UNIVERSITÀ DEGLI STUDI DI PADOVA

—

Dipartimento di Ingegneria Industriale

—

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA MECCANICA

SMITH PREDICTOR CORRECTION BY TIME DELAY ESTIMATION WITH COMMUNICATION DISTURBANCE OBSERVER

Relatore: Ch.mo Prof. Ing. GIULIO ROSATI

Correlatore: Tomoyuki Shimono (Yokohama national university)

Laureando: DANIELE CANTON

Matricola: 2004121

ANNO ACCADEMICO 2021-2022

Alla mia famiglia

Contents

Abstract

There are different conditions for controlling a robot system. We could have an almost zero-time delay when between controller and robot there is almost no distance, so the time that the signal needs to reach the robot and come back to the controller is almost zero, or negligible considering the other disturbances in the system. On the other hand, there are situations where the physical distance between controller and robot, or between master and slave considering a bilateral control case, is so large that the time the signal needs to arrive and produce feedback signal, produces not-negligible errors inside control system. Last condition needs not-standard control system algorithms, and more than one solution has been implemented as today. Nowadays new conditions where we need to compensate and solve delay problems are much more common due to online communication. Online communication, in opposite to offline one, needs to consider that communication time between two system is not stable and it is based on several factors. This instability and lack of knowledge on the time that we need to control the robot is a problem for the most common used control algorithm, that is internationally known as Smith predictor. This algorithm needs a stable time delay, so the idea is to start from the communication disturbance observer to identify the time delay and then use this value as input in the Smith predictor. The way to identify the time delay, that I will show you in this thesis, is a statistical method that do a comparison of input signal to the robot and the feedback signal, in order to identify the delay. This method could be used in several type of control system, allowing to use Smith predictor in that case of time delay where communication disturbance observer doesn't work, like bilateral control and force control.

Introduzione

There are different conditions for controlling a robot system. We could have an almost zero-time delay, when between controller and robot there is almost no physical distance, and so the time that the signal needs to reach the robot and come back to the controller is almost zero, or negligible considering the other disturbances in the system. On the other hand, there are situations where the physical distance between controller and robot, or between master and slave considering a bilateral control case, is so large that the time the signal needs to arrive and produce feedback signal, produces not-negligible errors inside control system. Last condition needs not-standard control system algorithms, and more than one solution has been implemented as today. We need to control and compensate time delay system under different condition. The most common is when we have a controller that is far from the robot itself, and this condition is quite common when we consider situation where human beings cannot access to a specific place because it is too dangerous. Common example are dangerous places like nuclear power plant after an accident, like 2011 Fukushima disaster, or the robot that are sent to the Moon or Mars, which cannot be controlled without delay. In the laboratory where I did my research the focus is on haptic system. Haptic system needs of a master robot and a slave robot. The slave should reproduce the position and velocity of master's side one. This control system is used, and its use will growth in a future, especially for teleoperation, so surgery operation in a place where there is no doctor but only instrument and the patient. Nowadays new conditions where we need to compensate and solve delay problems are much more common due to online communication. The spread of internet brings more possibility to control robot in a remote position, but this has its own problems.

Online communication, in opposite to offline one, needs to consider that communication time between two system is not stable and it is based on several factors. This instability and lack of knowledge on the time that we need to control the robot is a problem for the most common used control algorithm, which is internationally known as Smith predictor. This algorithm, as will be shown in next chapter, needs a stable time delay, so it is not suitable for online system. Another control method that works with an instable delay has been discovered in the last decades. This system solves the problem when we have a basic position control problem but not in condition of force control or bilateral control. In these two last cases the best control algorithm is the Smith's one that cannot be used online. The aim of my research is to show a way to find a way to use Smith predictor also under variable time delay. In this way in a future work, it would be possible to implement it to control a system based on force control or bilateral control also under variable time delay. Since Smith predictor requires the correct value of time delay to work, I will show a way to estimate it starting from the algorithm that is now working in a time variable environment, the communication disturbance observer. The thesis is organized as follows: in chapter 2 I will explain the basic theory of a control system based on position control. The current systems to control in case of time delay will be presented in chapter 3. In chapter 4 the theory behind my own method will be shown. In chapter 5 will be presented the results of simulation based on this method. In chapter 6 it will be shown the experiments conducted on a virtual time delay system implemented by myself. Finally, in chapter 7 the results and future works will be shown.

Chapter 1

Motion control

1.1 Introduction

Every generic robotic system needs a controller to perform every movement. In order to do so several algorithms have been developed, but almost everyone is based on errors between a target and current value. In this chapter three main way to control a robot will be explained: position control, force control and bilateral control. In my research only position control has been used, but the application of results could be applied also to different control system. Every type of robot has an electrical motor that has the goal to convert electric power to mechanical power, in order to generate a force to move the robot itself. There are two main types of electrical motor, the rotating and linear one. Both these type shares common electromagnetic properties, the only difference is in shape. In my research I have used only linear motor, but results are not related to the robotic system used, since based on control algorithms. Electric motor of small power, as the one used in robot application, are usually in direct current, at the opposite of high-power electric motors that are controlled in alternative current. Every control system needs of a controller and a driver. The controller is the component that receive the input function and feedback value and based on these generates a small voltage that is applied to the driver. The driver has the function of amplify the voltage that receive from the controller and generate a current for the motor. Drivers usually work at a voltage several time higher than controller, so their goal is to amplify, and they are commonly connected to electric network, so they have also a rectifier to work with direct current. For a normal electrical motor there is a factor of conversion from current to torque or thrust, known as K_I. The sources of this chapter are [1, 2, 3, 4]

1.2 Position Control

Position control is the most common way to control an automatic system. The idea is to minimize the error between target position and current position of the system. To find the current position of the robot we must know the encoder value and how to convert it to the real position of the robot in an environment. If we have a linear motor, as the case of this research, the conversion is just the encoder value multiply for a standard coefficient based on the distance between the single line of the encoder. A simple position control brings several problems. If we consider only the error between the two values of position, the system is not stable.

$$\mathbf{V} = K_P \times (X_{com} - X_{res}) \tag{1.1}$$

Equation (NB) shows the problems. Until $\mathbf{X}_{com}$ is higher than $\mathbf{X}_{res}$ the voltage will be positive, so the motor generates a torque. When $\mathbf{X}_{com}$ will be equal to $\mathbf{X}_{res}$, the value of the generates current will be zero, but it still has a velocity different by zero, since there was a torque until just before zero, so the position will be overcome. The velocity will bring the robot to move away from the target up to a position symmetrical to the starting one, for energy conservation and symmetry of equation. A graph to show this problem is reported in figure 1.1.One way to reduce this problem is to also consider the difference between target and system velocity. Also, the velocity could be estimated using the encoder, so it is a known value. The equation will be now as equation1.2 and it is known as PD control system.

$$\mathbf{V} = K_P \times (x_{com} - x_{res}) + K_V \times (\ddot{x}_{com} - \ddot{x}_{res}) \tag{1.2}$$

This system is more balanced than the previous one, and if K_P and K_V coefficient are well chose, it is easy to reach the right position.

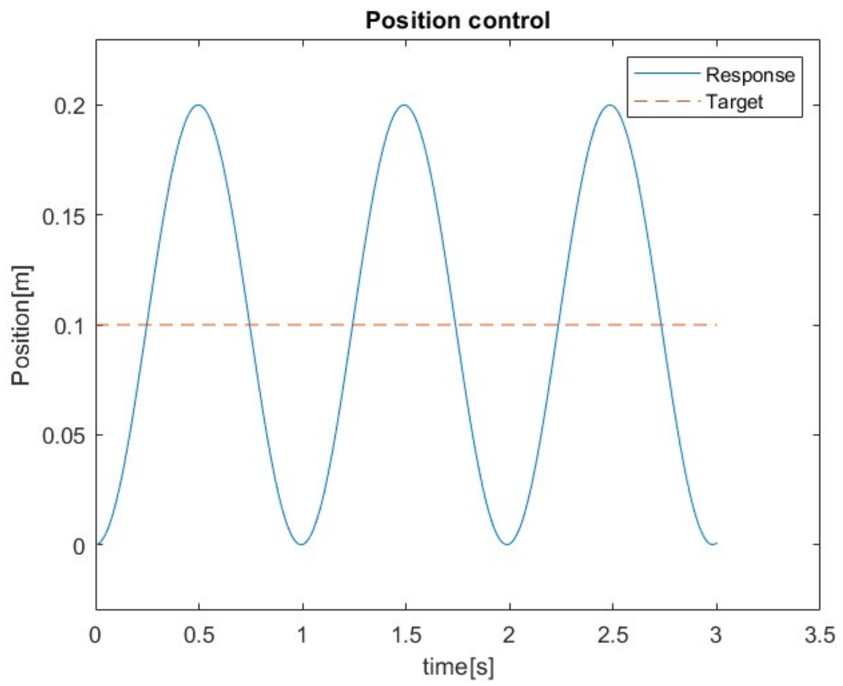

Figure 1.1: Instability of position of only position control as in equation 1.1

Results of a PD control system well balanced are shown in figure 1.2. The target in this figure is a constant position on time, one of the cases that will be studied with time delay problem. This figure wants to show what should be the target of an algorithm that can compensate time delay. A small error still appears in this system due to external force. But this is not a problem in this research since the focus is to compensate the error due to time delay, that is usual larger than the external force effect on PD control. The problem of position control is that using it you cannot control the real force that is applied to the robot, since it is only a position error-based system. This brings limitation in haptic applications, that are the focus of the laboratory where this research has been done.

1.3 Other control systems

In this section is illustrated other two different way to control a robot: force control and haptic control. The description of these two other control systems,

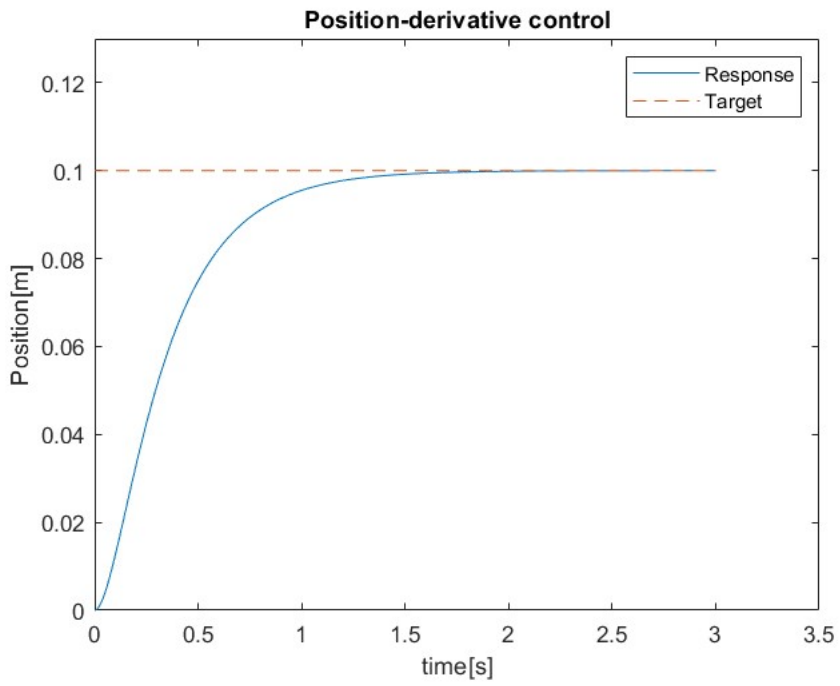

Figure 1.2: Position-derivative control response

despite they will not be used in this research, are important for future works based on this research's results.

1.3.1 Force control

A different conception of control system is that one based on force control. Input is now a force that robot must do in an environment. If there is no environmental reaction force, the robot will accelerate indefinitely. Force control could be used to control exactly the force that the system must do in an environment. This target cannot be reach by position control, that generates a torque or a thrust in function of the error between its own position and target one.

1.3.2 Bilateral control

Bilateral control is based on the presence of two robot. The first one is generally known as master, while the second is the slave. The master is generally controlled by an operator, that freely move the robot in an environment. The target is the slave does the same movement of the master, so it should replicate position and velocity of the master. In order to do such correlation both of the system have an encoder that tracks the position and usually a PD control system is implemented. The slave system could be seen as a simple PD control system that receive the information of target position and velocity from the master. Bilateral control could be improved if a system that estimate the reaction force from environment is implemented. A common way to estimate reaction force is the disturbance observer, that will be illustrated in the next chapter.

1.4 Disturbance observer

In control system a large problem has always been related to external force and how to compensate it in an efficient way. During 80s a new way to do so has been discovered in Japan. The idea is to find the difference between the force that the motor has generated on the robot, knowing the input current in the system and Kt, the torque coefficient, an electromechanical parameter that relates the current with the torque generated by the motor, and the actual acceleration that the robot had due to this force and the dynamics of the system. From dynamics and real acceleration is possible to calculate the force that should have generated this condition. For example, if we consider a linear motor, we have only a mass that moves linear, so the force that should have generated a certain acceleration follows the law $F = m \cdot a$. Now, we know the force that the motor has generated, since we know the control system, so we know voltage and current that the driver has produced, and we also now Kt. So the force generated by the motor is $F = K_t \times i$. Now, we should also consider the internal system force, such as friction or other small disturbance force not related to the environmental one. Now we could calculate the disturbance force, so generally the external force that worked on the system as in 1.3.

$$\mathbf{F}_{Dis} = F_{com} - m \times \ddot{x}_{res} \tag{1.3}$$

The value F_{com} is the force commanded by the controller, so the force that the motor should generate, while $\ddot{x}_{res}$ This force is known as disturbance force and the system is disturbance observer. Since it is common that encoder could produce some error due to its own precision, it is common to use a low pass filter to avoid false signal.

1.5 Application of disturbance observer

The disturbance observer as described in previous chapter could be very useful when we have every type of control system to correct the external force problem or to improve the general results.

1.5.1 Disturbance observer in position control

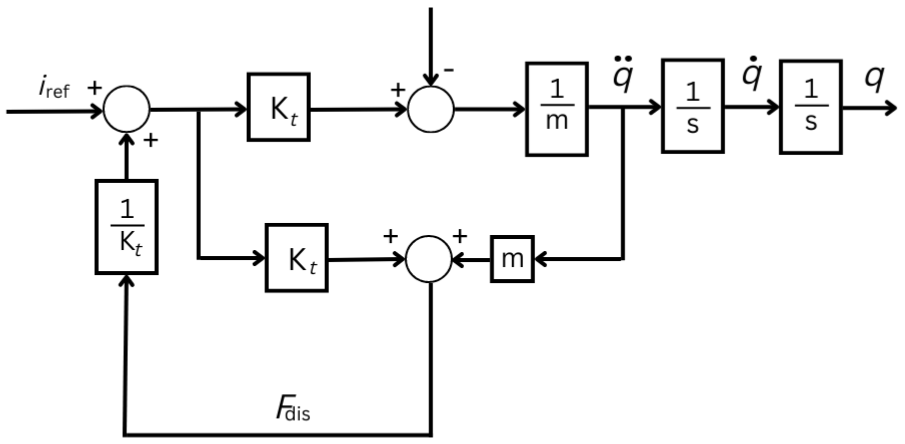

Figure 1.3: Disturbance observer block diagram for position control[1]

In position control system the disturbance observer is used to overcome the problem of external force. If we have an external force on the system, that is opposite than the force that is generated due to error position, we could arrive at the condition in which these two forces have the same value. In this condition the robot cannot reach the target position. So, the disturbance observer could solve the problem. If the system follows the one of figure 1.3, the disturbance force is equal to the external one, so if we sum to the current generated to position control algorithm, the current that the motor needs to generate a force equal to disturbance observer force, the environmental force is overcome, and the system can reach the right position. In the same way, if the external force accelerated the robot, the disturbance observer force will decrease the acceleration in order to achieve the same goal.

1.5.2 Disturbance observer in force control

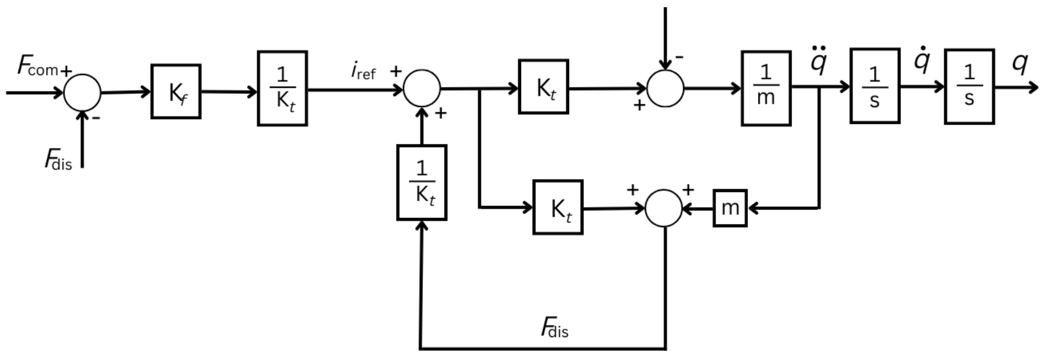

Figure 1.4: Disturbance observer block diagram for force control[1]

As we can see in 1.4, disturbance observer can be implemented also in a force control scenario. In this case disturbance observer has two different goals. The first one is to better improve the results, as in the position control scenario, but is has also a second effect of estimation of external force. In a classical force control system, the simple way to estimate environmental force is to use a force sensor that is not an external device. In opposite, here the force sensor is not necessary, since the external force is estimated as described in 1.4. In this way the robot is cheaper and at the same time could perform better results.

1.5.3 Bilateral control

In a bilateral control system a disturbance observer can be implemented as in 1.5. In this case we have two different types of effect thank to disturbance observation. The first one is to achieve better results in terms of position control, both in the master side and in the slave one. The second effect is the possibility of transmission of force from slave side to master side, in order to give force feedback on master

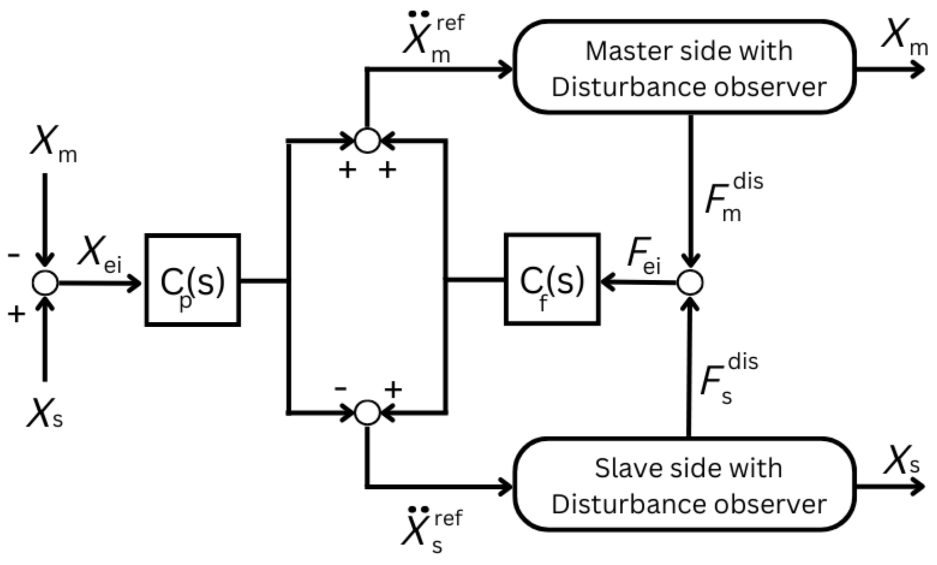

Figure 1.5: Disturbance observer block diagram for bilateral control[4]

side. Indeed, in this scenario if there is an environmental force in the slave side, that generates a force on the slave robot, this force is communicated to the master side to allow a haptic control. The force is generated as the difference between the disturbance force in master and slave side. So, if there is a force in slave side, but not in master's one, the operator that is moving the robot can feel this force as it is moving the slave side itself. This condition is important and interesting, since it is the base of telemedicine with haptic feedback, and the main focus of the laboratory where this research has been done. The aim of this research is to improve the time delay control algorithm to allows better results in this field.

