite Norm of the Sensitivity Function* M_s, that is the maximum absolute value of $S(i\omega)$ over all possible ω:

$$M_s \coloneqq \max_{\omega} |S(i\omega)| = \max_{\omega} \left| \frac{1}{1 + P_{\text{nom}} C(i\omega)} \right| \tag{4.5}$$

M_s is strongly important because $1/M_s$ is the distance of the open loop function $L(s)$ has form $(-1, i0)$ in the *Nyquist plot*. Indeed, the *Nyquist Stability Criterion* pointed out that, under appropriate hypothesis (see [44]), the unstable poles p_T of the close control loop transfer function $T(s)$ are equal to the unstable poles P_L of the open loop transfer function minus the numbers N of the anticlockwise lap around the point $(-1, i0)$ in the complex plane.

$$p_T = p_L - N \tag{4.6}$$

Hence, to achieve the stability condition $p_T = 0$, N has to be strictly 0, because in our case $p_L = 0$. For that reason, the distance measure of $L(s)$ from $(-1, i0)$ is also a robustness measurement of the control system to process variations. In control theory, there is the *Gain margin*, and the *Phase margin* that are usually used as specifications in the control design problem to guarantee a certain amount of robustness, like in the *Bode Design Method*, see [44], and [44]. We want to highlight these last two parameters are partially a way to measure the distance of $L(s)$ because they refer to the distance in the real axis and the angular distance when $|L(i\omega)| = 1$ respectively. Thus, for such a particular small variation of the process parameters, could happen $L(i\omega)$ encircles $(-1, i0)$, even though the Gain Margin and, or the Phase Margin are high. For this reason, we use M_s as the measure of the system robustness, see [16].
Therefore, the formulation of the *Robust Constraint* is the following:

$$f(k_p, k_i, k_d, \omega) \geq r^2 \tag{4.7}$$

Where $f(k_p, k_i, k_d, \omega)$ is the computed distance of $L(i\omega)$ from $(-1, i0)$, that is a function of the PID coefficients k_p, k_i, k_d, and of course the angular frequency ω. Hence, we can rewrite $f(k_p, k_i, k_d, \omega)$ as

$$f(k_p, k_i, k_d, \omega) = \left| L(i\omega) + 1 \right|^2 = \left| \left[k_p + i(k_d\omega - k_i/\omega) \right] P_{\text{nom}}(i\omega) + 1 \right|^2 \tag{4.8}$$

In the equation (4.7) r is the minimum distance that we want to guarantee in the design, so it is the radius of the circle around $(-1, i0)$ where $L(i\omega)$ can not enter. Basically, we chose $r = 1/M_s$, so the minimum distance is the maximum value of the *Sensitivity Function* allowed.
Out from literature, see [43], [4], [25], [2], and explicitly said in [24], we see that typical choices of M_s are in the range of $1 \leq M_s \leq 2$. For instance, in [38] they typically select $M_s = 2$ as robustness margin. $M_s = 1$ is for the maximum robust condition, and $M_s = 2$ is for more aggressive system (less robustness). Therefore, because we want to achieve the stability for a great variation of the parameters, in particular for τ that is not yet upper-bounded, we are looking at the small value of M_s. Furthermore, the condition $M_s = 1$ is too restrictive for our control system, so a good choice is

$$M_s = 1.1 \Longrightarrow r = \frac{1}{M_s} = 0.9\overline{09} \tag{4.9}$$

To analyze in detail the *Robust Constraint* condition, we have to make some more notation definitions. In particular, for the process transfer function expressing both with exponential and algebraic forms:

$$P_{\text{nom}} = \rho(\omega) e^{\varphi(\omega)} = \alpha(\omega) + i\beta(\omega) \tag{4.10}$$

Where in the definition (4.10) the last substitution is possible because obvious meaning of the terms: $\alpha(\omega) = \rho(\omega)\cos\big(\varphi(\omega)\big)$, and $\beta(\omega) = \rho(\omega)\sin\big(\varphi(\omega)\big)$.
Using the equation (4.10), and (4.8) we can make some calculations on equation (4.7)

$$\begin{aligned} f(k_p, k_i, k_d, \omega) &= \big|L(i\omega) + 1\big|^2 = \Big|\big[k_p + i(k_d\omega - k_i/\omega)\big]\big[\alpha(\omega) + i\beta(\omega)\big] + 1\Big|^2 \\ &= \Big|\alpha k_p + \beta(k_d\omega - k_i/\omega) + i\big[\beta k + \alpha(k_d\omega - k_i/\omega)\big] + 1\Big|^2 \\ &= (\rho k_p)^2 + 2\alpha k_p + \big[\rho(k_d\omega - k_i/\omega)\big]^2 - 2\beta(k_d\omega - k_i\omega) \\ &= \rho^2\left(k_p + \frac{\alpha}{\rho^2}\right)^2 + \frac{\rho^2}{\omega^2}\left(k_i + \frac{\omega\beta}{\rho^2} - k_d\omega^2\right)^2 \geq r^2 \end{aligned} \tag{4.11}$$

Hence, finally the *Robust Constraint* becomes:

$$\left(\frac{\rho(\omega)}{r}\right)^2\left(k_p + \frac{\alpha(\omega)}{\rho(\omega)^2}\right)^2 + \left(\frac{\rho(\omega)}{\omega r}\right)^2\left(k_i + \frac{\omega\beta(\omega)}{\rho(\omega)^2} - \omega^2 k_d\right)^2 \geq 1 \tag{4.12}$$

From expression (4.12) we can see that the *Robust Constraint* has a nice interpretation. For fixed ω and k_d it represents the exterior of an ellipse in the $k_p - k_i$ plane with the following parameters:

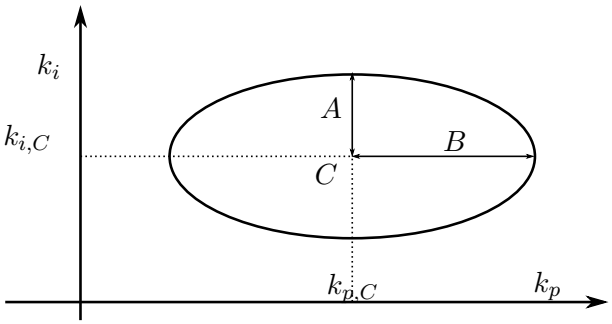


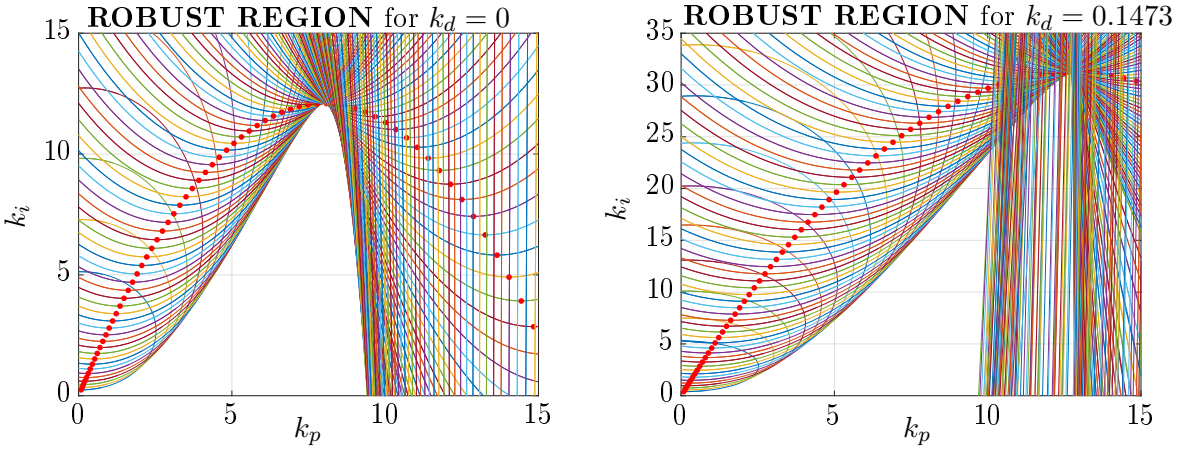
Figure 4.2: *Robust Constraint* becomes the external area of an ellipse for a fixed ω, and k_d

$$\begin{aligned} k_{p,C} &= -\frac{\alpha(\omega)}{\rho(\omega)^2} \\ k_{i,C} &= -\frac{\omega\beta(\omega)}{\rho(\omega)^2} + \omega^2 k_d \\ A &= \frac{\omega r}{\rho(\omega)} \\ B &= \frac{r}{\rho(\omega)} \end{aligned} \tag{4.13}$$

Where in equation (4.13) the coordinates of the center are $C = (k_{p,C}, k_{i,C})$. A is the vertical semiaxis, so it is the vertical distance from the center C to the lower vertex V_D, or the upper vertex V_U. B is the horizontal semiaxes and, in the same way, is the distance from the center C to the right vertex V_R, or the left one V_L. Figure 4.2 clarifies all the notations we have adopted. Varying ω and maintaining a fixed value of k_d, we can see from the equations (4.13) that all the parameters change. The result is the shifting of the centre of the ellipse C and the rescaling of its dimensions, see 4.4. It is worth highlighting the ellipse displacement due to ω variation, always with a fixed k_d, creates an envelope. Thus, the envelope fulfils these equations:

$$\begin{aligned} f(k_p, k_i, k_d, \omega) &= r^2 \\ \frac{\partial f}{\partial \omega}(k_p, k_i, k_d, \omega) &= 0 \end{aligned} \tag{4.14}$$

The envelope of all ellipses obtained for a fixed k_d could be *Smooth*, or *with Corner*. In the first case, see figure 4.3a, we see that the robust region, that goes from the k_p axes to the envelope, does not have corners. In the second case, see figure 4.3b, the robust region has a corner. Finally if we change the value of k_d the shape of the envelope changes. In figure 4.4 it is shown the different shape of the *Robust Region* caused by the changing of the envelope for a fixed value of k_d that grows from $k_d = 0$ to $k_d = 0.1473$. It is clear that the *Robust Region* does not exist for around $k_d \geq 0.1473$.
Now, what is the *Robust Region* that comes out from the *Robust Constrain* (4.12) is clear. Also, we have explained how its shape changes due to parameters variation. Then, we can define the optimal problem. In particular, the primary design goal is to achieve good rejection of load

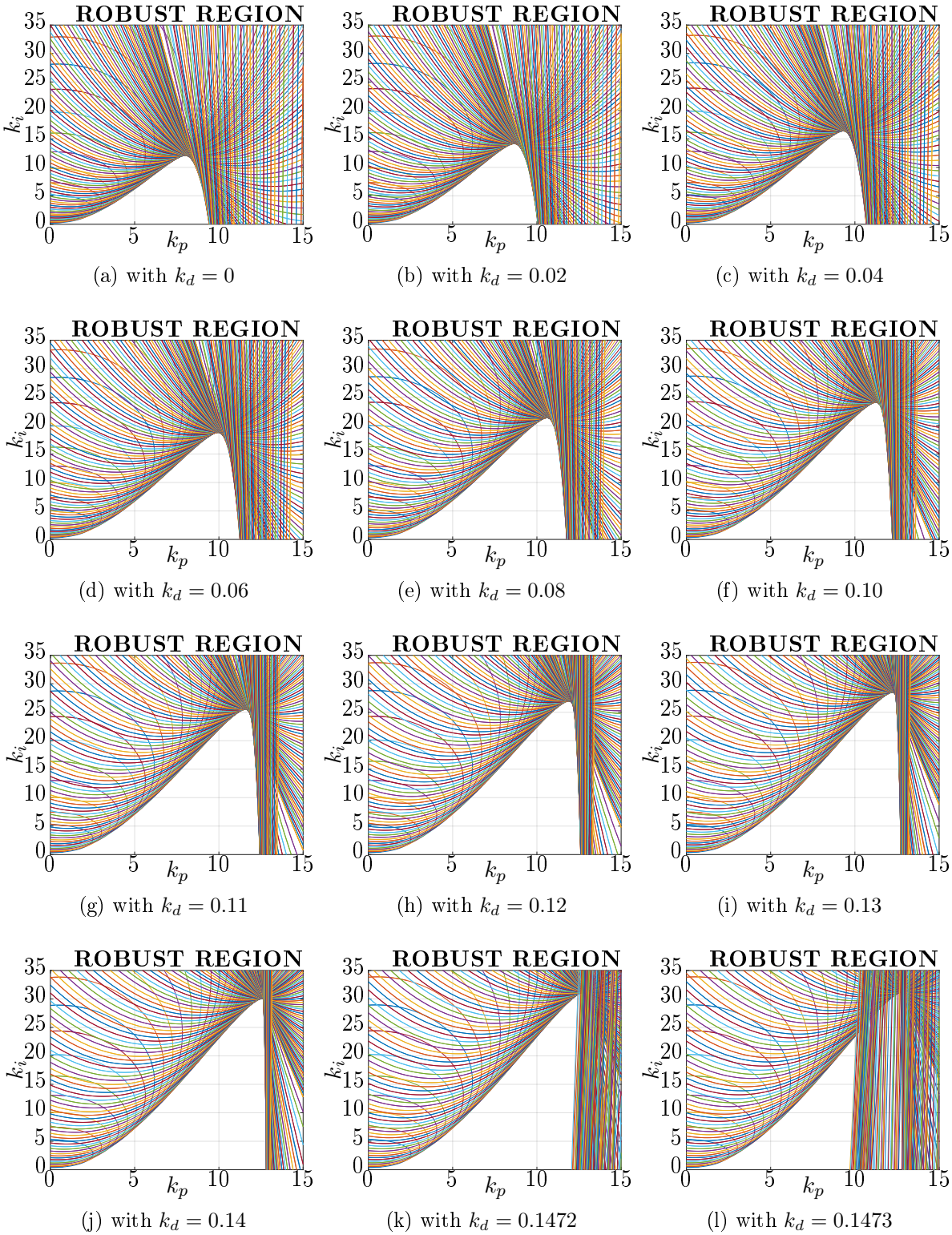


(a) *Smooth Envelope.* (b) *Envelope with Corner.*

Figure 4.3: *Robust Constraint* with $k_d = 0$ and $k_d = 0.1473$. With red dots are marked the lower vertex of the ellipses: V_D

disturbances where no detailed assumptions are made about the load disturbances except that they are low frequency. By maximizing the integral gain k_i, the effect of the load disturbance on the output x is minimized, see [24]. Indeed, in Åström paper [4] it is shown that maximizing the integral gain k_i is equivalent to minimizing the integral error (IE) for a step-change in the load disturbance. Hence, we remark the *optimal problem definition*:

Definition 4.1.1 (Optimal Design Problem)**.** Maximize the integral gain k_i of the PID controller respecting the *Robust Constraint* defined in (4.12) $\forall\omega|0 \leq \omega \leq \infty$



(a) with $k_d = 0$ (b) with $k_d = 0.02$ (c) with $k_d = 0.04$

(d) with $k_d = 0.06$ (e) with $k_d = 0.08$ (f) with $k_d = 0.10$

(g) with $k_d = 0.11$ (h) with $k_d = 0.12$ (i) with $k_d = 0.13$

(j) with $k_d = 0.14$ (k) with $k_d = 0.1472$ (l) with $k_d = 0.1473$

Figure 4.4: *Robust Region* for the process $P_{\text{nom}}(i\omega)$ at different value of k_d parameter of the PID controller

4.2 Selection of solutions approach

In this section, we explore how to solve the *Optimal Design Problem*, at the beginning from a generic point of view, and then intercalating into the specific problem for the process $P_{\text{nom}}(s)$. This way to expose the solution is useful for understanding the problem of our particular process and the choice made to arrive at the solution.
Looking again at the figure 4.3a, or 4.3b, we see that the solution of the problem is the point at the top of the *Robust Region* for a fixed k_d. In the case of *Sooth Envelope*, this point corresponds to the maximum of the function that represents the lower vertex of the ellipse V_D in the plane $k_p - k_i$. We can compute the locus of the lower vertical vertex simply using the relations in (4.13):

$$\begin{aligned} k_{p,V_D}(\omega) &= k_{p,C} = -\frac{\alpha(\omega)}{\rho(\omega)^2} = -\frac{1}{\rho(\omega)}\cos\Big(\varphi(\omega)\Big) \\ k_{i,V_D}(\omega) &= k_{p,C} - A = -\frac{\omega\beta(\omega)}{\rho(\omega)^2} + \omega^2 k_d - \frac{\omega r}{\rho(\omega)} = -\frac{\omega}{\rho(\omega)}\Big[r + \sin\Big(\varphi(\omega)\Big)\Big] + \omega^2 k_d \end{aligned} \tag{4.15}$$

To obtain the point in which occurs the maximum k_i we can differentiate $k_{i,V_D}(\omega)$ on the angular frequency ω:

$$\begin{aligned} \frac{\mathrm{d}k_{i,V_D}(\omega)}{\mathrm{d}\omega} &= \frac{\mathrm{d}}{\mathrm{d}\omega}\left[\frac{\omega\big(r+\sin(\varphi)\big)}{\rho}\right] + 2\omega k_d \\ &= \big(r+\sin(\varphi)\big)\left[\frac{\omega\dot{\rho}}{\rho^2} - \frac{1}{\rho}\right] - \frac{\omega\dot{\varphi}\cos(\varphi)}{\rho} + 2\omega k_d = 0 \end{aligned} \tag{4.16}$$

Where it is used this notation: $\dot{\rho} = \frac{\mathrm{d}\rho}{\mathrm{d}\omega}$, and $\dot{\varphi} = \frac{\mathrm{d}\varphi}{\mathrm{d}\omega}$. Hence, the equation (4.16) becomes:

$$h_{V_D}(\omega, k_d) = \frac{\mathrm{d}k_{i,V_D}(\omega)}{\mathrm{d}\omega}\frac{\rho}{\omega} = \big(r+\sin(\varphi)\big)\left[\frac{\dot{\rho}}{\rho} - \frac{1}{\omega}\right] - \dot{\varphi}\cos(\varphi) + 2\rho k_d = 0 \tag{4.17}$$

Therefore, to find the optimal solution in the case of *Sooth Envelope* we have to find the solution of equation (4.17) $\forall\omega, k_d | 0 \le \omega \le \infty, 0 \le k_d \le \infty$. Once the parameters $\omega_o, k_{d,o}$ are found, the other controller parameters are easily obtained from equation (4.15).
The solution of equation (4.17) is a two dimensional problem, hence, to not face the complexity, in [43] at page 212 it is suggested an algorithm that solves the problem easily:

Algorithm 4.2.1 (Controller Design for Smooth Envelope)**.**

1. *the value of k_d is fixed to* 0*;*
2. *the solution ω_o of equation* (4.17) *is found with bisection method;*
3. *compute the PID parameters using equations* (4.15) *with such k_d, and ω_o;*
4. *verify that the* Robustness Constraint (4.12) *is satisfied around ω_o;*
   (a) *if it is, increase the value of k_d and repeat from point 2;*
   (b) *if it is not, take the last value of k_d, k_i, k_p that satisfy the* Robust Constraint*;*
5. *to verify that it has been found the global solution of the optimal problem computing the Nyquist plot of the loop transfer function $L(i\omega)$. We have to check that $L(i\omega)$ it is out of the circle centered in $(-1, i0)$ with radius r.*

We highlight that the solution of the bisection method could differs for different search range of ω_o: $\omega_{\text{low}} \le \omega_o \le \omega_{\text{high}}$. Thus, it is important to choose it properly. In this regard, in [43] is proposed a strategy that we will see further on. The main problem of this approach is that it can not be used with a process transfer function that generates an *Envelope with Corner* for $k_d = 0$. Moreover, the iterations of algorithm 4.2.1 stop as soon as in the envelope appears a ridge, because, as we can see in figure 4.3b, when the envelope has a ridge, the maximum of the

lower vertex of the ellipse occurs in the forbidden region and not in the point of the solution. Hence, there is no guarantee that the maximum value of k_i is reached for all k_d, but only in the searched range. For this reason, we have to discuss how to find the solution in the case of the *Envelope with corner*.
Always looking at figures 4.3a, and 4.3b, we see that while in the case of the *Smooth Envelope* the solution is in the edge of one ellipse at the angular frequency ω_o, in the case of the *Envelope with Corner* the solution of the optimal problem is determined by two ellipses that have this point in common. Thus, these ellipses are obtained from equation (4.12) at two different angular frequencies $\omega_{o,1}, \omega_{o,2}$. Therefore, in this case, the problem can be reformulated as follows:

$$\begin{aligned} f(k_p, k_i, k_d, \omega_1) &= r^2 \\ \frac{\partial f}{\partial \omega}(k_p, k_i, k_d, \omega_1) &= 0 \\ f(k_p, k_i, k_d, \omega_2) &= r^2 \\ \frac{\partial f}{\partial \omega}(k_p, k_i, k_d, \omega_2) &= 0 \end{aligned} \tag{4.18}$$

The equations above are obtained applying the envelope equations (4.14) at two different angular frequencies $\omega_{o,1}, \omega_{o,2}$. These have to be true at the same time. Hence, the solution of the *Optimel Problem* in the case of the corner can be found solving these set of equations (4.18). Åström, respectively in [43], and [4] shows two ways to solve the problem:

1. Solve equation (4.18) using the Newton-Raphson method. The initial value for the Newton-Raphson iteration can be obtained by approximating the envelope by the loci of the right vertex V_R and the locus of the lowest vertex of the ellipse V_D.

2. Solve equation (4.18) using the Newton-Raphson method. The initial conditions are obtained by computing an approximate envelope for all values in a range that contain $\omega_{o,1}, \omega_{o,2}$. Manually looking at the corner it is possible to find approximate values for $\omega_{0,1}, \omega_{0,2}$. Then, it is possible to find the initial values for k_p, k_i out from $\omega_{0,1}, \omega_{0,2}$ computing the intersection of the two ellipses generated from equation (4.12).

Both of these approaches are affected by several problems. First of all, they are applicable for a fixed value of k_d, in fact, in [4] it is shown the design for PI control. Secondly, we have faced respectively the following issues:

1. The use of the vertex loci find good initial values for the Newton-Raphson algorithm only for particular transfer functions. For instance, the shape of the envelope changes radically if we add a time delay to a second-order transfer function. Furthermore, even with good initial conditions, the algorithm converges in most cases to a non-optimal solution, basically found $\omega_{o,1} = \omega_{o,2}$. A possible way to escape from this problem is to perturb the initial condition around the one found as before, compute iteratively the solutions using Newton-Raphson until the algorithm converges in the optimal one, so basically when $\omega_{o,1} \neq \omega_{o,2}$.

2. Computing the approximate envelope is time-consuming. Moreover, the manual check of the frequency is not an easy process, because of the hundreds of plots one over the other. We see that a good strategy is to compute the envelope with a large approximation, find the two frequency ranges where the critical frequency occurs, compute the locus again only in this shrink range of frequencies with a better approximation.

After all these considerations we see that the complexity of the *Optimal Problem* can be solved neither using the Algorithm 4.2.1 nor using the Newton-Raphson approach. We can use a more sophisticated algorithm, for example, the Matlab Optimisation Toolbox [8]. Now the issue is reduced to well pose the optimal problem and to give it good initial conditions.
Now, that we are ready to explore the method adopted in this work, we have to introduce a notation:

$$\omega_{L=\theta} \coloneqq \omega | \angle L(i\omega) = \theta \tag{4.19}$$

Algorithm 4.2.2 (Controller Design even if the envelope has corner)**.** *See Matlab Code in Appendix A:*

1. *the value of k_d is fixed to 0, and the initial searching range for ω_o is set to $\omega_{start} = \omega_{L=-90^\circ}$, $\omega_{stop} = \omega_{L=-180^\circ}$;*
2. *the approximated solution $\omega_{o,0}$ of equation (4.17) is found with bisection method in the interval $\omega_{start} \leq \omega \leq \omega_{stop}$;*
3. *a more precise solution ω_o of (4.17) is found using Levenberg-Marquardt algorithm with initial condition $\omega_{o,0}$;*
4. *compute the PID parameters using equations (4.15) with such k_d, and ω_o;*
5. *compute the* Robustness Constraint *(4.12) around ω_o, and verify that is satisfied;*
   (a) *if it is set $k_{d,min} = k_d$, and compute the new search range to $\omega_{start} = \frac{\omega_o}{2}$, $\omega_{stop} = \frac{\omega_{L=-180^\circ} + \omega_{L=-270^\circ}}{2}$;*
   (b) *if it is not set $k_{d,MAX} = k_d$*
6. *set the new $k_d = \frac{k_{d,MAX} - k_{d,min}}{2}$, and repeat from point 2 until the maximum k_d that satisfy the* Robust Constraint *is reached with the desired error;*
7. *solve the optimum problem using Matlab Optimisation Toolbox:*
   (a) *set the initial conditions: $k_{p,0}$, $k_{i,0}$, $k_{d,0}$ are the values obtained in the previous iteration, and also $\omega_{0,1} = \omega_{0,2} = \omega_o$;*
   (b) *define the constraints for the solution showed in equations (4.18), and define the semi-infinitely constrains that comes out setting the validity of the* Robust Constraint *(4.12) $\forall\omega|\omega_0 \leq \omega \leq \omega_\infty$, with $\omega_\infty = \omega|\omega \to +\infty$, and $\omega_0 = \omega|\omega \to 0^+$;*
   (c) *solve the* Optimal Problem *using* fseminf*, see [8]. The solution is the set of these parameters $k_{p,o}$, $k_{i,o}$, $k_{d,o}$, $\omega_{o,1}$, and $\omega_{o,2}$;*
8. *verify that it has been found the global solution of the optimal problem computing the Nyquist plot of the loop transfer function $L(i\omega)$. We have to check that $L(i\omega)$ it is out of the circle centred in $(-1, i0)$ with radius r. Furthermore, $L(i\omega)$ has to touch the circle in one point if $\omega_{o,1} = \omega_{o,2}$, otherwise has to touch in two points.*

In the Algorithm 4.2.2, from point 1 to point 6 the logic is almost the same as in Algorithm 4.2.1. The difference is in the variation of k_d from an iteration to the next one. In fact, while in the first process increment k_d with fixed step size, in the second case, the maximum k_d is found with a Bisection-method. We have benefits in terms of efficiency and precision. The necessary to reach the maximum k_d possible is due to observation. Looking at figures 4.4 we see that the maximum of k_i is reached for $k_d = 0.1472$. At the same time we see that the envelope is smooth from $k_d = 0$ to $k_d \simeq 0.14$. Hence, getting as close as possible to the solution of the problem when the envelope is still smooth, is a good initial condition for the Optimal Solver. In [25] it is proposed to use the solution get with $k_d = 0$ as the initial condition, but in this way, the solver is not able to find the global solution. We highlight that another approach that can be used to solve the problem and find the global solution is to use the *Global Optimization Toolbox* of Matlab, see [35], and [11] for further information.
The initial searching interval at step 1: $\omega_{\text{start}} = \omega_{L=-90^\circ}$, $\omega_{\text{stop}} = \omega_{L=-180^\circ}$ is due that the process P_{nom} has to be stabilized with the PID controller. For the considered nominal process P_{nom} of equation (3.14) we can verify that the following conditions are satisfied:

$$\begin{aligned} \frac{\mathrm{d}\angle P_{\text{nom}}(i\omega)}{\mathrm{d}\omega} &\leq 0 \\ \frac{\mathrm{d}|P_{\text{nom}}(i\omega)|_{\text{dB}}}{\mathrm{d}\omega} &\leq 20\left[\frac{\text{dB}}{\text{decade}}\right] \end{aligned} \tag{4.20}$$

In [43], and [4] the conditions (4.20) are called *Monotonicity Conditions*. For process that satisfy the conditions it is possible to shrink the searching initial interval to $\omega \leq \omega_{\text{stop}} = \omega_{L=-180^\circ} - \arcsin(r)$, see [4] for proof.
We want to mention that in both [43], and [25] is explained that maximising k_i could cause the closed-loop system has two pairs of complex poles with relative damping very low, which means that the time responses can be expected to be oscillatory. For that reason in [25] it is proposed to add the constraints below to the *Optimal Problem*:

$$\begin{aligned} \frac{\Re(\dot{L}(i\omega))\Im(\ddot{L}(i\omega)) - \Re(\ddot{L}(i\omega))\Im(\dot{L}(i\omega))}{\left[\Re(\dot{L}(i\omega))^2 + \Im(\dot{L}(i\omega))^2\right]^{3/2}} &< 0 \ \forall\omega \\ \frac{\partial \angle L(i\omega)}{\partial \omega} &< 0 \ \forall\omega \end{aligned} \tag{4.21}$$

With an obvious meaning of notation. The first constraint in (4.21) specifies that the Nyquist curve has negative curvature, and the second constraint prevents the controller from having excessive phase lead. For a process with integral action, as P_{nom} is, the second constraint is too restrictive, so we can neglect it. For P_{nom} we can verify that also the first condition is too restrictive, in fact, it respects only for the controller with neither derivative nor integral action. For instance, with $k_d = 0, k_i = 12.1, k_p = 7.9$ we obtain as maximum value 0.8857 that is not less than zero.
With the Algorithm 4.2.2 we are able to solve the *Optimal Problem* for the process transfer function under consideration P_{nom}. The solution is the following:

$$\begin{aligned} k_{p,o} &= 12.7534 \\ k_{i,o} &= 31.1783 \\ k_{d,o} &= 0.1472 \end{aligned} \tag{4.22}$$

We can verify in the Nyquist plot 4.5 that the solution respect the *Robust Constraint*, in particular that maximise the PID parameters until reach the bounder with tree points P_1, P_2, P_3, so at three frequencies.