Chapter 2

Time delay control methods

2.1 Introduction

A time delay control means an algorithm that allows to control a robot also in condition of time delay between the controller and the robot itself. A common case could be a robot that is far from the controller and so there is a time delay due to the time the information needs to be transmitted. For example, if we want to control a robot on a geostationary satellite that is at a distance of 36000 km from Earth surface, since the speed of light is almost 300000 km/s, the time the signal needs to complete a round trip is at least of 0.24s. If a classic position control would be implemented in such conditions, it cannot control the system in an accurate way. Indeed, every time the controller receives the feedback signal, so the position and velocity value of the robot, it is a value of the past, and not a current value. So, the controller calculates the current that should be applied to the motor using a wrong value and this could not bring any stability to the system, since it is based on a position of the past and not the real position. The sources for this chapter are [5, 6, 7, 8, 9]

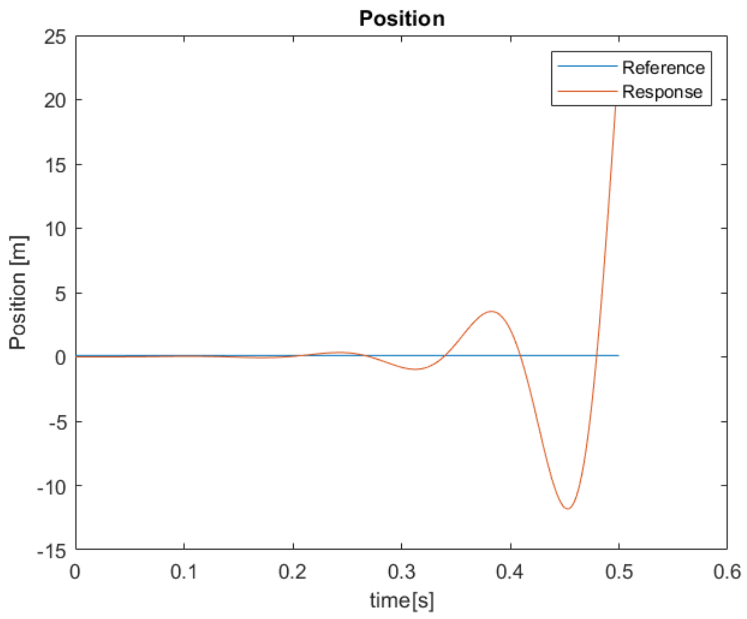

Figure 2.1: What happen if there is no delay control method

In the figure 2.1 there is a basic simulation of what happens if there is a time delay, but we try to use a simple position control algorithm. As can be seen, the system is completely not stable, after few seconds from when the signal is sent. Another case of time delay is on the Earth planet itself. Since nowadays communication can be handled using internet network, also a system of robot and controller can be implemented. Although the distance on the Earth surface is smaller than the distance between Earth and a satellite, the time a signal needs to be transmitted is not lower. The reason is that a signal needs to pass through several server and repeater to reach its destination, and every repetition needs some time. It is quite common that also inside a small continent like Europe the round trip of a signal is higher than 0.1 second, and it is enough to bring error and instability on the system. So, two main algorithms have been implemented as today, both of them with some limitations.

2.2 Smith predictor

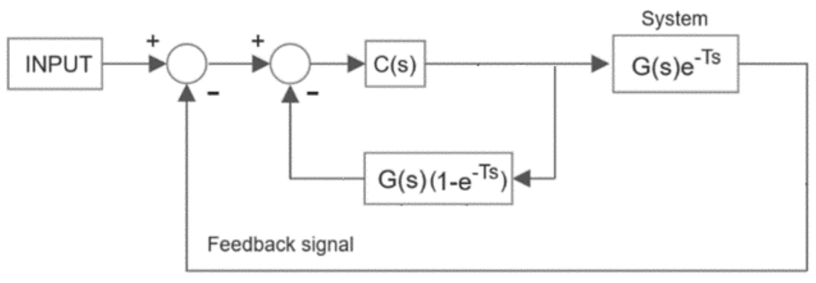

Figure 2.2: Smith control method block diagram [5]

First control algorithm that has been discovered to compensate time delay problem is Smith predictor. This is, as its name suggests, a predictor, so it calculates the position and velocity of the robot according to the input signal that has already been transmitted and to the time delay. In figure 2.2 there is a simple scheme of the Smith predictor in its basic idea. If we start from a known condition of velocity and position of the robot, we can estimate the position that the system have in this moment since we know how much time the signal needs to reach the position and the value of the signal itself. A small explanation of this block diagram should be provided before going on with the explanation of how it works. With input I refer to the target that is provided by the user, or any other type of target value or function, usually a position and velocity condition for a PD system. C(s) is the block with PD algorithm that calculate the current that should be applied to the motor. G(s) is the dynamics of the robot itself, so according to known mass and cinematic. The notation e-Ts is the Laplace representation of time delay in its domain, where s is the Laplace's variable and T is the value of round-trip time delay. With the expression (eq) it means that we are calculating the difference of distance and speed the system has reached to compare to the feedback signal. This happens since we are using the current that will be applied to the robot in a block and where we are also subtracting the position the system reached in this moment according to the current that has been applied. The difference in position and speed inside that block is compensated using the real value of position and velocity that is transmitted by the encoder with a time delay. This is so a fast-forward approach, since it knows the time delay and calculate what should be the position in this moment to give to the motor the right current to reach the input. This algorithm can be used also in force control and bilateral control with good results. The main problem is the limitation of fixed and known time delay. Since it is a fast-forward system if the value of time delay is different from the real value, due to variation of it or wrong calculation, the system is highly unstable.

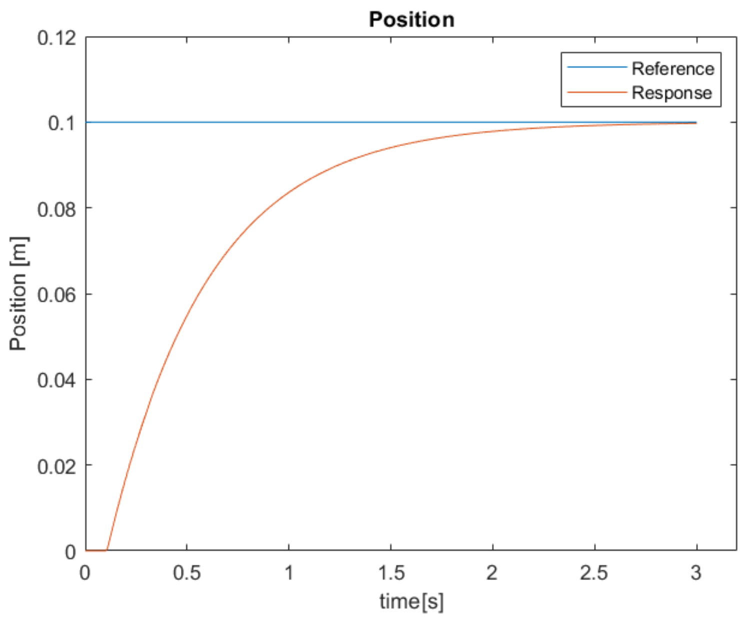

Figure 2.3: Smith control result for simple delay

In figure 2.3 there is a simulation of a time delay system compensated using Smith predictor, if there is right or wrong time delay value inside predictor block. The simulation has been done considering as target function a constant position of 0.1 m, and a speed of 0 m/s. The robot is supposed to be a linear motor with mass of 1 kg to simplify the system. Since it is a computer simulation a sample time of 1ms has been set for both simulations. In order to have perfect parallelism, also K_P and K_V are the same. As result of this simulation in the case of wrong time delay inside predictor block, the right position has never been reached and the system is highly unstable. The aim of this research is to find a way to use Smith predictor starting from a wrong value of time delay and basing on data analysis find the right time delay to correct the predictor in an automatic way. Nowadays a correction could be done by an operator, but is not an automatic one, and cannot work if there is a variation when the system is already working.

2.3 Communication disturbance observer

Another way to compensate time delay in a robotic system is communication disturbance observer (CDOB)

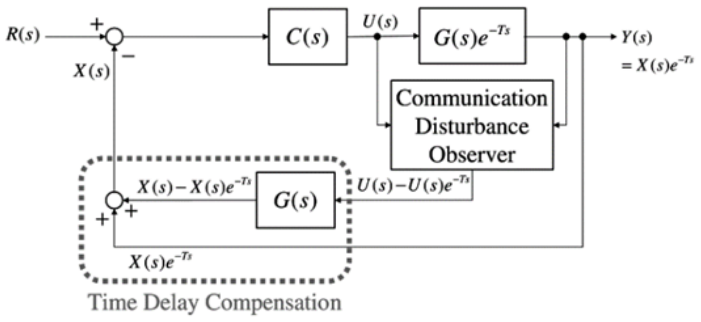

Figure 2.4: Block diagram for CDOB delay control[10]

The CDOB works in a different way compered to Smith predictor. If Smith predictor works in a fast-forward method, CDOB is based on a corrective method based on feedback signal. The name of communication disturbance observer is this one since it is related to the disturbance observer already seen in the previous chapter. The system works considering time delay as a disturbance. In previous chapter only force was considered as a disturbance, while here a disturbance could also be time delay. The communication disturbance observer block takes as input the input signal coming from the C(s) block, so the algorithm that calculate the current that should be applied to the motor, and the feedback signal of encoder that is delayed of a time delay unity. So, applying the same logic of previous chapter, we could identify an acceleration difference between what was applied to the system and the feedback. So, this allows to calculate the different in position and acceleration between the input value and real feedback one. If we sum this difference with the feedback value itself, we can obtain the current position of the robot. This is a feedback system that doesn't need the time delay value, in opposite of Smith predictor that needs this value. So, this is a system that could work also different condition of time delay, and when it is not known.

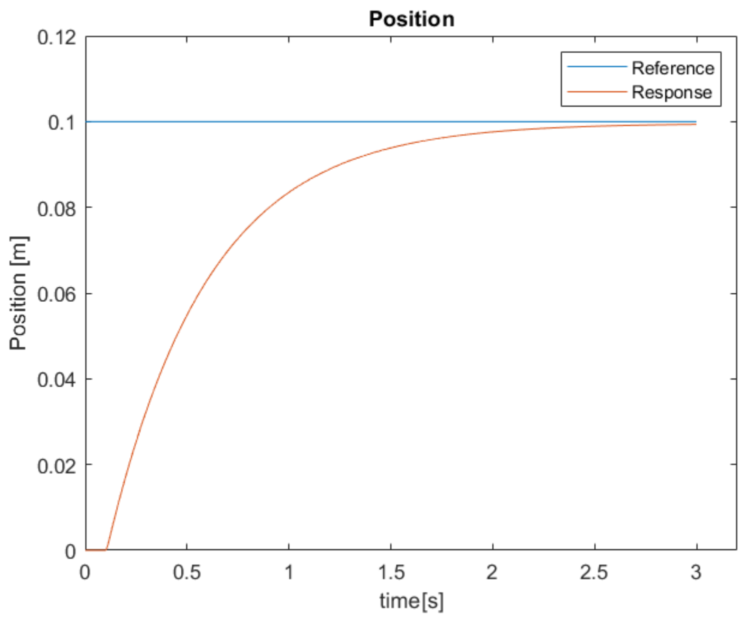

Figure 2.5: CDOB delay control response

In figure figure 2.5 there is a simulation of CDOB behaviour with the same input data of Smith's predictor. As can be seen, CDOB has the same behaviour of the right Smith predictor, but a far better one compared to the wrong Smith's predictor. This would suggest that we should abandon the Smith's predictor generally, since it is not showing any advantages. That is true when we are talking about position control, as in this case. If we start to consider different control algorithms, like bilateral one, or force control, CDOB stops working and only Smith's one works. So, the focus of this research is starting from CDOB advantages, so the possibility to work under different time delay conditions, to estimate a value of the delay and put this value inside Smith predictor, in order to have the possibility of working with variable delay with different control system.

Chapter 3

Proposed solution

3.1 Introduction

In order to use Smith predictor under different conditions of time delay, an automatic system to identify the delay itself must be found. In this thesis I will talk about two different methods that I try to estimate the delay. Both of two methods are based on CDOB. Since CDOB can works under time delay conditions, it means that maybe it is possible to find some useful information here. The two methods are different in approach and also in results. The first one, that will be shown in this chapter, is based on Laplace domain and wants to use a pure analytical method to estimate the delay. The second method, that will be shown in chapter(CH), is a statistical based method, that works using input signal and feedback one to estimate the delay under different conditions. To better understand different solutions already proposed this article

3.2 Laplace based method

The first method is based on Laplace domain so it should be an almost exact solution. If we look at how time delay is expressed in Laplace domain, we can find it is an exponential function as in equation 3.1.

$$U(t-T) = U(s) \cdot e^{-T \cdot s} \tag{3.1}$$

$$CDOB = U(s) - U(s) \cdot e^{-T \cdot s} \tag{3.2}$$

$$T = -\frac{1}{s} \cdot \ln \frac{U(s) - CDOB}{U(s)} \tag{3.3}$$

T is the time delay, and s is the variable of Laplace domain. We know that after communication disturbance observer block, the output is equation 3.2. If we knew this value and we also knew the U(s), we could do as in equation 3.3 and identify T. Now, this equation represents the time delay T as an integration of a logarithm and we should be able to calculate its value using this equation.

3.2.1 Simulation

The simulation works as follows: an input target function is chosen as sinusoidal in position, K_P and K_V are chosen since it will be used a PD control system, the system to control time delay problem is a CDOB, since it is the system that works, and it is the base for estimation. The system works as a classical CDOB, and the results in terms of position response will be correct. The target is to analyse the value of U(s) and CDOB, so the output of communication disturbance observer block, according to the equation equation 3.3 in order to try to estimate the delay. In figure figure 3.1 is a scheme of Simulink simulation for this scenario.

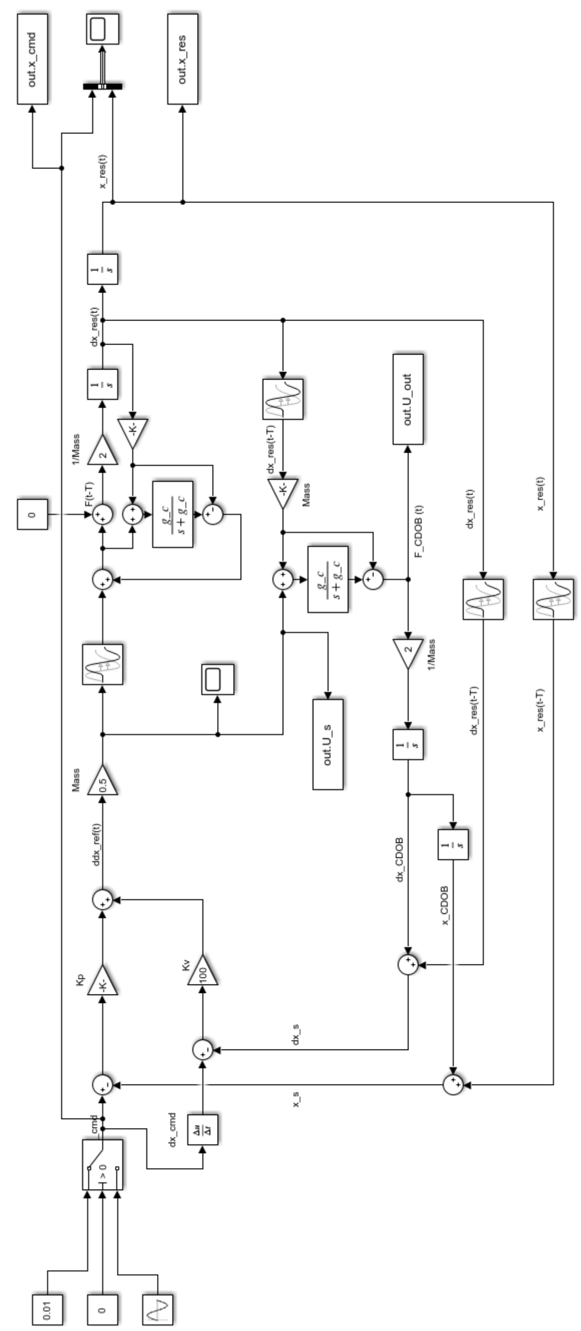

Figure 3.1: Simulink block diagram for Laplace simulation

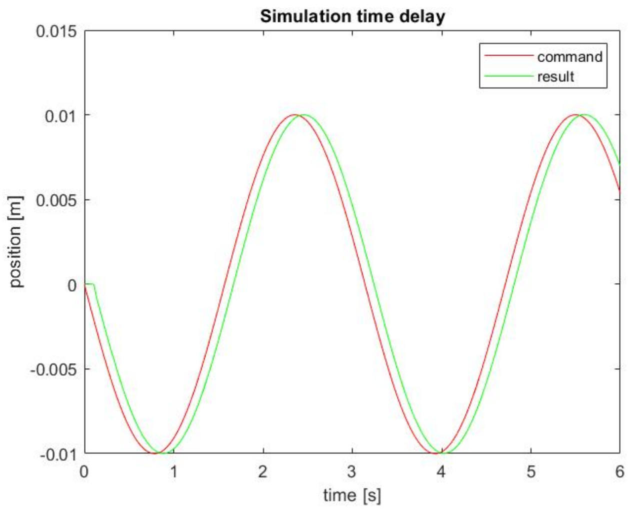

Figure 3.2: Position response in Laplace simulation

Using this block diagram we could simulate a time delay position with communication disturbance observer as compensation, in order to calculate also the value of U(s) and CDOB. Results of these two value are respectively in figure 3.2 and 3.3

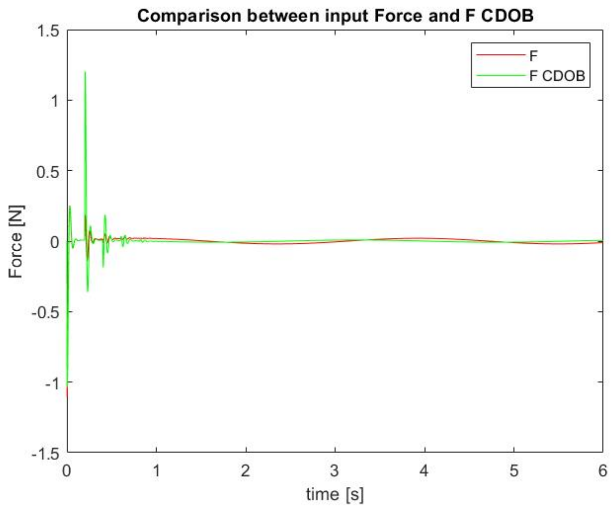

Figure 3.3: Input (F) and CDOB (FCDOB) results for this simulation

In figure 3.4 there is the results of time delay estimation based on integration. As can be seen the results is not optimal. The target is a constant time delay, while in this solution is variable with a sinusoidal function that has almost the same frequency of the target function.

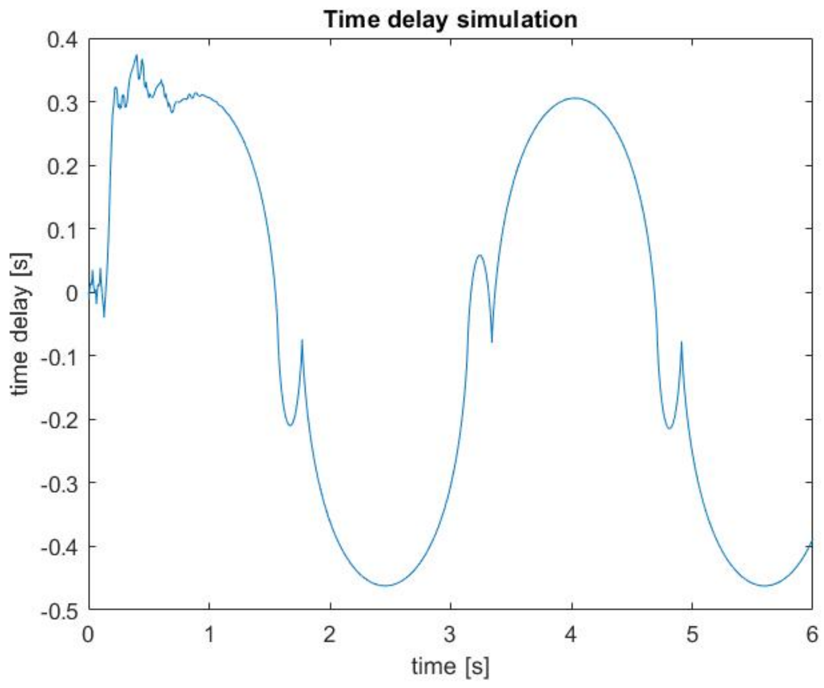

Figure 3.4: Time delay estimation response

3.2.2 Simulation response analysis

Looking at the equation 3.3, a notable thing can be seen. If we want the after a certain estimated time delay reaches a constant value, the integral should be constant and so the logarithm should be zero or should be a variable value with a mean equal to zero. In order to obtain such results, the value inside logarithm should be equal to 1, or variable around one. To obtain this the denominator and nominator of the fraction, so U(s) and $U(s)-U(s)\cdot e^{-T\cdot s}$, should be the same value. But if there is a time delay, the feedback value will be different from the input value, generally, since input value U(s) changes along the time, in order to control the system. Basing on this condition it is not possible to estimate the time delay from mathematical analysis point of view. This way should work from analytical point of view in Laplace domain, but in reality, it is not working. In Laplace domain we have that all value, so U(s), are monotonous growing value, but in reality, we have value of U(s) in function of time, not in Laplace domain. The real value we obtain from PD algorithm is in domain of time, as it is the output value from CDOB block. This doesn't allow to use the value in the equation 3.3, so maybe this is the reason why the time delay estimation it is not working. I have done some attempts to convert time domain value in Laplace domain, but it is not possible. Also, if it would be possible to convert time delay value in Laplace domain, maybe the results will not change. In Laplace domain value are monotonous growing, so the condition of $U(s) - U(s) \cdot e^{-T \cdot s}$, that is necessary to obtain a constant delay, in a simple case of constant delay, would be achieved only if $U(s) \cdot e^{-T \cdot s}$ is equal to zero, but this condition require infinite time delay or U(s) equal to zero. Both these conditions have few meanings in reality, so a different way to estimate time delay must be found. Analytical point of view in Laplace domain is not working, so it would be better to consider a way in time domain.

3.3 Statistical method

A different way to approach the same problem could be a data analysis of the signal. In the previous chapter there where an attempt to estimate the delay using the value of input and feedback signal in an analytical way, so take the single value of every step and using only it. A different way to estimate the delay could be working on the whole signal in a specific time windows. For example, if I can correlate the feedback signal with something that I already know, there is a possibility to estimate the delay. A common example could be found in previous research for delay estimation in different field. For example, a very popular estimation of time delay is done in acoustic field. The target is to estimate the delay in order to delete the feedback from different microphone in the same room. Since a instrument makes a sound, but it is recorded by different microphone, it could be a repetition of the same sound and it could generate a lower quality. The solution that has been adopted for this case is to try to correlate the signal in input in one microphone with the signal in input in another one. Then, it is possible to shift the data until there is a correlation between the two signals. The time that has been needed to obtain a correlation, using cross correlation method, is the time delay itself. Starting from this premise, I believe that the same idea could be applied also to our case of time delay between robot and controller, using input signal and feedback one. The target is to find a way to correlate the input signal, that is coming from the controller, and the feedback signal that is coming from the robot. The feedback signal is the output not of the position response or acceleration of the robot, but it is considered as feedback to do the comparison the CDOB value. If we consider the equation of communication disturbance observer value (3.2), we notice that we can easily calculate it, since we have feedback value of position and input value. But CDOB is also the delayed input value plus current input value. So from this we can calculate the delayed input value. Now, we have a input value and a delayed one. We could apply some tool to estimate the time delay. The main idea is expressed in follow figures.

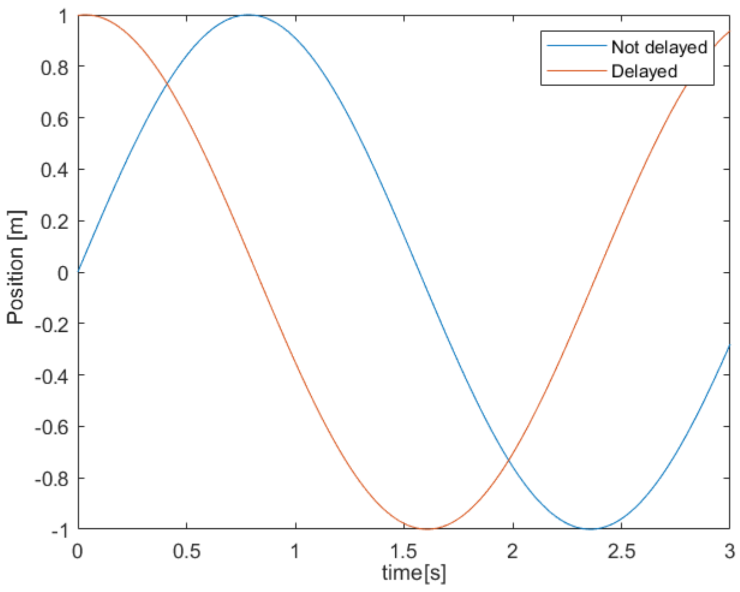

Figure 3.5: Two curves, one is the other delayed

In this figure we have two generic curves that I am using as example to explain the idea behind this method. In these two graphs we have the orange one that is the delayed signal and the blue that is the current time signal. We consider the time as seconds in the past. So, at zero we have the input signal generated in this time, while the delayed is the feedback signal that has just been received by the controller. So, if there would be a time delay of one second, as in this case, the orange has the same shape, but it is indeed delayed, so if the maximum is in zero second for the orange it would be in one second for the not delayed. If we have two curves that one it is supposed to be the other one delayed, it would be possible to find a sort of correlation between them. In other field a cross correlation is commonly used. The idea is to shift the delayed one and study the correlation as a function of the time that has been shifted.

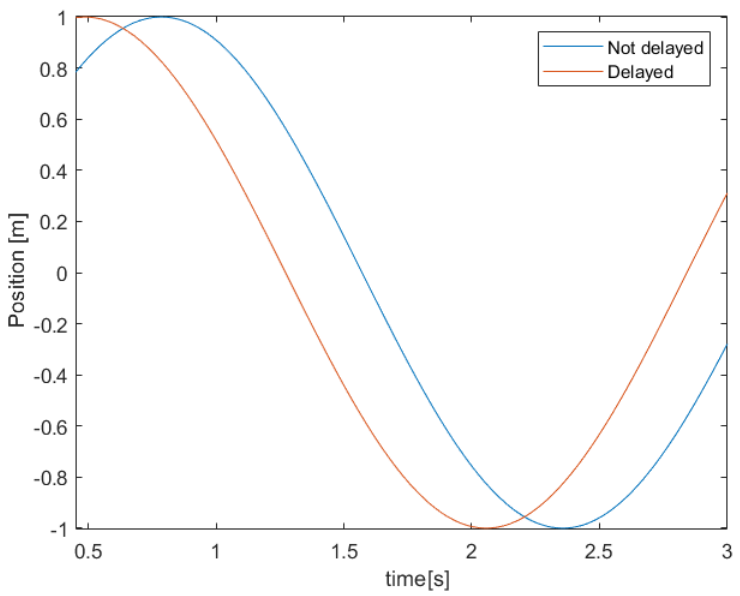

Figure 3.6: First shift to find delay

So for example in the figure 3.6 we have the delayed curve that has been shifted of 0.3 second compare to its initial condition. We want to find a way to understand where the two curves are perfectly overlayed. If we can know this, and we also know for how much time the curve has been shifted we could calculate the time delay.

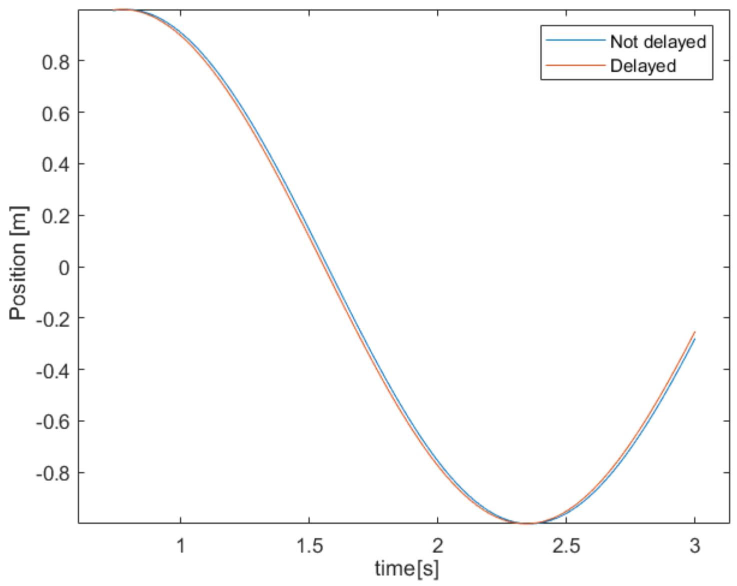

Figure 3.7: First shift to find delay

In figure 3.7 there it the representation of the goal: delayed and not delayed curve are overlayed. We want an automatic way to identify this condition in order to implement a self-control system. All analogical and digital system is handled using a sample time. So, for example the controller sends a signal every 1 ms, since it needs some time to elaborate the data. At the same time, it could read the feedback data at the same sample time. This allow to generate the curves as in the previous picture, since the sample time is constant. So, we can do the shift of the data in an easy way. We should store the value in a data structure used for the purpose of this shifting. It is good to have a certain time window where we operate this analysis, so it is better for example use only the last 3s of value, using a storage data that follows the FIFO rule, for both the curves. Since I want to evaluate a specific condition in every shift, in order to find the exact value of overlaying of the curves, the shift is done on step. Every step is done deleting the first value on the stack of the not delayed curve and the last value of the stack of the delayed curve. This has the same effect of a shifting, and we can control it easily. The shifting of the not delayed curve is done until the number of values inside the stack of the two curves is too small to help to estimate the point of overlaying.

Chapter 4

Statistical Method Verification

4.1 Further consideration

4.1.1 EVALUATION

In the previous paragraph it was explained how the shifting of the curves works, but it is important also to explain how it is possible to numerically estimate the point with the overlaying of the two curves. If for a human eye it would be easy to do this activity, since we don't do it using number, but using our insight, the same cannot be said for a computer. A numerical way using some algorithms must be found. Several strategies have already been used in different conditions, and the most effective for this condition I believe are based on cross correlation or a statistical correlation. The cross-correlation method is based on a comparison between the arrays that stores the data. If the value in a specific position on the first vector is the same value of the same position on the second vector, so there is a cross correlation. This could sound like a very good solution since when there is an overlay the value of the two vectors should be the same, and so the correlation would be perfect. But there is a problem with this method. It is not evaluating all the values, but it is a one-to-one comparison of single positions on the vector. So, if for example there is an intersection between the curves, like in figure 3.6 it is suggesting that here there is a correlation and that we should stop here, but there is a correlation almost in all shifting. So, this method has some problem because it is providing a lot of false positive. An extension of this method could be to evaluate for each step how many positions satisfy the condition of cross correlation, so the value is the same for both arrays. The point with max of this value would be the step with perfect overlay since it is the only step when there is a perfect correlation between all the value of the two arrays. Also, this approach has some problem if the signal is not clean. Indeed, if there is some noise, there could be a false result, due to the not perfect value. Moreover, it would probably doesn't find any point that follows the require condition, so it could not generate any estimation at all. But this method suggests that a correlation of some sort could help to solve the problem.

4.1.2 Statistical

A different way than compering single value on position of the array could be a more statistical method. In statistics a correlation is usually defined using standard deviation and mean. What I am talking about is generally known as correlation coefficient defined as equation 4.1.

$$\rho = \frac{cov(x, y)}{\sigma(x) \cdot \sigma(y)} \tag{4.1}$$

For each step in shifting the correlation coefficient is calculated. Then, since the maximum correlation should be when the two curves overlay, it just considers how many shifts has been necessary to find the maximum value. Then if we know how many shifts were needed and the rate of the shift, that could be equal or higher than the sample time, it is possible to estimate the delay with a simple equation 4.2.

$$\hat{T} = n \times ST \tag{4.2}$$

Where $\hat{\mathrm{T}}$ is the estimated time delay, n is the number of step necessary to obtain the highest correlation coefficient, and ST is the sample time of data collection. This method could be used also in condition of noise, since it doesn't depend on the exact value of the single position in the arrays, but it uses the mean and standard deviation. Of course, since it is a statistical approach, especially in conditions of high noise, there could be some problems to estimate the delay, or the estimation itself could not be so accurate. Other problem could be when there is a small variation between input and feedback signal. If for example we are in condition of no movement because we reached the right position, the correlation coefficient will be equal to one for every step, so it would be impossible to estimate the delay. An example of how it works can be done considering the previous three figures. The correlation coefficient of figure 3.5 and 3.6 is much lower than that of figure 3.6 and so we can say that here is the condition of overlay of the two curves, so we can estimate correctly the delay. But if we are in such conditions, it is usually not necessary to estimate the delay, since there is no movement, and the estimation could be done just when the robot starts to move. Another problem could be if there is a small error in delay estimation. Since it is a statistical approach, an error is possible and it will be studied what happen when the estimation is not perfect, to validate or not the solution.

4.1.3 Time window

Since we need some values to evaluate the system, we should store it in data structure that are generally arrays. These arrays should work following FIFO rule. So, the older values of CDOB and C(s) are deleted to make space to the new values. The dimension of both arrays must be the same to perform an estimation using a statistical method, since otherwise it could mathematically not work. Moreover, the estimation cannot be done until the stack array is completely full. So, the estimation of time delay will of course has a delay itself for this reason and this could generate problem when we will use it to compensate the system. The problems lay on the dimensions of such array. If the array's dimension is too small compared to the time delay, the estimation cannot be done, since the value of C(s) of the past has already been deleted. At the same time, if the dimension is too large a different problem happens. If time delay changes along time, the estimation as explained before works up to a certain point: we should have something like compared to find a correlation coefficient, so at least the shape of the curves must be similar. But if time delay changes, the phase of C(s) and CDOB is not constant. Both change in the same way but it is not constant, so where should be the point with higher correlation coefficient? The solution should be to use a time window for time delay estimation not too large compared to the delay itself. But since we don't know the delay we should suppose it, basing on first estimation, like the distance between robot and controller or basing on average time delay for that specific network connection. Large time stack generates a delay between when the delay had the value that is calculated and when the delay is estimated since we want to have the stack array full to perform an estimation. Moreover, a larger time window generates problem related to computational time. If the time window is large there are more steps where we want to estimate the correlation coefficient. For each step we are calculating the covariance and variance of two arrays. Both covariance and variance require the calculation of the mean of the array, that require a linear time, and then another sum that requires linear time. So, the simple calculation of covariance requires a time that is function of the array dimension power two, or $O(n^2)$. Since this operation is done for every step of the array, we could estimate that the computation complexity of time delay estimation is in order of $O(n^3)$. So, if we double the time window dimension it will requires eight time the time of the previous one. If we want to perform a calculation that is effective, we need to reduce the time that it needs to do the estimation, and so choose the time windows as smaller as possible. If array is too large, another problem lays that computational time will become larger than step time and so if the estimation and the controller are in series, there could be problems in control the system. A way to reduce the complexity is reduce the array dimension. For example, instead of saving the value of C(s) and CDOB every step, we could save it with a different rate compared to the sample time. If the saving rate time is not too large, the estimation of time delay should still be right, but the accuracy of it will be reduced. The precision of the estimation will be as large as the sample of stack array. So, if for example the stack array has a rate of collection of 5ms, the estimation of delay could have a resolution of 5ms, and so also the accuracy will be of 5ms.

4.2 Simulation of time delay estimation

Before applying this method in a real robot, it must be tested in a simulation. All simulations are done using MATLAB, since it is an easier programming language then other for doing simulation, and for the large number of available functions already built inside the code. All simulations are done under the same conditions, that are represented in table. The motor that is here simulated is a linear one, so the simplest one, and widely used to do such experiments. Here the simulations are done just to test if time delay can be estimated under different conditions using the communication disturbance observer and the correction is still not considered.

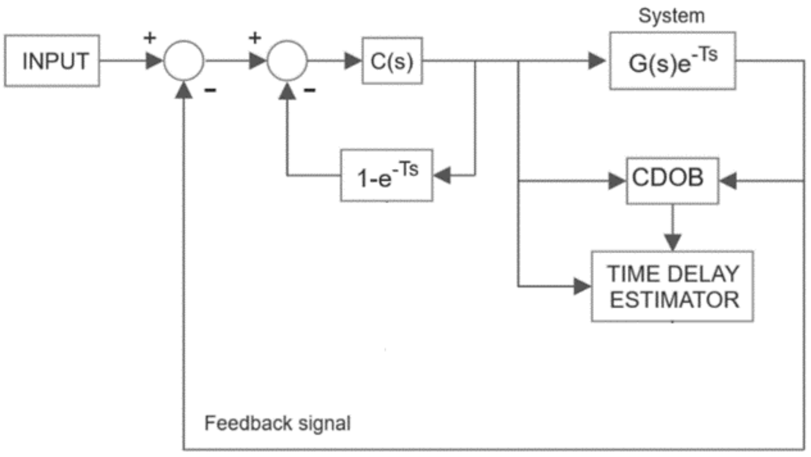

Figure 4.1: Block diagram for time delay estimation

In figure 4.1 there is a representation of the block scheme that want to represent how the system works. Here the compensation of time delay has been done using Smith predictor, as discussed in 2.2. The communication disturbance observer block is here used just as a predictor, so just to understand the value of CDOB, not to correct the system. The "TIME DELAY ESTIMATOR" block works as described in paragraph 4.1.2. It takes as input the value from C(s) that is the not delayed value of force after conversion, and the results of CDOB that should be the C(s) value with a delay of a round trip time. This block returns an estimation in seconds of the delay. The elaboration inside this block can be done with different frequency from sample time, also in function of the accuracy of the system.

4.2.1 Correct Smith's predictor

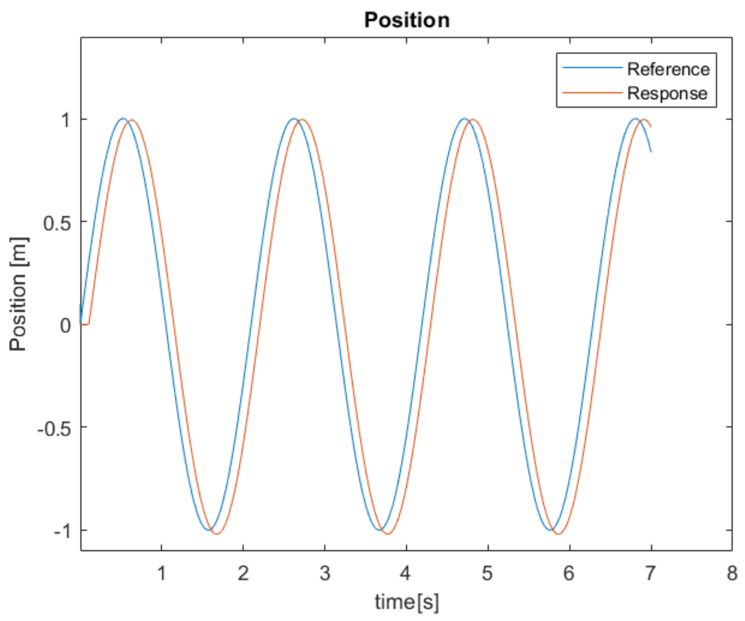

Figure 4.2: Position response for correct delay inside Smith's predictor

As first and simple case it is supposed to have a target position that changes with the time as a sinusoidal function, blue line, and we have a small time-delay of 0.1s. In this condition we have put the value of time delay inside Smith's predictor of 0.2s itself. In this way we are sure that we are calculating the time delay using a system that is already working and it doesn't need to be corrected. Starting from this consideration, we can also try to estimate the delay. We start from the value produced by C(s) and from CDOB block. The estimation is done using the value of figure 4.3 that shows these two values along the time.

The values have the same shape of input and response position, but it is a coincidence and not a general condition. The main reason is that since the delay is constant and the target position follows a constant function, also the results from C(s) is quite regular. Now using the estimation block we could try to estimate the delay. It is important to talk about the dimension of time windows, as described in paragraph 4.1.3. Here time window is set 0.3s in order to verify that also a small time-window works for our goal and to reduce computational time.