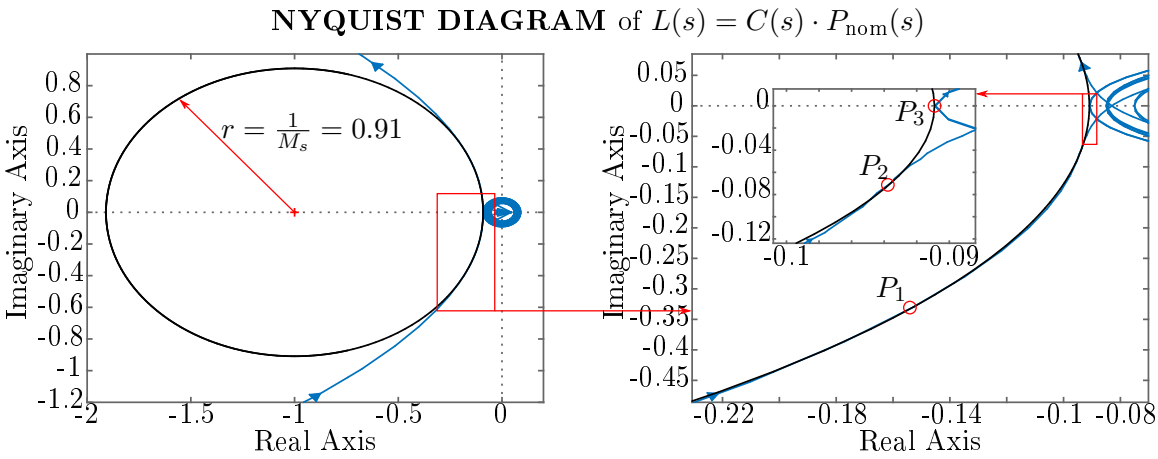


Figure 4.5: Nyquist diagram of the open loop transfer function $L(i\omega)$ with PID controller design using (4.22)

4.3 Sensitivity of PID control coefficients

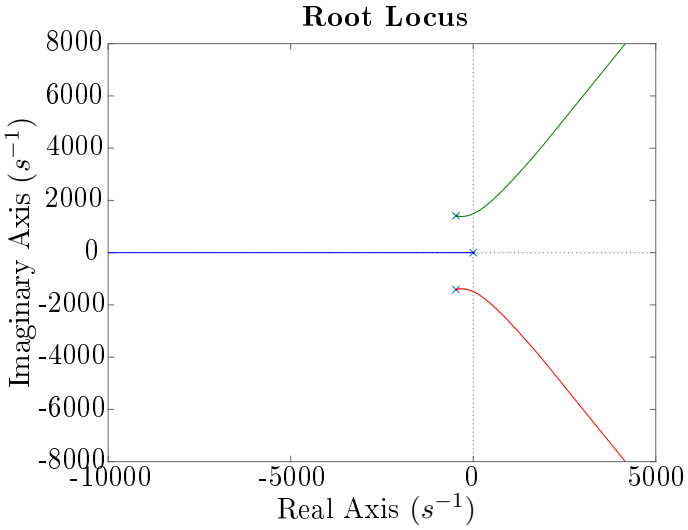


Figure 4.6: Root Locus of the nominal system without time delay: $P(i\omega)$

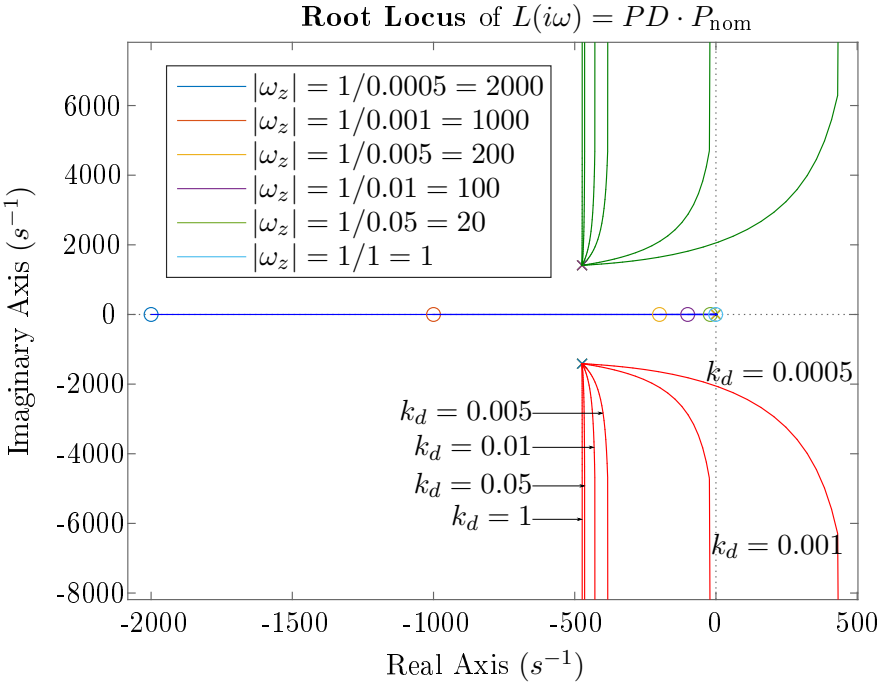


Figure 4.7: Root Locus of the loop TF using a PD control and the nominal system without time delay changing k_d.

In this section, we want to analyse the solution of the *Optimal Problem*, in particular, the goodness of the PID coefficients (4.22) found. To do that we have to take a step back to see what is the effect of each coefficient on the stability and performance of the system. We will use the Root Locus because is a good tool for this purpose.
Figure 4.6 it is shown the Root Locus of the nominal process under investigation. For simplicity and clearness we are not considering the time delay, hence $P(i\omega)$ 2.29 instead of $P_{\text{nom}}(i\omega)$ 3.14. We see that the system has a pair of complex poles at $p_{1,2} = -474.0[s^{-1}] \pm 1412.2i[s^{-1}]$, a pole in the origin $0 + 0i$, and no zeros. Furthermore, the system in the closed-loop is stable itself, and it remains stable until the gain of a P controller reaches around $k_p \simeq 2.6 \cdot 10^3$ when the branches of the complex poles cross the imaginary axes. In Figure 4.7 it is shown the Root Locus

of $P(i\omega) \cdot PD(i\omega) = P(i\omega) \cdot (k_p + i\omega k_d)$ for different values of k_d, respectively from 0.001 to 0.01 with a step of 0.001, k_p is fixed to 1. With a PD controller, we are adding a zero at the frequency of $\omega_z = k_p/k_d$. The closer this zero goes to the axes origin, the farther the vertical asymptote goes from the imaginary axes. Therefore, the k_d coefficient introduces a positive effect because, when the branches of the Root Locus lay on the left half complex plane the system remains stable for all process gain. Always from Figure 4.7 we see that the condition that allows having this positive effect on the stability of the system is the following:

$$|\omega_z| = \frac{k_p}{k_d} \leq \frac{1}{0.001}\left[\frac{\text{rad}}{s}\right] = 1000\left[\frac{\text{rad}}{s}\right] \tag{4.23}$$

For the set of parameters found in 4.22 we can verify that the condition (4.23) is verified:

$$\frac{k_{p,o}}{k_{d,o}} = \frac{12.7534}{0.1472} \simeq 86.6399\left[\frac{\text{rad}}{s}\right] \leq 1000\left[\frac{\text{rad}}{s}\right] \tag{4.24}$$

Hence, the combination of the coefficients $k_{p,o}$, and $k_{d,o}$ seems to be appropriate.
The optimal PID controller found shifts the poles and zeros of the closed-loop transfer function in the new position:

$$\begin{aligned} p_{1,2} &= (-0.4718 \pm 1.4539i) \cdot 10^3 \\ p_{3,4} &= (-0.0023 \pm 0.0024i) \cdot 10^3 \\ z_1 &= -84.1325 \\ z_2 &= -2.5179 \end{aligned} \tag{4.25}$$

From both equation (4.25) end figure 4.8 we can see that the new position of the poles in the closed-loop transfer function confirms the expectation described before discussing the figure 4.7. This placement of the roots of the closed-loop transfer function allows to improve the band-width of the control system, see figure 4.9. From the same picture, we can see that the first pair of poles shown in equation (4.25) are well-damped, and the other pair is not critical because has a gain under the unit.
Moreover, from the Bode plot of the open-loop transfer function, see figure 4.10, we see that the gain margin is $G_m = 20.8[dB]$, and the phase margin is $\varphi_m = 58.4°$. The phase margin usually is chosen in the design procedure around 60°, or 70°. We can assert that the obtained phase margin keep a good design value, especially because we have to take into account that in the nominal transfer function there is a time delay of $0.03[s]$.
Finally, both figures 4.9, and 4.10 show that such procedure drives the design solution to respect the standard requirements for a robust control design, see [29]:

1. $T(s)$ with wide bandwidth;
2. large loop gain $L(s)$ at low frequencies;
3. small loop gain $L(s)$ at high frequencies.

The frequency that divides the low-frequency range from the high one obviously is the critical frequency.

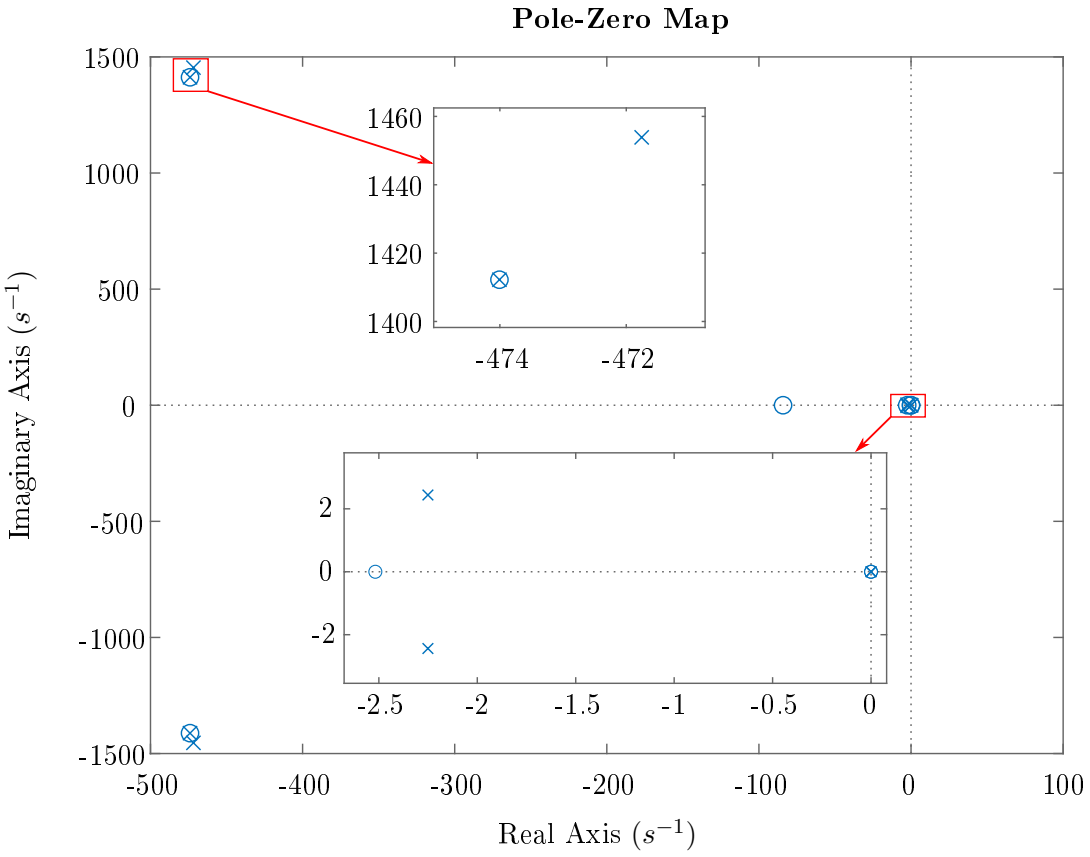


Figure 4.8: Poles and Zeros plot in complex plane for the closed loop transfer function $T(i\omega)$ obtained with the nominal process P_{nom} controlled by the optimum PID controller $C(i\omega)$

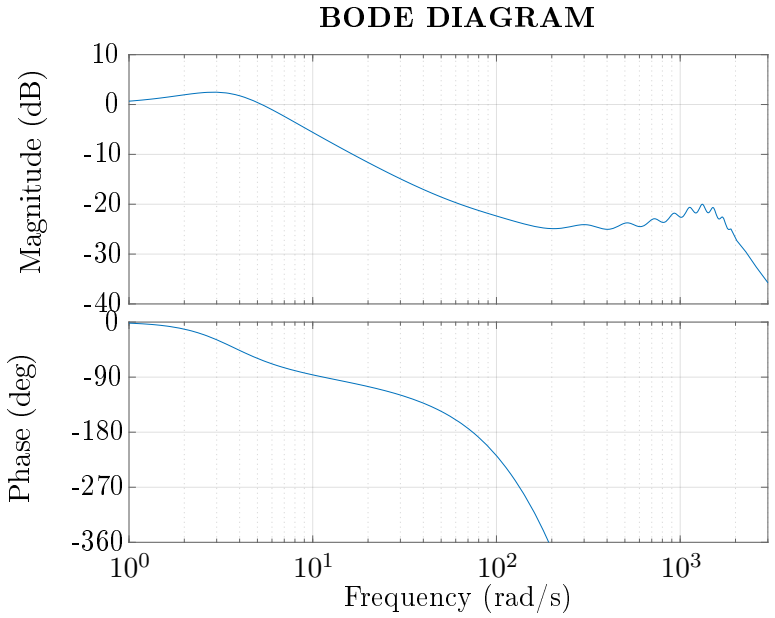


Figure 4.9: Bode diagram of the close loop transfer function $T(i\omega)$

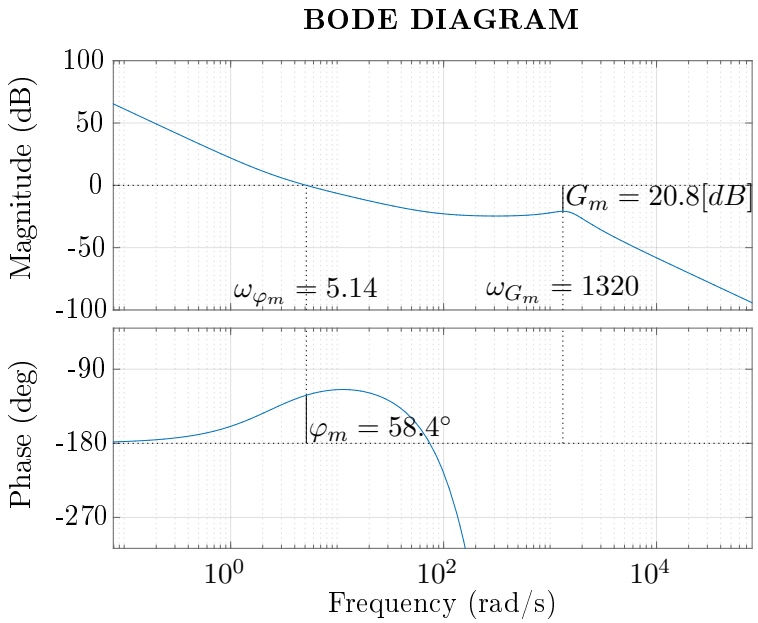


Figure 4.10: Phase margin and Gain margin of the open loop transfer function $L(i\omega)$

4.4 Final control structure design

The PID form of equation (4.1) is not physically achievable because of the pure derivative term. Furthermore, we want to implement a 2DOF controller to be able to achieve better performance in terms of set point response. Figure 4.11 points out the complete control structure. We can see starting from the position reference x_{ref} the set point filter F_{sp}, that produces a filtered version of the reference x^f_{ref}. Then, there is the 2DOF PID control structure. The goal of the structure is to get a time response of

$$u(t) = k_p\big[bx^f_{\text{ref}}(t) - x_{\text{mea}}(t)\big] + k_i\int_0^t \big[x^f_{\text{ref}}(\tau) - x_{\text{mea}}(\tau)\big]d\tau + k_d\Big[-\frac{dx_{\text{mea}}(t)}{dt}\Big] \tag{4.26}$$

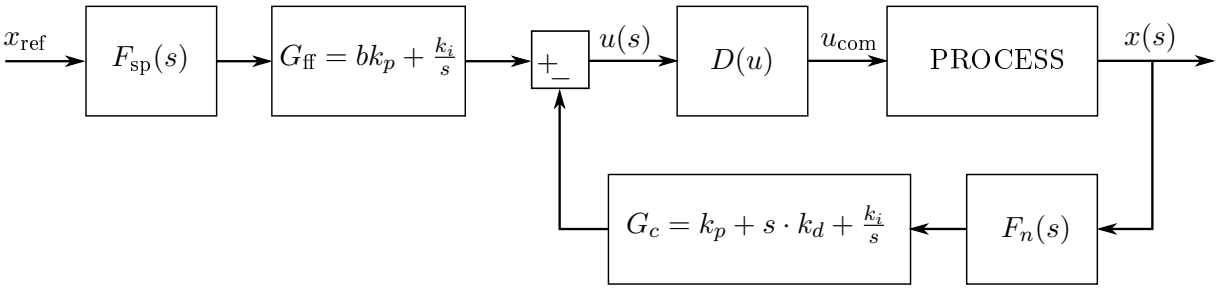


Figure 4.11: Full control structure block diagram

In equation (4.26) both the position signals $x^f_{\text{ref}}(t)$, $x_{\text{mea}}(t)$ are filtered. In the first case be the set point filter F_{sp} as discuss previously, in the second case by F_{n} for filtering the measurement noise. From equation (4.26) we can compute the 2DOF PID transfer function. In particular, we can separate the Feed-Forward action $G_{\text{ff}}(s)$, from the Feed-Back one $G_{\text{c}}(s)$:

$$G_{\text{ff}}(s) = bk_p + \frac{k_i}{s} \tag{4.27}$$

$$G_{\text{c}}(s) = k_p + \frac{k_i}{s} + k_d s \tag{4.28}$$

From equation (4.28) we can see that the noise filter F_{n} has another function: combined with $G_{\text{c}}(s)$ makes the latter physically achievable. The control signal $u(t)$ that comes out from the PID controller has to be scaled by the function $D(t)$, $u_{\text{com}} = D[u(t)]$. Basically, $D(t)$ has to compensate the non-linearity of the dead-zone that we have not considered in the nominal process $P_{\text{nom}}(s)$.
This chapter points out the design procedure of all the remaining parts of the control system.

4.4.1 Noise filter

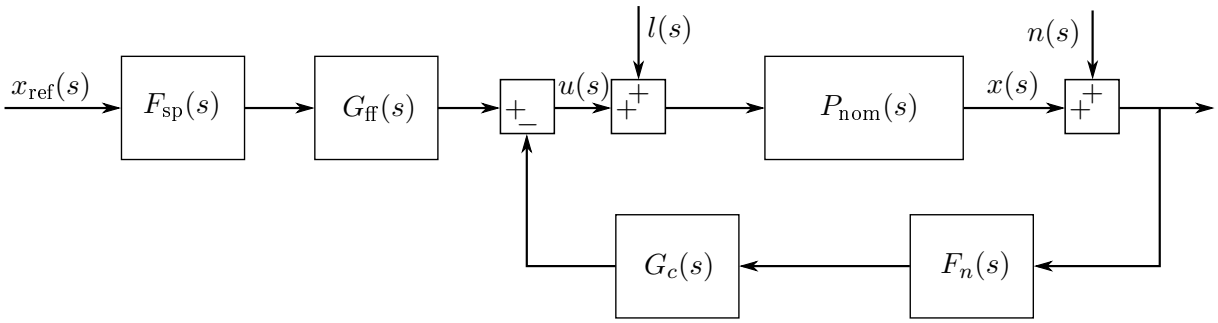


Figure 4.12: Control structure block diagram with measurement noise $n(s)$ and load disturbance $l(s)$

The noise filter has to reduce the disturbance that comes out from the linear encoder, and as we said, make the derivative term physically achievable. For this reason seems to be good to have a small cut off frequency: $\omega_f = \frac{2\pi}{\tau_f} \downarrow$. The drawback is the amplification of the load disturbance, indeed:

$$\begin{aligned} \frac{x(s)}{l(s)} &= \frac{P_{\text{nom}}(s)}{1 + G_{\text{c}}F_{\text{n}}P_{\text{nom}}(s)} \\ \frac{x(s)}{n(s)} &= -\frac{G_{\text{c}}F_{\text{n}}P_{\text{nom}}(s)}{1 + G_{\text{c}}F_{\text{n}}P_{\text{nom}}(s)} \\ \frac{u(s)}{n(s)} &= -\frac{G_{\text{c}}F_{\text{n}}(s)}{1 + G_{\text{c}}F_{\text{n}}P_{\text{nom}}(s)} \end{aligned} \tag{4.29}$$

using the first of (4.29), we see that the sensitivity of the output $x(s)$ to the load disturbance $l(s)$ increases in we reduce the cut off frequency of the filter. The solution is a trade off between reduction of load disturbances $\frac{x(s)}{l(s)} \downarrow$ and filtering of measurement noise: $\frac{x(s)}{n(s)} \downarrow$, and $\frac{u(s)}{n(s)} \downarrow$.
We explore the performances of a first and second order low pass filter:

$$F_{\text{n}}(s) = \frac{1}{(1 + s\tau_f)^{\text{ord}}} \tag{4.30}$$

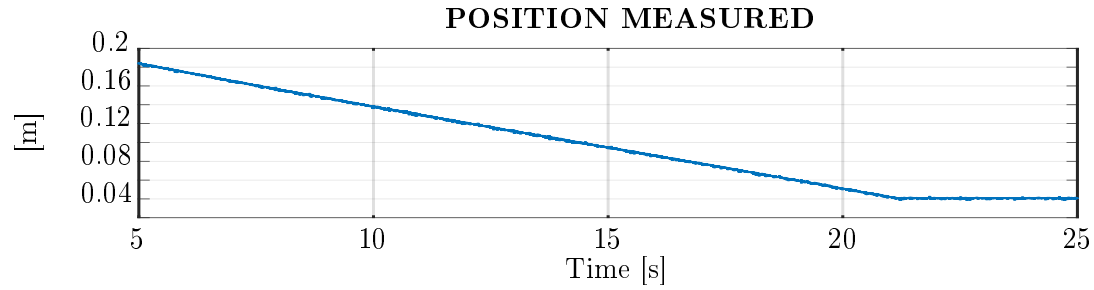
The cut off frequency has to be less than the critical frequency ω_o discovered with the *Optimization Algorithm* and equal to the frequency where $|S(s)|$ has its maximum: $\omega_o = \omega |\ |S(i\omega)| = M_S$. Typical choices are the following:

$$\tau_f = \begin{cases} \frac{1}{N\omega_o}, & \text{for first-order filter: ord} = 1 \\ \frac{1}{2N\omega_o}, & \text{for second-order filter: ord} = 2 \end{cases} \tag{4.31}$$

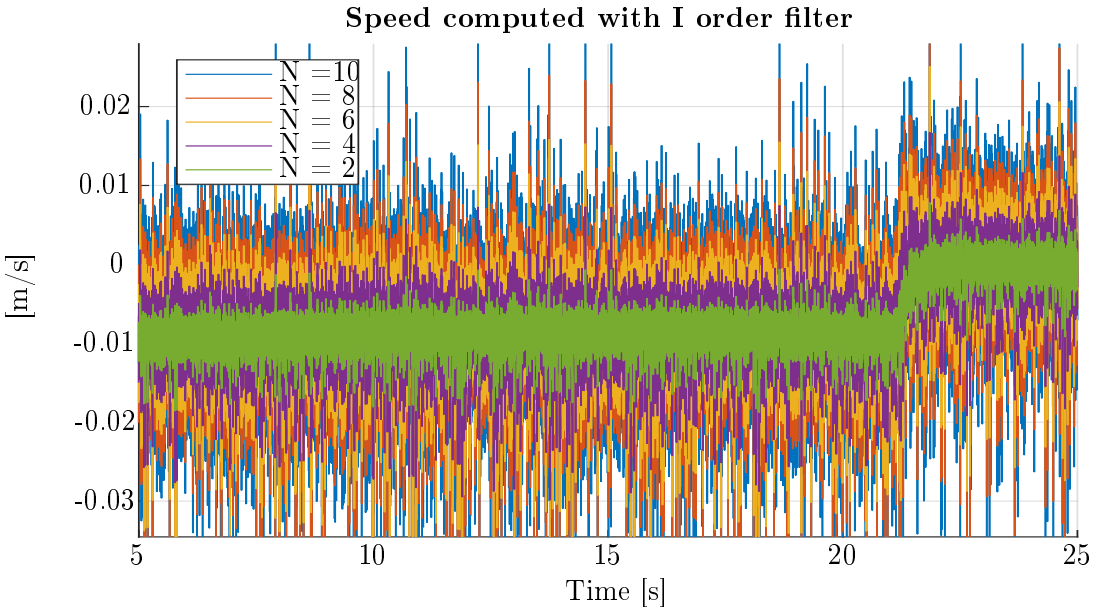
With N from 2 to 10. Implementing the filter of (4.30) with the time constant showed in (4.31) we obtained the responses sowed in Figure 4.13b, and 4.13c. In these two figures are used respectively filters of first and second order. In both the images, at the top we see the filtered position of an experiment, at the bottom, is computed the speed using a derivative filter with a pole at the same frequency as the respective F_{n}. The last is made to see the result of the derivative action in the PID controller. We can notice that the speed responses are still very noisy for all the sets of first-order filters. Looking at the value of the time constant τ_f in table 4.1 we see that all the values are much greater than the sample time of the system interface $t_s = 0.0005[s]$. Finally, at the end of all the considerations, and based on simulation results that we will see later, we choose $\tau_f = 0.0431[s]$, so basically $N = 5$.

Table 4.1: τ_f for the noise filter F_{n} used in Figures 4.13b and 4.13c.

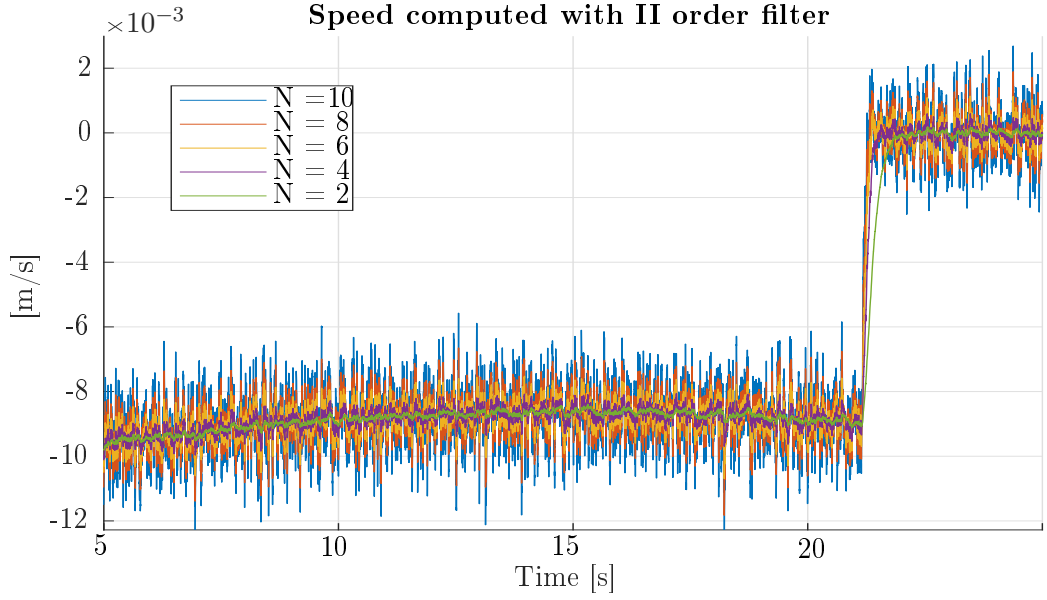
N	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
ord=2	0.1291	0.0861	0.0645	0.0516	0.0431	0.0369	0.0322	0.0287	0.0258
ord=1	0.2583	0.1722	0.1291	0.1033	0.0861	0.0738	0.0645	0.0574	0.0516



(a) position measured



(b) first order filters: $F_{\mathrm{n}}(s) = \frac{s}{(1+s\tau_f)}$, for five values of N



(c) second order filters: $F_{\mathrm{n}}(s) = \frac{s}{(1+s\tau_f)^2}$, for five values of N

Figure 4.13: Speed computation with a filtered derivative. A set of filters with different time constant are used. The values are computed using the equation 4.31.

4.4.2 Set point response

For the design of the reference filter F_{sp} and the set point weight b, there are many possibilities. One is to consider the transfer function from set point to output and to make sure that the maximum magnitude of this transfer function is not larger than 1. That gives a set point response without overshoot for the nominal process considered. To do that, first of all, we have to compute the overall transfer function:

$$W(s) = \frac{x(s)}{x_{\text{ref}}(s)} = \frac{G_{\text{ff}} P_{\text{nom}}(s)}{1 + G_{\text{c}} F_{\text{n}} P_{\text{nom}}(s)} \tag{4.32}$$

In equation 4.32 the only parameter that is not yet specified is b. Hence, we have to proceed iteratively: we have to compute the value the infinity norm of $W(s)$, $M_W = \max_\omega[|W(i\omega)|]$, for a fixed value of b. Then, if M_W is greater than 1 we have to reduce b otherwise we can increase. Therefore the set point filter F_{sp} is necessary only in the case $M_W > 1$ also for $b = 0$. In that case F_{sp} is designed to make the infinity norm of $F_{\text{sp}}W(s)$ equal to 1.
For the process $P_{\text{nom}}(s)$, the Optimal PID designed, and the second-order noise filter F_{n} obtained with $\tau_f = 0.0431[s]$, we see that even with $b = 0$, we have $M_W = 1.0379 > 1$. Hence, we have to design the set point filter F_{sp}. The filter is found with some algebra:

$$F_{\text{sp}}(s) = \frac{1}{1 + s\frac{2\pi}{\omega_{\text{sp}}}\sqrt{M_W^2 - 1}} = \frac{1}{1 + s\tau_{\text{sp}}} \tag{4.33}$$

Where ω_{sp} is the angular frequency in witch M_W occurs.
The following Matlab code computes the set point parameters b, and $F_{\text{sp}}(s)$ in both the cases described before:

Listing 4.1: Matlab code to compute the set point parameters

```
% define transfer function:
s = tf('s');
Gc_tf = PID.kp +PID.ki/s +PID.kd*s;
Fy_tf = noiseF.Fy_tf(PIDsp.NoiseFilterChosen);
b = 0;
Gff_tf = b*PID.kp +PID.ki/s;
W_tf = Pnom_tf*Gff_tf/(1+Pnom_tf*Gc_tf*Fy_tf);

% find infinity norm and frequency where it occours:
w_vector = linspace(-3,3,100000);
[mag_W, phase_W] = bode(W_tf, w_vector);
mag_W = squeeze(mag_W);
InfNorm_W = max(mag_W)
w_InfNorm_W = w_vector(find(mag_W==InfNorm_W,1))

if InfNorm_W > 1
    PIDsp.b = 0;
    PIDsp.Fsp_T = sqrt((InfNorm_W)^2-1)/abs(w_InfNorm_W/2/pi);
    PIDsp.Fsp_tf = 1/(1+s*PIDsp.Fsp_T);
else
    while InfNorm_W <= 1
        b = b + 0.1;
        Gff_tf = b*PID.kp +PID.ki/s;
        W_tf = Pnom_tf*Gff_tf/(1+Pnom_tf*Gc_tf*Fy_tf);
        [mag_W, phase_W] = bode(W_tf, w_vector);
        mag_W = squeeze(mag_W);
        InfNorm_W = max(mag_W);
        w_InfNorm_W = w_vector(find(mag_W==InfNorm_W,1));
    end
    PIDsp.Fsp_tf = 1;
    PIDsp.b = b - 0.1;
end
```

Then, we can compute the overall transfer function from the reference position to the system rod end position. We obtain the followings:

$$W_{\text{tot}}(s) = \frac{x(s)}{x_{\text{ref}}(s)} = F_{\text{sp}} \cdot W(s) = F_{\text{sp}} \cdot \frac{G_{\text{ff}} P_{\text{nom}}(s)}{1 + G_{\text{c}} F_{\text{n}} P_{\text{nom}}(s)} \tag{4.34}$$

In figure 4.14 are shown the step response numerically computed of the system expressed by equation 4.34. We highlight that is the response of the nominal process P_{nom} controlled with the full-control scheme. We compute the unitary step response with different set point designs. The set point controller is designed respectively in the three following ways:

1. designed with $b = 1$, and $F_{\text{sp}}(s) = 1$;
2. designed with $b = 0$, and $F_{\text{sp}}(s) = 1$;
3. designed with $b = 0$, and $F_{\text{sp}}(s)$ found with equation (4.33).

We see that we obtain the desired effect, but applying the step point filter we are reducing drastically the control system band-width. Thus, we can take at least the last two design solutions, the ones with $b = 0$, to check first the performance in the simulated system and the real system after. Indeed, we know that we have made a lot of simplifications linearising the model, most of all for the friction model.

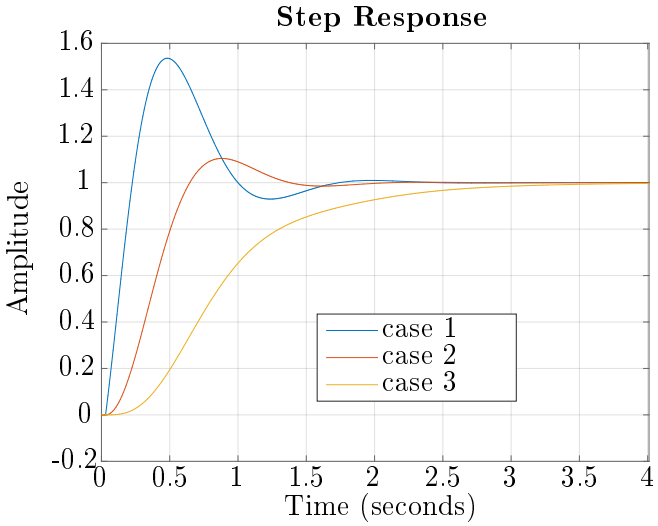


Figure 4.14: Numerically computed step response of the nominal process P_{nom} controlled by the 2DOF PID that gives the system of equation 4.34. The controller are designed respectively 1) with $b = 1$ and $F_{\text{sp}}(s) = 1$; 2) with $b = 0$ and $F_{\text{sp}}(s) = 1$; 3) with $b = 0$ and $F_{\text{sp}}(s)$ found with equation (4.33)

4.4.3 Feed forward dead zone compensation

In the end, the last thing still remaining to do for completing the controller design is the Feed-Forward action. Basically, we can implement a piecewise linear function that compensates for the non-linearity introduced by the dead-zone. The aim is that $z(t) = u(t)$, so $D(u)$ has to be the inverse of the dead zone function (2.3). Thus, we get:

$$u_{\text{com}} = D(u) = \begin{cases} u - 0.1, & \text{for } u \leq -0.0042 \\ 25u, & \text{for } -0.0042 < u \leq 0.0042 \\ u + 0.1, & \text{for } u > 0.0042 \end{cases} \tag{4.35}$$

In figure 4.15 we can see the function of the dead-zone plus the saturation and the inverse function $D(u)$. An important thing, also detectable from equation (4.35), is that at the origin, the slope is very high but still not ∞. This prevents, in case of oscillation on the response, the appearance of a high-frequency limit cycle that can damage the directional control valve.

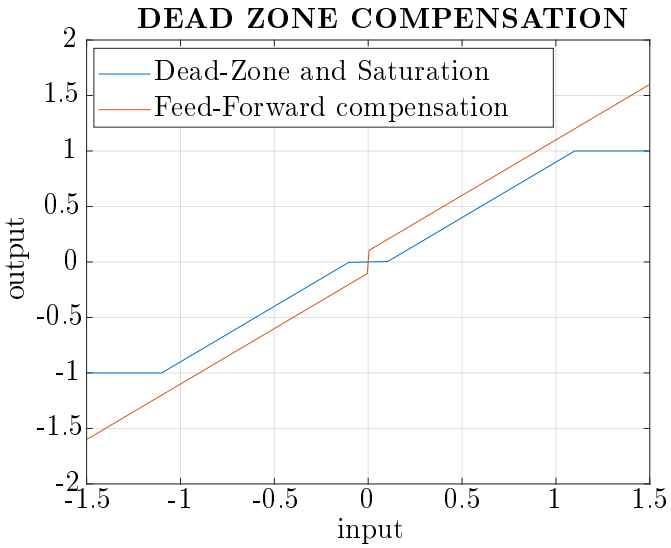


Figure 4.15: The orifice dead-zone and saturation function (shown in blue) and the dead-zone inverse $D(u)$ (shown in red).

Chapter 5

Robust condition

In this chapter, we want to analyse the robustness of the control system designed. To do that, we have to evaluate a model that represents the uncertainty discussed in chapter 3. Then, we need to find the stability range of the system because we have to guarantee a safe operation in certain conditions.

5.1 Representing uncertainty

In chapter 2 we have seen the *Full Order Model* of the system, and the *Reduced Order Model*. We have said that we are going to use these models for simulations. Then, to find a linear model for the control design we have had to linearised the *Orifice Equations* on an operative point, so to get the coefficients $\hat{C}_q$, and $\hat{C}_{qp}$. Furthermore, the communication through a WIFI bridge introduce in the control loop a variable time delay τ. In this way the transfer function computed P_τ (3.10) is a good approximation of the *Reduced Order Model* only in the particular operative point, gives from the choice of $\hat{C}_q$, $\hat{C}_{qp}$, and $\hat{\tau}$. Hence, after several considerations, we have choose the nominal value of these parameters $C_{q,\text{nom}}$, $C_{qp,\text{nom}}$, and τ_{nom} so we compute the nominal process transfer function: $P_{\text{nom}}(s)$, see 3.14.
Finally, we have discussed that we can consider for simplicity only the gain uncertainty that the before mentioned parameters produce, still maintaining a good approximation. So this section aims to find a way to model the gain uncertainty of the process transfer function.
First of all, we can consider our system affected by a so-called *Multiplicative Perturbation*. In fact

$$P_\tau(i\omega) = \big[1 + \Delta_U(i\omega)W_U(i\omega)\big]P_{\text{nom}}(i\omega) \tag{5.1}$$

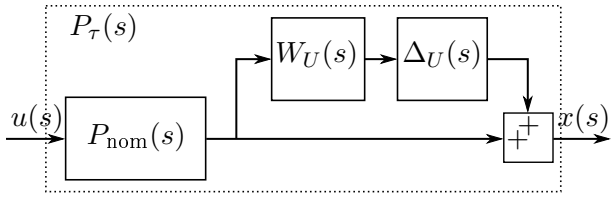


Figure 5.1: Block diagram of the perturbed process modelled with multiplicative uncertainty

so the gain of the transfer function $P_{\text{nom}}(i\omega)$ could vary from the nominal value, in a range that is defined by $\Delta_U(i\omega)W_U(i\omega)$. $W_U(i\omega)$ is a fixed, stable transfer function that, as we will see later, has the duty of weighing the uncertainties. $\Delta_U(i\omega)$ is a variable stable transfer function satisfying $\|\Delta_U(i\omega)\|_\infty < 1$ with the purpose of perturbing the weighting function $W_U(i\omega)$ so that the transfer function is unknown. The model of the *Multiplicative Perturbation* is represented with a block diagram in figure 5.1. Moreover, because $\Delta_U(i\omega)$ is defined as $\|\Delta_U(i\omega)\|_\infty < 1$, then we have the following relation:

$$\left|\frac{P_\tau(i\omega)}{P_{\text{nom}}(i\omega)} - 1\right| \leq |W_U(i\omega)|,\ \forall\omega,\ \forall(\hat{C}_q, \hat{C}_{qp}, \tau) \tag{5.2}$$

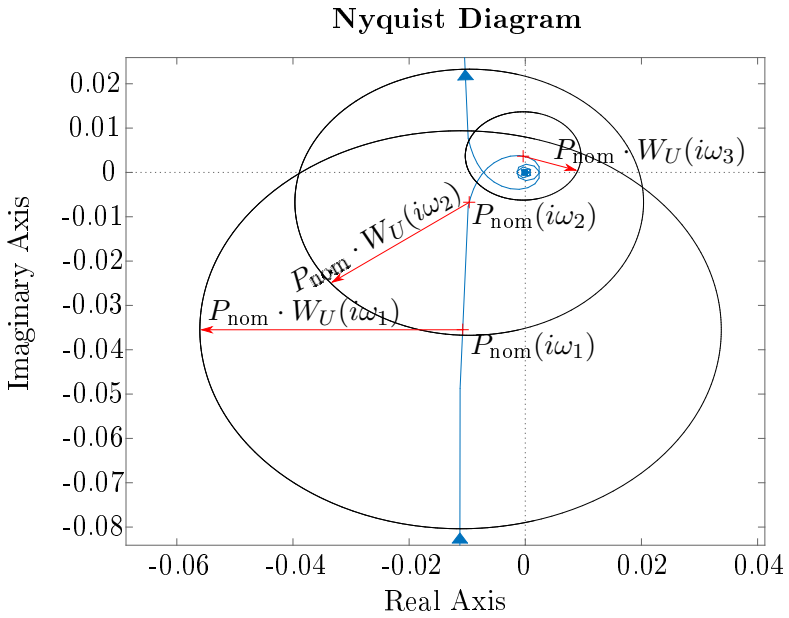


Figure 5.2: Nyquist plot of the nominal process and of three points of the perturbed model

The last equation is very important because it shows clearly that $|W_U(i\omega)|$ provides the uncertainty profile. Furthermore, it shows a practical way to obtain the weighting function. With this uncertainty model, there are two important considerations to take in mind.

1. The transfer functions $W_U(i\omega)$ and $\Delta_U(i\omega)$ are frequency depending, so they are modelling the gain uncertainty with different weights from low to high frequency. Typically uncertainty increase with the frequency [16].

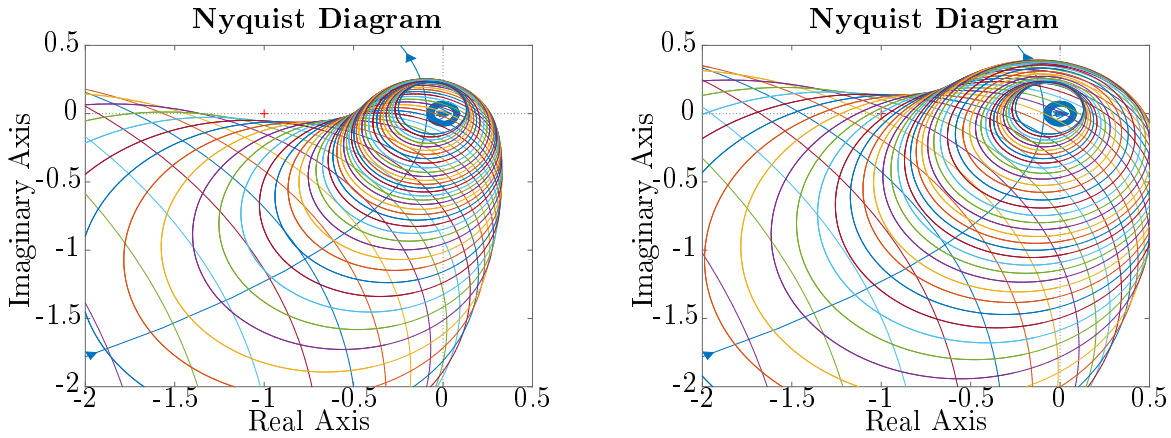
2. We are modelling the uncertainty with a disk-like shape in a complex plane. Hence, we are assuming that the Nyquist plot of the uncertain process $P_\tau(i\omega)$ at a fixed frequency $\hat{\omega}$ is inside a disk of radius $W_U(i\omega)$ and centered in $P_{\text{nom}}(i\hat{\omega})$, see figure 5.3.

The last point introduces a hypothesis necessary for simplifying the evaluation of the *Robust Stability Condition*, but the drawback is a more conservative approach. In fact, generally, the combination of the uncertain parameters gives an uncertain region that has an undefined shape. Therefore, the disc surrounds the uncertainty area, with a center and radius that depend on the choice of the nominal model. Hence, when the perturbed system gives an unstable closed-loop transfer function, we can not guarantee that the real process is stable, but neither we can say that the real process is stable. For instance, in figure 5.3a is shown the open-loop transfer function obtained with the optimal PID designed in the previous chapter. It is also plotted over the uncertain region gives by the disc with radius $W_U(i\omega) \cdot L(i\omega)$ and center in $P_{\text{nom}}(i\omega)$, for a range of ω. In particular $W_U(i\omega)$ is obtained considering $\tau_{\text{MAX}} = 0.10[s]$. We see that both the nominal open-loop transfer function and the sum of all the uncertain regions are encircling the critical point $(-1, i0)$. Then, thanks to the Nyquist criterion, we can conclude that the control system is stable also for the perturbed model $P_\tau(i\omega)$. In figure 5.3b is plotted the same, but this time $W_U(i\omega)$ is obtained considering $\tau_{\text{MAX}} = 0.15[s]$. We see that at some frequency the disc of the uncertain region surrounds the critical point. Then, we can not say anything about the stability of the system.

5.2 Robust stability condition

To understand well what *Robust Stability* means, we have to see together the definitions of the *Nominal Stability*:

Definition 5.2.1 (Nominal Stability)**.** Is guarantee if the controller $C(i\omega)$ internally stabilize the nominal model of the process $P_{\text{nom}}(i\omega)$.



(a) The uncertain radius is computed with $\tau_{\text{MAX}} = 0.10[s]$. The critical point $(-1, 0)$ is outside the uncertain region.

(b) The uncertain radius is computed with $\tau_{\text{MAX}} = 0.15[s]$. The critical point $(-1, 0)$ is inside the uncertain region.

Figure 5.3: Nyquist plot of the close loop transfer function $L(i\omega)$ with the uncertain disc region at some frequencies.

Definition 5.2.2 (Robust Stability)**.** Is guarantee if the controller $C(i\omega)$ internally stabilize every possible perturbed plant of the process $P_\tau(i\omega)$.

From the above definitions, we can see that the *Robust Stability* condition is much more restrictive than the *Nominal* one. In fact, in the set of all possible perturbed plant $P_\tau(i\omega)$, in our case defined through the multiplicative uncertainty, there is a configuration of the parameters that give the most difficult plant to stabilize, so we call worst-case plant. For instance, in the set of $P_\tau(i\omega)$ there is the plant with the maximum time delay, that impose for achieving *Nominal Stability* a lower bend-width in the control system, compared to the one with the nominal time delay.
In [38] is derived the criterion for *Robust Stability*, so-called *Robust Stability Condition*, with three method. These proofs are valid for SISO systems. While in [42] there is a full demonstration of the *Robust Stability Condition* for MIMO systems using the *small gain theorem*. Instead, we want to see the validity of the condition from an intuitive point of view.
In figures 5.3a, and 5.3b we have seen that, for a multiplicative uncertainty, a simple criterion for the *Robust Stability* is that the discs are not encircling the critical point. The distance from the Nyquist plot of the nominal loop transfer function $L(i\omega)$ and the critical point $(-1, i0)$ is $|-1 - L(i\omega)| = |1 + L(i\omega)|$. The radius of the disc is basically $C(i\omega) \cdot W_U P_{\text{nom}}(i\omega) = W_U L(i\omega)$, hence

$$\begin{aligned} &\textit{Robust Stability Condition} \iff |W_U L(i\omega)| < |1 + L(i\omega)|, \ \forall\omega \\ \iff &\left|\frac{W_U L(i\omega)}{1 + L(i\omega)}\right| < 1, \ \forall\omega \iff |W_U T(i\omega)| < 1, \ \forall\omega \\ &\iff \|W_U T(i\omega)\|_\infty < 1 \end{aligned} \tag{5.3}$$

Therefore, the last implication is the criterion we were looking for: the infinite norm of the complementary sensitivity function multiplied by the weighting function of the uncertainties has to be less than one. Thus, in the condition, the choice of $W_U(i\omega)$ has an important part.

5.3 Weighting function

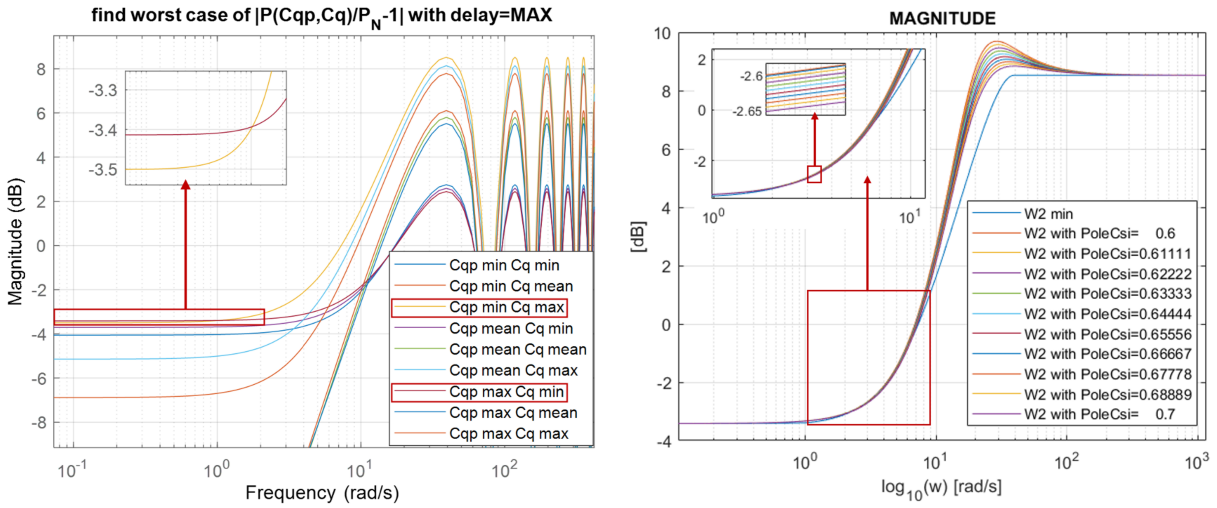
The choice of the weighting function for the multiplicative perturbation of the nominal process has to take into account three main ideas:

1. has to respect equation (5.2);
2. has to be less conservative as possible;

3. has to verify the *Robust Stability Condition* (5.3).

To achieve all these objectives, we want to proceed in this way: we are going to design an algorithm that finds the best W_U possible for a defined τ_{MAX}. Then, we will check the *Robust Stability Condition* and, if it is not verified, we decrease the value of τ_{MAX} until the condition is satisfied. We remind that τ_{MAX} is the upper bound for the RTT (round travel time) in the control system. Hence, find the value of τ_{MAX} that satisfy the *Robust Stability Condition* means find where the safe operation of the control system is guaranteed.
To find W_U we start computing the left side of the inequality (5.2). We have to calculate the maximum possible value of the term for all possible combinations of the parameters and for each frequency. Hence, we fixed the value of τ to τ_{MAX}, and we evaluate the modulus $\left|\frac{P_\tau(i\omega)}{P_{\text{nom}}(i\omega)} - 1\right|$ for all the combinations of C_q, C_{qp} respectively in the set $C_{q,\text{min}}, C_{q,\text{mean}}, C_{q,\text{MAX}}$, and $C_{qp,\text{min}}, C_{qp,\text{mean}}, C_{qp,\text{MAX}}$. The result for $\tau_{\text{MAX}} = 0.11[s]$ is shown in figure 5.4a. In the figure it is highlighted that there is not only one particular combination of C_q, C_{qp} that gives the maximum magnitude, but in this case happens for two: at low frequency (around up to $1\left[\frac{\text{rad}}{s}\right]$) for $C_{q,\text{min}}, C_{qp,\text{MAX}}$, and at high frequency for $C_{q,\text{MAX}}, C_{qp,\text{min}}$. Hence, the left part of inequality (5.2) has to be computed frequency by frequency. We call this magnitude calculation $W_{U,\text{min}}(i\omega)$, that is shown in figure 5.4b with a blue line. Now we have to find $W_U(i\omega) \geq W_{U,\text{min}}(i\omega)$. In [38] it is shown with an example how get the weighting function a third-order TF for a first-order system with time delay. Furthermore, in [16] for a double integrator system with time delay it is used a first-order weighting function with a zero in the origin. Usually $W_U(s)$ is of low order to simplify the controller design [38] so, because a first order TF fits $W_{U,\text{min}}(i\omega)$ not good enough, we use a second-order transfer function:

$$W_U(s) = k_W \frac{\frac{s^2}{\omega_z^2} + 2\xi_z \frac{s}{\omega_z} + 1}{\frac{s^2}{\omega_p^2} + 2\xi_p \frac{s}{\omega_p} + 1} \tag{5.4}$$



(a) Bode plot of the left term of (5.2) for different combinations of $(\hat{C}_q, \hat{C}_{qp})$ and with $\tau = \tau_{\text{MAX}}$

(b) $W_U(s)$ fit attempt of $W_{U,\text{min}}(s)$ varying ξ_p from $\xi_p = 0.6$ to $\xi_p = 0.7$

Figure 5.4: Evaluation of the weighting function lower bound: $W_{U,\text{min}}(s)$ and fit attempt with a second order transfer function varying xi_p.

The idea is to find $W_U(s)$, so find k_W, ω_z, ξ_z, ω_p, and ξ_p with an optimal algorithm that minimize the error $|W_U(i\omega)| - |W_{U,\text{min}}(i\omega)|$. As usual, the problem is give to the algorithm the proper initial condition. With simple rules of the bode diagram we can fixed:

$$\begin{aligned} \omega_p &= \omega_{\text{resonance}} = \omega_{\max[W_{U,\text{min}}]}\sqrt{1-2\xi_p^2} \\ k_W &= |W_{U,\text{min}}(i0)| \end{aligned} \tag{5.5}$$

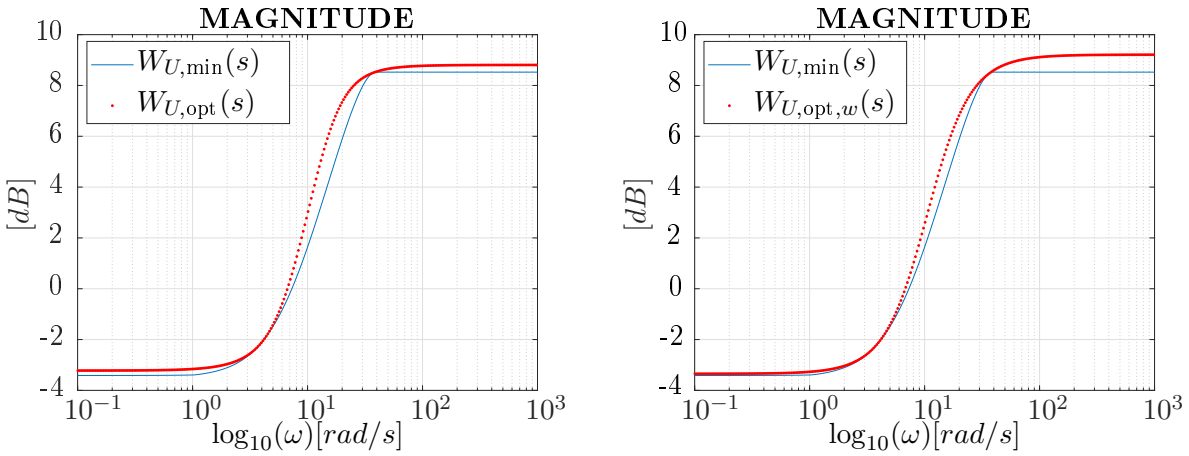
where $\omega_{\max[W_{U,\min}]}$ is the frequency in which the first relative maximum value of $|W_{U,\min|}$ occurs. We can fixed the values of the damping factors, looking at figure 5.4b we see that as first attempt a good value for ξ_p is $\xi_p = (1/\sqrt{2} + 0.5)/2 \simeq 0.604$, and $\xi_z = 1$. Then the position of the zero is computed so that the gain at $\omega = \infty$ is equal to the one of $|W_{U,\min}|$:

$$\omega_z = \hat{\omega} | \; |W_U(i\infty)| = |W_{U,\min}(i\infty)| = \max_{\omega} |W_{U,\min}(i\omega)| = |W_{U,\min}(i\omega_{\max[W_{U,\min}]})| \tag{5.6}$$

After some algebra, we see that the constraint (5.6) is equal to fix the ratio ω_p/ω_z to the constant 1.9882. With a better look of figure 5.4b we see that we can reduce the resonance peak increasing the value of ξ_p until is not making $|W_U(i\omega)| < |W_{U,\min}(i\omega)|$. Hence, we obtained $\xi_p = 0.6111$. Now that a good set of initial conditions is found we can proceed with the optimization algorithm. The cost function to minimize is the integral square error (ISE):

$$\text{cost function} = ISE = \int_0^\infty \Big[|W_U(i\omega)| - |W_{U,\min}(i\omega)| \Big]^2 d\omega \tag{5.7}$$

$$\text{constraint: } |W_U(i\omega)| \geq |W_{U,\min}(i\omega)|, \; \forall\omega \tag{5.8}$$



(a) Bode plot of $W_{U,\text{opt}}$, red dots, and $W_{U,\min}(s)$ blue line

(b) Bode plot of $W_{U,\text{opt}W}$, red dots, and $W_{U,\min}(s)$ blue line

Figure 5.5: Optimized fitting of the second order weighting function over $W_{U,\min}$.

Table 5.1: Weighting function parameters computed for two time delay upper-bounds $0.11[s], 0.15[s]$, using the optimization algorithm and the same weighted

$\tau_{\mathbf{MAX}}[\mathbf{s}]$	0.11		0.15	
$\mathbf{W_U}$	$W_{U,\text{opt}}$	$W_{U,\text{opt}W}$	$W_{U,\text{opt}}$	$W_{U,\text{opt}W}$
$\xi_\mathbf{z}$	0.8497	0.8987	0.845	0.9221
$\xi_\mathbf{p}$	0.802	0.9045	0.796	1
$\mathbf{k_W}$	0.6902	0.6803	0.6919	0.6816
$\omega_\mathbf{z}$	7.3061	7.3061	4.8749	4.8749
$\omega_\mathbf{p}$	14.6012	15.0527	9.7187	10.3189

Solving such problem for $\tau_{\text{MAX}} = 0.11[s]$ we get the transfer function $W_{U,\text{opt}}$ showed in figure 5.5a. We see that the solution found fit really well the objective $W_{U,\min}(i\omega)$.
We can now compute the *Robust Stability Condition*. Figure 5.6a shows the main transfer function involved in the calculation of the robust constraint using the sampled version of $|W_{U,\min}(i\omega)|$ so the best possible $W_U(i\omega)$. We see that the critical frequency for satisfying the *Robust Stability*

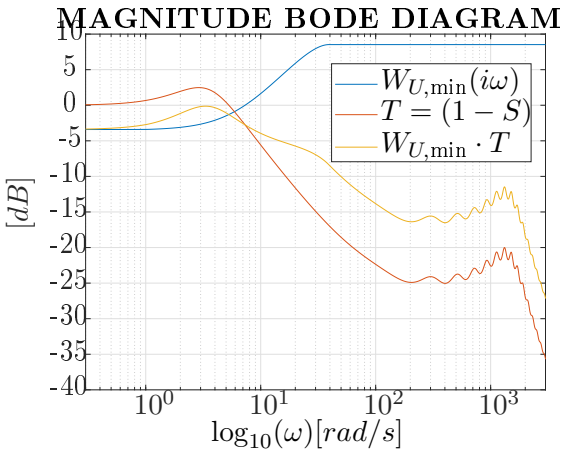
Condition are the low frequency up to $10\left[\frac{\text{rad}}{s}\right]$. Hence, if we modify the optimization algorithm adding a bigger weight for this critical frequency range, we should have a better calculation of the *Robust Stability Condition*: closer to the one made with $|W_{U,\min}(i\omega)|$. Below we show the Matlab code that performs this calculation:

Listing 5.1: Matlab code that solves the optimization problem to evaluate $W_{U,\text{opt}}$ or $W_{U,\text{opt}W}$

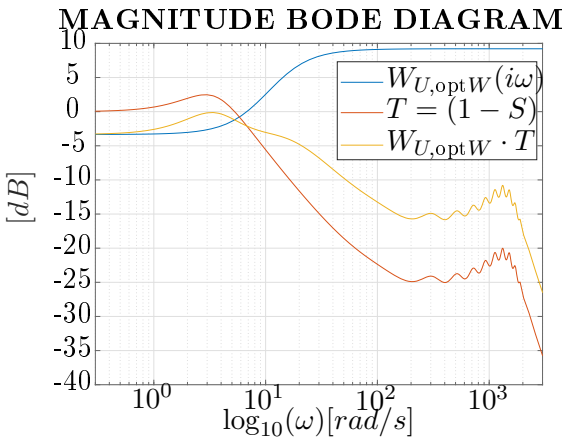
```
%% OPTIMIZATION PROBLEM TO GET W2IIorder:
prob = optimproblem('ObjectiveSense','min');
KK = optimvar('KK','LowerBound',W2min(1),'UpperBound',W2min(1)*10^(10/20));
WP = optimvar('WP','LowerBound',w_0dB_k,'UpperBound',w_max_k);
CSIP = optimvar('CSIP','LowerBound',0.1,'UpperBound',1);
WO = optimvar('WO','LowerBound',0,'UpperBound',w_0dB_k);
CSIO = optimvar('CSIO','LowerBound',0.1,'UpperBound',1);

syms w wp w0 csi0 csip GAIN real positive
W2OPT = GAIN*(((1i*w)/w0)^2 + 2*csi0*(1i*w)/w0 + 1)/...
            (((1i*w)/wp)^2 + 2*csip*(1i*w)/wp + 1);
W2OPT_abs = (sqrt(real(W2OPT)^2+imag(W2OPT)^2));

dec = 20;
w_vec_IAE = w_vec(1:dec:end);
W2min_IAE = W2min(1:dec:end);
IAE = subs(W2OPT_abs,w,w_vec_IAE)' - W2min_IAE;
% weighting vector for the critical frequency
Weight_IAE = ones(1,length(IAE));
for i=1:length(IAE)
    if w_vec_IAE(i)>0 && w_vec_IAE(i)<10 % w_critic interval
        Weight_IAE(i) = 10; % weight
    else
        Weight_IAE(i) = 1;
    end
end
IAE_con = optimconstr(length(IAE));
IAE_cost = optimexpr(length(IAE));
for i=1:length(IAE)
    IAE_fn = matlabFunction(IAE(i)); % @(GAIN,csi0,csip,w0,wp)
    IAE_con(i) = fcn2optimexpr(IAE_fn, KK, CSIO, CSIP, WO, WP) >= 0;
    IAE_cost_fn = matlabFunction(IAE(i)*Weight_IAE(i)); % ...
       @(GAIN,csi0,csip,w0,wp)
    IAE_cost(i) = fcn2optimexpr(IAE_cost_fn, KK, CSIO, CSIP, WO, WP);
end
prob.Constraints.Grater = IAE_con;
prob.Objective = IAE_cost'*IAE_cost;
clear x0
x0.KK = gain_0;
x0.WP = wP_0;
x0.CSIP = CSIP_0;
x0.WO = wO_0;
x0.CSIO = CSIO_0;
[x, cost] = solve(prob, x0)
```

In figure 5.5b is shown the weighting function $W_{U,\text{opt}W}$ obtained weight the frequency range $0 \leq \omega \leq 10\left[\frac{\text{rad}}{s}\right]$ ten times the other. We see that the fit at high frequency is worse than before but, the calculation of $\|W_U T(i\omega)\|_\infty$ is closer to the minimum value, see table 5.2. In table 5.1 are summarised all the weighting function parameters obtained with these two versions of the optimal algorithm, for two upper-bound of τ. Also the evaluation of the *Robust Stability Condition* is made for the two maximum value of τ and shown in table 5.2.