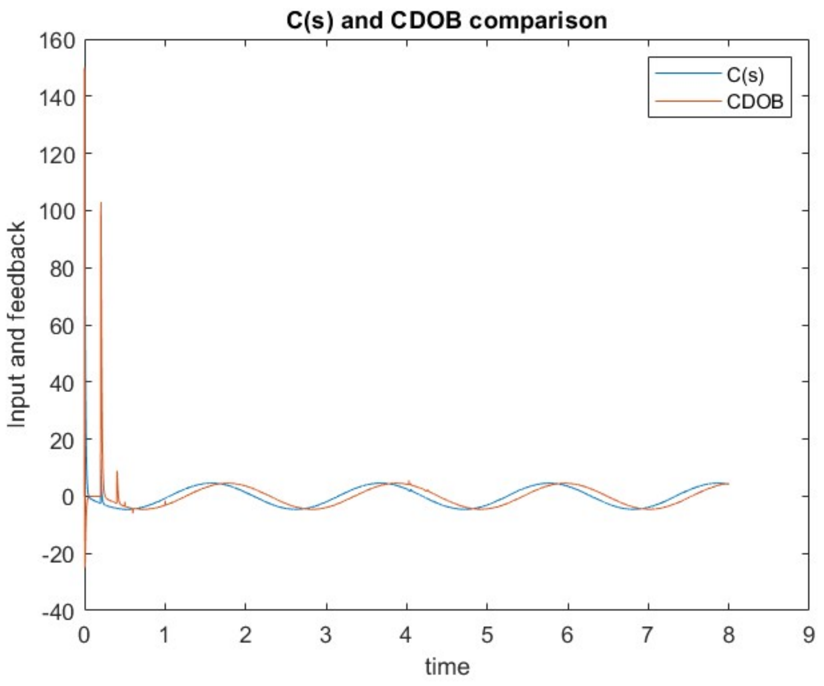

Figure 4.3: C(s) and CDOB values for correct delay condition

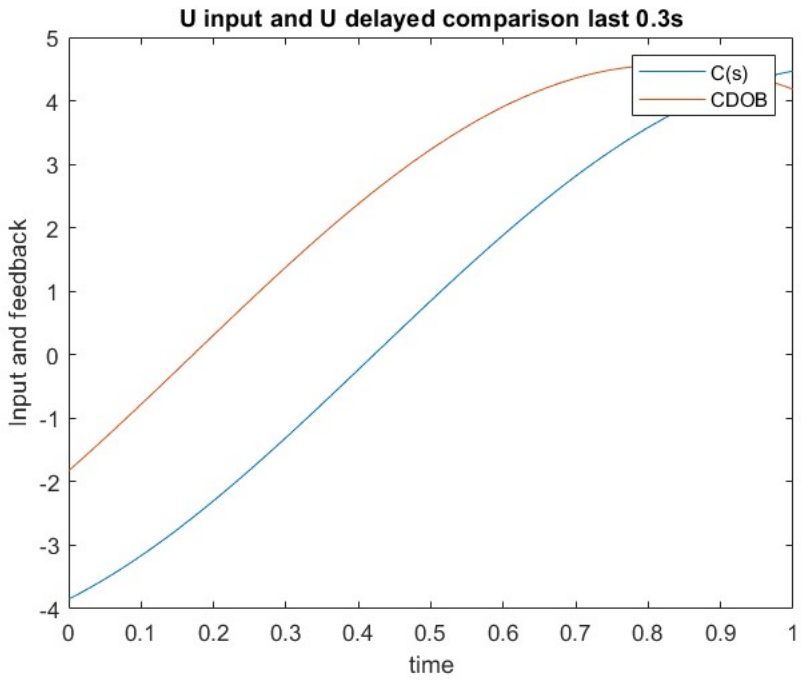

Figure 4.4: Last 0.3s before shifting

In figure 4.4 there is the time windows of last estimation just to show what the estimator algorithm is using.

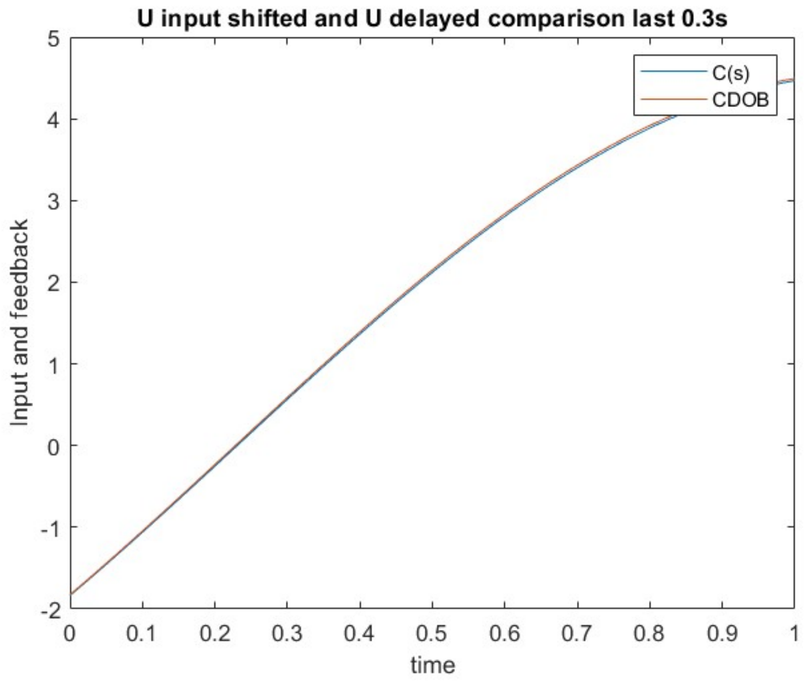

Figure 4.5: Identification of point of higher correlation coefficient

In the next figure there is the instant when estimator block believes there is a perfect overlay and so the point that it uses to calculate the time delay. We can observer that there is an almost perfect overlay, and so the estimation is working well.

In figure 4.6 we can see the estimation of delay. The result is quite good, since after around 1s after the first estimation the estimated delay and real one has the same value. A small error is in the first second, but it is very small compared to the real value, so the system looks working, at least with a constant delay. This value of time delay is not small, if we remember the problem that caused a smaller one, that is represented in figure 2.1, so I believe that at least it is working in this condition. Of course, the estimation shows an error. The estimation is start at 0s of time, but it is not possible, due to time windows dimension and problems discussed in 4.1.3. Here we have an estimation of the delay at the same time the delay is generated. Indeed it is not possible to estimate the delay of the present,

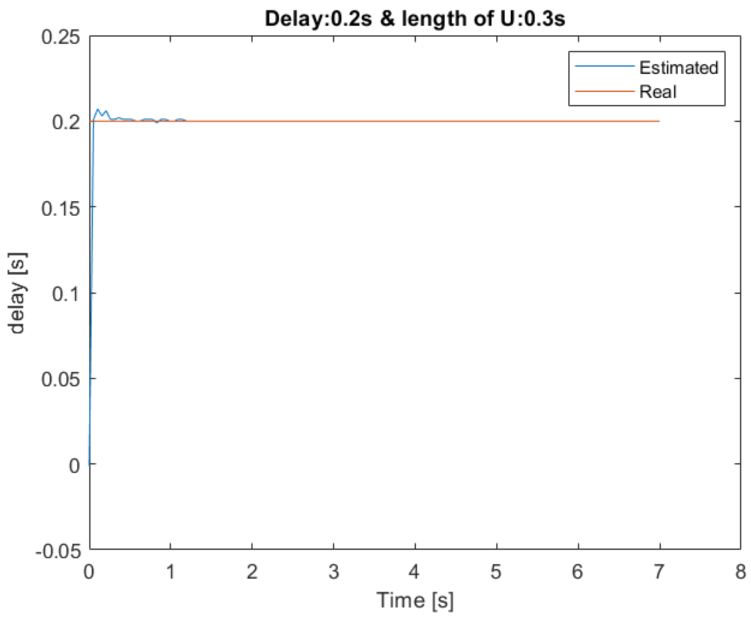

Figure 4.6: Time delay estimation

but only a past delay can be estimated. So the estimation starts at a time as large as the time needed to fill the stack to estimation, but here it is shown as the time delay when it was present. So for example if we have a stack time of 0.5s, we can do the estimation only after 0.5s and not before, since we don't have fill up our stack that we believe is necessary to do the estimation. So if we wanted to plot the time delay in function of the time when it produced that estimation, so we ignore the time window in the representation, otherwise we could include a dead zone for the first stack-time seconds. Anyhow the estimation is working so it's time to test different condition.

4.2.2 Wrong constant delay

In the second scenario we have a time delay inside the Smith's predictor block that is different by the real time delay. It is the real condition that I image for the application of this control system. Supposed delay has a roundtrip of 0.18s while the real time delay is of 0.2s. The difference between real and supposed is of 0.02s of the real one.

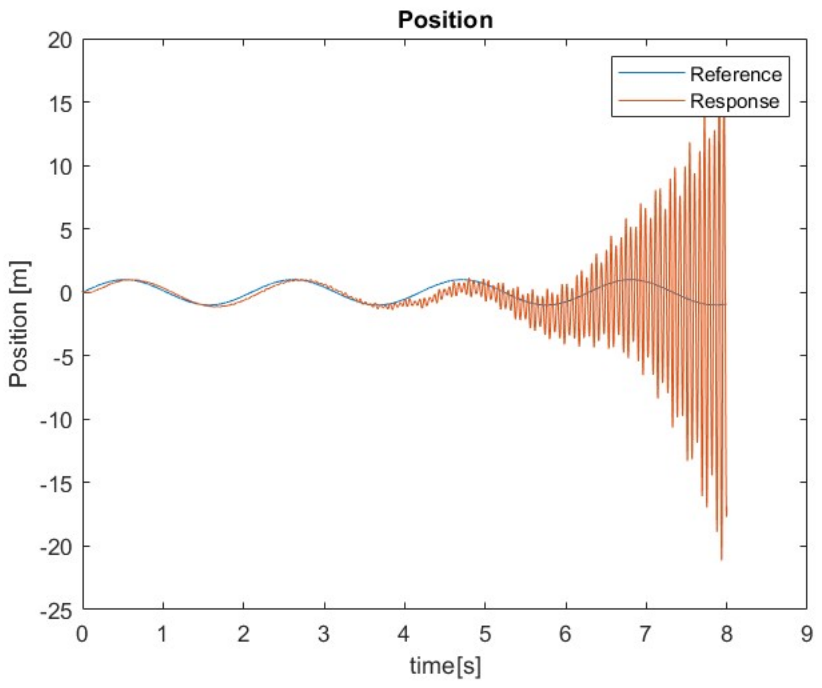

Figure 4.7: Position response for wrong value of constant time delay

In figure 4.7 there is the position response under this condition. As we can see, the position control cannot be achieved by the Smith's predictor under this condition and the response, if the system would be free to move without mechanical constriction, would have an error very large, since just after seven second the position would be of several meter. It would be interesting to understand if the delay estimator as described in previous chapter can or cannot estimate the time delay also if there is a so large error.

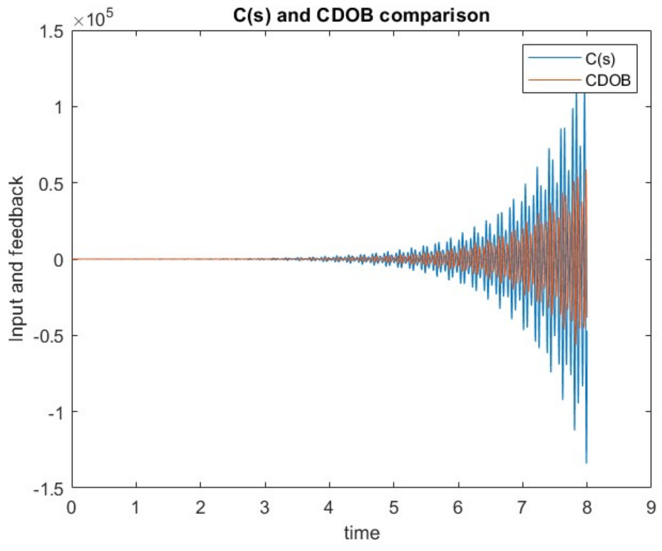

Figure 4.8: C(s) and CDOB for wrong constant time delay

If we look to the value of the value of C(s) and CDOB, we can see that it looks that a not so large variation occurs, in different between position response and target. Moreover, the most important thing is that the amplitude can change, but the shape must be conserved between C(s) and CDOB, in order to do the estimation with a statistical approach.

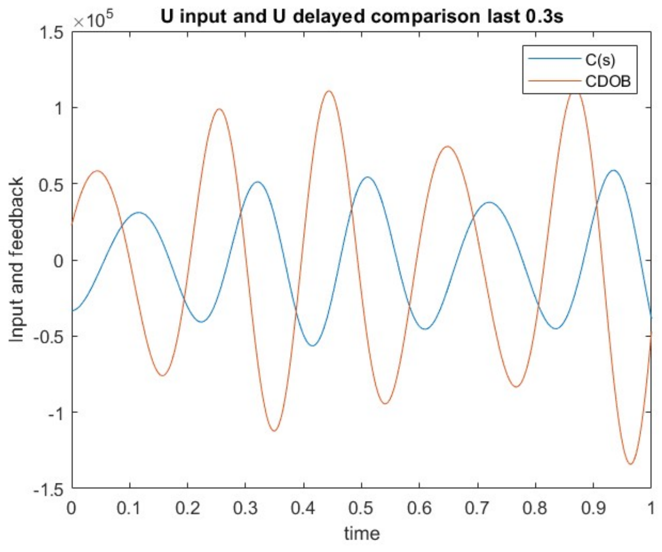

Figure 4.9: C(s) and CDOB value in time window for a generic time

Looking at time windows, it looks like it is possible to work using correlation coefficient, but the results could not be optimal maybe, due to the repetition of same pattern.

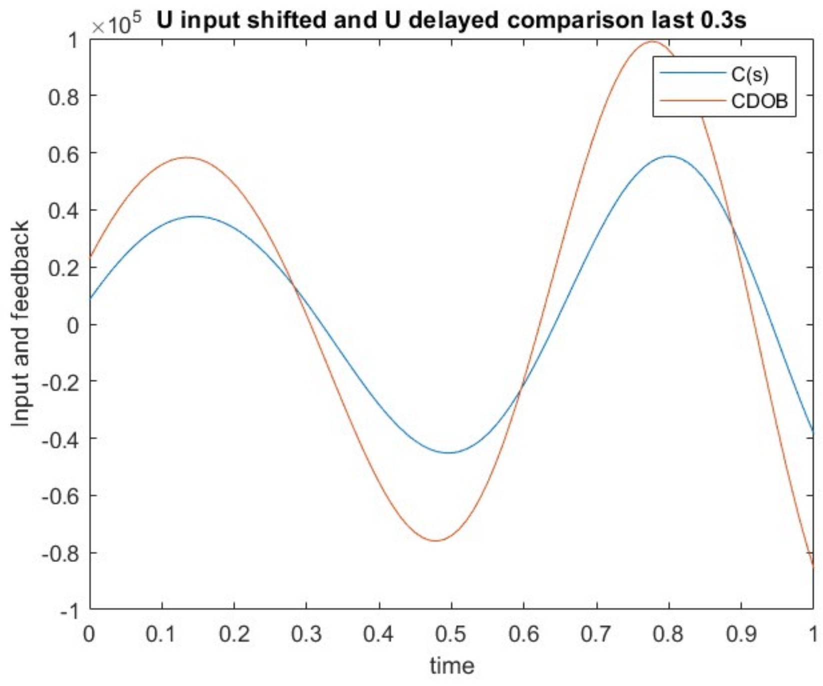

Figure 4.10: C(s) and CDOB overlay with automatic delay estimation

The automatic algorithm suggests, for this case, that the maximum correlation coefficient would be obtained with this translation and the results of time delay estimation suggests the results is almost correct with a small error, as can be seen in figure 4.11

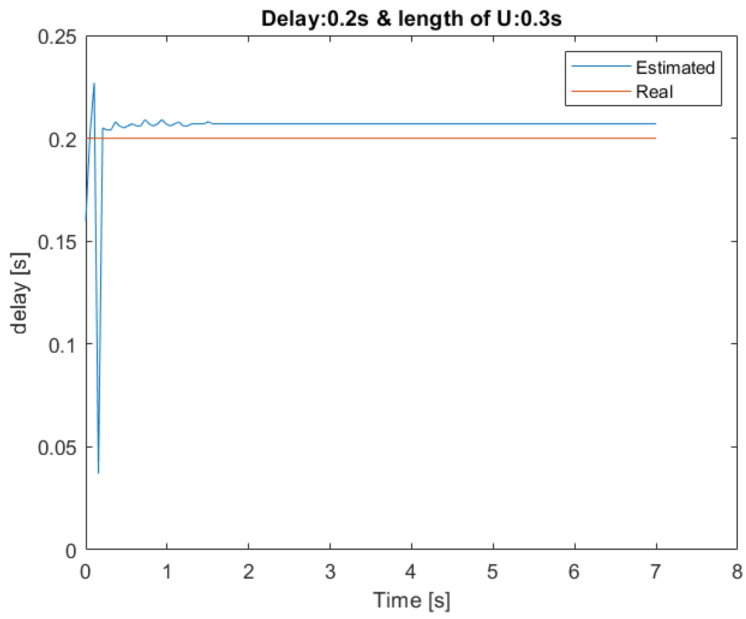

Figure 4.11: Delay estimation for wrong delay inside predictor

The estimated delay, blue line, is almost right with an estimated value of 0.21s and an error of 0.01s of the right value. This estimation brings a good result also if the position response is completely wrong. This is due to the method that is used and that the value considered are C(s) and CDOB that strictly related to each other depending on the time delay mainly. So, the system is robust if also in such conditions can bring a right estimation. The small error maybe could be reduced if the system can self-correct, this condition will be explained in next chapter.

4.2.3 Wrong variable delay

A possible condition is also when time delay is variable along the time. Time delay is influenced by few or no factor if the communication has a stable channel, but nowadays information is commonly transmitted through internet. Using internet brings an instability in delay, that is usually noted as ping, so the time that an information needs to do a round trip. Since it is common and the target of this research, I will try to estimate here also the case where delay changes long the time. We can consider that a step change is the same case discussed before.

Indeed, in the last paragraph inside Smith's predictor there is a constant and know time delay, but it is different by the real value that is constant too. This is the perfect simulation of a step time delay variation, but not a simulation of a time delay that changes along time, maybe now increasing and then decreasing. I believe that this could be the condition of when for some reason the time delay gradually increase and then decrease, maybe due to overwork of the network or technical problems. I tried to represent it with a simple function like equation 4.3.

$$T(t) = 0.07 + 0.35 \cdot sin(t) \tag{4.3}$$

So, time delay changes along the time with a sinusoidal and we want to understand if the estimator block can find the right delay also in this condition. Also, in this condition inside Smith predictor there is of course a wrong time delay, since the purpose is to show the robustness of the algorithm under this condition.

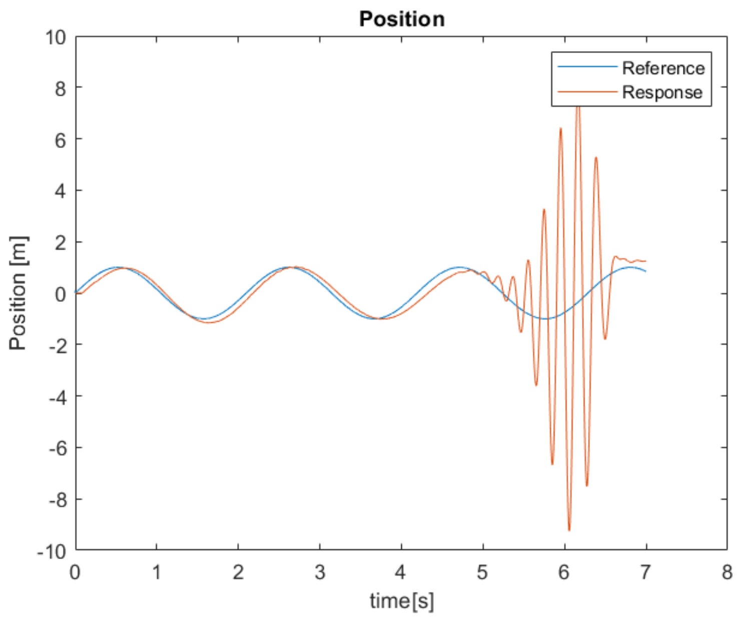

Figure 4.12: Position response for wrong value of variable time delay

Also in this case, since there is no compensation, the position response isn't right. It looks more stable than previous case, but after 5s starts to deviate too.