From table 5.2 we see that the *Robust Stability Condition* is satisfy for $\tau_{\text{MAX}} = 0.11[s]$ but not for $\tau_{\text{MAX}} = 0.15[s]$ as we expected looking at figure 5.3b. Hence, we can fixed the upper-bound for



(a) Bode plot of $W_{U,\min}(s)$, $T(s)$ and $W_{U,\min}T(s)$ in blue, red and yellow lines respectively. $W_{U,\min}(s)$ is computed with $\tau_{\text{MAX}} = 0.11[s]$.



(b) Bode plot of $W_{U,\text{opt}W}(s)$, $T(s)$ and $W_{U,\text{opt}W}T(s)$ in blue, red and yellow lines respectively. $W_{U,\text{opt}W}(s)$ is computed with $\tau_{\text{MAX}} = 0.11[s]$

Figure 5.6: Robust stability condition verification for $\tau_{\text{MAX}} = 0.11[s]$.

Table 5.2: *Robust Stability Condition* verification for two τ_{MAX} using different W_U

$\tau_{\mathbf{MAX}}[s]$	$\mathbf{W_{U,min}}$	$\mathbf{W_{U,opt}}$	$\mathbf{W_{U,optW}}$
0.11	0.9850	0.9871	0.9854
0.15	1.1144	1.1451	1.1236

the variable time delay to $\tau_{\text{MAX}} = 0.11[s]$. Thus, in table 5.3 we see that the frequency for having a time delay less or equal to the upper bound is around 2σ for the first four tests and around 3σ for the last six tests. The frequency values are computed evaluating the cumulative distribution function out of the density distribution function as it is shown in figure 5.7. Therefore, in the same condition of these ten tests, we can expect that the controller achieves the *Robust Stability Condition*.

Table 5.3: Values of the cumulative distribution function of the time delay at the upper-bound $\tau_{\text{MAX}} = 0.11[s]$ for the respective tests defined in table 3.2

TEST NAME	**CD at 0.11[s]**
Test1	0.9514
Test2	0.9684
Test3	0.9368
Test4	0.9552
Test5	0.9974
Test6	1
Test7	0.9988
Test8	0.9941
Test9	0.9926
Test10	0.9801

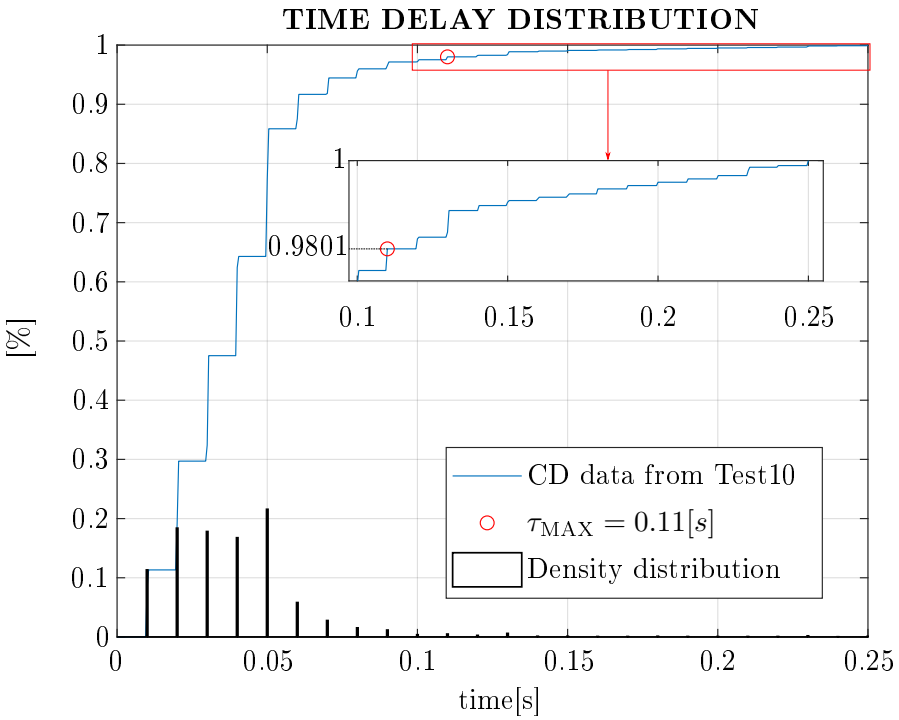


Figure 5.7: Time delay distribution for *Test10*, see table 3.2. In blue the cumulative function, in black the density and with a red circle the value of the cumulative distribution at the upper-bound $\tau_{\text{MAX}} = 0.11[s]$

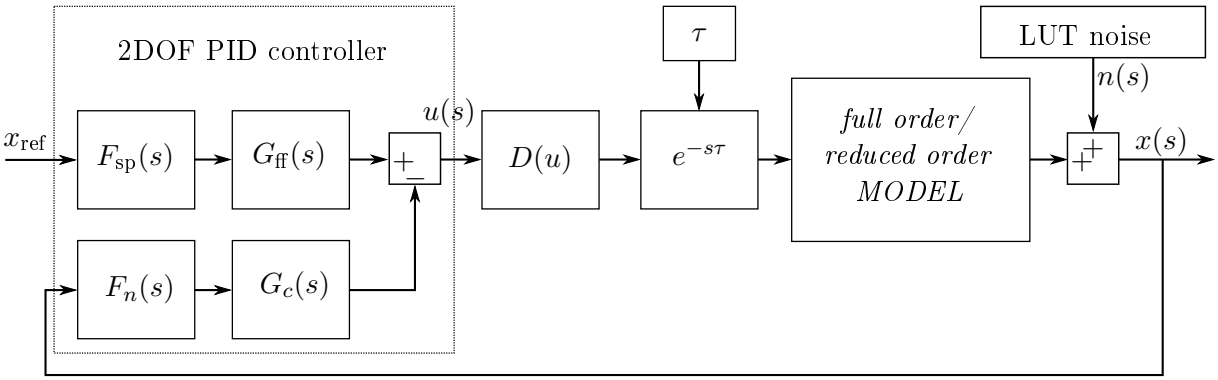
Chapter 6

Simulation results

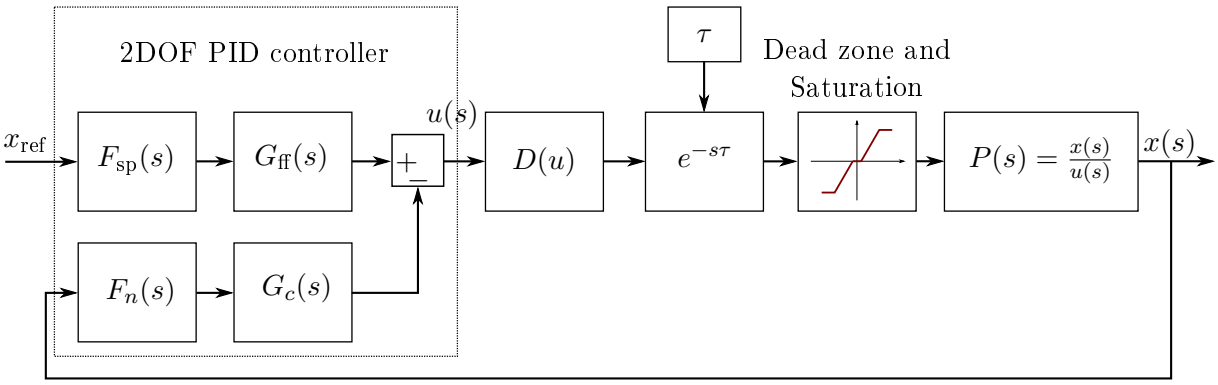
In this chapter, we show the numerical results obtained with such a designed controller. We use for the simulation the Matlab Simulink software. The model of the system is composed of the two hydraulic cylinder models developed in chapter 2: *Full-order model*, and *Reduced-order model*. The overall simulation model is still not complete because we have not yet represented the variable Round Trip Time and the measurement noise. We want to clarify these important points to analyse the results correctly. Hence, we discuss separately one by one.

6.1 Simulation models

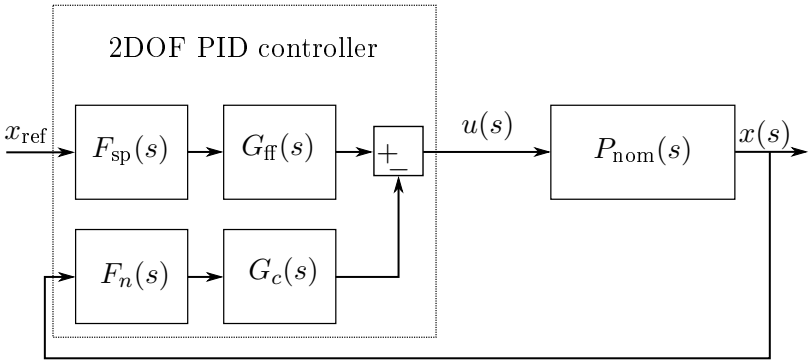
In the numerical simulations, we use at the same time, for each test, four models. The first two are obtained using the *Full-order model*, and the *Reduced-order model* to represent the hydraulic cylinder. We highlight that in the case of the *Full-order model* we choose to use the transfer function with less band-width to represent the directional control valve dynamic. With this choice, we are operating in the worst-case condition, in fact, less band-width means less phase margin in the control system hence, the system is closer to the instability. The block diagram of the overall control system for these two models is shown in figure 6.1a. We see that we introduce in the control loop the noise signal $n(s)$ and the time delay $e^{-s\tau}$. It is shown the time delays τ is the input of the delay transfer function just to remark that τ is a time variable. The two models obtained in this way are called respectively with the same name of the cylinder representation: *Full-Order Model*, and *Reduced-Order Model*
With the name: *Linear Model* we mean the same control scheme as the previous but, with the transfer function of equation (2.29). Thus, in this case, we have a linear model linearised around an operative point. We define this transfer function as uncertain thanks to the *Robust Control Toolbox*, see [5]. In this way, the point is randomly chosen by the program. The overall control structure is shown in figure 6.1b. We point out that there is no more noise signal but still the variable time delay because the latter is not included in the function of $P(s)$ (2.29).
Lastly, we call *Nominal Model* the implementation of the scheme shown in figure 6.1c. In this last case, the control system is composed solely by the 2DOF controller without dead-zone compensation, and the nominal transfer function $P_{\text{nom}}(s)$ (3.14). We see we avoid the insertion of the variable time delay because we have already considered inside the process transfer function. The reason behind the choice to use these four models for the simulations is simple: with the *Full-Order Model* we want to have the best representation of the real system possible. With the *Reduced-Order model* we can evaluate the goodness of the symmetry approximation. With the *Linear Model* we can see the influence of the linearisation point used for the *Orifice Equation*. With the *Nominal Model* we want a confirmation of the performance that we expect from the controller design.



(a) Block diagram with the *Full-Order Model* or the *Reduced-Order Model*.



(b) Block diagram using a *Linear Model* with random values of the parameters C_q, C_{qp}, τ still in the already defined ranges.

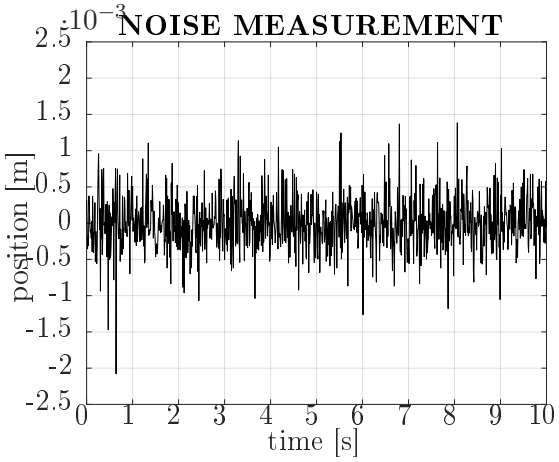


(c) Block diagram using the *Nominal Model*.

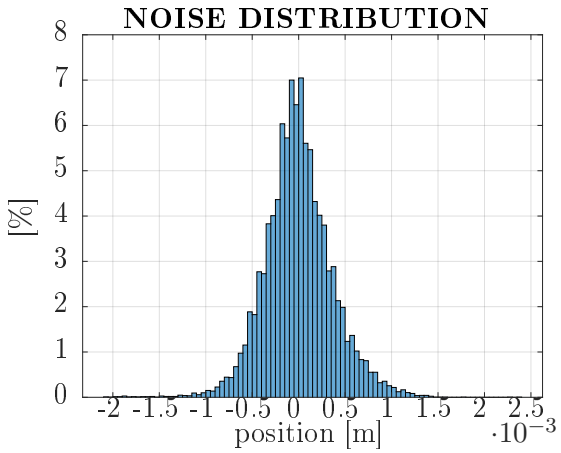
Figure 6.1: Block diagram of the models used in the simulations.

6.1.1 Measurement noise

For the *Full-Order Model*, and the *Reduced-Order Model* we have to generate a noise signal to use during the simulations $n(t)$. We can achieve this purpose with two strategies. One is to use directly the measurement noise data collected during the experiments. The experiment has to be longer than the simulation time because it has to avoid the repetition of the signal. In this case, once we have chosen $n(t)$, we keep the same value for all the simulations. Another approach is to estimate the statistical properties of the noise and generate $n(t)$ during the simulation run. In this manner, we always have different noise data.
We choose the first approach because it is easier and guarantees the repeatability of the results. Furthermore, avoid approximations and assumptions on the statistical distribution. We take the set of data collected with the experiment previous called *Test3*, see table 3.2. In this test, the true piston position is fixed because the piston is at the end of the stroke. Hence, we get directly the noise signal $n(s)$ out from the position measurement after taking out the mean value. We have more than $210[s]$ enough to perform all the necessary simulations. Figure 6.2a it is shown the first seconds of such computed noise signal. The frequency distribution of $n(t)$ is evaluated and shown in figure 6.2b. As expected, the distribution presents the typical Gaussian-shape. Indeed, the measurement noise is the result of many aleatory processes. First of all, electromagnetic interference affects the electric signal. Therefore, applying the *Central Limit Theorem* we get a Gaussian process. Finally, we see that the noise amplitude is inside the range of $-2.5 \cdot 10^{-3} \leq n(t) \leq 2.5 \cdot 10^{-3}[m]$, so produce a relative error for a step size of $10[cm]$ of $\frac{2.5 \cdot 10^{-3}}{0.1} = 2.5\%$.



(a) Rod end position measured at the beginning of the experiment. The piston is fixed. Data from experiment *Test3*.



(b) Position measurement noise frequency distribution. Data from experiment *Test3*.

Figure 6.2: Position measurement noise in experiment *Test3*, see table 3.2.

6.1.2 Time delay model

Regarding the representation of the time delay, we have the same possibilities as the noise signal. Indeed, we can generate the signal τ both with a set of measured data and using the γ-*distribution* found in chapter 3.2. In this case, we chose the second option because we want to get a general result and not a particular one. Indeed, the value of the time delay strongly changes the performance achieved and using a fixed set means to observe only one particular response without any aleatory.
Thus, for this reason, we generate the τ signal with a γ-*distribution* obtained with the parameters of table 6.1. The parameters come out from the fit of the time delay data computed in the experiment *Test10*. We choose *Test10* because it represents the goal operative condition of the control system.
We implement the transfer function $e^{-s\tau}$ with the *Variable Transport Delay* block. The block

Table 6.1: Parameters of the γ-*distribution* that fit the time delay of *Test10*.

TEST NAME	**a**	**b**
Test10	2.91554	0.013567

uses the Padè approximation and has a sample time of $0.01[s]$ because this is the communication sample time. The result of such implementation is shown in figure 6.3. With a dashed line, we see the input signal. That signal is shifted to obtain the output signal with a dot-dashed line. We highlight that the variable time delay can cause the losses of some data points. In reality, this happens because if the delay of the previous data point plus the sample time is larger than the delay of the next data point, then the previous is overwritten by the new one and it does not produce an effect on the control. The shifted signals observed in the ten experiments performed for evaluating the time delay, see 3.2, present the same shape as the one in figure 6.3, see also the experimental result for comparison.

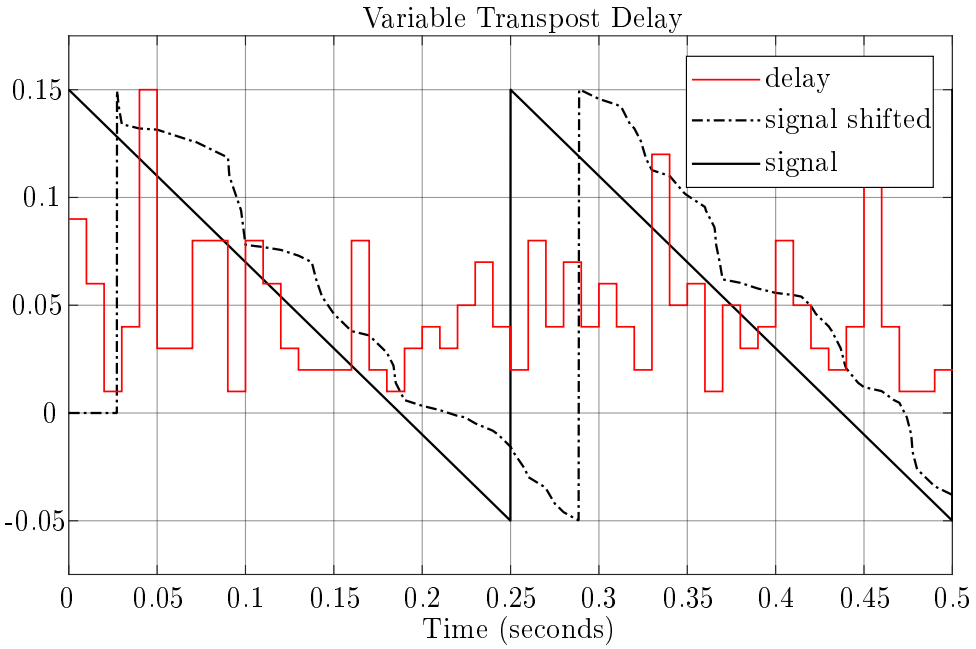


Figure 6.3: Effect of the variable transport delay in the simulation for a demonstrative signal.

6.2 Numerical tests

For clearness, we list all the evaluated simulations to remark the conditions and the controller used. We will refer to a particular numerical test with a label shown in table 6.2 under the column *Simulation Name*. In the same table are listed the time delay condition used for the test, for instance, with γ-*dist* we refer to the implementation discussed in the previous section. Evaluating the response of the control system to a fixed time delay is worthy. We are going to set $\tau = 0.0005[s]$ because it is the sample time of the system interface, *Speedgoat* in our case, that is the minimum time delay possible, achievable only neglecting the TCP-IP communication and implementing the controller directly in the interface. We are also going to set $\tau = 0.03[s]$, $\tau = 0.11[s]$, and $\tau = 0.15[s]$ that are respectively the nominal time delay, the upper-bound, the value where the *Robust Stability Condition* is not satisfy. Fixing the time delay value will not make the system more likely, but allow us to make important observations regarding the system stability. Furthermore, the simulation tests differ for the controller implementation. We have seen in section 4.4.2 that we can design the controller to reduce the overshoot through the set-point parameter b, and a reference pre-filter F_{sp}. Then, there are three interesting combinations to evaluate: first, the control with both pre-filter, so noted in the table with *yes*, and set-point

parameter $b = 0$. Secondly, the control without pre-filter, so noted in the table with *no*, and set-point parameter $b = 0$. Lastly, we design the controller without the set-point tricks, so there is no pre-filter and $b = 1$. For all cases, we are going to test the response to a step in the position reference. The reference wave is designed with a square shape for all the tests except the ones with $\tau = 0.15[s]$. The square wave has a first initial step from 0 to $0.05[m]$ to operate than in the middle of the piston stroke. Then, for each $5[s]$, there is a step forward and back of $0.1[m]$ of amplitude. For the last three simulations, so from 13th to 15th, the reference position signal is only one long step of $10[s]$ because we expect an unstable response, or at least very oscillatory.

Table 6.2: Evaluated simulations and names

Simulation Name	τ	**set-point parameter b**	**set-point filter $\mathbf{F_{sp}}$**	**test type**
simulation1	0.0005	0	yes	square wave
simulation2	0.0005	0	no	square wave
simulation3	0.0005	1	no	square wave
simulation4	0.03	0	yes	square wave
simulation5	0.03	0	no	square wave
simulation6	0.03	1	no	square wave
simulation7	γ-dist	0	yes	square wave
simulation8	γ-dist	0	no	square wave
simulation9	γ-dist	1	no	square wave
simulation10	0.11	0	yes	square wave
simulation11	0.11	0	no	square wave
simulation12	0.11	1	no	square wave
simulation13	0.15	0	yes	step
simulation14	0.15	0	no	step
simulation15	0.15	1	no	step

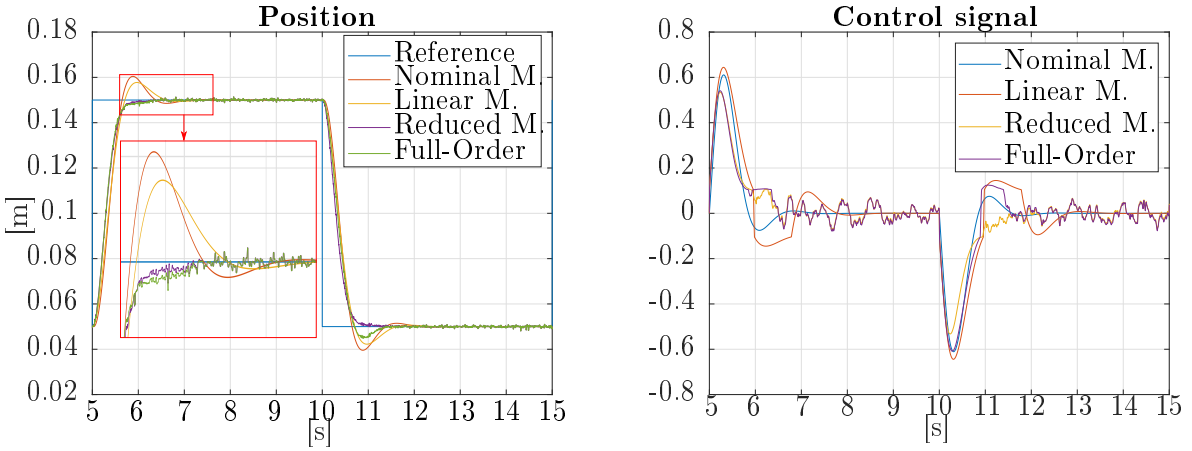
As previously said in section 3.1, we highlight that in the simulation model, in both the *Full-Order Model*, and the *Reduced-Order Model* there is an algebraical loop caused by the pressure feedback, P_A, and P_B in the first case, and P_L in the second. Such a loop can drive into non-physical pressure values, for instance, greater than the supply pressure. In particular, the model works perfectly for a simple step response but could be problematic if the system has to follow a sawtooth trajectory. To solve this problem we limit the value of P_A, and P_B, to $\pm P_S$ for the *full order model*, and $0 \leq P_L \leq P_S$ for the *reduced order model*. Furthermore, we see that a fixed step size solver guarantees more likely results, in particular, we use *ODE3* with a sample time of $0.0001[s]$.

6.2.1 Without time delay

Here we group all the simulations evaluated with the lowest time delay possible $\tau = 0.0005[s]$, so the first three tests are listed in table 6.2. In figure 6.4a is shown the step response for the test called *simulation2*. We see that for all the evaluated models the time constant remain the same because all the responses have the same rise time around half a second. On the other hand, while the *Nominal Model*, and the *Linear Model* show the expected overshoot, the *Reduced-Order Model*, and the *Full-Order Model* have almost no overshoot. This behaviour is not weird, indeed, the result pointed out the great difference between the two friction models used, the linear one in the first two cases and a more complex and non-linear one for the last two. In particular, if we remember how was evaluated the linear friction parameter in the section 2.3.2, we know that at high speed the linear model produces a bigger counteracting force than the so-called *Stribeck friction model*. The result is a slightly faster response at the beginning of the step. Then, when

the speed goes down, the Stribeck model makes almost twice the force value, producing the stiction effect. In this case, this phenomenon gives a positive contribution because it reduces the settling time. In the results, it is also evident the asymmetry of the *Full-Order Model*. Indeed, we get different step responses for the forward and back directions. In particular, for the step back the pressurised chamber faced with a bigger piston area, then in the other case, because of piston plunger presence: $A_A > A_B$. Thus, the control produces a greater force that causes a little overshoot.

In figure 6.4b are shown the control signals acquired out of the controller. We remark that the amplitude of the signals confirms the observations regarding the friction model and the asymmetry of the *Full-Order Model*. Furthermore, we can see some oscillation of amplitude $\simeq 0.1$ for the non-linear models. These are caused by the combination of the noise signal injected $n(t)$, and the feed-forward compensation introduced by the controller. Indeed, we have seen that the noise amplitude arrives at $2.5 \cdot 10^{-3}[m]$, then with only the proportional part of the PID controller $k_p \simeq 12.7$, we get a control signal of $2.5 \cdot 10^{-3} * 12.7 = 0.03175$. The obtained value is one order of amplitude greater than the threshold of the dead-zone compensation 0.0042, see equation 4.35. Hence, even though the noise is filtered, we easily go over the dead-zone compensation threshold then, we get a value over 0.1.



(a) position reference and response signals of all the models.

(b) control signals applied to all the models defined.

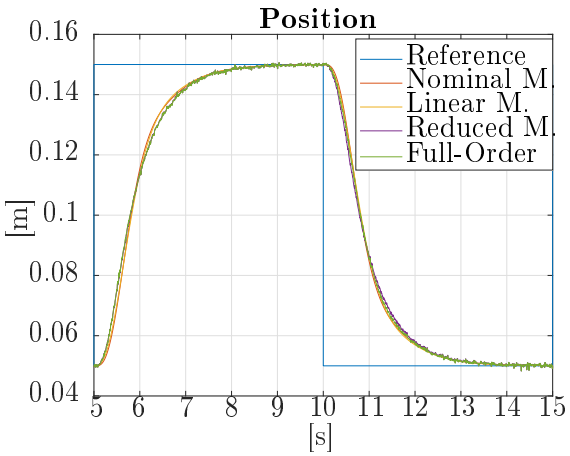
Figure 6.4: Results of the simulation called: *simulation2*

We report next to the simulation in the same time delay condition but with the controller with the set-point filter inserted as first, and then without any set-point trick. We are talking respectively about *simulation1*, and *simulation3*. In particular, in figure 6.5a, and 6.5b we can see the results for the first, and in figure 6.6a, and 6.6b we can see the results for the second. In the first case, we obtain a slower step response, caused by the insertion of the set-point filter that has dominant dynamics. For the same reason, all the position signals have a very close trajectory. Regarding the control signal, the observations are the same as in the previous case. In the *simulation3* the main idea is to neglect the set-point response design to evaluate the benefits. We see that the step response has a big overshoot, around 50%. Looking at the control signals in figure 6.6b we see that go over ± 1 that is the saturation level. Then, wind-up phenomena probably increase the overshoot amount.

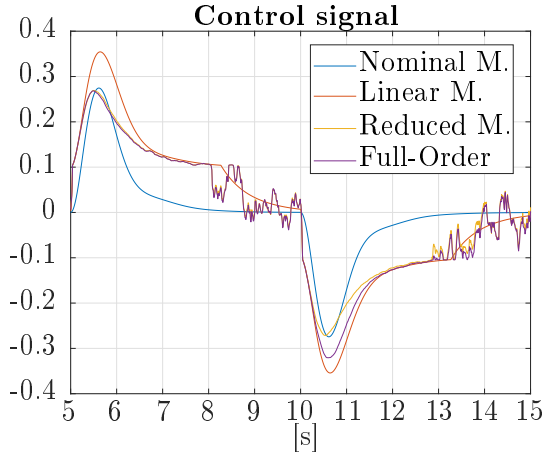
After this first set of simulations is evaluated in the best condition in terms of time delay, we can make some considerations:

1. all the models show a very similar dynamic because they have the same rise time and a very close settling time;

2. the friction non-linearity increases the performance in the case $b = 0$, and $F_{\text{sp}} = 1$, resulting a response without overshoot and with almost no oscillation;

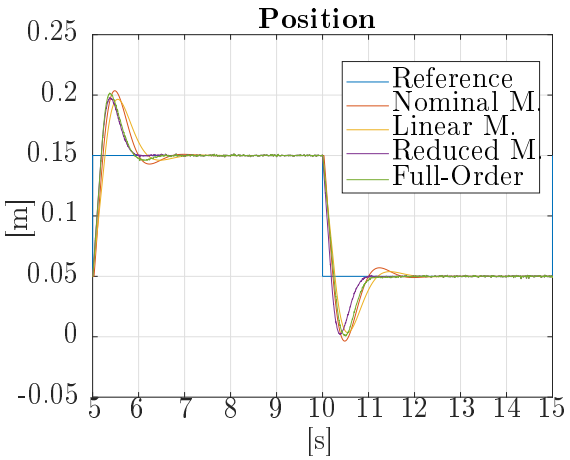


(a) position reference and response signals of all the models.

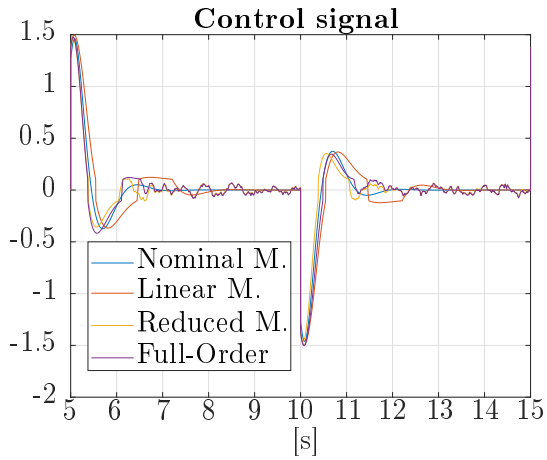


(b) control signals applied to all the models defined.