The more stability is maybe due to the variation of time delay, that starts with the same value inside predictor, so there is no large error initially.

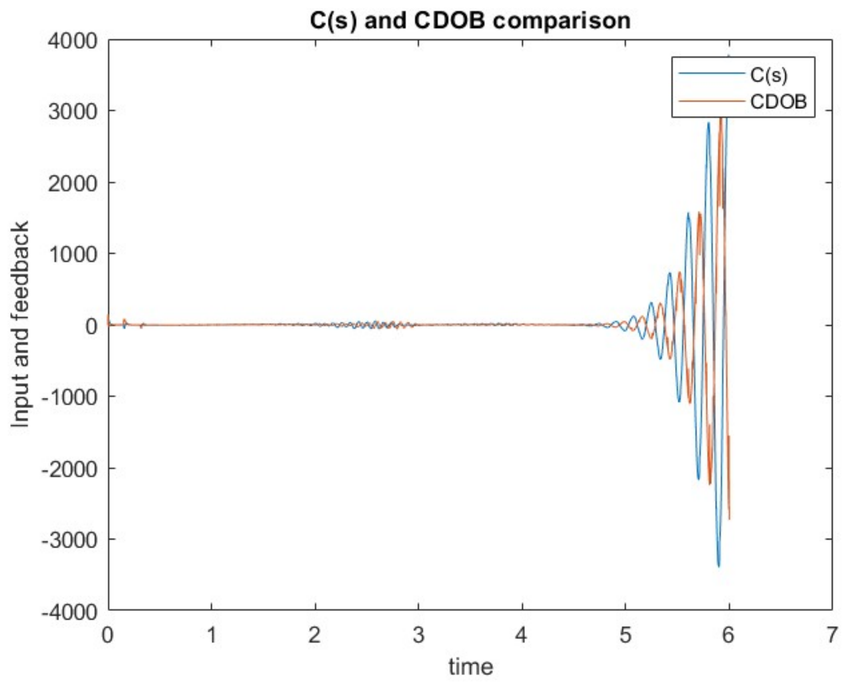

Figure 4.13: C(s) and CDOB for wrong variable time delay

Looking at the value of C(s) and CDOB, also here we can see the same pattern of previous case, and so it will be probably possible to estimate the delay. Here the value of both input and feedback are lower compared to the case of static delay, but there is also here a variation that looks easy also without any calculation to estimate the delay, just watching the distance in time between top value of each wave.

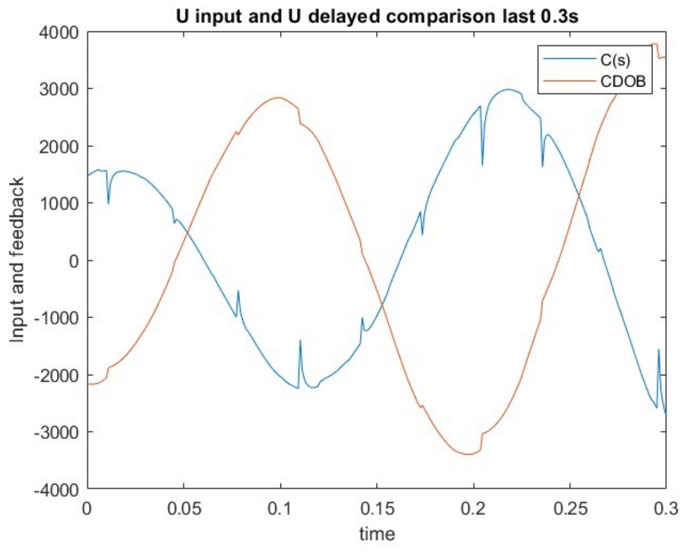

Figure 4.14: C(s) and CDOB value in time window for a generic time

In figure 4.14 we can see the last 0.3s that is the time window that the calculator is using to estimate the delay. We can see a lack of continuity in both the curves, but it should not be a problem, since it is not necessary to have a perfect correlation, but just a high one. Only average shape must me similar, not point by point, so it would be possible to correctly estimate it.

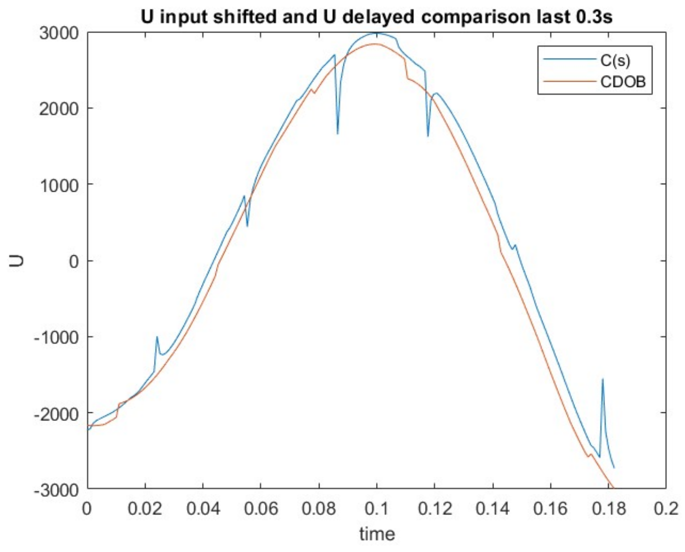

Figure 4.15: C(s) and CDOB overlay with automatic delay estimation

Here in figure 4.15 we can see and almost perfect overlay, also if there are a lot of discontinuity in both the curves. This result could not be achieved with other method not based on statistics, because there is not a perfect correlation between point. Here there is the estimation of the delay. As discussed in paragraph

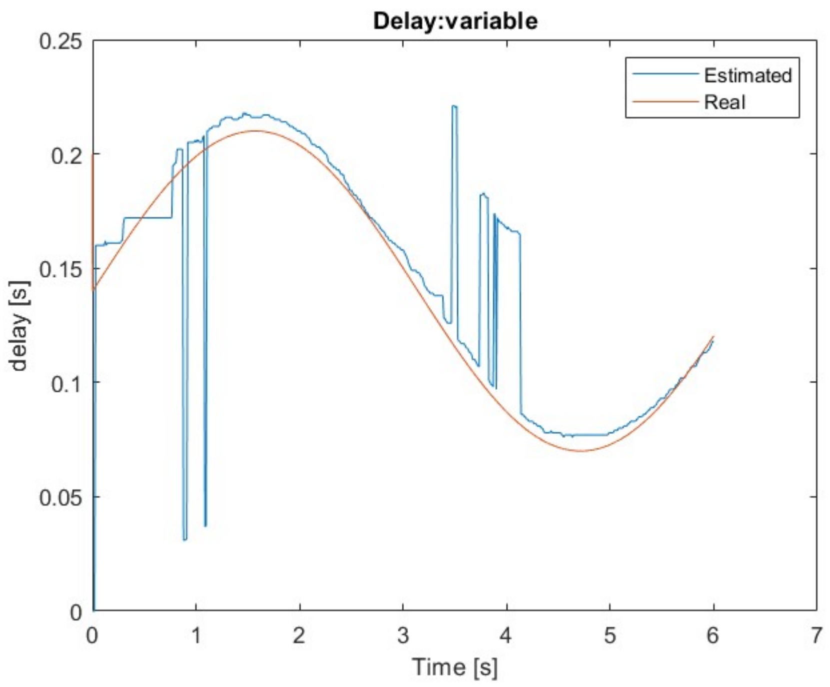

Figure 4.16: Delay estimation for wrong delay inside predictor

4.2.1, the estimation is referred to the time when that delay affected the system, not when it is estimated. We can see here that there is some point where the estimation is not as close as in the other. The average error, if we could ignore this point, would be very low, as in the other scenarios. The estimator can find the right value of the delay also when it is strongly variable with the time and this shows a robustness of the algorithm, at the same time there is an error for almost one second. It would be interesting to understand if this wrong estimation bad affect the control when there would be a self-correction. It will be study in the next chapter. It is important to notice that when the position starts to have problems, after 5s, the estimator is still able to find the right delay.

Chapter 5

Simulation of correction

5.1 Introduction

In this chapter I will explain the results of the integration of the estimator, that we have understood is able to find the real delay of the system, and the Smith predictor, in order to use it also in non-standard condition. The Smith's predictor works using the known delay, so at every step the time delay inside the predictor will be update. So, for example, if the delay increases compared to the value inside the predictor, the predictor when calculate the velocity and position difference, as explained in chapter (CH), will take the value of a different point in the past. So now the predictor cannot work using a FIFO for the value expressed by $C(s){\cdot}e^{-T\cdot s}$, since the time delay is not constant. Now it will be necessary to save more position and velocity value than a normal predictor and the time delay value will be used to find the right value to use in each step. For example, if for the classic Smith's predictor, with an estimated delay of 0.1s it would be enough to store 100 value with a sample rate of 1ms, and use a FIFO system to make it work, now it is necessary to have a larger stack of value, since we don't know the exact delay. Moreover, we cannot pick just the first value put in the stack, like a classic predictor, but it is necessary to take the value in a specific time in the past, so the algorithm itself is more complex. Moreover, the time needed to do this operation is slightly higher, since we have a larger memory and so the store and delete of older value, that now it would be consider not necessary, needs more time. The aim of this chapter is to check the feasibility of the self-correction before testing it in a real robot. If the simulations will bring good results the next step will be to test it in a real environment. The chapter is organized as follows: the first paragraph is a simulation of a static error delay, the second paragraph will be a simulation of a variable delay, while in the last I will do a simulation of a contact with an environment, to understand if can or cannot work, at least from theoretical point of view.

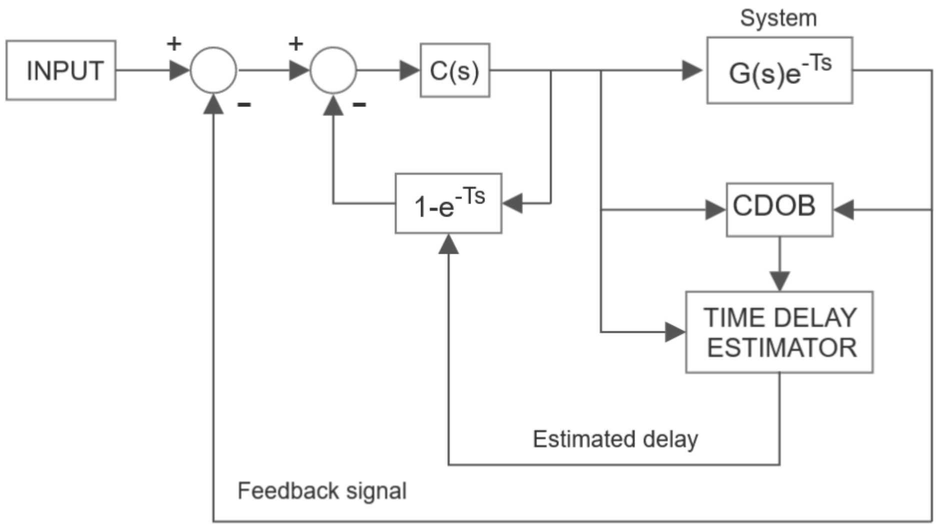

Figure 5.1: Block diagram for this simulation with self-correction

5.2 Constant delay

As mentioned before, in this paragraph will be analysed the behaviour of this control system with self-correction in a case where the delay is constant but the delay value inside Smith's predictor is wrong, the same case of paragraph 4.2.2. Now the difference is that the value inside the predictor will change with the value returned by time delay estimator block. In this case I will show you first the position response of the robot, and then the input and feedback value, and finally the delay.

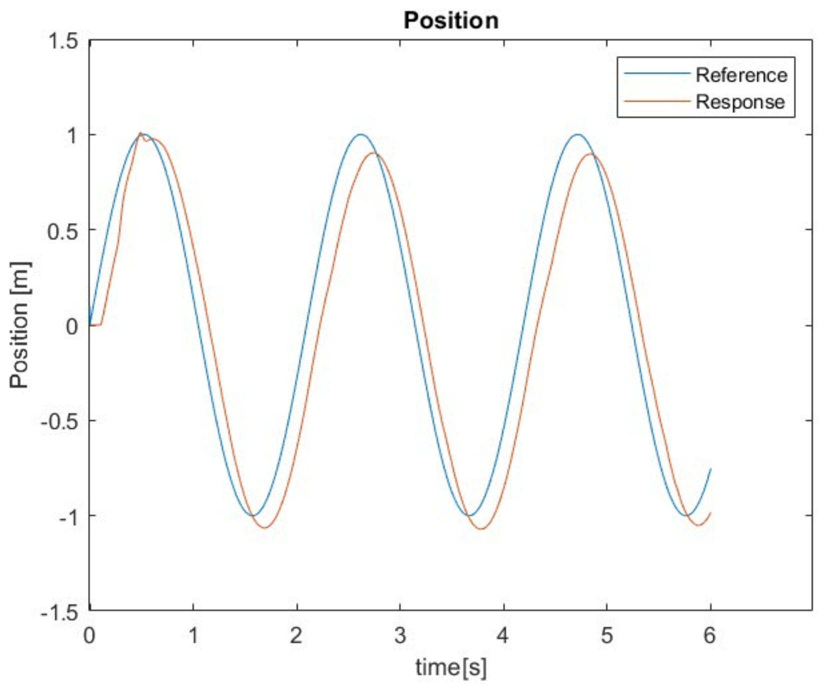

Figure 5.2: Position response for simulation of correction with constant delay

In figure 5.2 it can be seen a great improvement compared to the figure 4.7 of the previous chapter, when the delay was not correctly compensated and it brought an instable system. So it looks working quite well despite still there is a small error.

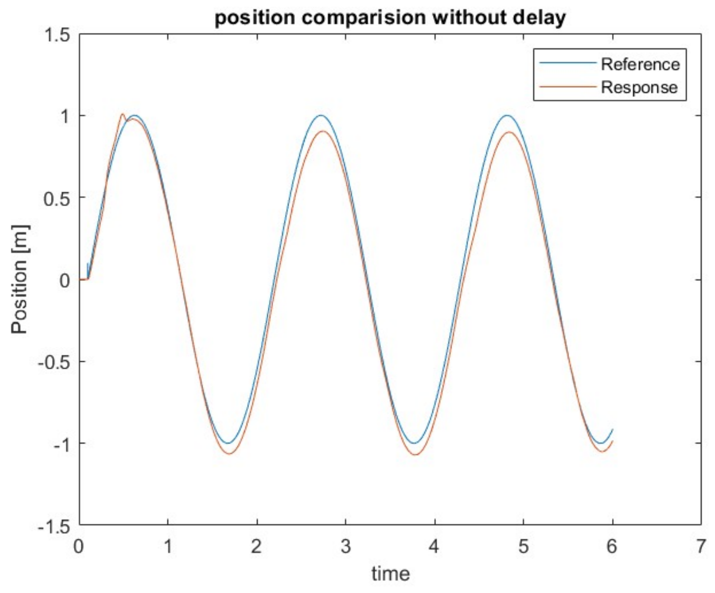

Figure 5.3: Evaluation of position response with translation

To better understand what the real delay is, in this simple situation of constant delay, I have tried to also shift the reference position function of the real delay value. It can be seen that despite there is a position error it is not so large, and it follows the input function almost correctly.

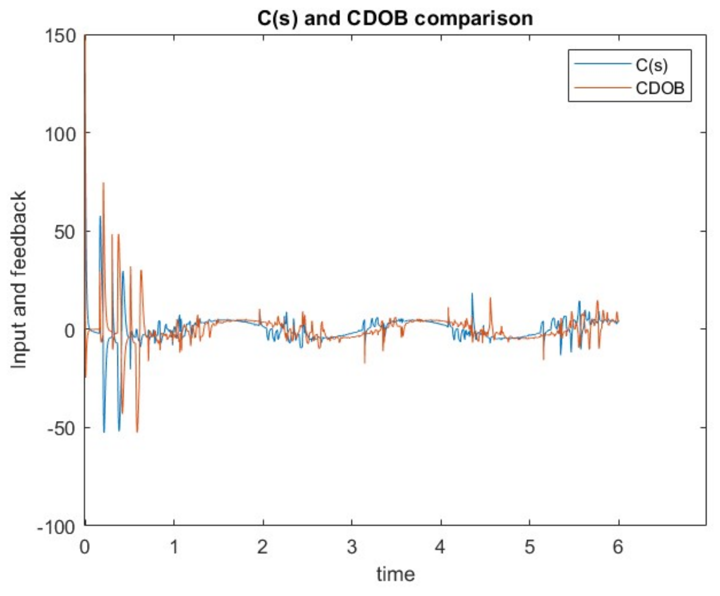

Figure 5.4: C(s) and CDOB response for correction if delay is constant

In figure 5.4 there is a representation of input and communication disturbance observer value. We can see that when the disturbance and input function has little or absent noise, the position in figure 5.2 is almost the same for the two curves, so the system is working well. At the same time, when the disturbance has higher value and high noise coincide with the higher difference in position between reference and response value how it should be, since we are considering time delay and different in position as disturbance.

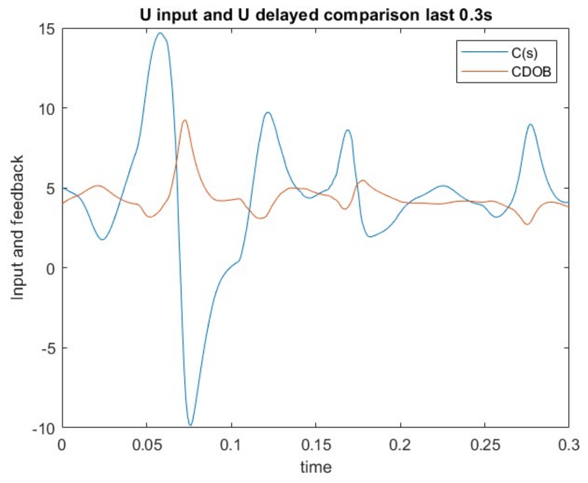

Figure 5.5: C(s) and CDOB in a general time windows

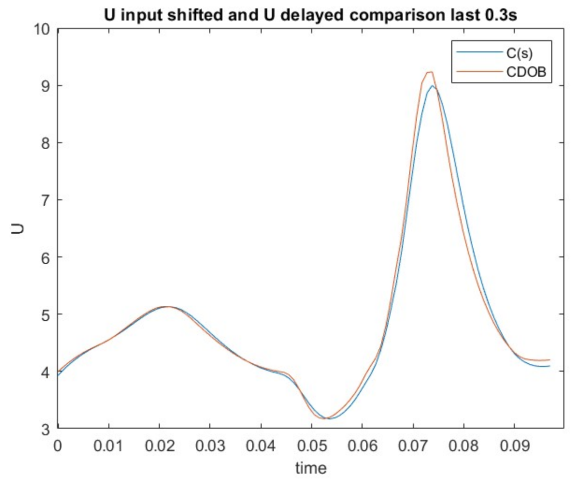

Figure 5.6: C(s) and CDOB overlay using highest correlation coefficient

In figure 5.5 and 5.6 we can see that also with correction the system of estimation using statistics is working. I have set the same time window of previous simulation, so 0.3 s and it is still perfect. The great similarities between input and CDOB allow to estimate the delay almost for all time, despite figure 5.4 could suggest some problem when there is a lot of noise.

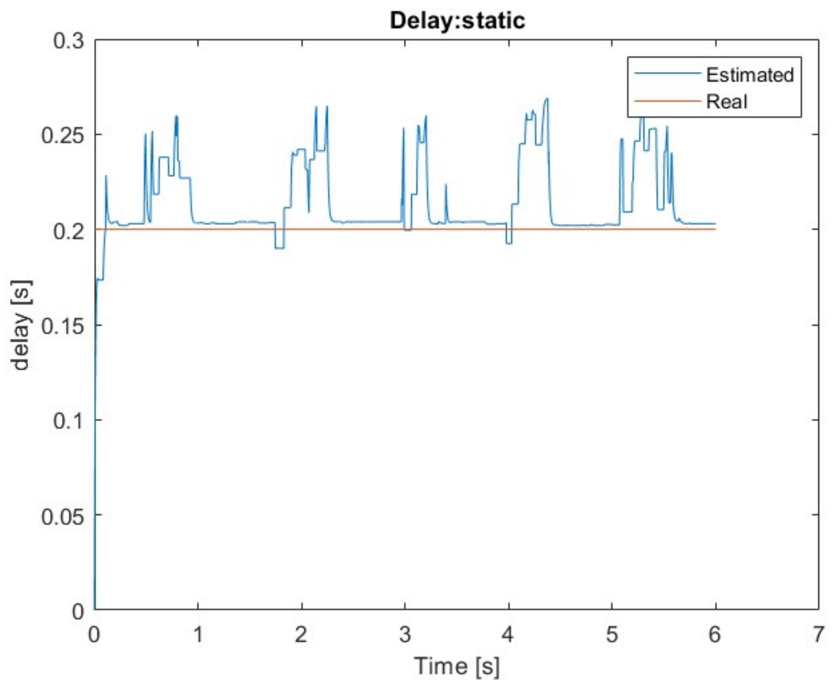

Figure 5.7: Delay response for self correction with delay constant at 0.2s

Observing the estimate time delay it is possible to understand why there are some points when the position response has a low error and point when the error is larger. Where the estimation of time delay is worse, also the position response has worse results. And delay estimation has problems when the disturbance has much noise. An attempt to improve the situation in next configuration will be to set a low pass filter to reduce, as much as possible, the noise in input and CDOB values. However, although the estimation is not perfect the position response has improved a lot compared to the case without self-correction.