Figure 6.5: Results of the simulation called: *simulation1*



(a) position reference and response signals of all the models.



(b) control signals applied to all the models defined.

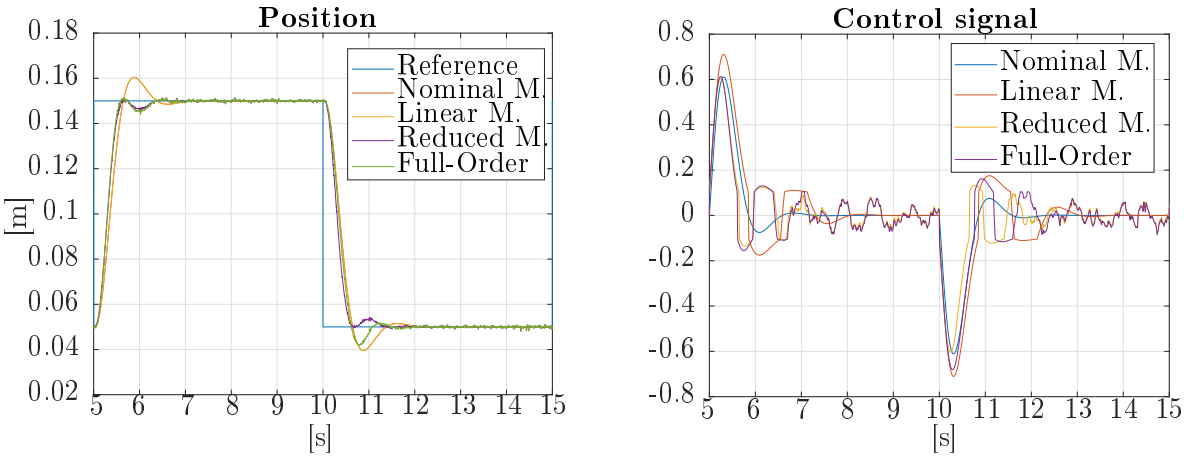
Figure 6.6: Results of the simulation called: *simulation3*

3. the controller designed without set-point tricks, so in *simulation3*, gives an unacceptable response.

We have seen that the first controller design, so with the choice of $b = 0$, and $F_{\text{sp}} = 1$ gives the best step response. At the same time, both the simulations *simulation1* and *simulation3* present strong drawbacks: very slow control for the first or too aggressive for the second. Then, it will be much more interesting in future tests to focus the analysis on the first controller design.

6.2.2 Nominal time delay

In this case, the simulations are evaluated with a fixed time delay set to the nominal value $\tau = 0.03[s]$. The results of a two sides step are shown in figure 6.7a, and figure 6.7b. We see almost the same response from the models, and all the observations made before are still valid. On the other hand, for the non-linear models appears an oscillation while the system is trying to reach the reference position. With a close look, we can see that also the nominal model increased slightly the undershoot.



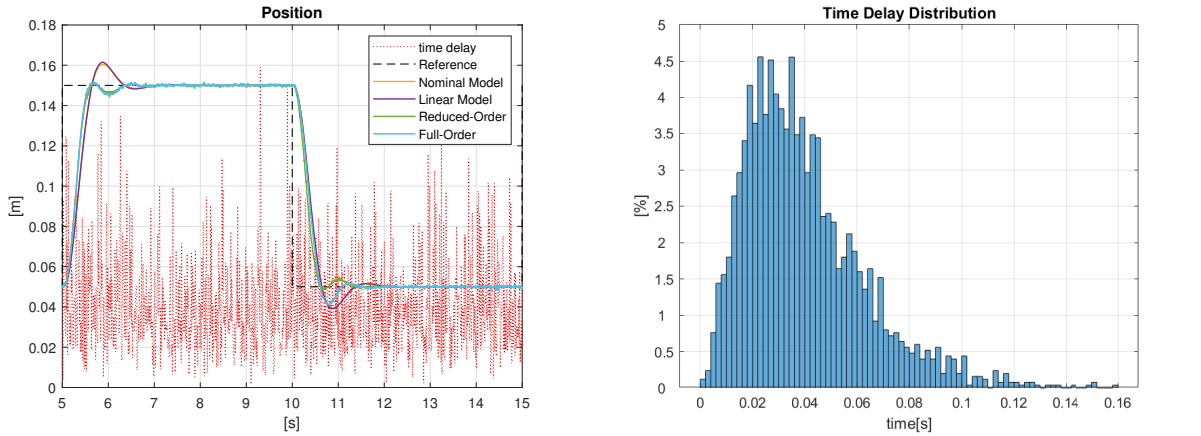
(a) position reference and response signals of all the models.

(b) control signals applied to all the models defined.

Figure 6.7: Results of the simulation called: *simulation5*

6.2.3 Gamma distributed model of time delay

Now we point out the result obtained with the γ-*distributed* model of the time delay discussed before. In particular, as the first attempt, the distribution is computed without quantization so basically, the time delay could have a sub-multiple value of the sampling time $0.01[s]$, and also could be 0. In figure 6.8a we see that the step responses of all the systems are still very close to *simulation5*, and it is confirmed by the shape of the control signal in figure 6.9a. Indeed, we remember that the γ-*distribution* was fitted over a data set with the mean value around 0.03, that is the nominal value. In figure 6.8b we see the frequency distributions of the time delay of the simulation.

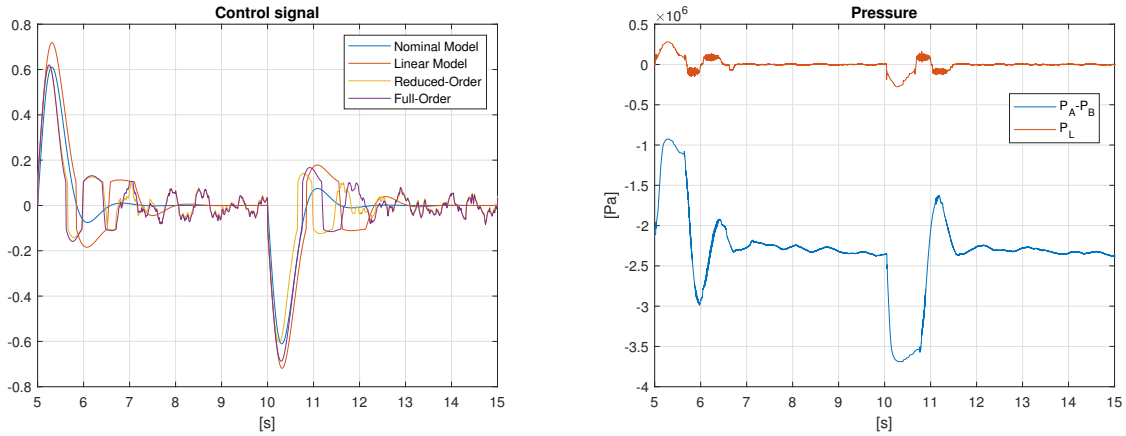


(a) position reference and response signals of the all models in the test *simulation8*. With dotted red line is sown the time delay in seconds of the relative sample point.

(b) continuous γ-*distribution* of the time delay in *simulation8*

Figure 6.8: Results of the simulation called: *simulation8*

It is interesting to see the behaviour of the load pressure P_L for the *Reduced-Order Model* or equivalently the computation of $P_A - P_B$ for the *Full-Order Model*, see figure 6.9b. There is a quite large difference between the two signals due to the asymmetry. What is interesting to note is that the oscillations at a steady-state are very tiny. Furthermore, the assumed values are far below the maximum allowed: $P_S = 1 \cdot 10^7[Pa]$. The latter guarantees that the assumption made during the uncertainty analysis $P_{L,\text{MAX}} = 0.95 \cdot P_S$ is valid.
Finally, for completeness, we evaluate the same simulation (*simulation8*) with a more likely time

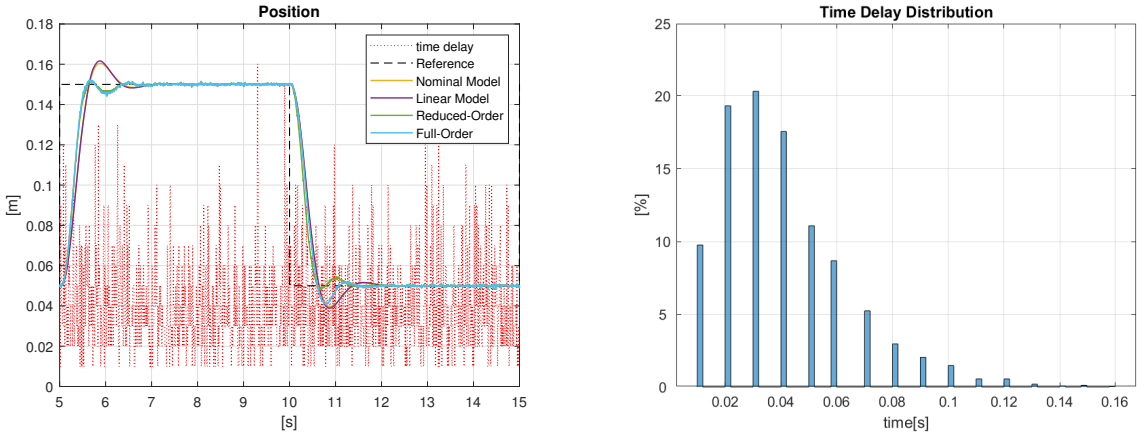


(a) control signals of all the models

(b) load pressure signals for the non-linear models

Figure 6.9: Internal signals in the models during the simulation called: *simulation8*

delay distribution. In fact, as we said, the time delay can have only multiple values of $0.01[s]$, and as minimum $\tau_{\min} = 0.01[s]$. Then, the continuous distribution can be corrected for having quantized values and a minimum of $0.01[s]$ instead of 0. The result with such a procedure is shown in figure 6.10a. In figure 6.10b is shown the obtained quantized γ-*distribution*. As expected, the result is exactly the same as with the continuous distribution.



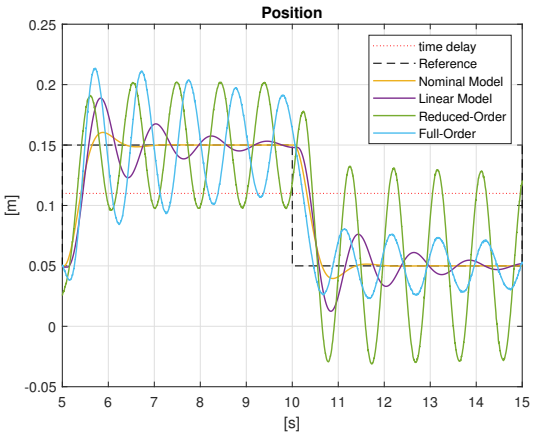
(a) position reference and response signals of the all models. With dotted red line is sown the time delay in seconds of the relative sample point.

(b) discrete γ-*distribution* of the time delay, the minimum value is $0.01[s]$

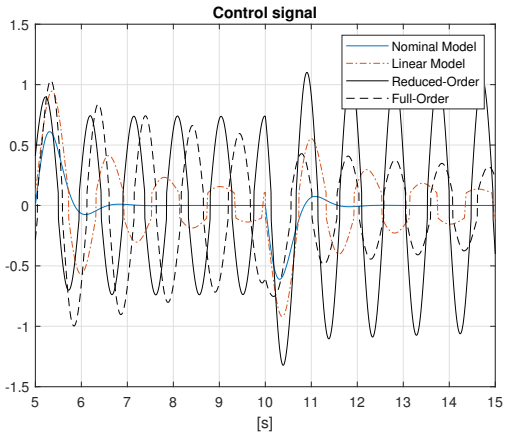
Figure 6.10: Results of the simulation called: *simulation8bis*

6.2.4 Maximum time delay

Even though we have designed the simulations with the γ-*distribution* model of the time delay for being most likely possible, it is interesting to verify the *Robust Stability Condition* evaluated at chapter 5. To do that, we first simulate the models with the fixed maximum time delay $\tau_{\text{MAX}} = 0.11[s]$, and then we repeat the test with a greater value. We choose $\tau_{\text{MAX}} = 0.15[s]$ for the second attempt because it violates the *Robust Stability Condition*, as we have seen in chapter 5. In figure 6.11a, and 6.11b we see that the close loop system is still stable with the maximum time delay, but the response is highly oscillatory. It is clear that the system can not operate in such conditions.
In figure 6.12a, and 6.12b is shown the results of *simulation13*. In particular, we highlight that this simulation is obtained with both the set-point tricks: set-point filter inserted and set-point parameter set to 0. We observe the system instability, even though it is in the least aggressive



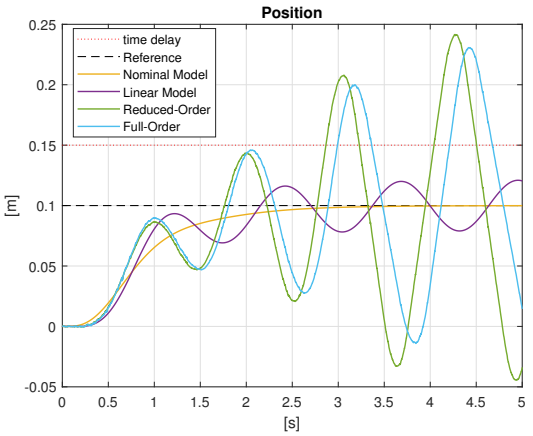
(a) position reference and response signals of all the models. With dotted red line is sown the time delay in seconds



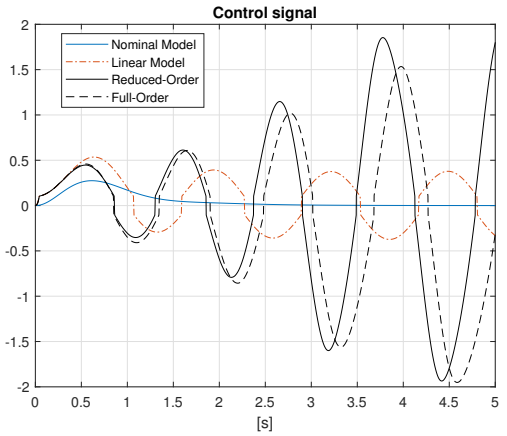
(b) control signals of all the models

Figure 6.11: Results of the simulation called: *simulation11*

configuration. Then, the time delay $\tau_{\text{MAX}} = 0.15[s]$ can not guarantees the stability, as we have concluded in chapter 5.



(a) position reference and response signals of the all models. With dotted red line is sown the time delay in seconds



(b) control signals of the all models

Figure 6.12: Results of the simulation called: *simulation13*

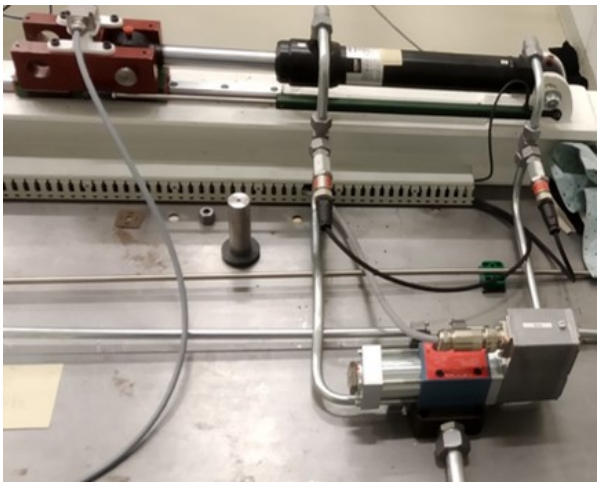
After all the simulations evaluated, we can make some conclusions:

1. All the models give the expected results, which means that the reduction and linearization process is consistent.
2. We achieve good performance at least till the γ-*distributed* model.
3. The choice of *Multiplicative Perturbation* for describing the gain uncertainty of the model is a good approximation and drives to correct results.

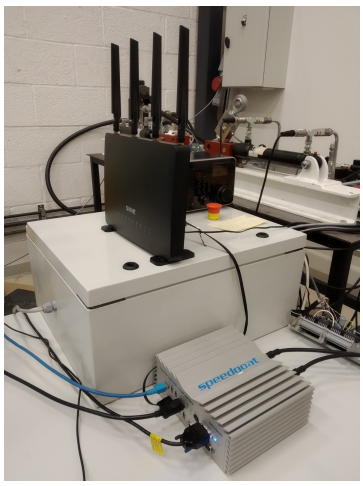
All these conclusions encourage us to continue with the implementation of the designed controller in the real system under study.

Chapter 7

Experimental set-up and control implementation



(a) hydraulic system under tests. The cylinder, the directional control valve and the linear encoder are visible.



(b) system interface components: the RT Speedgoat micro-controller, the power unit, and the first WIFI router for the TCP-IP connection.

Figure 7.1: Experimental set-up: the hydraulic system and the interface. The remote controller is not shown.

In this chapter, we point out how we practically set-up the controller of the system shown in figure 7.1a. In particular, we focus on the implementation of the remote control through a TCP-IP communication with a WIFI connection or an Ethernet connection. Furthermore, we explain how the time delay is evaluated to make the analysis at section 3.2 possible. In the end, we describe the operative system states focusing on the initialization procedure necessary for the synchronization of the communication channel, the homing procedure to get the absolute position value, the control strategy to guarantee safety during the operation.

In table 7.1 are listed the devices used in the control system. In particular, we remark that the system interface supports the Matlab Simulink environment. It is possible to program Speedgoat with a host computer thanks to the application always provided by Matlab: Simulink-Real-Time-Explorer. Therefore, in Speedgoat and in the host computer is installed Matlab R2018a, while in the controller computer is installed Matlab R2021a. There are compatible problems because the two devices (Speedgoat and the laptop) are running separately two programs, and they only communicate signals with the TCP-IP protocol.

Description	Model number
Cylinder	D633 R16KD1M0NSM2
Moog servo valve	CD25-40 25x200-SS-HC-SSN-NNN
Celesco linear-pot.	CLP-250
System interface	Speedgoat, Baseline real-time target machine-S(4043)
WIFI routers	Svive Cirrus Router
Controller laptop	Asus VivoBook S530F

Table 7.1: Installed components of experimental system, [27]

7.1 Implementation of the TCP-IP communication

In figure 7.2 we see the scheme of the all control system. From the hydraulic system, we see the two connections for getting the control signal and for sending the position measured to a power drive. The power drive was designed and built in a previous job. Basically, the aim is to transform the control signal acquired from Speedgoat into a power signal. Speedgoat has two Ethernet connections. The first is with the host device, so we will call *host port*, the second is with the first WIFI router, therefore we will call *router port*. Then, the first router shared data with the second router through a WIFI bridge. Finally, the second router is connected to the laptop where the control program is running through an Ethernet connection. To be precise, the laptop we have used has not the Ethernet port, so we use an Ethernet-USB adapter but, this does not change in any way in the following discussion. To the best of our knowledge, there is not a similar configuration already established with the Speedgoat device in the literature. We can find some useful information on how IP addresses work in the Acromag white-paper [7].

First of all, with the host device, we built a program that receives the sensor information and sends the control signal to the power device. For these purposes, we use the specific blocks of analogue input and output provided by the Speedgoat Real-Time Libraries. The same libraries also provide the blocks for the TCP-IP communication, called TCP Send and TCP Receive, with the obvious meaning. These blocks are configured using a third block called TCP Server Configure that needs the network information and the parameters of the slot PCI (Periferica Component Interconnect) in with the used network card is installed. In particular, we need the IP address, the network Mask, Gate, port, and for the PCI slot, we need the Bus number, the Slot number, and the Function number. To set a proper IP address, we choose 30.30.30.1, so a compatible network mask and gate are respectively 255.255.255.0, and 0.0.0.0. To get the PCI information of the used Ethernet port, we use the command: `speedgoat.showInstalledIoModules('Type', 'Ethernet')`, and with some communication attempts we can find the correct numbers. Finally, we choose the port number 5027.

Similarly, in the controller computer, we implement the TCP-IP communication with some Simulink blocks. We find some useful information about the block on [40]. In this case, the blocks are provided by the Instrument Control Toolbox. In these blocks, we have only to set the IP address with which we want to establish the connection, so 30.30.30.1, and the port number 5027. Furthermore, we have to set the network address of the laptop properly so that is in the same network as Speedgoat. The controller computer is running Windows10 home, so this operation is easily done by changing the configurations in *Ethernet properties* → *Protocol Internet (TCP/IPv4) Properties*. We have to choose a fixed IP address instead of automatically getting it. A proper choice is IP: 30.30.30.2 and subnet mask: 255.0.0.0. The WIFI connection between the routers is obtained by implementing a bridge. We have to manage the router's configurations, but the details are left because they can be easily found in the literature.

To sum up, in figure 7.2 we can see also the details regarding the Network address and information of each device port.

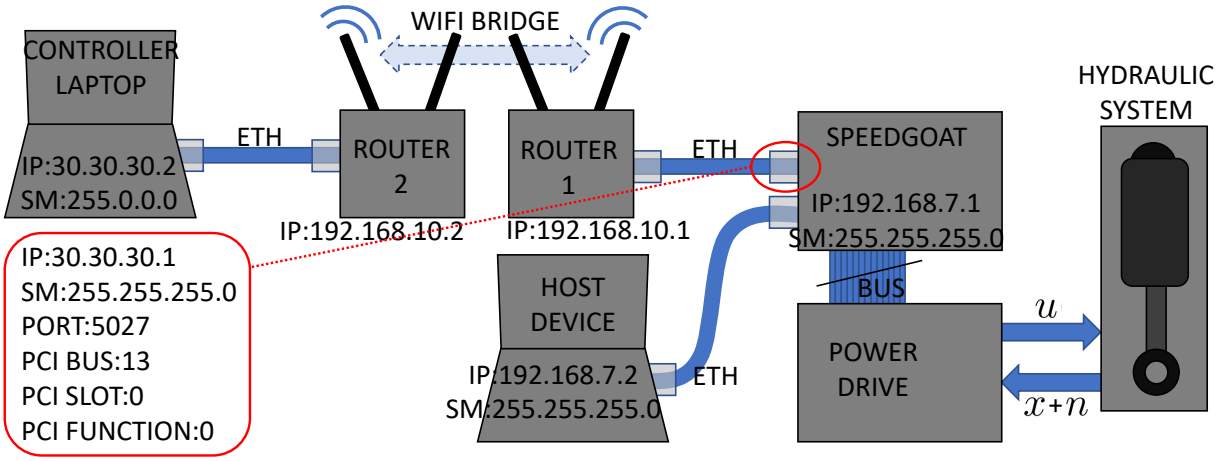


Figure 7.2: Connections diagram on the control system. The network address is highlighted in each device.

7.2 Time delay evaluation

In section 3.2 we have defined the *Round Trip Time* (RTT), and we have analysed the distribution of the data acquired in several tests. Now we want to explain how we to get this measure most consistently.
Always in section 3.2, there is the scheme of figure 3.3 in which we can see that the RTT affects all the signals that make the round trip between the controller (laptop), and the system interface (Speedgoat). Then the idea shown in that figure 3.3 is to generate another signal on Speedgoat, let it make the round trip and compare the *signal sent* with the *signal received*. Basically, we can get the time delay from the phase shift but, there is a simpler way. Thence, in this section, we see two approaches for evaluating the time delay. Both of them use the same signal, but while we compute the first real-time on Speedgoat, we calculate the second with a data post-process. The additional signal has to be useful for an easy phase shift comparison. Hence, we choose a saw-tooth signal of amplitude 10, and period $10[s]$, with the lowest value at 1, see figure 7.3. The idea behind the choice of the saw-tooth signal is to simplify the data samples identification. Indeed, we only have to detect the signal at the same amplitude. In figure 7.3 we see the saw-tooth sent and received by Speedgoat. We see that the time delay is detected even by a simple look.

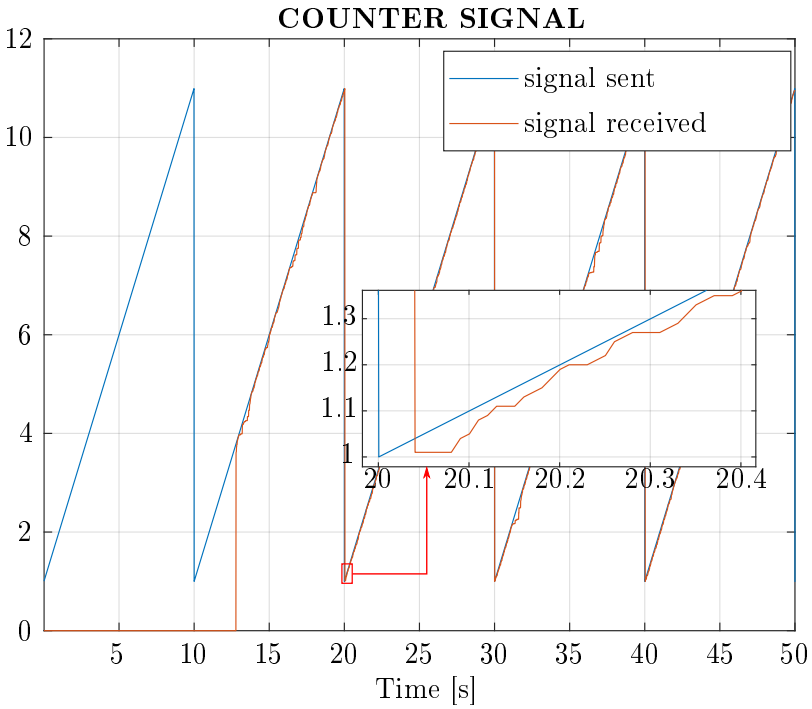


Figure 7.3: Counter signal used to evaluate the time delay sent and received by the server.

7.2.1 Real Time evaluation

It is necessary to compute the time delay with a real-time process because the safe operation of the system, as we have seen, strongly depends on the time delay in the communication. If the time delay goes upper the bound for too long, the system becomes unstable. Thus, we have to monitor constantly the amplitude of the delay to stop the system under certain conditions later explained.
Looking at the top of figure 7.4 we see a zoom of the same signals of figure 7.3. We can recognise the sample time of the communication because the received signal change value is each $0.01[s]$. Moreover, we see that not all the sample points are received by Speedgoat, for instance, at $61.03[s]$ the value is the same as $61.02[s]$, and only at $61.04[s]$ it changes. We guess that this happens because the time delay of the sample at $61.03[s]$ is bigger than the next one, so it is overwritten by the next. This phenomenon makes the online calculation more complex because not all the data points have a respective received value. Then, the simplest way to compute the time delay is to use the property of the designed saw-tooth: there is a 1 to 1 ratio between the time axis and the amplitude axis. Hence, we can compute the delay by simply subtracting the signal received to the signal sent at each time sample. This approach is evident by looking at figure 7.4. Below is reported the function implemented in the program running on Speedgoat.

Listing 7.1: Real-Time evaluation function to compute the time delay

```
function dT = TimeDelay(count_sent, count_received, enable)
if (enable~=0) && (count_received~=0)% condition to have the communication
    dT = count_sent-count_received; % because the counter slope is 1
    if dT < 0
        dT = dT + 10; % 10 is the amplitude of the counter
    end
else
    dT = 0;
end
```

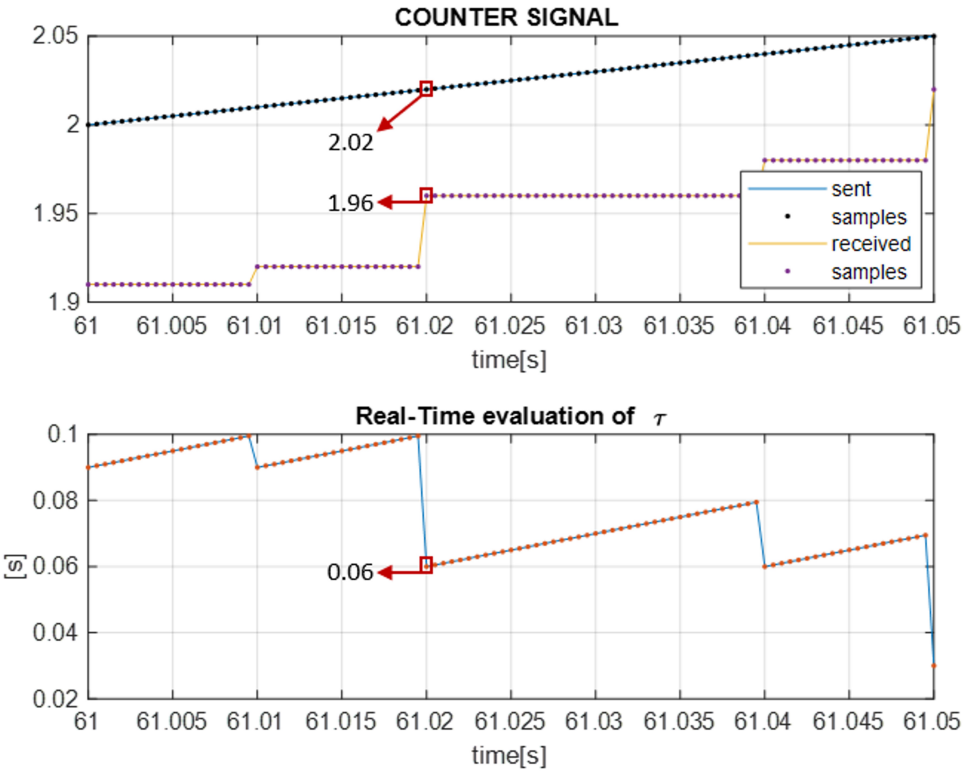


Figure 7.4: Real-time evaluation of the time delay using the difference of the counter signal amplitude.

7.2.2 Evaluation form data post processing

As we said, some data points are overwritten by the following ones, so it is easier to use the amplitude difference of the signal sent and received. Unfortunately, this is not exactly the same thing. We can understand the difference looking at figure 7.5. We see that all the points from $61.05[s]$ to $61.07 - 0.0005[s]$ have the same value (2.02). Then, we can compute the delay by the amplitude subtraction. We see that we get an increasing function. The time delay calculation is different if we think about the time shift of the same data points. The first point, the one at $61.02[s]$ has a time shift of $0.03[s]$, then the next one, so at $61.02 + 0.0005[s]$, has a time delay greater of the previous one, instead of lower. Indeed, the delay of the point at $61.02 + 0.0005[s]$ is the time of the sample point that overwrites it, so $61.07[s]$. Thus, we have a delay of $61.07 - (61.02 + 0.0005) = 0.0495[s]$. We know that the difference between the two approaches is very little. Nevertheless, having the best possible set of measures is very important because we want to fit the statistical distribution. Furthermore, this calculation could be done by running an off-line program, basically, post-processing the two counter signals. We highlight that we use the time delay real-time evaluated only for checking the safety operation of the system and the time delay computed with the post-process of the data for all the rest.