5.3 Variable delay

In this scenario the configuration is similar to the previous one, with the only difference that delay is now variable following the equation 4.3.

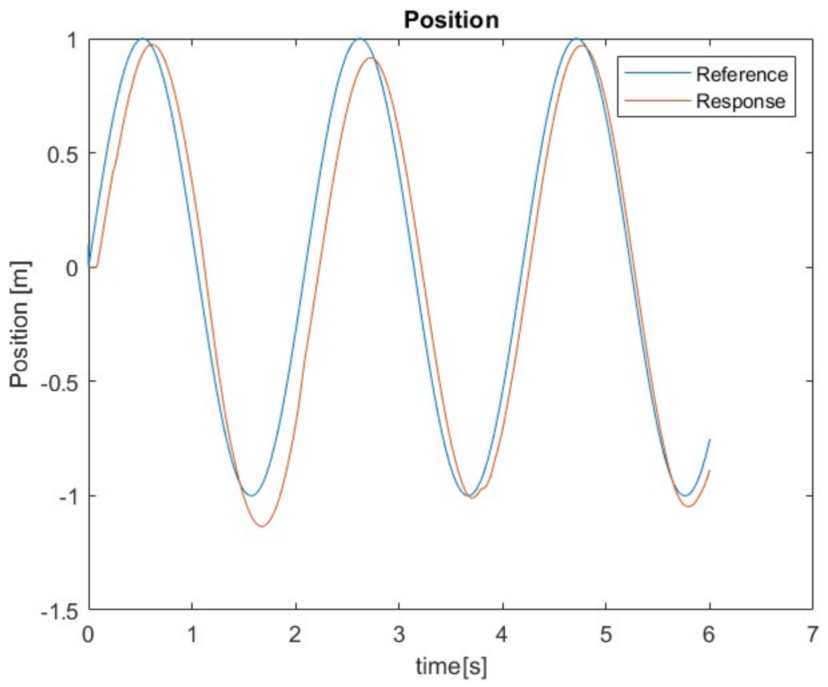

Figure 5.8: Position response for simulation of correction with variable delay

The position response is better than the case without compensation, so it is working. There are still some problems on position control, but it is overall correct, with a large error only after 1.5s and at 4s.

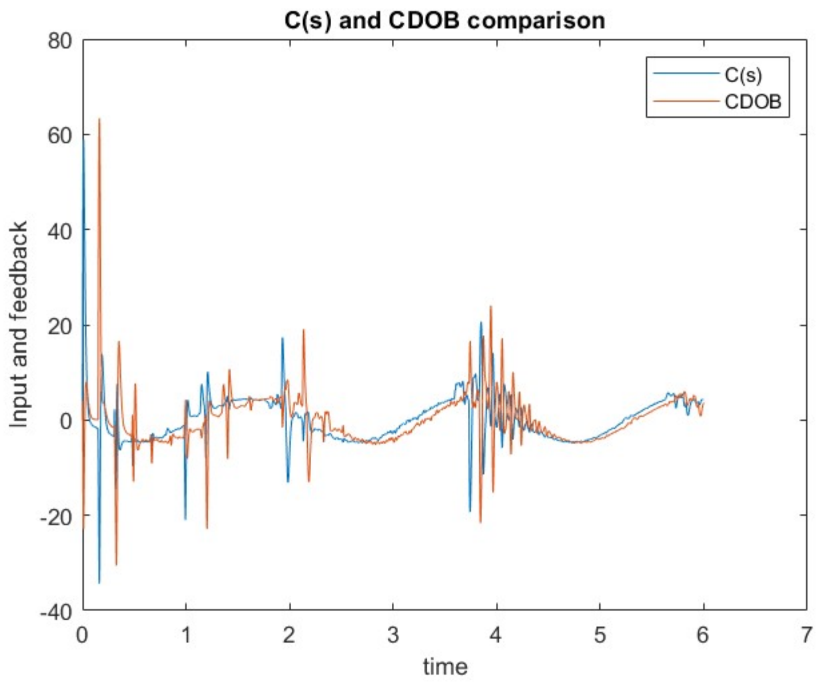

Figure 5.9: C(s) and CDOB response for correction if delay is variable

An analysis of the input and CDOB shows that despite a low pass filter, still there is a lot of noise. If we look at the correlation between error in position response and noise in figure 5.9, we can see some correlation. In the point of larger noise there is some difficult to estimate the delay for my method, so the position is affected by this error.

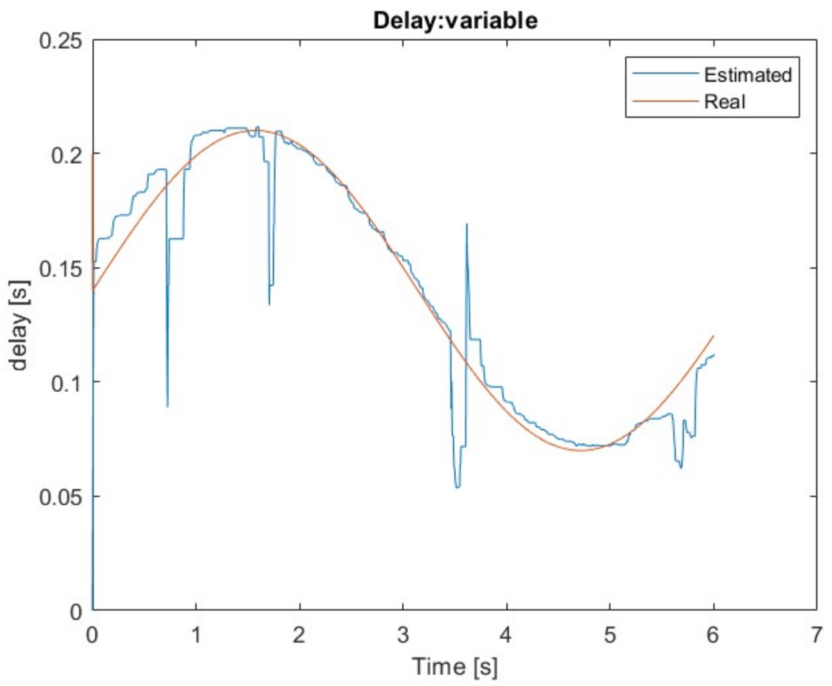

Figure 5.10: Delay response for self correction with delay variable following a sinusoidal function

Time delay estimation worked very well, better than the case of constant delay. The main reason why it improved is probably the low pass filter in C(s) and CDOB, that helped to do the correlation. For the reason explained before there is some error in time delay estimation that brought an error in position itself. The overall result is positive for me, the system is able to self-correct and to correctly estimate the delay.

5.4 Position reaching

In previous paragraph has been studied what happen if the system moves, but not what happen if the system reach a position and then remain constant. So, in this paragraph I am studying if time delay correction works in this condition. The position input will be the same of before, with the main difference that after 3s the position is constant, and the velocity is zero. Both cases are with a time delay that is variable long the time, with the same equation of before.

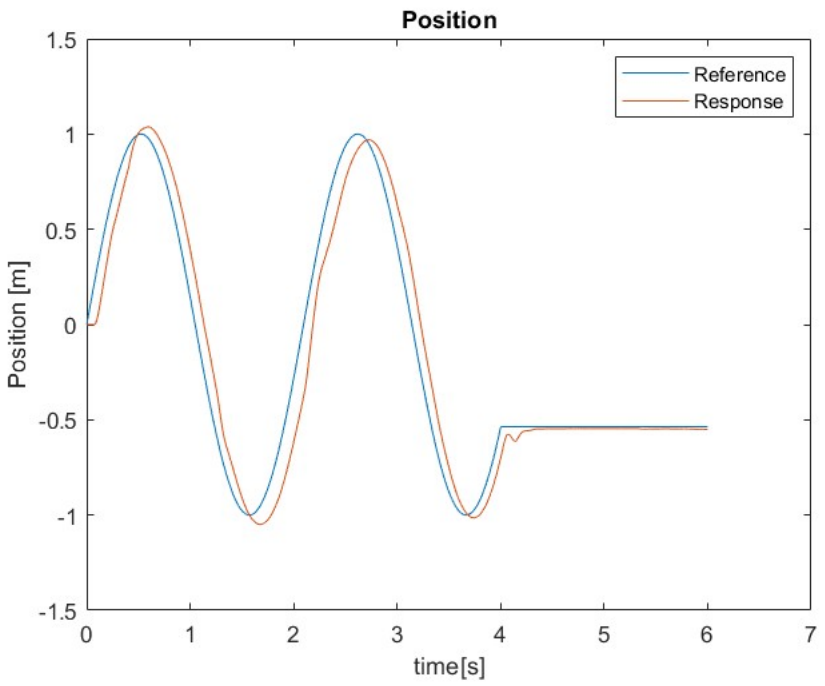

Figure 5.11: Position response for variable delay but stable position at the end

In figure 5.11 we could see the response in condition of variable delay and the position is constant after a certain time. The result is satisfying and allows also to reach a position, not only to follow a path. Signals value can be seen in figure 5.12 and delay response in figure 5.13.

The estimation of time delay still works also when the position is reached since actually there is a small error, and this bring a value of input and feedback that allow the estimator to work. Also delay estimation works well, as expected from good position response.

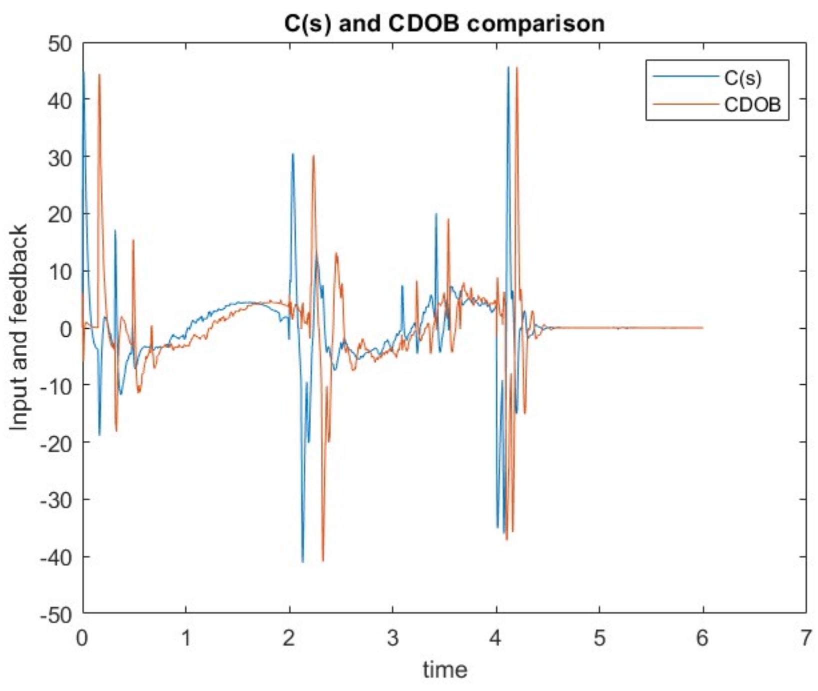

Figure 5.12: Response of C(s) and CDOB in case of position reaching

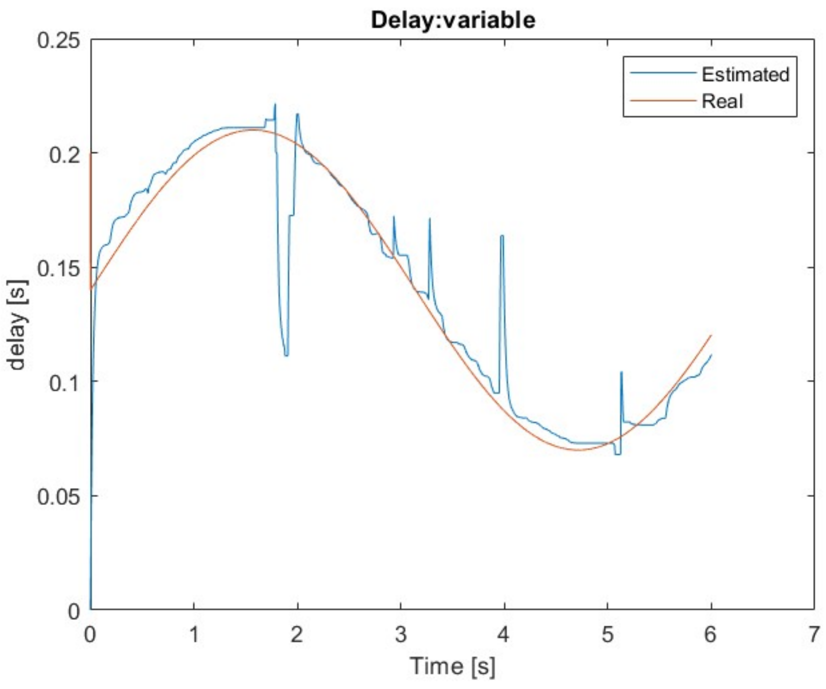

Figure 5.13: Delay response in case of position reaching

5.5 Contact with environment

Since this method of delay estimation and correction has the aim to be used also in a haptic control in a future, it should be important to understand if it can work also if there is an external force, and in this case also if it is possible to estimate the reaction force from the environment. To estimate the external force and compensate it, a classic disturbance observer as described in chapter 1.4 has been implemented. So here I will simulate an environment where the robot is free to move for the first 0.5m and after there is an object with elastic and viscosity properties. The object should decrease the velocity of the robot, but don't stop it.

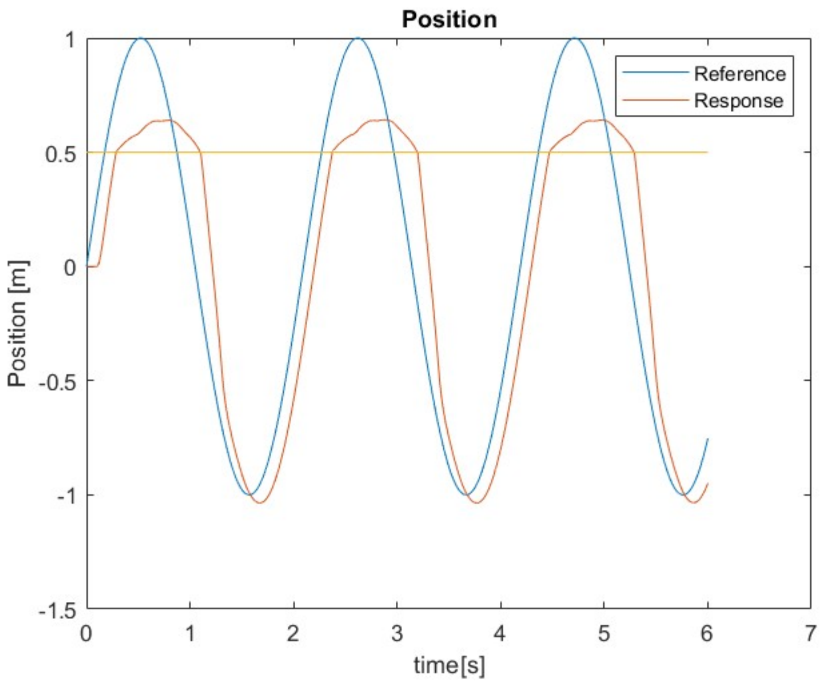

Figure 5.14: Position response for case of right delay inside the predictor

Here also the position input, that ignores the object, is a sinusoidal function as the previous cases. The position response should be ideally the same of figure 5.14, that has been simulated considering right time delay inside Smith's predictor. The yellow line should represent the position after that there is the environment.

5.5.1 Contact for constant delay

In this paragraph will be studied the case with constant delay of 0.2s, to do a first check if it could work with simple delay correction. An interesting thing could be do a comparison of what happen with or without the correction. In figure5.15 there is a simulation of the behaviour with a wrong value of delay in the predictor.

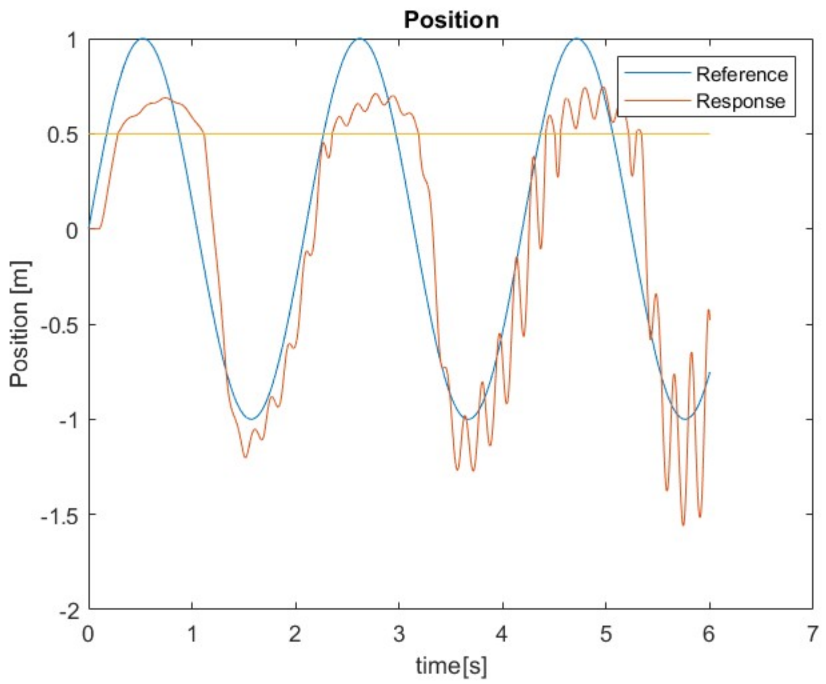

Figure 5.15: Position response for delay constant but wrong value inside predictor

We can see that without a correction the position response is not following the input function and moreover it is completely wrong compared to the case with right delay in the predictor.

There is a large difference also compared to the case of wrong delay but without contact with an object, like figure5.16: the difference is that the object limits the position of the robot with an elastic and viscosity force, and it doesn't allow to the system to be too unstable.

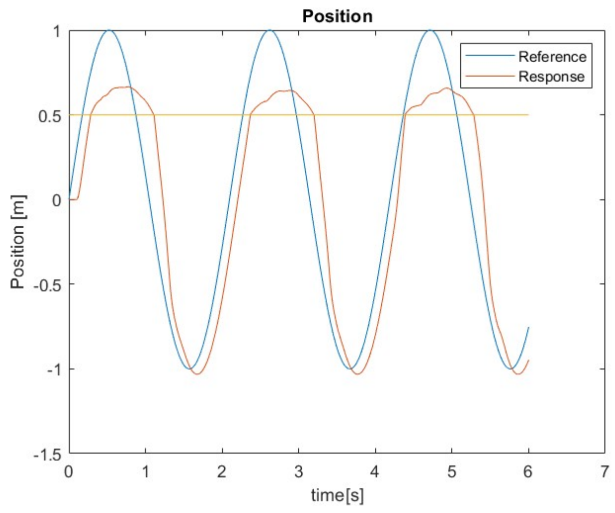

Figure 5.16: Position response for delay constant and correction

Here we can see the results with the corrector. The position response is working well since the result is similar to the case of figure5.14. We can notice that there is some point where the position is not perfect, but I think it could be a success. It would be interesting to look the value of C(s) and CDOB, since now we have two different disturbance sources: delay and environment. The worries lay in the overlay of two different disturbances that could generate some problem to delay estimation.