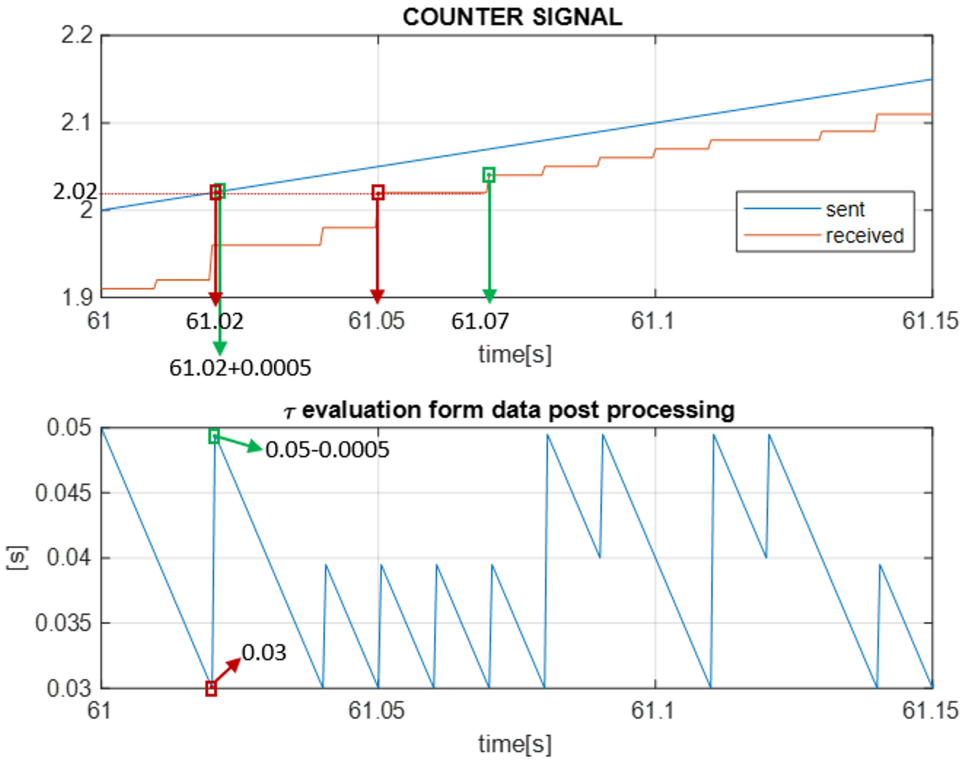


Figure 7.5: Evaluation of the time delay using the phase shift of the counter signal.

7.3 Operative system states

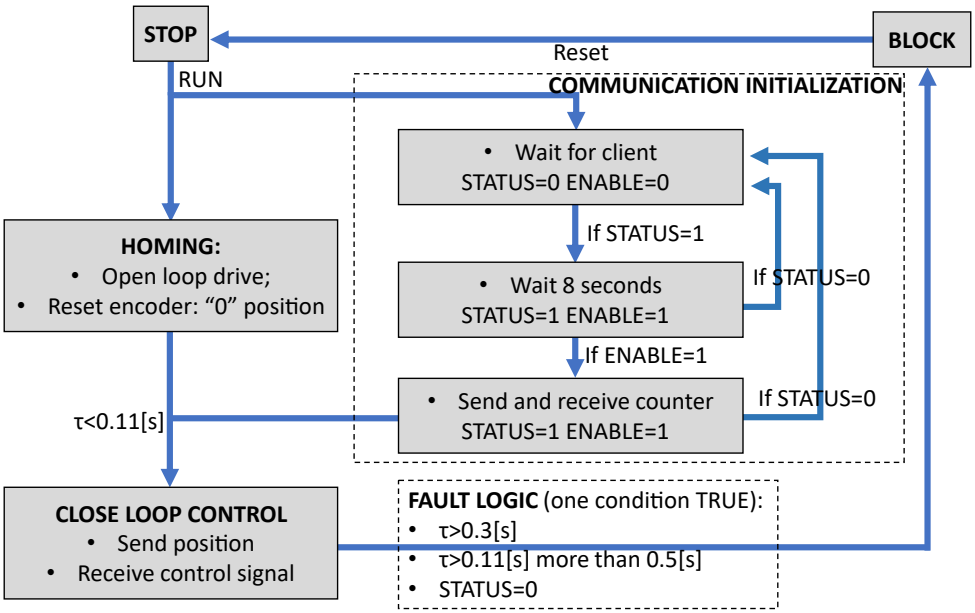


Figure 7.6: Block diagram of the control system states.

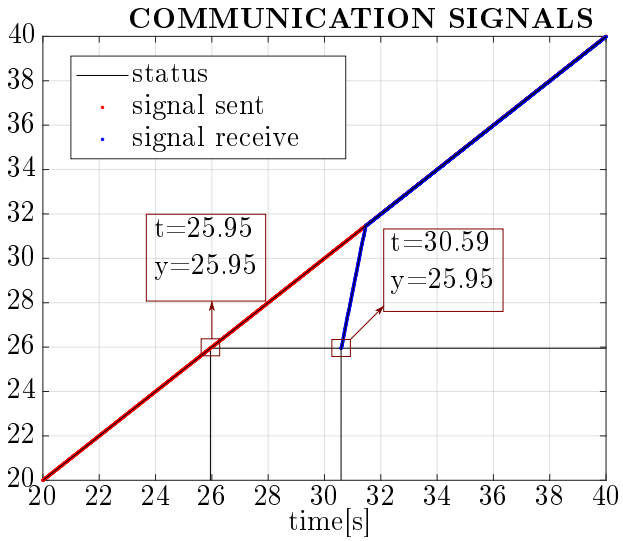
To be able to evaluate the controller design with experiments we have to implement some basic machine states. The principal goals are:

1. make the homing procedure for initialising the position measured and for start from a "safe" position;
2. to manage the connection between server and client and guarantee that the communication has no problems;
3. manage the transition between open-loop control and close loop control;
4. to implement a fault logic that guarantees a safe operation.

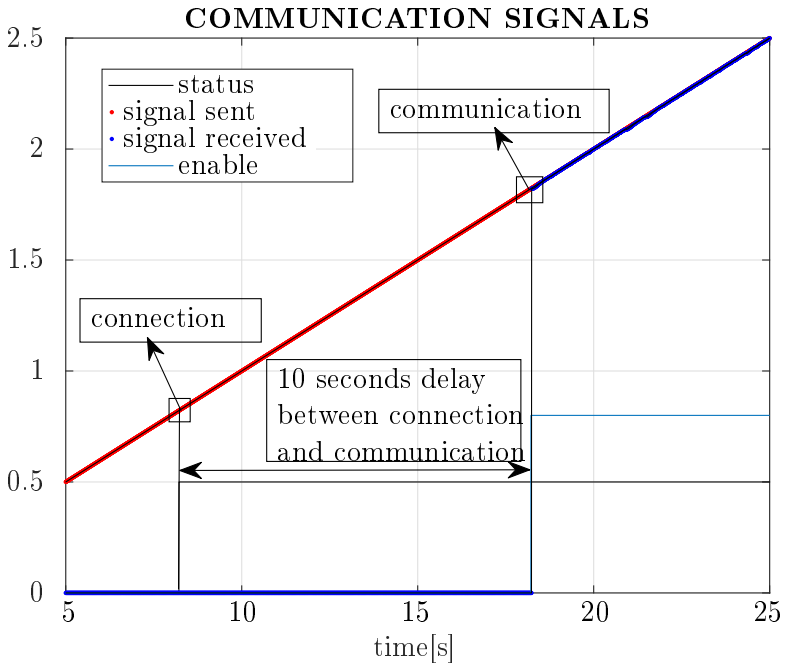
We highlight that we are considering the control system divided into two subsystems: the system interface with all the power units and hydraulic, and the laptop with the controller. From a communicational point of view, before we talked respectively about *server*, and *client*. Thanks to the *Robust Stability Condition* we can say that the controller has not any impact on the safe operation of the machine, if the time delay is under the upper-bound $\tau_{\text{MAX}} = 0.11[s]$, and if the system is driving in close loop. Then, all the above objectives have to be implemented by the server, so in our case on Speedgoat.
Our design choice is shown in figure 7.6. We see inside the grey blocks the implemented machine states. Basically, starting from the initial condition indicated with the state *stop*, running the program, the system interface starts doing at the same time the *homing*, and the *communication initialization*. Then, under the condition $\tau < 0.11[s]$ we start the *close loop control*. From the latter state, the machine goes into *block* under the conditions detected by the *fault logic*. From the *block* state, we can return to the initial state of *stop* only with a reset of the machine. Now we are going to explain in detail how we implement this procedure.

7.3.1 Communication initialization

The state *communication initialization* has to establish the connection between server and client and start the evaluation of the time delay in the loop. In particular, the time delay must always have consistent values. Indeed, after the *homing* procedure finishes, it gives the condition for starting the *close loop control*. Unfortunately, we face a problem if we wait until we have established the connection, and then, we communicate the counter signal used for the time delay evaluation. We explain that problem with figure 7.7a. If we look at figure 7.7a we see that we have the connection established at 25.95[s]. We know that because we use an internal signal called *status* that gets a boolean value based on the connection status, the figure is multiplied by a constant to see better. In this case, as suggested before, at the same time, we also start with the communication of the signal called *signal sent*, and we acquire the same after it makes the round trip, so we call it *signal received*. We see that we start receiving the signal with a huge delay of $30.59 - 25.95 = 4.64[s]$, and then the acquisition catch up in around 1[s] to achieve the time delay analyzed in section 3.2. This phenomenon is due to the communication initialization of both client and server and it is not deterministic, so we do not know exactly what is the initial delay. This phenomenon is not acceptable because could happen the initial delay is greater than the counter amplitude, the result is the overflow of the τ calculation. Thus, the wrong computed τ could make the condition $\tau < 0.11[s]$ true and activate the close control loop. To avoid such a problem, we can basically delay the communication from the connection. We see in figure 7.7b that a delay of 10[s] between the connection and the communication allow us to get the desired result. We control the connection start using another internal signal called *enable* that also assume binary values. The experience provided by several tests allows us to say that a delay of 8[s] is enough for our purpose. Therefore, we can implement the initialization procedure as shown in the figure 7.2. First, we wait until the server and client are connected, so *status* became 1. Then, we wait for 8[s], but we go back to the first state as soon as the connection is lost. Then, after these 8[s] in which the status signal is always 1, *enable* becomes 1, and the communication of the counter starts. Also, in this final state with *enable*=1, we go back to the first state as soon as the connection is lost, so *status*=0. Only in this final state with *enable* and *status* both equal to 1 we are computing the communication time delay.



(a) Test signal that makes the round trip from server. We can see the effect of client initialization.



(b) In this case the initialization effect in the communication disappears because of the 10[s] imposed delay.

Figure 7.7: Counter signal from the server point of view during the communication initialization.

7.3.2 Homing and open loop control

When we start running the program in the machine interface, and the *communication initialization* procedure is in progress, the *homing* procedure begins at the same time. In particular, this machine state aims to drive the piston in a safe starting position for the closed-loop control. Moreover, zeroing the position measurement when the piston reaches the new position. We set up the following steps:

1. bring the piston to the beginning of the cylinder stroke;
2. drive forward for a while, just to detach the piston from the side;
3. reset the measure read by the encoder.

The first step is necessary to reach a known position because basically when we turn on the machine we do not know where the piston is left. We can do that by applying a small negative control signal as it is visible in figure 7.8. The amplitude has to be greater than the dead-zone level, so ± 0.1, but also it has to be very little to avoid a strong hit that could damage the system. For this purpose, we choose -0.15 for a time length of $15[s]$.
The idea behind the second step resides in the fact that when we close the control loop the oscillations we have seen in the simulations due to the measurement noise will make the piston hit the cylinder continuously. Hence, we simply give a small control signal of 0.15 for a short amount of time $0.5[s]$. In this way, we know that the control produces a tiny displacement of less than 0.03[m].
Finally, with the third step, we reset the position value to zero, so we can simply drive the system with the close-loop control without any effort on the initial condition of the reference signal. Looking at figure 7.8 we remark that the signal we send to the client is constantly zero, only when we are in the *close loop state*, as we will see, we can send the now zeroed position measure.

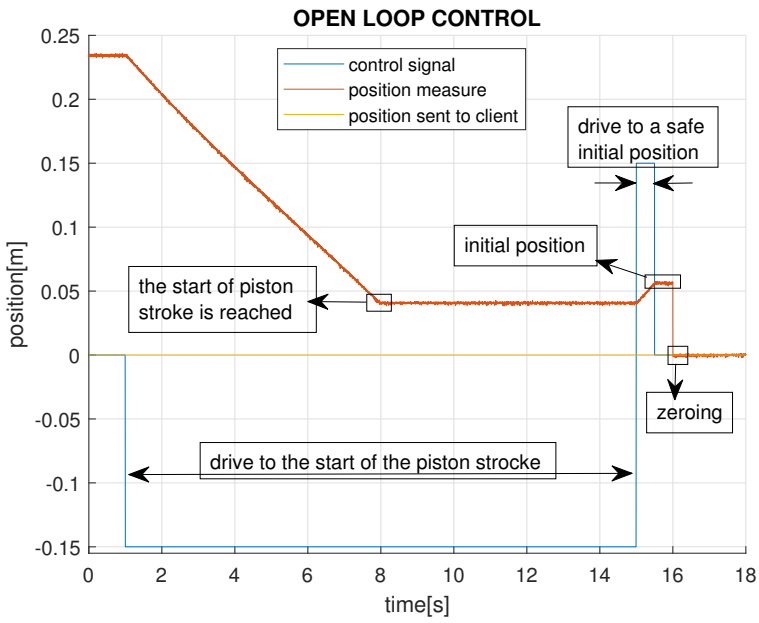


Figure 7.8: Position and control signals during the open loop procedure.

7.3.3 Close loop procedure

Always looking at picture 7.6 we see that we can reached the *close loop control* state only after both the *homing*, and *communication initialization* are well finished. Moreover, the time delay has to be less than the upper-bound $\tau < \tau_{\text{MAX}} = 0.11[s]$. Hence, when all these conditions are satisfied, we send to the client the position measure, as it is visible in figure 7.9. Before that, the position sent to the client is set to zero to avoid the PID controller integrates the error. Figure 7.9 shows both the server and the client signals during this state transition. We see that the *homing* procedure ended after we have established the communication. All the conditions for getting into the *close loop control* state are satisfied when the *homing* procedure terminates. For precaution, always in the picture, we see that we wait some seconds before effectively starting the close control.

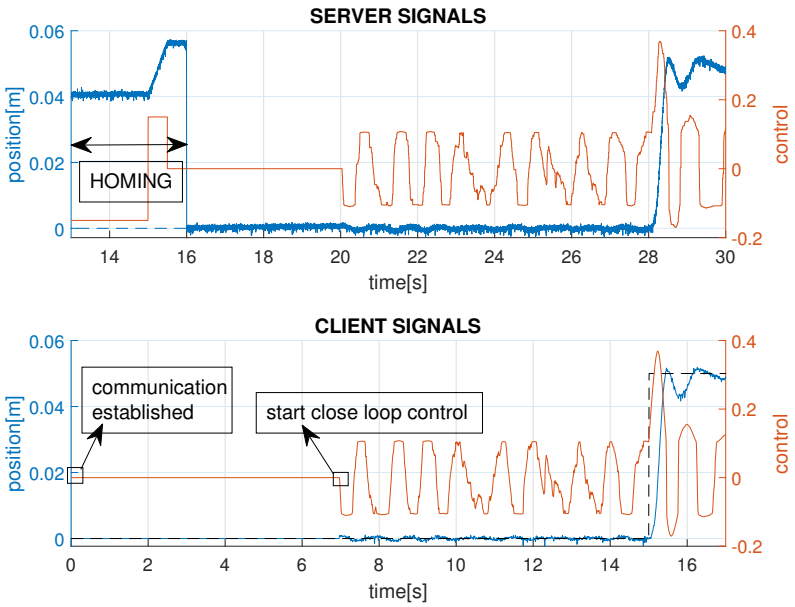


Figure 7.9: Server and client signals during the transition from the open loop to the close loop control.

We remark that the two subsystems have a different time axis, basically because we have to run the server first, and then we can run the client that search the connection on the specified IP address and port. Due to the variable time delay in the communication, it is impossible to perfectly synchronise the axis. We make a good approximation if we assume the variable time delay for one way is half of the Round Trip Time. In the next chapter, we are going to present the results of the designed controller. In that case, we need to have in the same time axis reference and control signal generated by the client, and the measured position and time delay from the server. Then, we will use for practice the client time axis so we shift the server signals using the already mentioned approximation.

7.3.4 Safe operation logic

We remember that this machine states project aims to be able to evaluate the experimental results of the designed controller. Hence, evaluating a *safe operation logic* that guarantees to operate under safe conditions is important, but having not too precautionary conditions is likewise necessary. Indeed, too precautionary conditions make the evaluation procedure impossible.
With the *Robust Stability Condition* we guarantee that the close loop control system remain stable if the time delay keeps under $\tau_{\text{MAX}} = 0.11[s]$. Analysing the time delay distribution we have seen that it keeps under the upper bound from 93.68% to 100% of the cases, see table 5.3. Thus, even if it would be better to activate the fault condition as soon as the real-time computed time delay goes upper the bound, it is necessary to find a less restrictive condition. Indeed, if we do not do that, statistically the system will stop one time each $0.0071[s]$, clearly not long enough to make any test.
The choice we make is to activate the fault procedure if one of the following conditions become true:

1. the time delay goes upper 0.3[s];
2. the time delay keep over the bound $\tau > \tau_{\text{MAX}} = 0.11[s]$ for at least 0.5[s];
3. there is no more connection between client and server: *status*=0.

With the first condition, we avoid extremely high delay in the communication that may cause fast divergence in the response. With the second, we avoid that the system keeps in an unstable condition for too long, and of course, if the connection is lost, there is no point to maintain the closed-loop configuration. We choose the values of the first two conditions by means of the analysis of the experiments already done.
When the fault condition is activated, the system stops the closed-loop control and instead drive the hydraulic with zero control, so the piston simply stops immediately. Hence, we arrived in the sate called *block* in figure 7.6. From this state, the only way to become operative again is to reset both the server and the controller.

Chapter 8

Experimental results and discussion

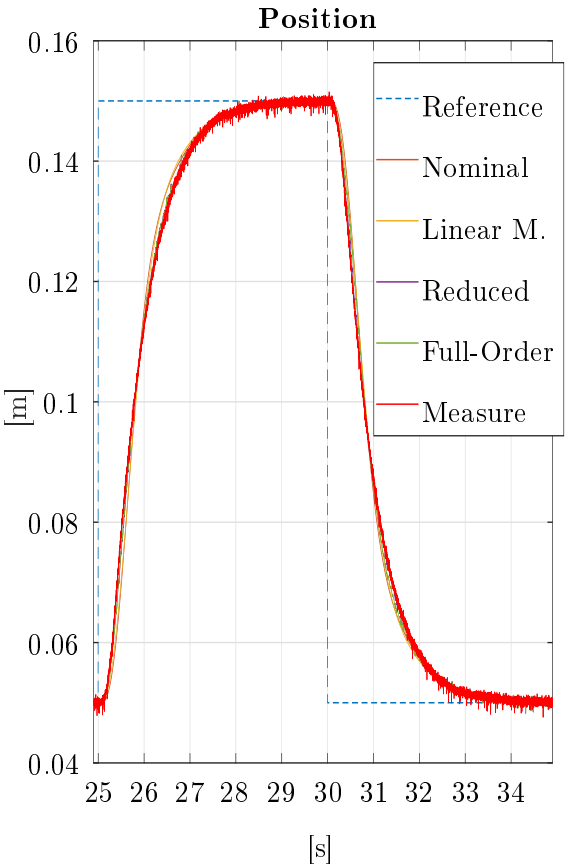
We project twelve experiments to evaluate the controller design on the real hydraulic system of UiA laboratory. These tests are listed in table 8.1. In the table we can see from the first column to the last: the label of the experiment, the set-up configuration, the set-point parameter b, and the set-point filter F_{sp}. The set-up configuration clarifies where is the controller. For instance, in the first three tests with *ETH cable*, we mean that the controller program is running on the laptop, and the TCP-IP communication from the controller to the system interface is through an Ethernet cable. Then, by *WIFI* $d_{WIFI} = 0.2[m]$ we mean that the TCP-IP communication takes place via the WIFI connection at a specified distance d_{WIFI}. We choose d_{WIFI} to reproduce the condition of the preliminary tests made for evaluating the time delay in the communication, see chapter 3.2. In particular we will talk about *Close WIFI distance* for $d_{\text{WIFI}} = 0.2[m]$, *Middle WIFI distance* for $d_{\text{WIFI}} = 2[m]$, and *Great WIFI distance* for $d_{\text{WIFI}} = 10[m]$. Finally, by *deterministic RT-C* we mean that the controller is running Real-Time directly on the system interface, then without TCP-IP communication. For the set-point parameter b, and for the set-point filter F_{sp} we use the same notation of table 6.2, in chapter 6. We use a square wave as a position reference of $0.1[m]$ of amplitude and $0.05[m]$ shifting because we want to make exactly the same tests we have made in the simulations. Hence, the comparison between experimental results and simulation results is the easiest.

Table 8.1: Set of experimental tests, names and properties

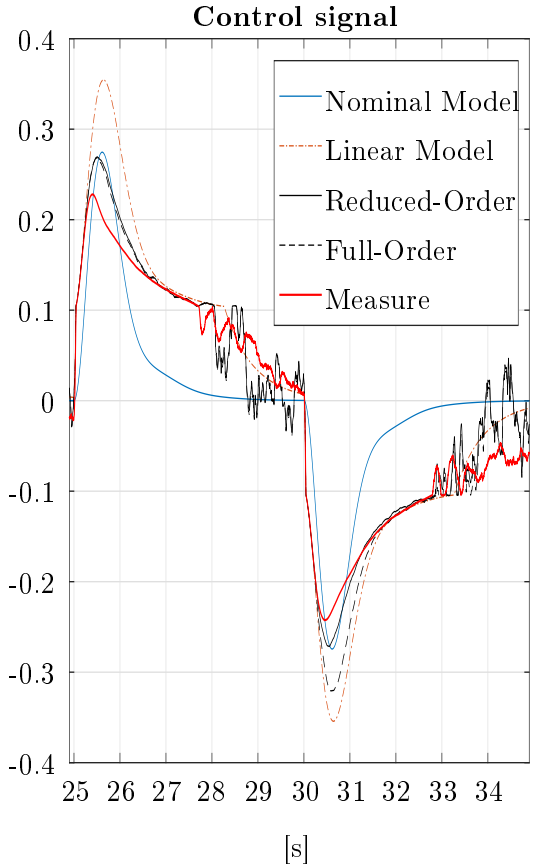
Experiment Name	**Configuration**	**b**	**F_{sp}**
test1	ETH cable	0	no
test2	ETH cable	0	yes
test3	ETH cable	1	no
test4	WIFI $d_{\text{WIFI}} = 0.2[m]$	0	no
test5	WIFI $d_{\text{WIFI}} = 0.2[m]$	0	yes
test6	WIFI $d_{\text{WIFI}} = 2[m]$	0	no
test7	WIFI $d_{\text{WIFI}} = 2[m]$	0	yes
test8	WIFI $d_{\text{WIFI}} = 10[m]$	0	no
test9	WIFI $d_{\text{WIFI}} = 10[m]$	0	yes
test10	deterministic RT-C	1	no
test11	deterministic RT-C	0	no
test12	deterministic RT-C	0	yes

8.1 Deterministic real time control

Now we present the results obtained with the control implemented on Speedgoat. The step responses for the experiment *test12*, and the simulation called *simulation1* are shown together in figure 8.1a, and 8.1b. In this case, we use all the set-point tricks. We see that the measured position has almost the same trajectory as all the simulation models. In particular, this similarity becomes more pronounced with the *Full-Order Model*. Furthermore, we can say the same by looking at the control signals. Again in the control signals graph, we see the measure has less amplitude than the simulation signals. That means there is a difference between some models parameters and the real system, that does not affect the dynamic behaviour. We guess the main difference is in the friction model. Therefore, the real friction force is strongly dependent on the temperature, lubricant condition, dust, and many other non modelled factors. Moreover, the experiments for friction identification have been made more than one year before.



(a) position acquired in Speedgoat and reference generated in the remote controller for the experimental test. Comparison with the result of the simulation with different models.

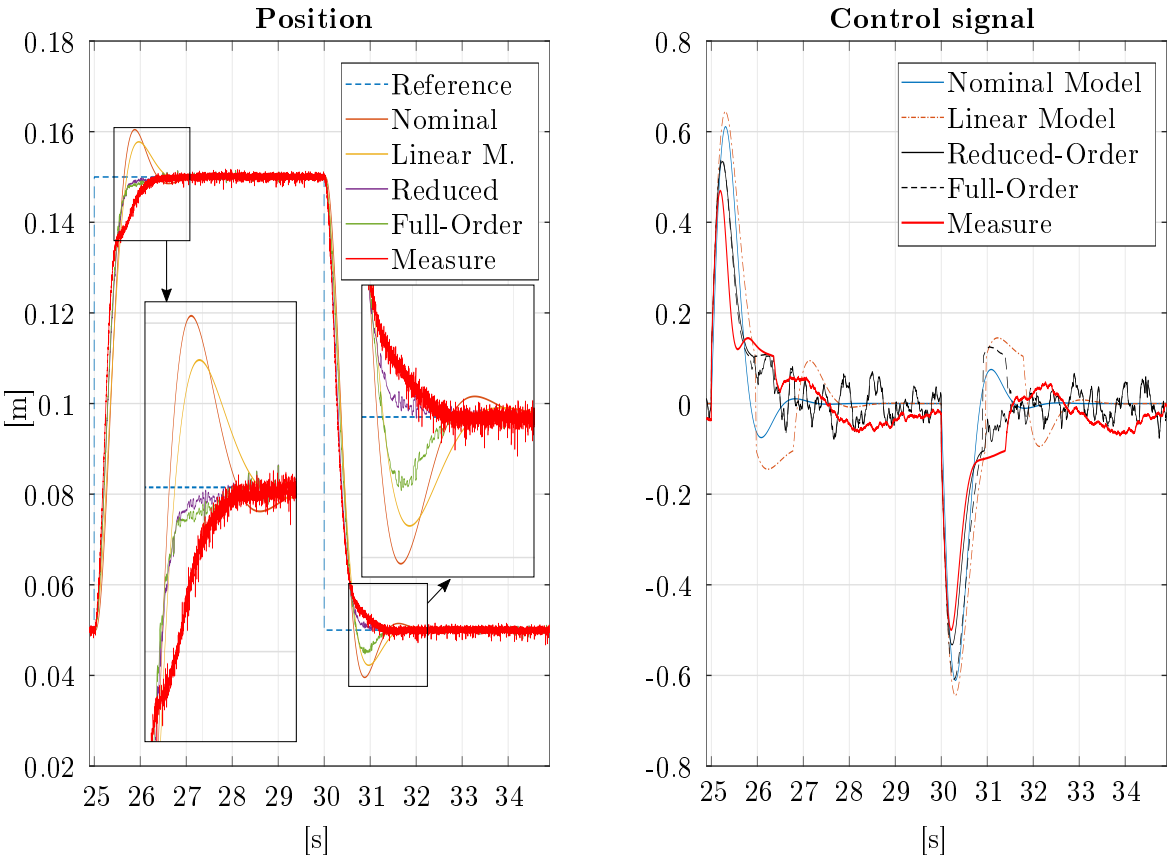


(b) control signal acquired in Speedgoat and control signal in the simulation models.

Figure 8.1: Results of the experimental test called *test12* compared with the results from *simulation1*

In figure 8.2a, and 8.2b are reported together the step responses for the experiment *test11*, and *simulation2*. In this case, the set-point filter is not used, so we are in the best controller configuration. We see that the step response is still very close to the simulation, in particular in the step back with the *Full-Order Model*. The only difference we can note is when the piston in the step forward is approaching the reference. We see a huge increment of the counteracting force that reduces drastically the speed, so the controller tries to compensate by increasing the control signal value. We suggest that this difference is caused by both the variation of the friction behaviour discussed before and the friction non-linearity of the drove mass position on the piston

guide.



(a) position acquired in Speedgoat and reference generated in the remote controller for the experimental test. Comparison with the result of the simulation with different models.

(b) control signal acquired in Speedgoat and control signal in the simulation models.

Figure 8.2: Results of the experimental test called *test11* compared with the results from *simulation2*

8.2 Remote control

We evaluate all the following experiments with the control implemented in the laptop. Thence, there is TCP-IP communication in the system loop. The idea is to start with the best possible case, so the direct Ethernet connection between controller and system interface. Then, with the WIFI implementation, we expect the increment of the time delay, so a worse operative condition.

8.2.1 Direct ETH cable connection

The connection from the laptop to Speedgoat is made with an Ethernet cable around 4 meters long. The results for *test2* are shown in figure 8.3. We highlight that *simulation1* is evaluated with the minimum time delay. In the upper plot of the figure, we see the measured position, the signals from *simulation1*, and the computed time delay. In the second plot are shown the respective control signals. We see that the time delay causes some oscillations in the response, but the trajectory is still very close to the expectation.

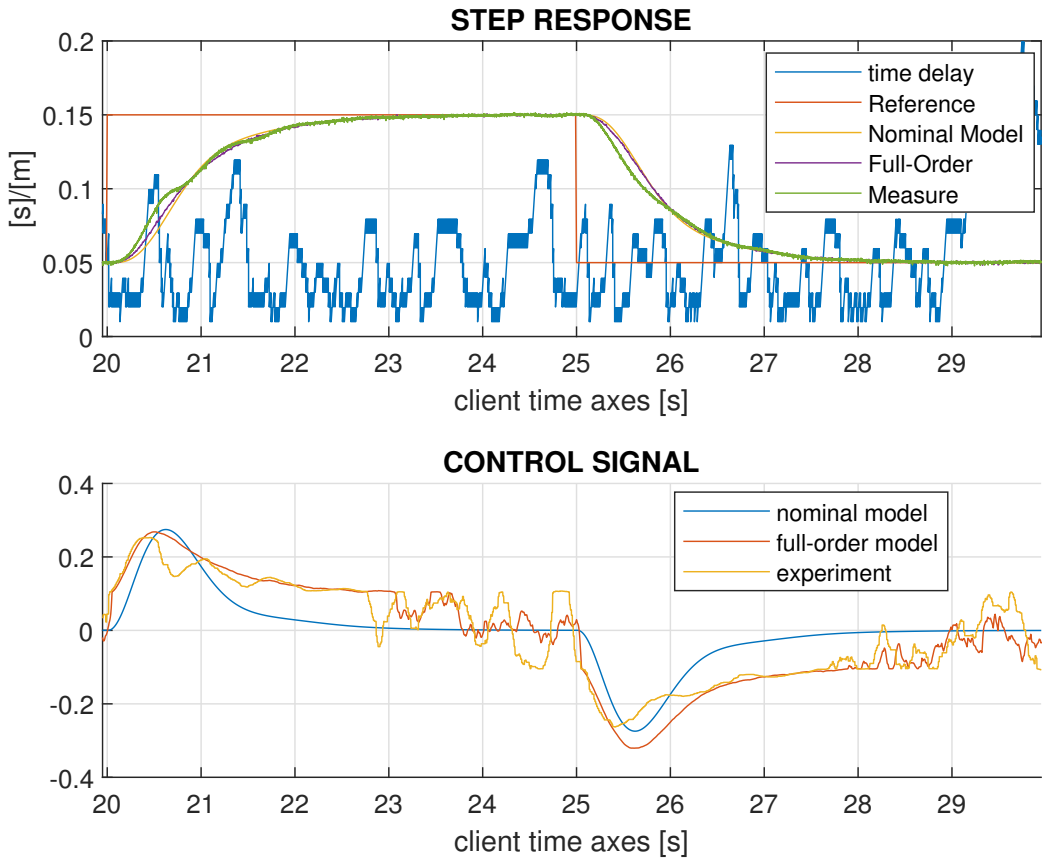


Figure 8.3: In the upper plot there are the position reference, the measured data from *test2* and the time delay. The measured data from *test2* are compared with the simulation responses obtained in *simulation1*. In the second plot there are the respective control signals.

Figure 8.4 is a zoom out of the first graph in figure 8.3. The large time span of more than 40 seconds makes visible the difference in the step response from period to period. For instance, at around 51 seconds with a red circle is marked a deep oscillation in the measured position. At the same time, we see that the time delay reaches $0.19[s]$ that is far over the maximum upper-bound $\tau_{\text{MAX}} = 0.11[s]$. We see that, even though the stability of the system is no more guaranteed, the response is still convergent to the reference. Then, if the time delay exceeds the upper bound for a small amount of time, the system is still stable. This observation supports the discussion made in the previous chapter and the choices regarding the fault condition.
In Figure 8.5 are also shown the position measured in *test1* and the response of *simulation2* with

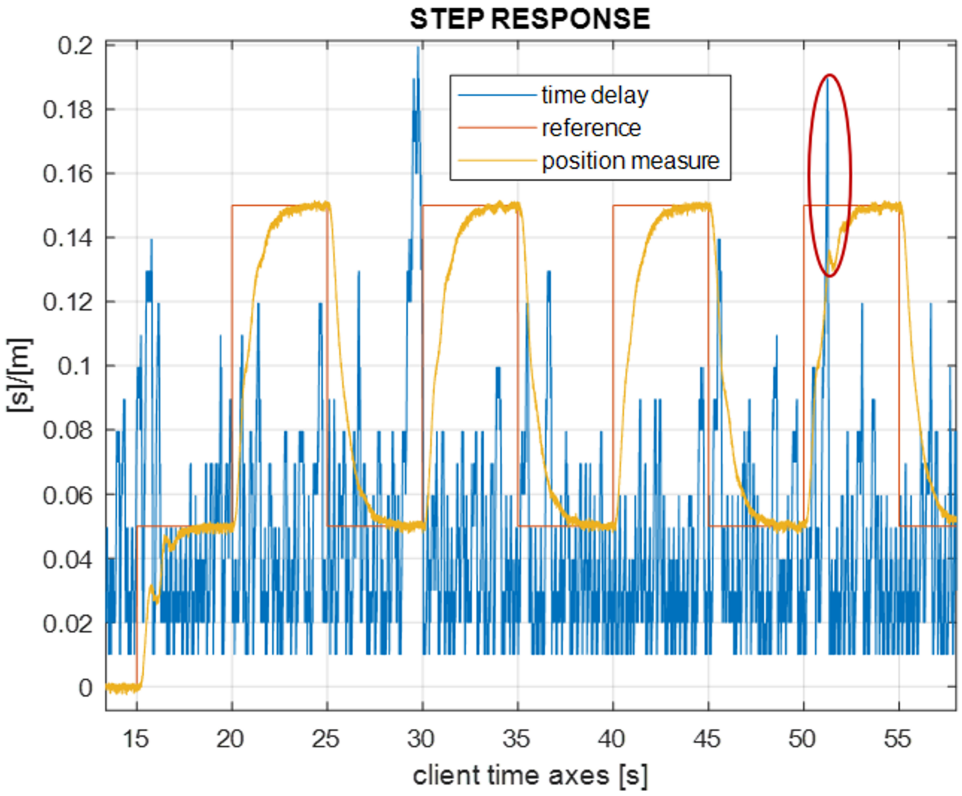


Figure 8.4: Plot of the position reference, the measured data from *test2*, and the time delay with a large time span.

the respective control signals. We divide the figure to have a better view of the step forward and step back. The result is good because, if the time delay keeps below the threshold of $0.11[s]$, we have a very small rise time $\simeq 0.3[s]$ and a good settling time $\simeq 1.3[s]$, with low overshoot.

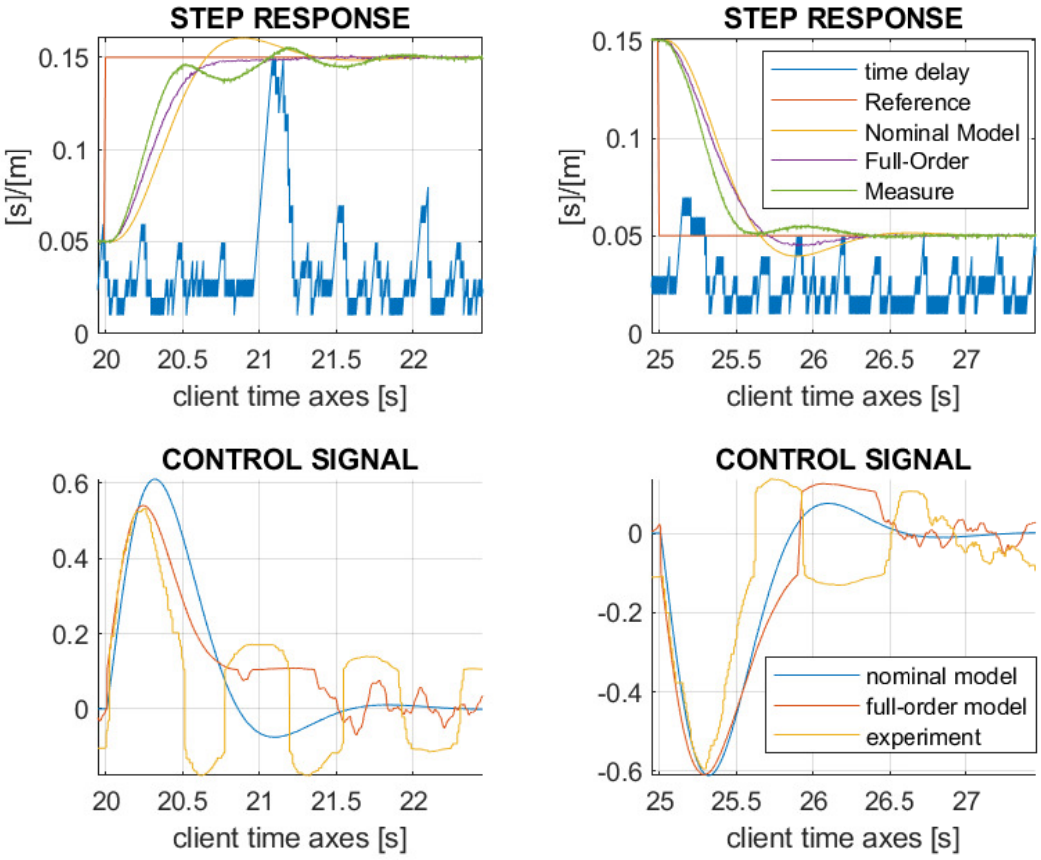


Figure 8.5: In the upper plots there are the position reference, the measured data from *test1* compared with the simulation responses obtained in *simulation2*, and the time delay. In the lower plots, there are the respective control signals.

8.2.2 Close WIFI distance

The good results reached with the direct Ethernet cable connection allow us to test the implementation of the WIFI bridge. In this section, we show the results obtained with a small distance through the routers: $d_{\text{WIFI}} = 0.2[m]$. In particular, we analyse the response of the system designed with $b = 0$ and without a set-point filter, so it is the *test4*. Figure 8.6 shows the position results separately for the step forward and back. In this case, we compare the simulated results evaluated using the γ-*distributed* time delay. We see that the response start to differ from the *Full-Order Model*, but still, the rise time and the settling time remain the same. Instead, the amplitude of the oscillation is bigger in the experimental data.

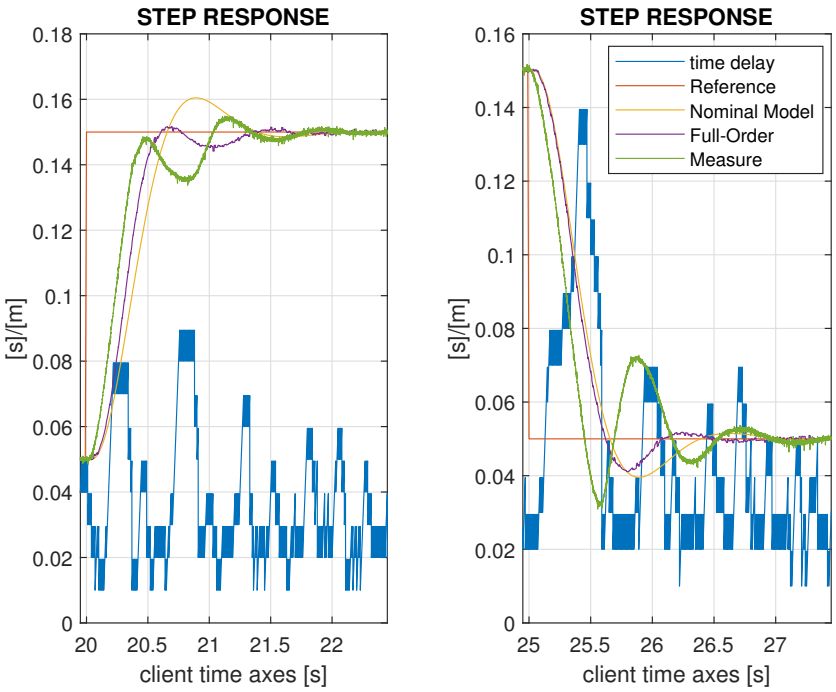


Figure 8.6: In the upper plot there are the position reference, the measured data from *test4* compared with the simulation responses obtained in *simulation8* and the time delay.

8.2.3 Middle WIFI distance

In this section, follow the results obtained with a distance through the routers of $d_{\text{WIFI}} = 2[m]$. In particular, we analyse the response of the system designed with $b = 0$, and without a set-point filter, called *test6*. In figure 8.7 are shown the position results in the upper plot and the relative control signals on the bottom one. We are still comparing the simulated results evaluated using the γ-*distributed* time delay. We see that the response strongly differs from the *Full-Order Model*. Furthermore, now the convergence to the reference signal is reached in a few cases. Indeed, the oscillation continues if the time delay keeps high values.

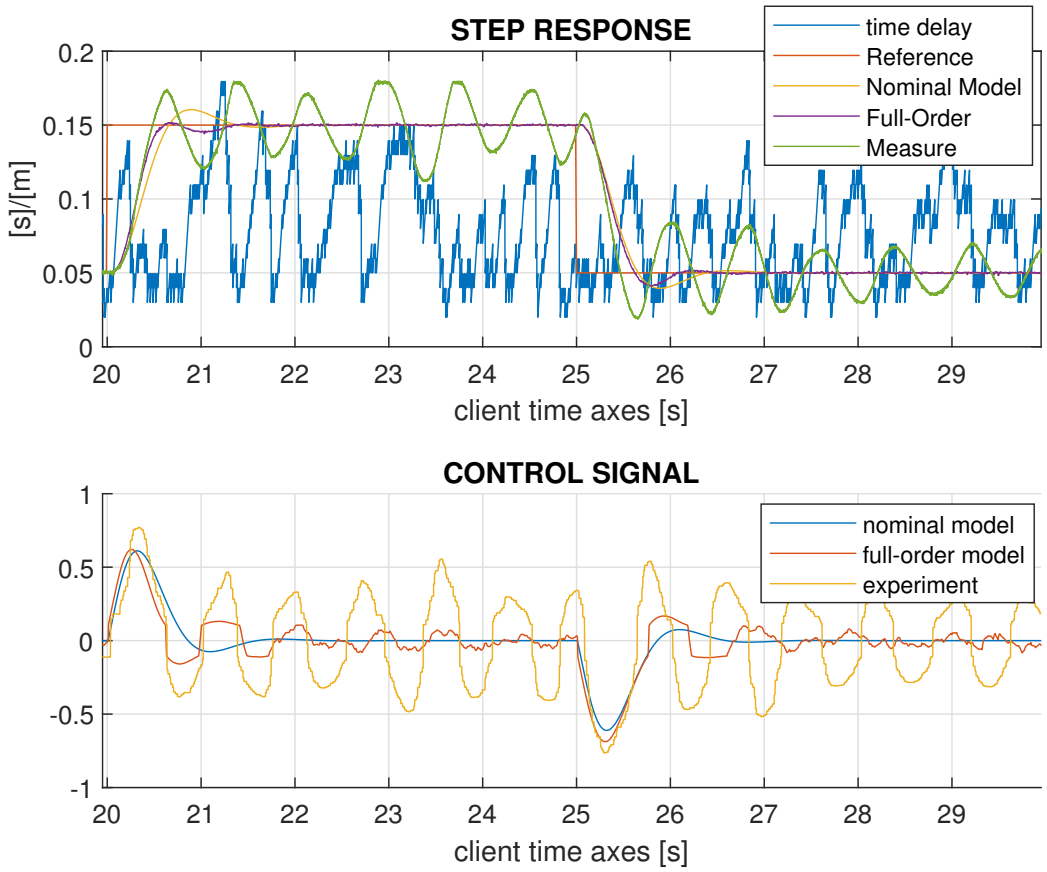


Figure 8.7: In the upper plot there are the position reference, the measured data from *test6* compared with the simulation responses obtained in *simulation8* and the time delay. In the second plot, there are the respective control signals.

8.2.4 Great WIFI distance

As last, we implement the WIFI bridge over a distance of $d_{\text{WIFI}} = 10[m]$. In figure 8.8 w eshow the result for the *test8* compared with the *simualtion11*. In this case, the simulation is evaluated using the maximum allowed time delay: $\tau = 0.11[s]$.

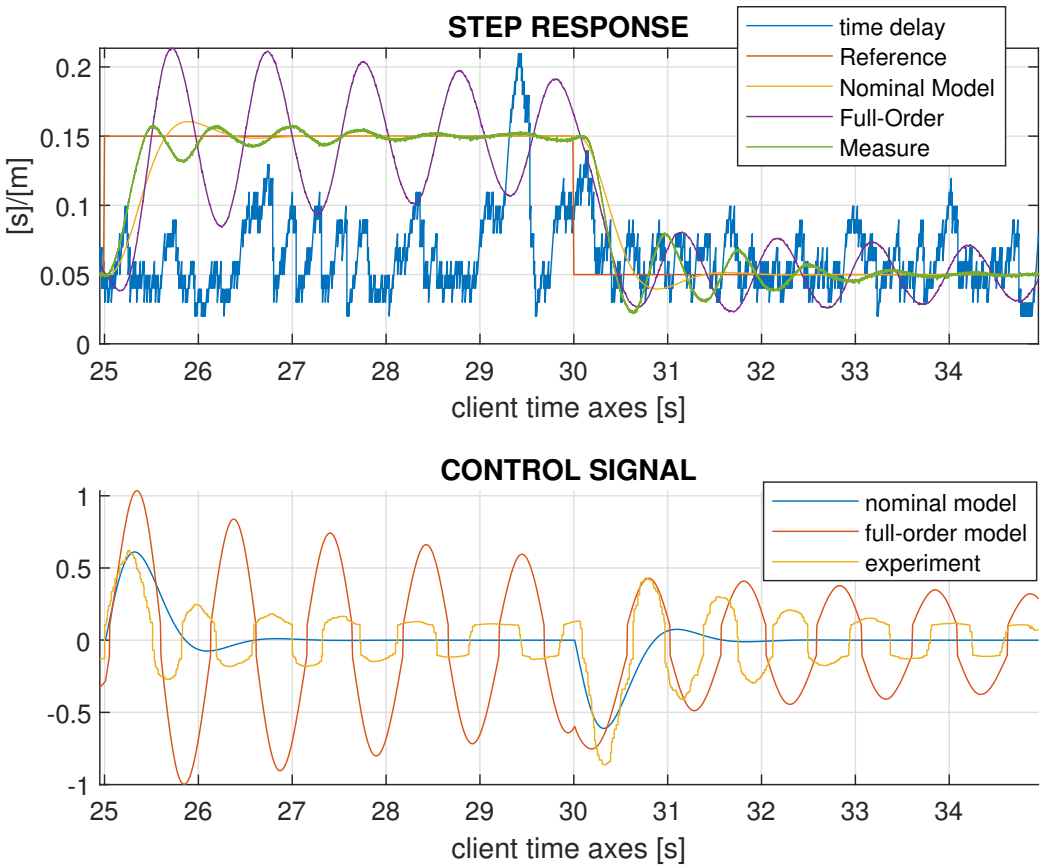


Figure 8.8: In the upper plot there are the position reference, the measured data from *test8* compared with the simulation responses obtained in *simulation11* and the time delay. In the second plot, there are the respective control signals.

8.2.5 Analysis

To sum up, in figure 8.9 there are the experimental performances obtained changing the type of connection or the WIFI distance. In particular, in the figure, the controller is implemented without the set-point filter and with the set-point parameter $b = 0$. It is clear the performance of the control system is satisfying with the Ethernet connection, but the oscillations become bigger for a "middle" and "far" distance between the routers with the wireless connection.

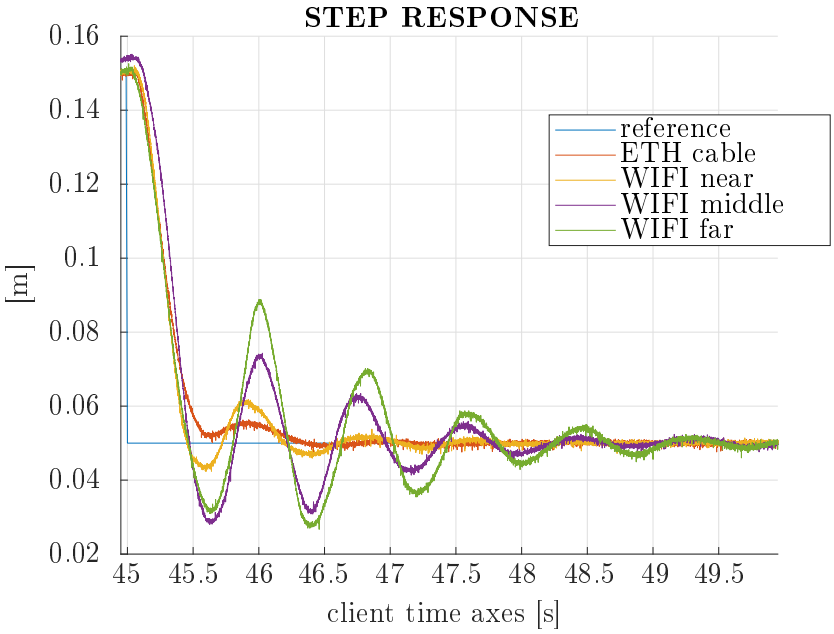


Figure 8.9: System step back response under different control connections.

To judge the goodness of the design approach adopted in this work, we have to analyze the results by looking at the statistical parameters of the time delay, which is the kernel of the control problem. In table 8.2 are listed some fundamental statistic parameters computed from the time delay in the respective experimental tests. We have also computed the amount of time delay points under the upper bound of $0.11[s]$ and shown in the last table column. The values are normalized by 1. We see that the number of values under the bound are far from the $3 \cdot \sigma$ interval. Furthermore, only in the first test, the mean value is close to the design value of $0.03[s]$. From *test6* to *test9* the value becomes greater than twice.

Table 8.2: Main statistic parameters of the time delay during the experimental tests

[seconds]	max	min	mean	mode	var	$\tau \leq \tau_{\mathbf{MAX}}$
test1	0.2395	0.01	0.0349	0.021	8.33E-04	0.9683
test2	0.2195	0.01	0.0432	0.021	8.82E-04	0.9613
test3	0.1495	0.01	0.041	0.021	6.71E-04	0.9738
test4	0.2595	0.01	0.0474	0.021	9.55E-04	0.9554
test5	0.2195	0.01	0.0431	0.021	7.73E-04	0.968
test6	0.2195	0.02	0.0678	0.0475	9.57E-04	0.9021
test7	0.2395	0.01	0.067	0.0475	0.001	0.9085
test8	0.2295	0.02	0.0683	0.0475	9.81E-04	0.9065
test9	0.2795	0.01	0.0662	0.0475	0.0011	0.9029
nominal model	0.11	0.01	0.03	/	/	1

Hence, we fit the time delay distributions obtained in the respective tests to see if it is still comparable to the one adopted in the modelling process. Such result is shown in figure 8.10. We

see that the fitted model used in the simulations is close to the distributions fitted for *test1*, to *test5*. Then, from *test6* to *test9* there is a clear shift to higher time delay value. In table 8.3 are listed the two parameters of the γ-*distribution* that fits the data for each experiment. Moreover, at the bottom, there are the parameters of the γ-*distribution* we have chosen for the simulations. Also, the shape and scale factor highlights the difference in the distributions.

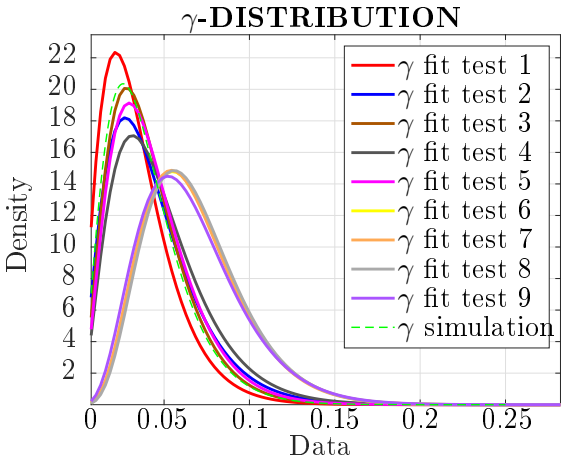


Figure 8.10: γ-*distributions* fits of the time delay τ

Table 8.3: Parameters of the fitted γ-*distributions*

parameters	a	b
test1	2.622685	0.013319
test2	2.690141	0.016058
test3	3.139806	0.013056
test4	2.964207	0.015988
test5	3.157358	0.013662
test6	5.323636	0.01273
test7	5.246183	0.01277
test8	5.488174	0.012445
test9	4.777977	0.013856
simulation	1.6575	0.026533

The computing of the control signal energy of both the experimental tests and simulations is also worthy. Indeed, we want to see if the control action of the simulation is around the same as the experimental test under equivalent conditions. To do that, basically, for the simulations we evaluate the energy separately for the response to the step-up and step-down. We highlight that we use only the *full-order model*. We compute the energy in a five-second interval for all the cases because we have five seconds between the step-up and step-down in the simulations. We list the results in table 8.4. In the same table, we can see also the mean of the energy. As we expect, we see that for good step response, so for the small-time delay in the control loop, we have low energy values.

Table 8.4: Control signal energy in the simulation evaluated over the five seconds of the step response, and the mean value.

SIMULATION	STEP-UP	STEP-DOWN	MEAN
simulation1	0.0885	0.1165	0.1025
simulation2	0.1108	0.1615	0.1362
simulation3	0.5018	0.6133	0.5576
simulation4	0.0898	0.1183	0.1041
simulation5	0.1350	0.1899	0.1624
simulation6	0.7067	0.8012	0.7540
simulation7	0.0900	0.1186	0.1043
simulation8	0.1379	0.1931	0.1655
simulation9	0.6304	0.8667	0.7485
simulation10	0.3286	0.1504	0.2395
simulation11	17.748	0.6297	12.022
simulation12	46.957	58.934	52.946

We compute the control signal energy of the experimental tests in a five seconds interval because it is the same we have used for the simulations. Hence, we can make a comparison. In this case, we evaluate a mean between the energy of five step-ups and five step-downs to have a fair value.

We list the results in table 8.5. We also compute a mean of all steps, up and down, as for the simulations. We see that the step-down has generally a bigger energy value than the step-up for the same test. That is due to the system asymmetry we have already explained.

Table 8.5: Experimental control signal energy evaluated over the five seconds of the step response. The value sown is a mean value.

TEST	MEAN STEP-UP	MEAN STEP-DOWN	TOT MEAN
test1	0.1367	0.1510	0.1439
test2	0.0758	0.0890	0.0824
test4	0.1892	0.1827	0.1859
test5	0.0825	0.0882	0.0853
test6	0.3548	0.4457	0.4003
test7	0.1223	0.1012	0.1118
test8	0.2899	0.3537	0.3218
test9	0.0924	0.1005	0.0965

We compare some control signal energy obtained in the simulations with the one obtained in the experimental tests. We can see the energy difference in table 8.6. We compare simulations and tests, which we have made with the same control structure. The energy difference is little in the first case. Indeed, the absolute difference is 0.0077, and the relative difference is only $0.0077/0.1439 \cdot 100 = 5.3\%$. That means the *full-order model* is close to the experimental set-up when the time delay is low. In the other cases, the relative difference is bigger. That happens because the mean of the time delay in the experimental tests becomes farther from the one modelled in the simulations. Moreover, the dead-zone compensation introduced in the control structure causes a nonlinear behaviour of the control signal. Hence, a tiny difference in the system dynamic leads to a big difference in the control signal energy.

Table 8.6: Control signal energy comparison between some simulations and tests.

COMPARISON		DIFFERENCE
SIMULATION	EXPERIMENT	
simulation2	test1	0.0077
simulation1	test2	-0.0201
simulation8	test4	0.0204
simulation8	test6	0.2348
simulation11	test8	-11.7002

In the end, we see that the design model is close only to the preliminary tests made with the direct cable connection through the system and the controller. The simulation model seems to be valid also for the experiments evaluated with $d_{\text{WIFI}} = 0.2[m]$. The tests with $d_{\text{WIFI}} = 2[m]$, and $d_{\text{WIFI}} = 10[m]$ have a time delay distribution greater than the expected one. We guess the reason is due to the different programs used in the time delay evaluation tests, and the experimental tests. The implementation of the controller may load the laptop resources and reduce the performance of the TCP-IP communication.
Apart from that, the controller design gives good performance and guarantees stability under the conditions defined in the design procedure. The fact we get the expected performance under the same time delay condition means that the model used in the simulation is very likely.

Chapter 9

Conclusions

To sum up, we can say that we have reached all the objectives of the work. Indeed, we have completed the following points:

1. we have implemented a TCP-IP communication with a WIFI connection between the controller and the hydraulic system;
2. we have evaluated a perturbed model of the process which allows verifying the robust stability condition;
3. we have robustly designed a 2DOF PID controller optimizing the integral action;
4. we have tested the control system performance with both simulations and experiments in the hydraulic plant of UiA.

We have simulated the control process using four models. The two most important are the so-called *nominal model* and the *full-order model*. The latter tries to emulate the experimental system as much as possible. The *nominal model* is a linear transfer function that approximates the dynamic of the hydraulic system. We have obtained the model by simplification and linearization of the *full-order model*. We have discussed why we have chosen a particular operative point for the linearization. Briefly, the *nominal model* contains the average of the variable parameters. We see in the simulations that the controller reaches good performance because the nominal model gives a rise time in a step reference of around 0.3[s] and a settling time of 2[s] with low oscillations and overshoot. Furthermore, with the simulation of the full-order model we get even better results. Indeed, the settling time, the overshoot and the oscillations are reduced compared to before because of the friction non-linearity. That is a good result if we think that in the control loop a time delay gamma-distributed from a minimum of 0.01[s] to a maximum of 0.16[s] and a mean of 0.03[s] is modelled. Moreover, we verified numerically the validity of the robust stability condition evaluated on the parameter's uncertainties. In particular, we found the upper bound of 0.11[s] of time delay.
Hence, we can conclude:

1. The robustly designed controller achieves nominally satisfactory performance.
2. The uncertainty analysis and the evaluation of the perturbed model produce a reliable condition for robust stability.

We tested the controller in the hydraulic plant, with a direct Ethernet connection and with several distances of the WIFI connection. Furthermore, we tested the system with the real-time implementation of the controller, so without the TCP-IP communication, just to compare. With the latter configuration, we obtain a response that is very close to the full-order model simulated without time delay. The only difference is in the friction behaviour at low speed. Thus, we get the same rise time and settling time. Then, with the TCP-IP communication, we get still a good response even though the mean time delay is yet over the expectation: 0.035[s]. With the implementation of the WIFI connection, for a short distance between routers, the delay increases

to an average of 0.041 / 0.047 [s] and for a long distance, the delay increases to a value of 0.068 [s]. Then, the system response becomes worse. What is interesting to see is that the system is still stable even with only 90% of the sample points with a time delay lower than the upper-bound 0.11[s]. Therefore, we can conclude the followings:

1. The full-order model describes the hydraulic plant faithfully.
2. The linearization of the model and the uncertainty analysis give a dependable nominal model for the control design.
3. The controller has the expected robustness to parameter variations.

We remark that we saw an increase in the time delay in the loop compared to the one obtained with the identification tests. Hence, one should consider improving the performance of the wireless connection and communication if anyone is interested in the practical implementation of the controller structure discussed in this report.