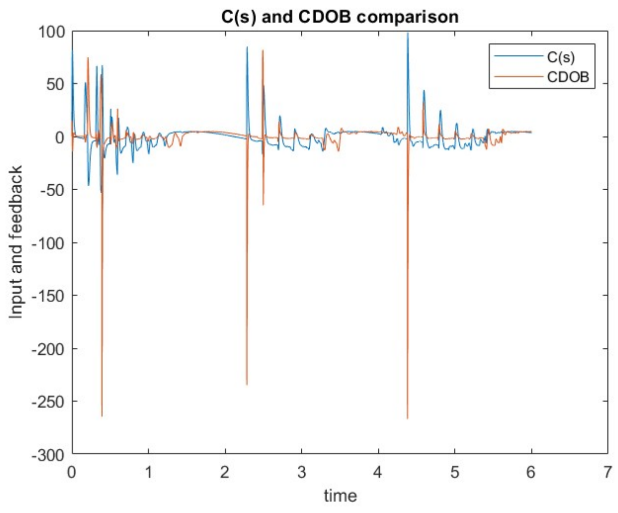

Figure 5.17: C(s) and U(s) response for contact and correction

It is hard to identify the single components that generate the two disturbances, but we can easily identify when the robot touch the object, since there is an higher value of both of them.

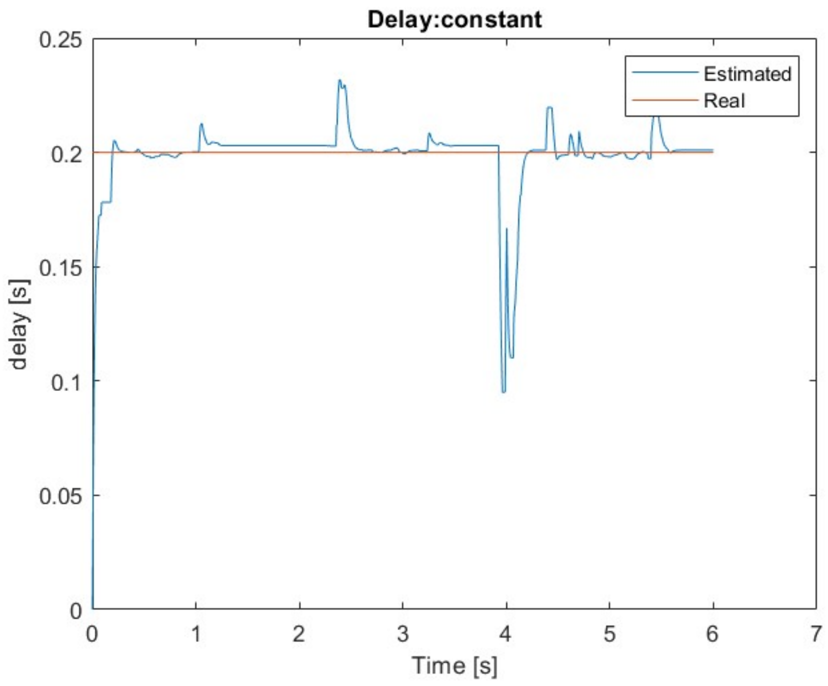

Figure 5.18: Delay response for contact with self correction

However, as can be seen in figure5.18 delay estimation is working correctly although there are two diverse sources of disturbance. Another important target of this analysis was to understand the feasibility of force estimation. As described before it would be useful in future work if the system will be applied to an haptic control.From figure5.19 it is clear that the reaction force can be successfully estimated by classic disturbance observer inside the system.

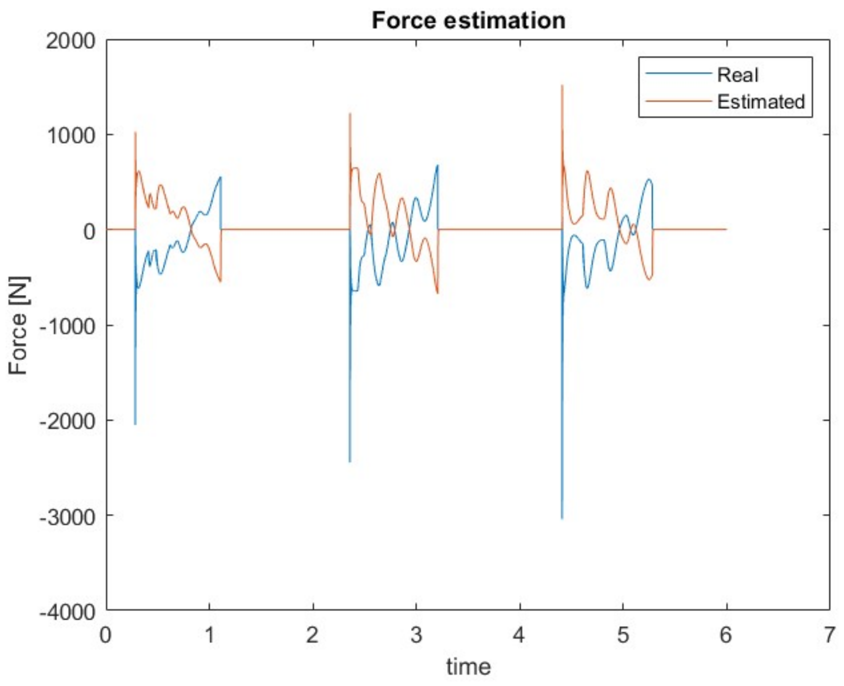

Figure 5.19: Force estimation response and real force applied to the robot

5.5.2 Contact for variable delay

A similar experiment of previous one has been conducted with the difference in time delay behaviour. Here we are considering a variable delay to understand if the system is still stable under this condition. To modify a little more the experiment now the target will be a different function, in order to understand if the target position can or cannot influence the result: the position function is a sinusoidal function that is constant after 1.5s and the object position is the same of previous case, so 0.5m from the initial position.

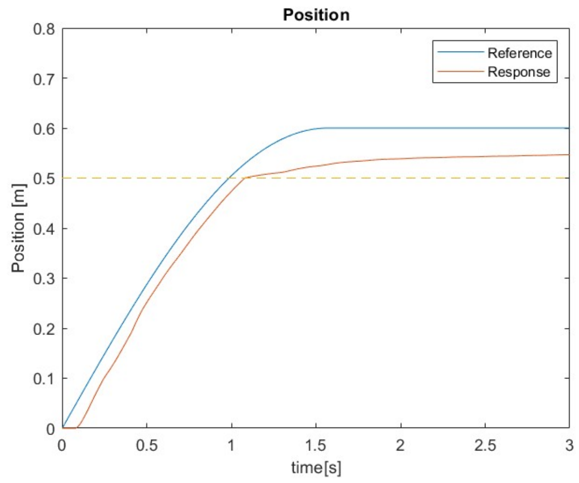

Figure 5.20: Position response for variable delay and correction

In figure 5.20 there is the position response for this scenario. The result is how expected. Since there is a object with viscosity and elastic properties, the right position cannot be reached and the position will be constant after a certain point, since the input force and reaction force are the same. The system is able to work under in this condition of constant position of reference after a certain time without any instability. Another time, this is important for a case of haptic system, where the master moves in a certain position after an object and the slave can move without instability.

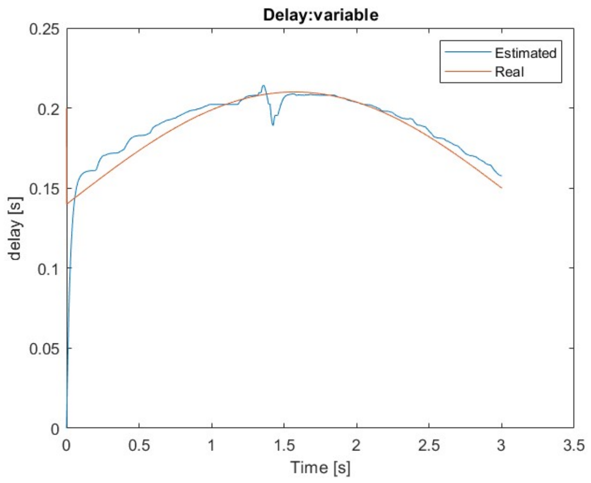

Figure 5.21: Delay response for contact with self correction and variable delay

Also here the delay has been correctly estimated from the estimator, also if the system reach a constant position after a certain time and also if there are two different type of disturbances that affect the system.

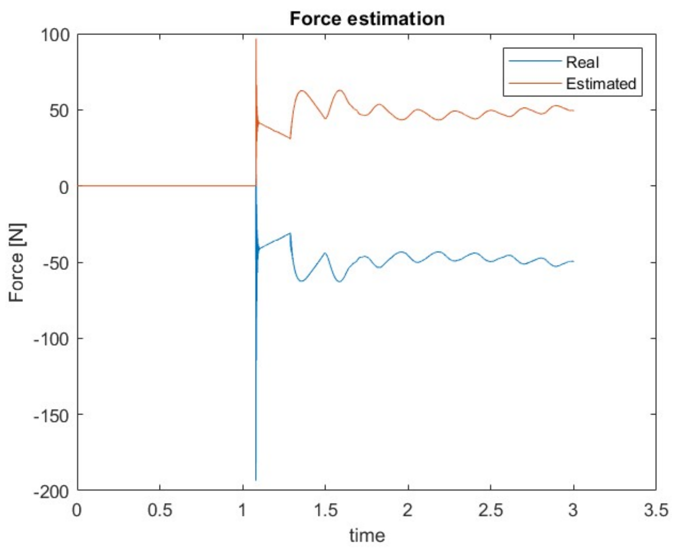

Figure 5.22: Force estimation response and real force applied to the robot

Force estimation looks working also here. When the system reaches a constant position, the force should be constant since there is a perfect balance between input and reaction force. In this case more time is needed to achieve this goal, due to the properties of material. However, the target has been achieved also in this scenario with an outstanding position response and force estimation.

5.6 Simulation conclusions

The simulations brought satisfactory results for position response, that was the main target. Since the estimation and compensation can work together the next step will be trying to do experiment based on this algorithm. Also time delay estimation is working well and this suggests there is the possibility to extend the system also to different control method, like position control and haptic control. Haptic control, moreover, could be carried out since also the reaction force looks working well also under these conditions.

Chapter 6

Experiments

Finally in this chapter a series of experiments will be conducted to verify the simulations results. The importance of doing experiments lays in the real feasibility of the algorithm and method that I have implemented. Also, if something is working in a computer simulation, this doesn't mean that it will work in the real world, since some variables could be not considered in the simulation process. If the experiments will bring the expected results, so something similar the simulations results, I could say that this system is working, at least in the simple case of position control. In the case of not satisfactory results, it will be necessary to identify the source of error and how the simulations are different from reality.

6.1 Experiments setup

The experiments have been done using the linear motor of Shimono's laboratory in Yokohama National University. This linear motor has an overall length of 0.1m and a mass of 0.5kg. Figure 6.1 is the picture of the motor that has been used for the experiments: it is a classic linear motor and the principle of how it works will be now briefly explained.

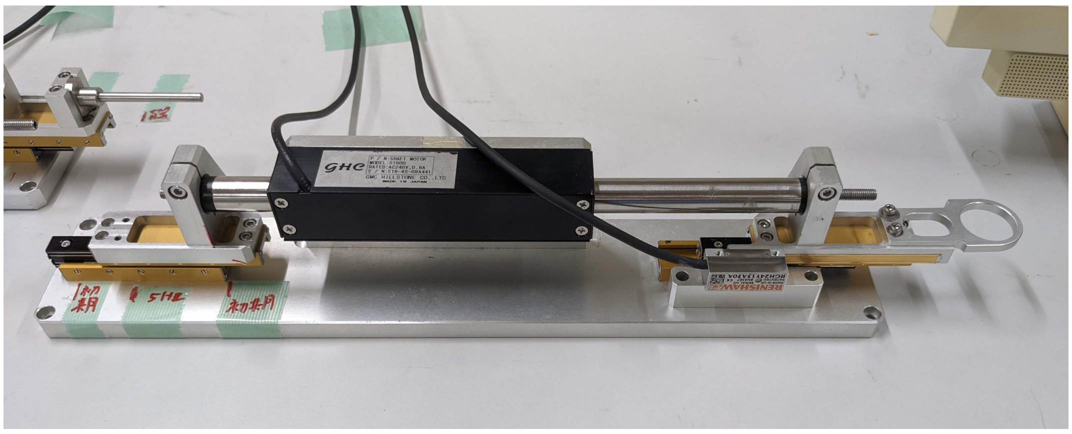

Figure 6.1: The linear motor used for these experiments

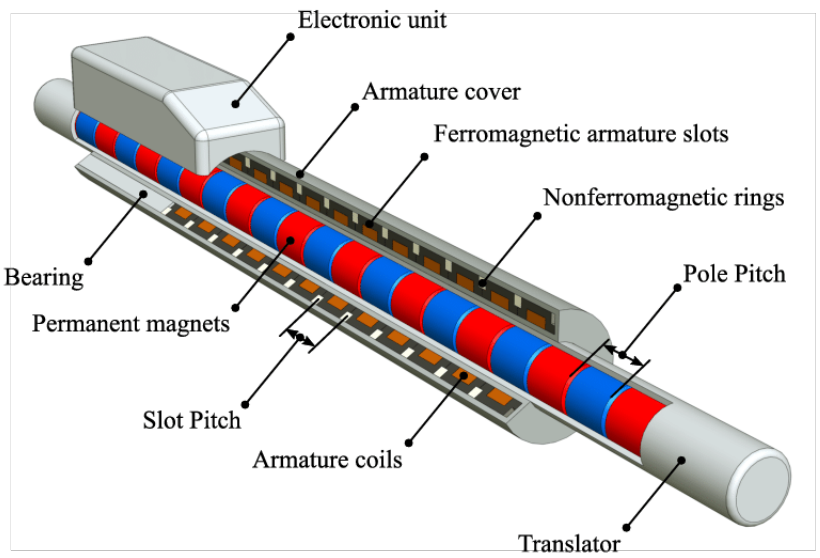

Figure 6.2: A scheme of the component of the linear motor[11]

The figure6.2 is a schematic representation of the linear motor I used, but where we can better see the different parts. The metallic cylindrical components in my linear motor is what has been called translator in the schematic figure. Inside this part, that can also be called mover, there are a series of permanent magnets, with alternate different polarity. The black box that can be seen linear motor is the stator, that in the schematic representation is made by coils and armatures. The current that passes though the coils generate a force on the magnets due to Lorentz law. The principle is the same of a classic rotative motor, with the only difference that the shape of magnets and coils is cylindrical. The force that is here generated is directly proportional to the current that flows inside the coils. There is a linear relationship and the parameter K_a, noted as force constant or torque constant in rotative motor, depends only on magnets properties and geometry. In the picture we can also see the presence of a linear encoder that measure the position of the mover. To control the motor is enough to regulate the current and it is done using a servo drive, figure6.3. Servo drive has the

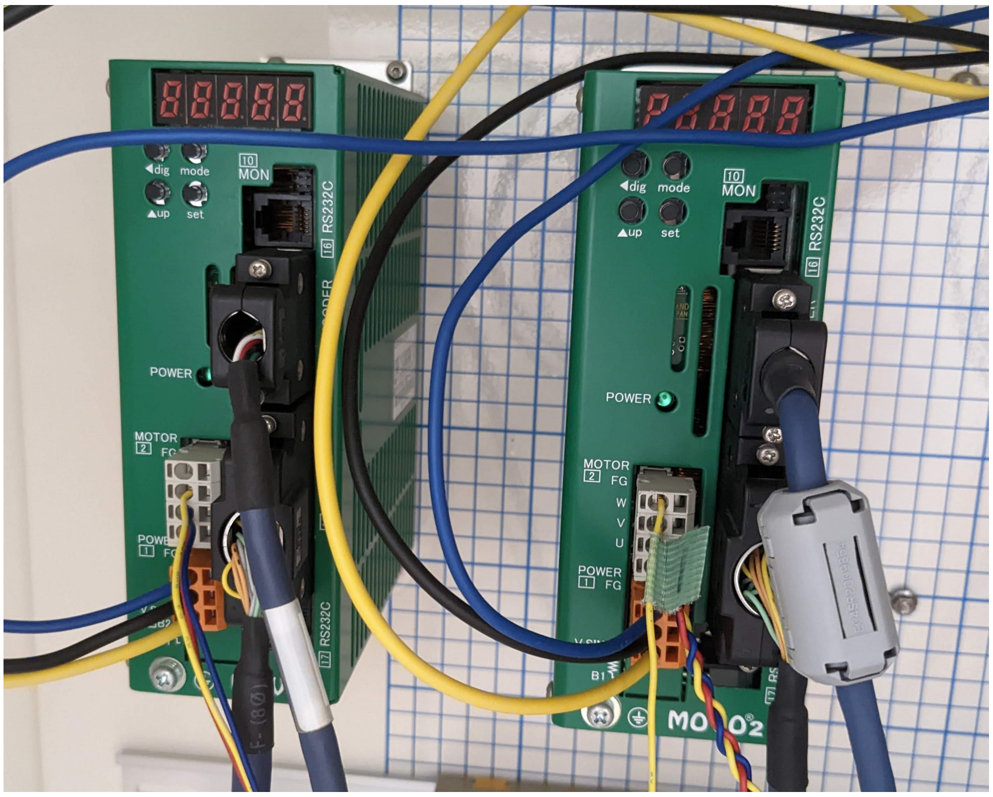

Figure 6.3: Servo driver used to control the motor

function of amplifying the signal from controller to the motor and to receive the feedback signal from the encoder and sent it to the controller. This servo drive is not connected directly to the power network, but there is also a power supplier that provide a direct current at 5V. The controller itself is a computer which runs the Linux distribution CentOS. The programming language that has been used to control the robot is C99.

6.1.1 C Code

As said in previous chapter the program that has been used to do these experiments was written in C, and so not a robot specific programming language. This program has already been written by other students in the previous years, while my goal was to modify it to adapt to my specific target. In particular, none has never studied how to simulate time delay in this experiment setup, so I studied a way by myself. The already been written code, allowed me to read the position from the encoder and to create my own code for position control, or other control, and to choose a force value, the other conversion, for example from encoder value to position or from force to voltage, was in the code part that I didn't have to change. Since the I knew the sample rate that controller uses to receive and transmit the signal to the servo drive, and I also can modify it, I decided to set it to 1ms, while for normal experiments it was set at 0.1ms. The choice to change the sample rate are due to computational time required to perform the analysis: since at every step computer needs to perform time delay estimation, that is a function of O(n3) as described in chapter (CH), if the sample time was set too small, the time that this estimation needs would be larger than sample time itself. This is the reason for the high value of sample time, but also the simulations have been done under this rate and they brought satisfactory results. The code didn't have any way to handle the delay communication time, and my way to solve such problem was to create a FIFO array for the send and received value. For each step the first element of array is sent to the servo drive, then all value shift of one position and the new current value calculated by the computer based on Smith predictor and feedback value, is inserted in last position. This method works well in condition of static time delay, while it shows its problem when it changes. For the variable delay a method based on the same principle but that evaluated the difference between the real time that has passed, and the delay has been implemented. The system, after some attempt and correction, brought a real simulation of the delay. For what concern the time delay estimator, differently by what has been done in the simulation, where it was possible in MATLAB to use built-in function to do statistical analysis, now it was necessary writing all algorithms. This allowed me to write more efficient algorithms than the MATLAB one, and that resulted in less required computational time. After the experiments the program allows the user to save the needed data in a txt file. The data elaborations, like plotting, will be done in MATLAB to have a simpler tool.

6.2 Experiment to test the delay

The first experiment is that of constant delay without compensation but with the right value on the Smith's predictor. This first case has been studied to verify if the delay simulation is actually working and to verify also if delay estimation works or not. So the block scheme that is used here to control the motor is the same of figure5.1, without a self-correction. Here there are some differences by the simulation. The position is provided by an encoder, so also velocity and acceleration used in CDOB, that are derived from position, are a sort of encoder's value. Since the encoder is a linear one with its own resolution, could happen that velocity and acceleration assume strange value. For this reason, a low pass filter has been implemented here, to avoid point when velocity or acceleration are zero. Other problem that generated some errors, are related to the time the system needs to perform time delay estimation. If it, for some steps, requires more time than the sample rate, it could bring instability, since servo drive doesn't receive any new value of voltage from computer. The target position follows a sinusoidal function as in the previous simulations. But since here the dimension bounds are relevant, the new function is as in equation $x(t) = 0.01 \cdot sin(0.04t)$. Several attempts have been done to create an algorithm that can satisfactorily simulate the delay in a real system, since there where no other way to do so.