Appendix A

Matlab Code

Listing A.1: Algorithm:4.2.2

```
%% PID design with Optimization Toolbox
% PROCESS WITH FREE INTEGRATOR AND TIME DELAY
% need functions: @hitcallback @bisectionMethod
clear all; close all; clc;

w_vec = logspace(-1,4,10000);
% check robustness constraint around w_crit
w_check = [linspace(0.8,1.2,10), linspace(1.3, 20, 40)];
% Newton Raphson perturbation
num_perturb_NewRap = 3; % num^4
dx0 = [0.1 0.1 0.1 0.1]'; % wheiting function (Kp Ki w1 w2)
% design specifications
Ms = 1.1;
r = 1/Ms;
c = 1;

%% Load data from: A_Model_Initialization

% 'P' = symbolic model without time delay
% 'P_nom' = symbolic model with nominal time delay
% 'usys' = uncertain transfer function without time delay
% 'Pnom_tf' = nominal transfer function with nominal time delay
% 'usysPade' = uncertain transfer function with Padè approximation of dt
load('Data Variables/A_PxFy_forDESIGN.mat');

%% define transfer functions
s = tf('s');
syms w real positive
syms w1 real positive
syms w2 real positive
syms K real
syms Ki real
syms Kd real

figure;
bode(Pnom_tf);
xlim([10^-5,10^4]);
ylim([-360 -45]);
grid on;
title('Process Nominal Transfer Function');

ro = sqrt(real(P_nom)^2+imag(P_nom)^2);
phi = angle(P_nom);
rod = diff(ro,w);
```

```
phid = diff(phi);
alpha = ro*cos(phi);
beta = ro*sin(phi);

%% SMOOTH CONSTRAINT %%%%%%%%%%%%%%
        PID = K + Ki/(1i*w) + Kd*(1i*w);
        L = PID*P_nom;
        real_L = real(L);
        imag_L = imag(L);
        real_L_dot = diff(real_L, w);
        imag_L_dot = diff(imag_L, w);
        real_L_dot_dot = diff(real_L_dot, w);
        imag_L_dot_dot = diff(imag_L_dot, w);
        smooth1 = (real_L_dot*imag_L_dot_dot-real_L_dot_dot*imag_L_dot)/...
            (real_L_dot^2+imag_L_dot^2)^(3/2);
        smooth2 = diff(angle(L),w);
        smooth1_fnc = matlabFunction(smooth1); % @(K,Kd,Ki,w)
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% STEP 1: DETERMIN THE SEARCH RANGE w_low w_high

kdPI = 0;
less_w = 0;
cross_0 = 0;
hpid = (r+c*sin(phi))*(rod/ro-1/w)-c*phid*cos(phi)+2*ro*kdPI;
hpidfnc=matlabFunction(hpid); % @(w)
[~, phaseP] = bode(Pnom_tf, w_vec);
i=1;
while phaseP(i) > -90
    i = i+1;
end
w90 = w_vec(i);
i=1;
while phaseP(i) > -(180-asin(r/c)*180/pi)
    i = i+1;
end
w180asin = w_vec(i);
if hpidfnc(w90)*hpidfnc(w180asin)<0
    cross_0 = 1;
end
while cross_0 == 0
    w_lin = linspace(w90-less_w,w180asin-less_w,100);
    figure;
    plot(w_lin,hpidfnc(w_lin));
    grid on;
    less_w = input('INSERT THE LEFT TRASLATION OF\n THE INTERVAL FOR 0 ...
       CROSSING: (w>0):');
    if hpidfnc(w90-less_w)*hpidfnc(w180asin-less_w)<0
        cross_0 = 1;
    end
end
w_lin = linspace(w90-less_w,w180asin-less_w,100);
figure;
plot(w_lin,hpidfnc(w_lin));
grid on;

%% STEP 2: solve for PI:
% design PI controller to find initial conditions
% [kp ki kd=0] and [w0 w180 w270]

kdPI = 0;
```

```
hpid = (r+c*sin(phi))*(rod/ro-1/w)-c*phid*cos(phi)+2*ro*kdPI;
dh = diff(hpid,w);
hpid_fun = matlabFunction(hpid);
wpi0 = bisectionMethod(hpidfnc,w90-less_w,w180asin-less_w,1e-5);
options = optimoptions(@fsolve,'Algorithm','levenberg-marquardt');
wpi = fsolve(hpid_fun,wpi0,options);
phiPI = subs(phi,w,wpi);
roPI = subs(ro,w,wpi);
kpPI = double(-c/roPI*cos(phiPI));
kiPI = double(-wpi/roPI*(r+c*sin(phiPI))+wpi^2*kdPI);

% verify that the robustness constraint is satisfied arpund w*
f = abs(c+(kpPI+1i*(kdPI*w-kiPI/w))*P_nom)^2;
ROBUST_CONDITION = true;
for j=1:length(w_check)
    if double(subs(f,w,wpi*w_check(j)))<r^2
        ROBUST_CONDITION = false;
        break;
    end
end
ROBUST_CONDITION_PI = ROBUST_CONDITION;

if ROBUST_CONDITION_PI==0
    corner = 1;
    ('ENVELOP WITH CORNER OR SEE ROBUST CONDITION')
else
    corner = 0;
    ('SMOOTH ENVELOP (at Kd=0) OR SEE ROBUST CONDITION')
end

%% STEP 3A: SMOOTH ENVELOPE:

if corner==0
    % repeat until the envelope is smooth to find initial condition for
    % optimization problem (@fseminf())
    kp_smooth = kpPI;
    ki_smooth = kiPI;
    kd_smooth = kdPI;
    w_smooth = wpi;
    PIDtf = kp_smooth + ki_smooth/s + kd_smooth*s;
    Ltf = PIDtf*Pnom_tf;
    [~, phaseL] = bode(Ltf, w_vec);
    % phaseL = phaseL-360;
    i=1;
    while phaseL(i)<-180
        i = i+1;
    end
    w180 = w_vec(i);
    i=1;
    while phaseL(i)>-270
        i = i+1;
    end
    w270 = w_vec(i);
    w_start = w_smooth/2;
    w_stop = (w180+w270)/2;

    kp_smooth_0 = kp_smooth;
    ki_smooth_0 = ki_smooth;
    kd_smooth_0 = kd_smooth;
    w_smooth_0 = w_smooth;
    delta_kd = 1;
```

```
n_iterations = 0;
bisect_kd_MAX = 1;
bisect_kd_min = 0;
ROBUST_CONDITION_kdMAX = true;
f = abs(c+(kp_smooth+1i*(bisect_kd_MAX*w-ki_smooth/w))*P_nom)^2;
for j=1:length(w_check)
    if double(subs(f,w,w_smooth*w_check(j)))<r^2
        ROBUST_CONDITION_kdMAX = false;
        break;
    end
end

if ROBUST_CONDITION_kdMAX == false
    % BISECTIONAL METHOD FOR SEARCHING MAX KD THAT SATISFY ROBUST
    % CONSTRAINT

    while (bisect_kd_MAX-bisect_kd_min) > 0.00001 && ...
       n_iterations<50    % EXIT CONDITIONS

        kd_smooth_0 = (bisect_kd_MAX + bisect_kd_min)/2;
        n_iterations = n_iterations + 1;

        hpid = ...
            (r+c*sin(phi))*(rod/ro-1/w)-c*phid*cos(phi)+2*ro*kd_smooth_0;
        hpid_fnc=matlabFunction(hpid); % @(w)
        w_smooth_0=bisectionMethod(hpidfnc,w_start,w_stop,1e-5);
        options = ...
           optimoptions(@fsolve,'Algorithm','levenberg-marquardt');
        w_smooth_0 = fsolve(hpid_fnc,w_smooth_0,options);
        phiPID = subs(phi,w,w_smooth_0);
        roPID = subs(ro,w,w_smooth_0);
        kp_smooth_0 = double(-c/roPID*cos(phiPID));
        ki_smooth_0 = double(-w_smooth_0/roPID*(r+c*sin(phiPID))+...
        w_smooth_0^2*kd_smooth_0);

        % verify that the robustness constraint is satisfied arpund w*
        f = abs(c+(kp_smooth_0+1i*(kd_smooth_0*w-ki_smooth_0/...
        w))*P_nom)^2;
        ROBUST_CONDITION = true;
        for j=1:length(w_check)
            if double(subs(f,w,w_smooth_0*w_check(j)))<r^2
                ROBUST_CONDITION = false;
                break;
            end
        end
        if ROBUST_CONDITION == false
            bisect_kd_MAX = kd_smooth_0;
        elseif ROBUST_CONDITION == true
            bisect_kd_min = kd_smooth_0;
            kp_smooth = kp_smooth_0;
            ki_smooth = ki_smooth_0;
            kd_smooth = kd_smooth_0;
            w_smooth = w_smooth_0;
        end

    end
    f = abs(c+(kp_smooth+1i*(kd_smooth*w-ki_smooth/w))*P_nom)^2;
    ROBUST_CONDITION = true;
    for j=1:length(w_check)
        if double(subs(f,w,w_smooth*w_check(j)))<r^2
            ROBUST_CONDITION = false;
```

```
                break;
            end
        end
        kp_smooth
        ki_smooth
        kd_smooth
        w_smooth
        n_iterations
        ROBUST_CONDITION

    else
        ("for KD = MAX the robust constraints are satisfied")
    end
end % SMOOTH ENVELOPE

%% STEP 4: initial values for solve the optimum problem:

kp0 = kpPI;
ki0 = kiPI;
kd0 = 0;
if corner==1
    w0 = w1PI;
elseif corner==0
    w0 = wpi;
end
PIDtf = kp0 + ki0/s + kd0*s;
Ltf = PIDtf*Pnom_tf;
[~, phaseL] = bode(Ltf, w_vec);
% phaseL = phaseL-360;
i=1;
while phaseL(i)<-180
i = i+1;
end
w180 = w_vec(i);
i=1;
while phaseL(i)>-270
i = i+1;
end
w270 = w_vec(i);
w_start = w0/2;
w_stop = (w180+w270)/2;

% initial condition for seminfcon
Pnom_syms = P_nom;
radius = r;
save('Data Variables/B_parameters_for_seminfcon.mat','Pnom_syms','radius');
clear Pnom_syms radius smooth_nyquist1 smooth_con

%% SOLVE OPTIMIZATION PROBLEM USING: fseminf
% x = fseminf(fun,x0,ntheta,seminfcon,A,b,Aeq,beq,lb,ub,options)
% x = [kp ki kd wc1 wc2]
clear x x0
objfun = @(x)-x(2);
x0 = [kp_smooth ki_smooth kd_smooth w_smooth w_smooth];
ntheta = 1;
A = [];
b = [];
Aeq = [];
beq = [];
lb = [12, 30, 0.14, w_start, w_start];
ub = [20, 40, 1, w_stop, w_stop];
```

```
[x,fval,exitflag,output,lambda] = fseminf(objfun,x0,ntheta,@seminfcon,...
                                                A,b,Aeq,beq,lb,ub);
x
exitflag
output

kp = x(1);
ki = x(2);
kd = x(3);

%% verification of the solution
PIDtf = kp + ki/s + kd*s;
figure(9);
hold on;
nyquist(PIDtf*Pnom_tf);
circle.x = cos(w)/Ms;
circle.y = sin(w)/Ms;
fplot(circle.x-1,circle.y,'r');
hold off;

Ltf = PIDtf*Pnom_tf;
Wtf = Ltf/(1+Ltf);
figure(10);
hold on;
step(Wtf);
grid on;
hold off;



function [c, ceq, K1, s] = seminfcon(x,s)
% No finite nonlinear inequality and equality constraints
syms w real positive
syms K real
syms Ki real
syms Kd real

load('Data Variables/B_parameters_for_seminfcon.mat');
P_nom = Pnom_syms;
r= radius;
% smooth1 = smooth_nyquist1;

ellipse = (real(1+(K+1i*(Kd*w-Ki/w))*P_nom))^2+...
                    (imag(1+(K+1i*(Kd*w-Ki/w))*P_nom))^2;
ellipse_diff = diff(ellipse,w);
ellipse = matlabFunction(ellipse); % @(K,Kd,Ki,w)
ellipse_diff = matlabFunction(ellipse_diff); % @(K,Kd,Ki,w)

ellipse_fnc1 =  ellipse(x(1),x(3),x(2),x(4))-r^2;
ellipse_fnc2 =  ellipse(x(1),x(3),x(2),x(5))-r^2;
ellipse_diff_fnc1 =  ellipse_diff(x(1),x(3),x(2),x(4));
ellipse_diff_fnc2 =  ellipse_diff(x(1),x(3),x(2),x(5));

ceq = [ellipse_fnc1, ellipse_fnc2, ellipse_diff_fnc1, ellipse_diff_fnc2];

% Sample set
if isnan(s)
    s = [1 0];
end

t1 = 4.98:s(1):51.48*50;
```

```
K1 = -ellipse(x(1), x(3), x(2), t1) +r^2;

end
```

Appendix B

Figures

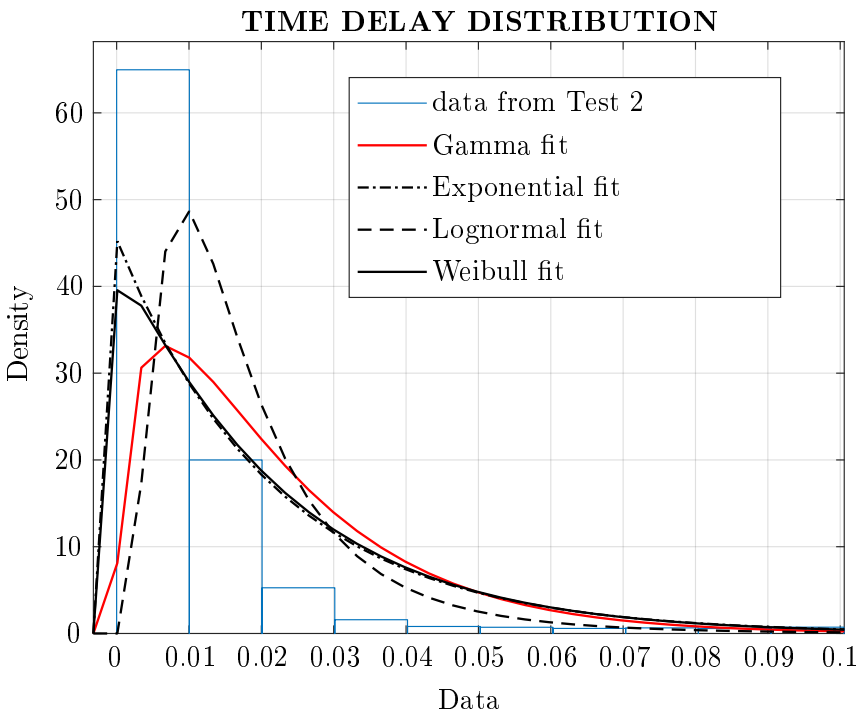


Figure B.1: Various models used to fit the distribution of the RTT. The data are from Test2, see 3.2

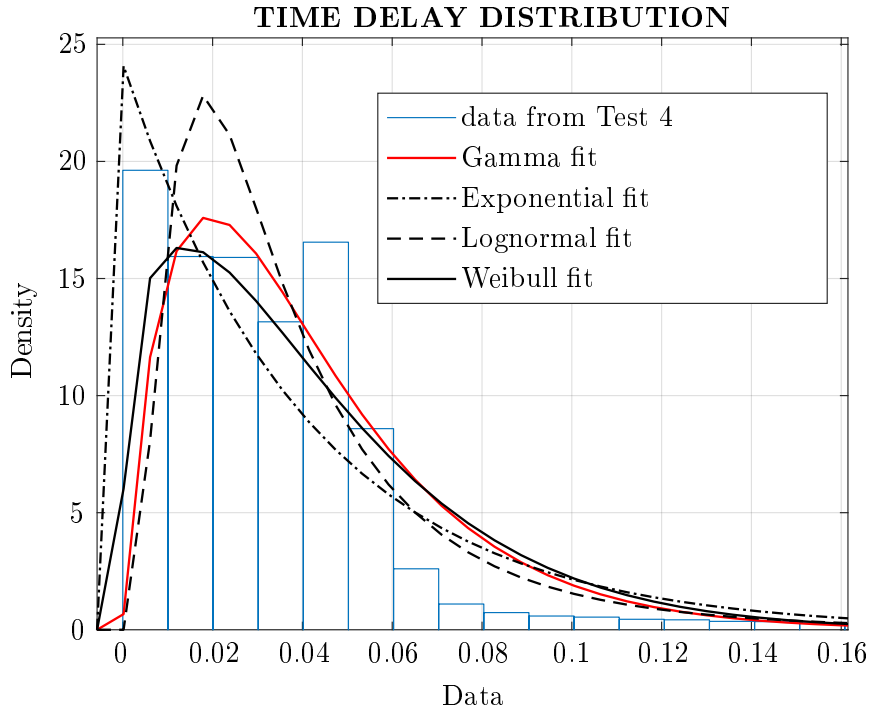


Figure B.2: Various models used to fit the distribution of the RTT. The data are from Test4, see 3.2

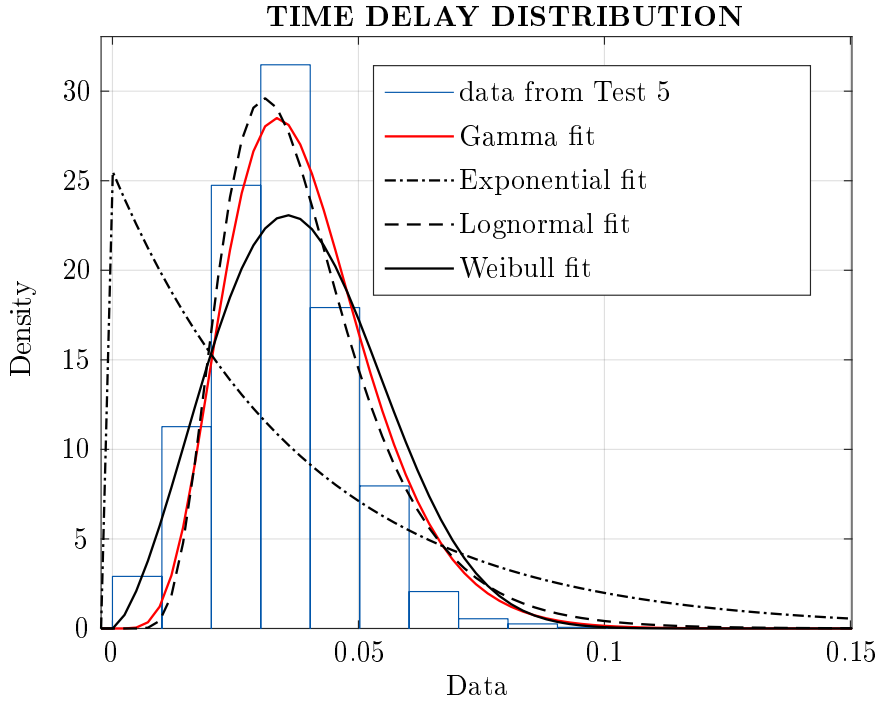


Figure B.3: Various models used to fit the distribution of the RTT. The data are from Test5, see 3.2

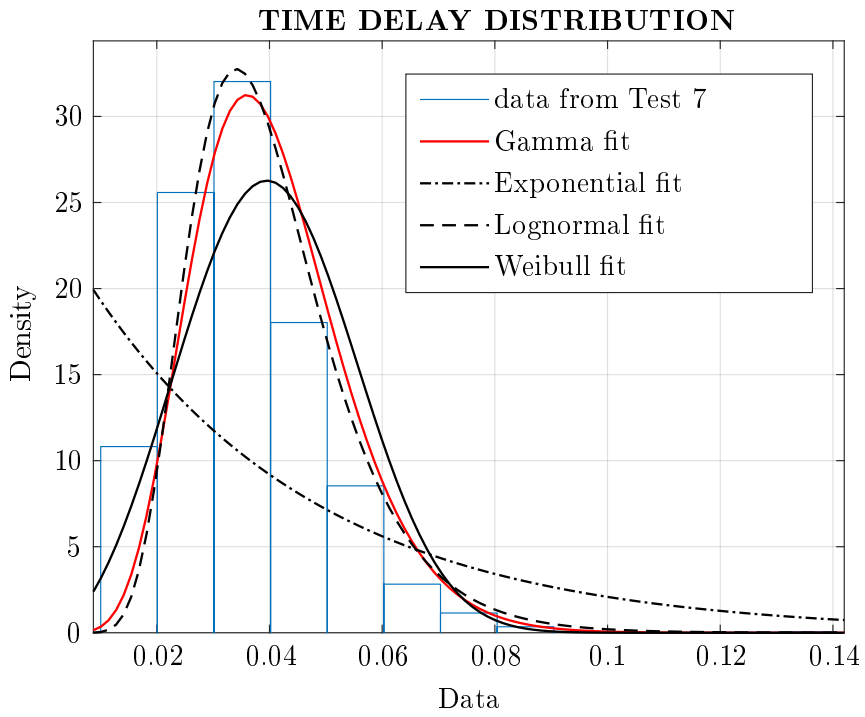


Figure B.4: Various models used to fit the distribution of the RTT. The data are from Test7, see 3.2

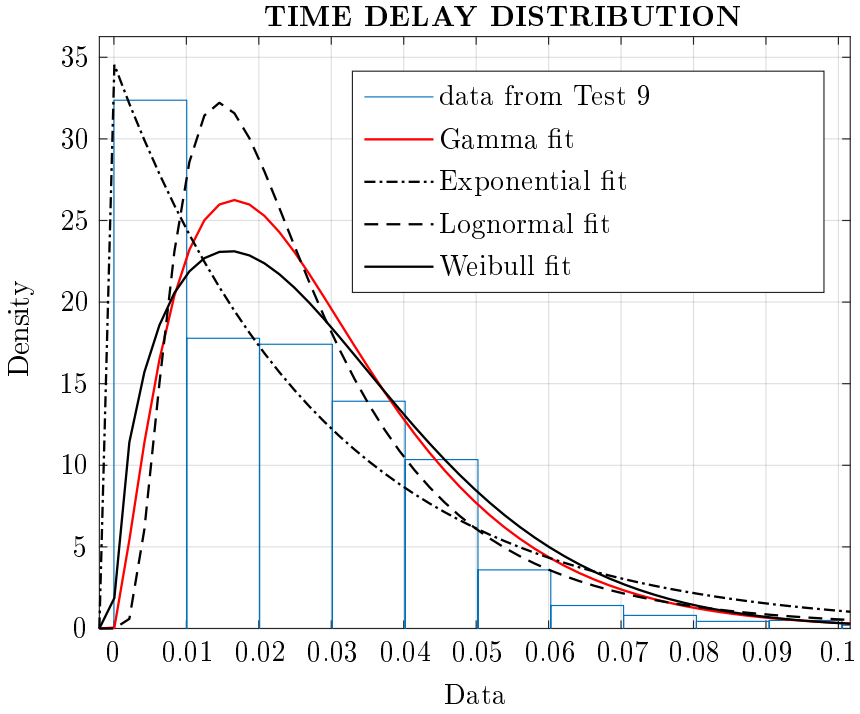


Figure B.5: Various models used to fit the distribution of the RTT. The data are from Test9, see 3.2

Bibliography

[1] Najah Abu Ali et al. "Measured delay distribution in a Wireless Mesh Network test-bed." In: *2008 IEEE/ACS International Conference on Computer Systems and Applications*. 2008, pp. 236–240. DOI: 10.1109/AICCSA.2008.4493540.

[2] Karl Johan Åström and Tore Hägglund. "Revisiting the Ziegler–Nichols step response method for PID control." In: *Journal of process control* 14.6 (2004), pp. 635–650.

[3] Karl Johan Åström and Tore Hägglund. "The future of PID control." In: *Control engineering practice* 9.11 (2001), pp. 1163–1175.

[4] Karl Johan Åström, Hélène Panagopoulos, and Tore Hägglund. "Design of PI controllers based on non-convex optimization." In: *Automatica* 34.5 (1998), pp. 585–601.

[5] Gary Balas et al. "Robust Control Toolbox;[user's guide]." In: *The Math Works, Inc., Tech. Rep* (2021).

[6] CJ Bovy et al. "Analysis of end-to-end delay measurements in Internet." In: *Proc. of the Passive and Active Measurement Workshop-PAM*. Vol. 2002. 2002.

[7] Senior Design Engineer Bruce Cyburt. *How to Connect to an Ethernet Device for Communication*. white-paper. Acromag, Inc, 2018.

[8] Thomas Coleman, Mary Ann Branch, and Andrew Grace. "Optimization Toolbox;[user's guide]." In: *The Math Works, Inc., Tech. Rep* (2021).

[9] Tamas Elteto and Sandor Molnar. "On the distribution of round-trip delays in TCP/IP networks." In: *Proceedings 24th Conference on Local Computer Networks. LCN'99*. IEEE. 1999, pp. 172–181.

[10] Abbas Emami-Naeini Gene F. Franklin J. David Powell. *Feedback Control of Dynamic Systems*. Seventh Edition. Pearson Education Limited, 2015. ISBN: 9781292068909.

[11] *Global Optimization Toolbox;[user's guide]*. 2021. URL: https://www.mathworks.com/help/pdf_doc/gads/gads.pdf.

[12] Chriss Grimholt and Sigurd Skogestad. "Should we forget the smith predictor?" In: (2019).

[13] Tae-Jun Ha, Jaeyoung Lee, and Jong Hyeon Park. "Robust control by inverse optimal PID approach for flexible joint robot manipulator." In: *2007 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE. 2007, pp. 336–341.

[14] Vladimir Hanta and Aleš Procházka. "Rational approximation of time delay." In: *Institute of Chemical Technology in Prague. Department of computing and control engineering. Technická* 5.166 (2009), p. 28.

[15] Norbert Hohenbichler and Jürgen Ackermann. "Synthesis of robust PID controllers for time delay systems." In: *2003 European Control Conference (ECC)*. IEEE. 2003, pp. 1169–1174.

[16] Allen Tannenbaum Jhon Doyle Vruce Francis. *Feedback Control Theory*. Macmillan Publishing Co., 1990.

[17] Mehmet Karakaş. "Determination of network delay distribution over the internet." MA thesis. Middle East Technical University, 2003.

[18] G.P. Liu and S. Daley. "Optimal-tuning nonlinear PID control of hydraulic systems." In: *Control Engineering Practice* 8.9 (2000), pp. 1045–1053. ISSN: 0967-0661. DOI: https://doi.org/10.1016/S0967-0661(00)00042-3. URL: https://www.sciencedirect.com/science/article/pii/S0967066100000423.

[19] M. Vidyasagar Mark W. Spong Seth Hutchinson and. *Robot Modeling and Control*. First Edition. JOHN WILEY and SONS, INC.

[20] Athina Markopoulou, Fouad Tobagi, and Mansour Karam. "Loss and Delay Measurements of Internet Backbones." In: *Computer Communications* 29.10 (2006). Monitoring and Measurements of IP Networks, pp. 1590–1604. ISSN: 0140-3664. DOI: https://doi.org/10.1016/j.comcom.2005.07.011. URL: https://www.sciencedirect.com/science/article/pii/S0140366405002720.

[21] Constantin-Irinel Morărescu, Silviu-Iulian Niculescu, and Keqin Gu. "Stability crossing curves of shifted gamma-distributed delay systems." In: *SIAM Journal on Applied Dynamical Systems* 6.2 (2007), pp. 475–493.

[22] Amarnath Mukherjee. "On the dynamics and significance of low frequency components of Internet load." In: (1992).

[23] Roger C. Fales Noah D. Manring. *Hydraulic Control Systems*. Second Edition. John Wiley and Sons, Inc., 2020. ISBN: 9781119416470.

[24] H Panagopoulos, KJ Astrom, and T Hagglund. "Design of PID controllers based on constrained optimization." In: *Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251)*. Vol. 6. IEEE. 1999, pp. 3858–3862.

[25] Hélène Panagopoulos, KJ Astrom, and T Hagglund. "Design of PID controllers based on constrained optimisation." In: *IEE Proceedings-Control Theory and Applications* 149.1 (2002), pp. 32–40.

[26] Philipp Pasolli and Michael Ruderman. "Hybrid State Feedback Position-Force Control of Hydraulic Cylinder." In: *2019 IEEE International Conference on Mechatronics (ICM)*. Vol. 1. 2019, pp. 54–59. DOI: 10.1109/ICMECH.2019.8722829.

[27] Philipp Pasolli and Michael Ruderman. "Linearized Piecewise Affine in Control and States Hydraulic System: Modeling and Identification." In: Oct. 2018, pp. 4537–4544. DOI: 10.1109/IECON.2018.8591572.

[28] Pasolli Philipp and Ruderman Michael. "Hybrid Position/Force Control for Hydraulic Actuators." In: *2020 28th Mediterranean Conference on Control and Automation (MED)*. 2020, pp. 73–78. DOI: 10.1109/MED48518.2020.9183305.

[29] Robert H. Bishop Richard C. Dorf. *Modern Control Systems*. Thirteenth Edition. Pearson Education Limited, 2017. ISBN: 978292152974.

[30] Otto Roesch and Hubert Roth. "Remote control of mechatronic systems over communication networks." In: *IEEE International Conference Mechatronics and Automation, 2005*. Vol. 3. IEEE. 2005, pp. 1648–1653.

[31] Alessandro Vespignani Romualdo Pastor-Satorras. *Evolution and Structure of the Internet. A Statistical Physics Approach*. CAMBRIDGE UNIVERSITY PRESS, 2004. ISBN: 9780521826983.

[32] Michael Ruderman. "Full- and reduced-order model of hydraulic cylinder for motion control." In: *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*. 2017, pp. 7275–7280. DOI: 10.1109/IECON.2017.8217274.

[33] Michael Ruderman. *Modeling and control of kinetic friction for robotics; [Guest lecture in TTK4195]*. 2021. URL: https://home.uia.no/michaeru/20210413_TTK4195_guest_lecture_.pdf.

[34] Michael G Safonov. "Origins of robust control: Early history and future speculations." In: *Annual Reviews in Control* 36.2 (2012), pp. 173–181.

[35] Loren Shure. *Solving Optimization Problems with MATLAB*. 2018. URL: https://www.matlabexpo.com/content/dam/mathworks/mathworks-dot-com/images/events/matlabexpo/us/2018/master-class-solving-optimization-problems-with-matlab.pdf.

[36] Guillermo J Silva, Aniruddha Datta, and Shankar P Bhattacharyya. "New results on the synthesis of PID controllers." In: *IEEE transactions on automatic control* 47.2 (2002), pp. 241–252.

[37] Guillermo J Silva, Aniruddha Datta, and SP Bhattacharyya. "Robust control design using the PID controller." In: *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.* Vol. 2. IEEE. 2002, pp. 1313–1318.

[38] Sigurd Skogestad and Ian Postlethwaite. *Multivariable Feedback Control. Analysis and design*. Second Edition. JOHN WILEY and SONS, INC., 2001.

[39] Andrei M Sukhov and Natalia Kuznetsova. "What type of distribution for packet delay in a global network should be used in the control theory?" In: *arXiv preprint arXiv:0907.4468* (2009).

[40] Martin Sysel. "MATLAB/simulink TCP/IP communication." In: July 2011, pp. 71–75.

[41] Yunbo Wang, Mehmet C. Vuran, and Steve Goddard. "Cross-Layer Analysis of the End-to-End Delay Distribution in Wireless Sensor Networks." In: *IEEE/ACM Transactions on Networking* 20.1 (2012), pp. 305–318. DOI: 10.1109/TNET.2011.2159845.

[42] Kemin Zhou, John C. Doyle, and Keith Glover. *Robust and Optimal Control*. USA: Prentice-Hall, Inc., 1996. ISBN: 0134565673.

[43] Karl Johan Åström and Tore Hägglund. *Advanced PID Control*. ISA-The Instrumentation, Systems, and Automation Society, 2006. ISBN: 1556179421.

[44] Karl Johan Åström and Richard M. Murray. *Feedback systems. An Indroduction for Scientists and Engineers*. Second Edition. Priceton University Press, 2021. ISBN: 9780691193984.