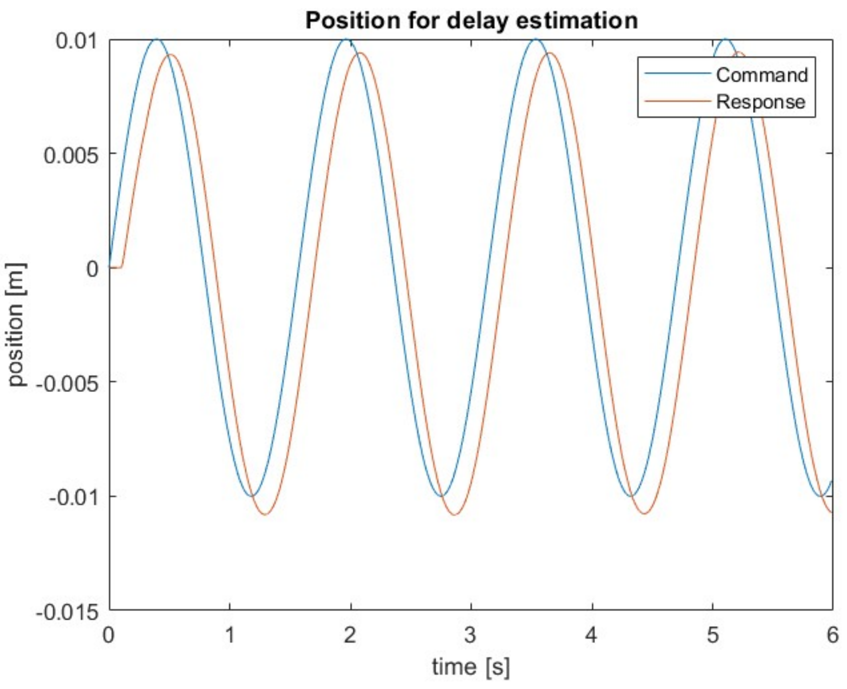

Figure 6.4: Position response for simple delay simulation experiment

In figure6.4 there is the results of position response from real experiment when inside Smith's predictor there is the right value of time delay, in this case 0.2s of roundtrip delay. As can be seen in the graph there is a small error in position response, due to a bad compensation of an external force and to a bad behaviour of disturbance observer that has the target to delete this external force. This force is due to a problem with the motor itself and it happen also when the motor is in idle, so when the program has not already been lunched. To compensate such force a classic disturbance observer, as described in paragraph1.4 has been implemented, but the correction is far to be optimal. By the way, the delay has been satisfactory simulated, so the target has been achieved. Now it is time to check the value of C(s) and CDOB to understand if their behaviour is right.

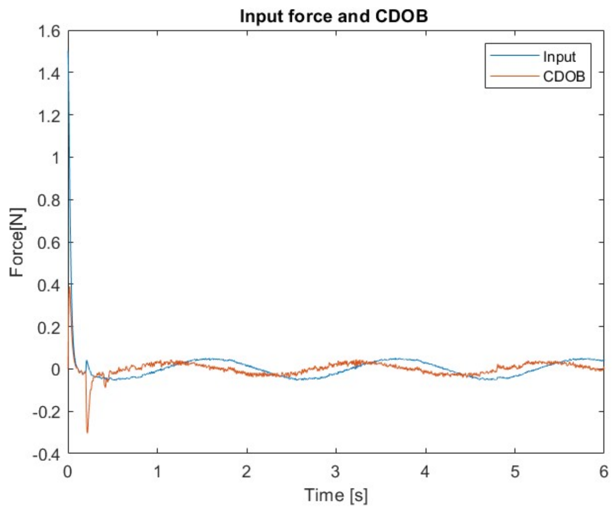

Figure 6.5: C(s) and CDOB response for delay test

Look at the figure6.5, we can see that the average values of input and CDOB are similar to the simulation value of case with right value inside Smith's predictor, with the only difference the noise. In the first experiments the noise was much larger than in this figure and the estimation wasn't working. To solve this problem, a low pass filter has been implemented and it looks work smoothly, despite it is not perfect.

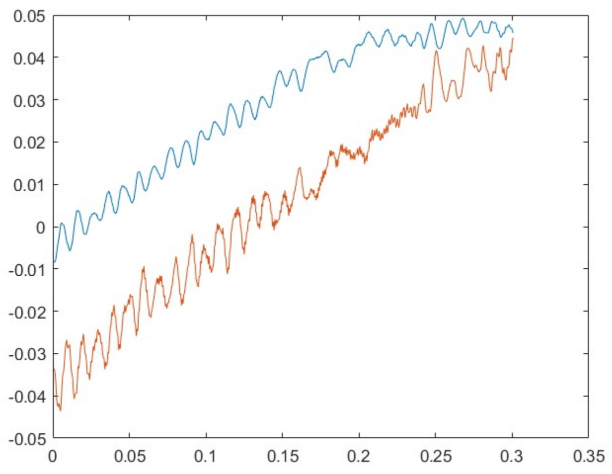

Figure 6.6: Zoom on the last 0.3s of input and CDOB values

Considering the only last 0.3 seconds, so the time window that has been considered for this test, we can see that the noise is high, but that the average could still be used to estimate the delay, as already seen in previous chapter.

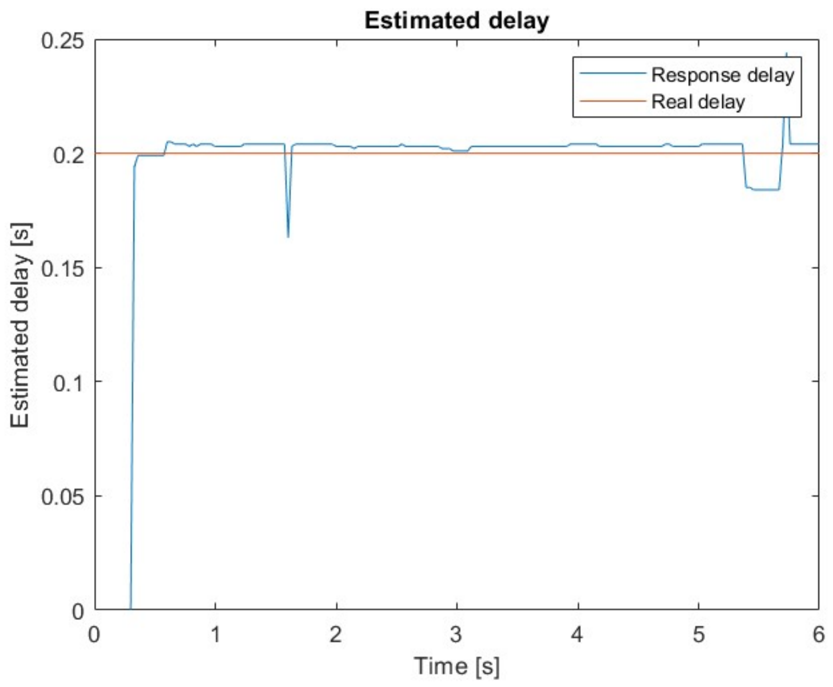

Figure 6.7: Delay test for this simple experiment

Here there is the delay estimation made by program when it was also controlling the robot position. As can be seen, the delay estimation worked satisfactorily, with just two points when the delay estimation is not accurate, and this could be related to noise mainly. The graph here has a different aspect from the simulation's one. Here the time delay has been represented when it is estimated and not when it has been generated, as in simulation. This is due to how I decided to save the data to use it in the program rather than an actual technical reason. The result is anyway enough good to attempt to do the self-correction, the target of this research.

6.3 Self correction for constant delay

The first case where I am going to test the self-correction in a real world is with a constant delay, as the scenario studied before. Also here the delay is set constant at 0.2s and the position target is the same as before.

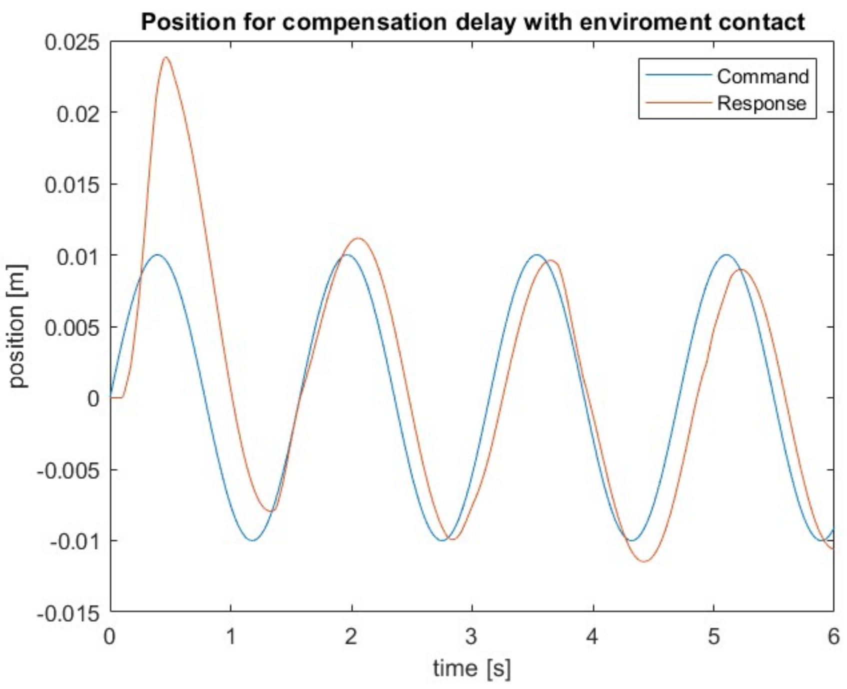

Figure 6.8: Position result for compensation experiment with constant delay

In figure6.8 we can see the result of position control in this experiment. As can be seen, there is a large error in the first second, while later the position almost corresponds to the target value. This result could be seen in two different ways. The upside is that after the first second the position command and response follows the same path and the average error is low, also if compared to the previous scenario where there was a small position error, despite it was the best scenario. This result is suggesting that meanly the algorithm and theory behind it are working satisfactorily. At the same time there is a large downside if we consider the first second, where the error between target and response are too large to be consider acceptable. The problems could be in some coding error in conversion between MATLAB's simulations and C code, while some my mistakes could have caused an error in programming. Another problem could be that in simulations something has not been considered. It looks a problem related to when the correction really starts.

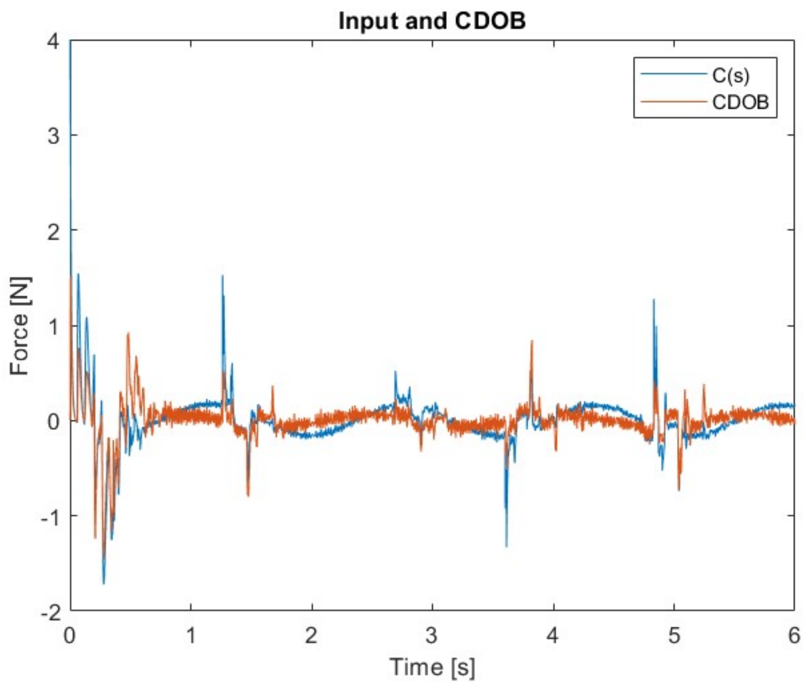

Figure 6.9: C(s) and CDOB results for constant delay experiment

Considering the value of input force and CDOB, we can see that there is much more noise in this case, especially for what concern communication disturbance observer's value. Such noise is due to the higher disturbances, indeed, that are related to the position that is not following accurately the target value. We can particularly see that in the first few steps, the absolute value of both function is very high, and it is a symptom that something is not working. However, after the first second, there is a more stability and both curves look follow a regular wave, as we expect to have a correct estimation of the delay. From both graphs the suggestions is that the delay estimation is probably working well, and it is the next step to analyse.

From the graph of the delay, we can see that estimation is working quite well, as expected from the good results of previous graph. As the previous delay analysis, also here the estimated delay starts at 0.3s since this is the dimension

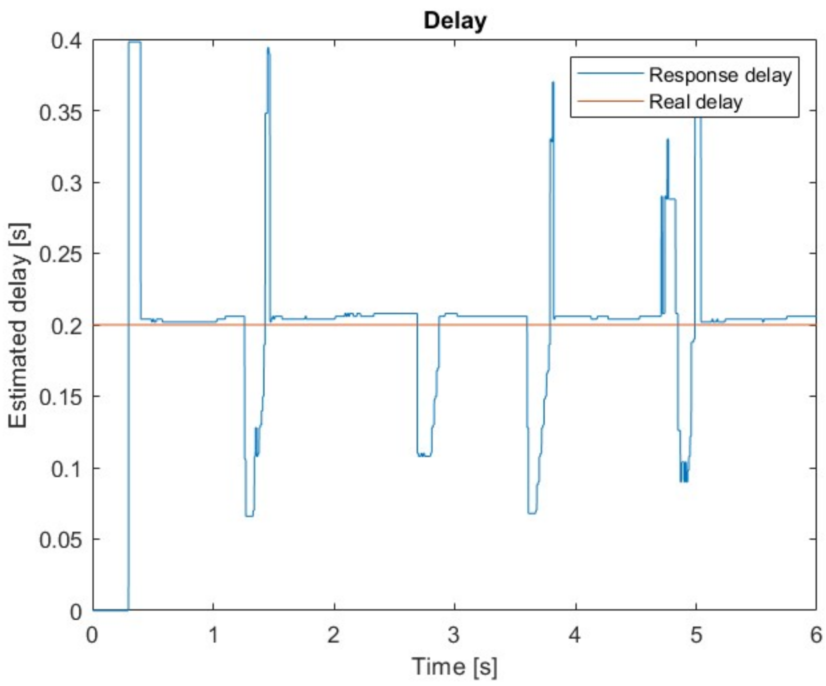

Figure 6.10: Delay response for experiment with constant delay at 0.2s

of chosen time window. Moreover, in the next few tenths of second, the estimation fails, and this could be another source of the wrong behaviour in position. However, after this first error, the value tends to be almost right, with few mistakes. The presence of this mistakes looks not generate too many problems to the general result, so the system looks have stability.

6.4 Experiment for variable delay

Since the first experiments brought an almost satisfactory result, the next step following my previous simulations, is now to test a condition with variable delay. As in the case studied in simulations, also here the delay will increase and then decrease regularly, following a sinusoidal curve. The goal of this experiment is to verify if the robot can follow the target position also in this condition, that I think could happen also in a real world, also if with different function. It is especially interesting since the main problem of the previous scenario happened when the value inside the predictor was wrong compared to the real one. But since the delay now continually changes, the value inside the predictor is always wrong, since the time when it does the estimation must be a half roundtrip late and some time needs to do the estimation due to a smaller time window but higher the communication delay, compared to the real value of the delay. Although in simulations this didn't generate any problem, maybe here the things will change, as happen in previous experiment. Firstly, I will evaluate the position response as always.

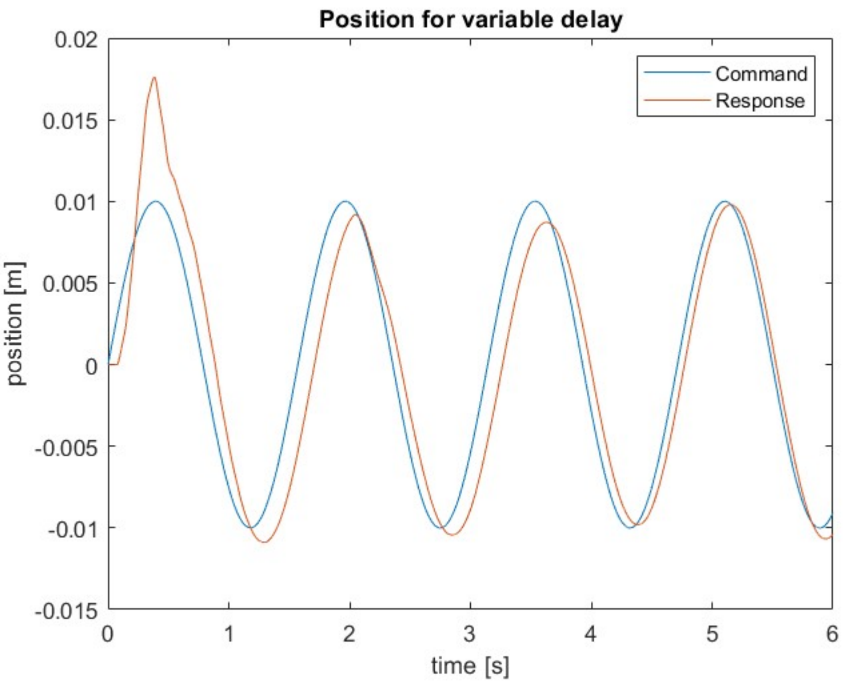

Figure 6.11: Position result for compensation experiment with variable delay

The position response brought satisfactorily results, despite the worries related to the continuous late between estimated and real delay. Also here the problem lay in first second but the relative error position is less than the previous case. The reason could be that the difference between delay inside the predictor and real delay when the experiment starts is equal to zero, and only after it start to changes. By the way, the position response has a good behaviour after this first second, showing that the system is generally stable and able to follow changes in delay.

Although the presence of a low-pass filter, looking at the value of CDOB,

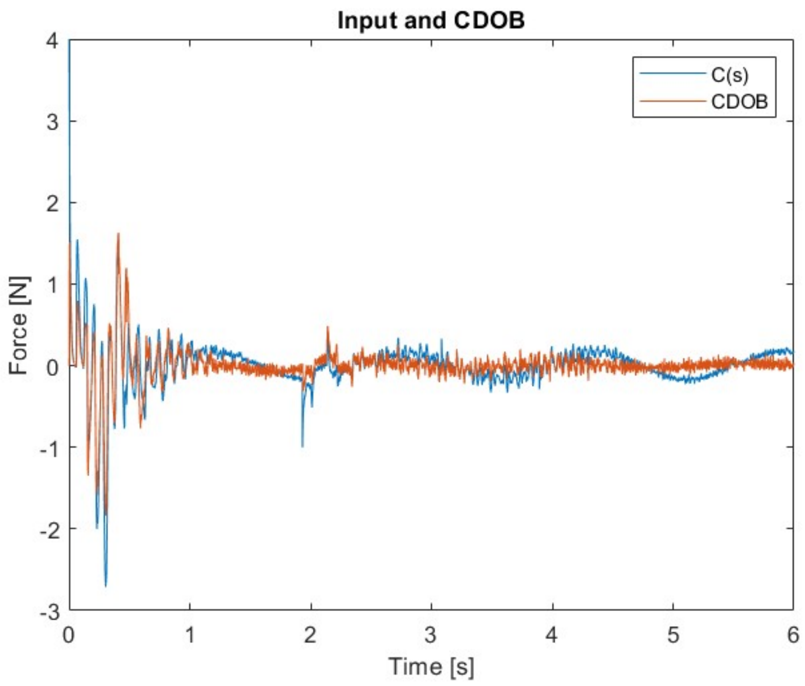

Figure 6.12: Input and CDOB response for variable delay experiment

we can see that the noise is much higher than previous cases. The reason is the increasing of disturbance due to the variable delay, that doesn't allow the stability of previous case, neither after a certain amount of time. As expected, the force value for both the signals are higher in the first second, when the disturbance due to error in position is high. It looks like that the estimation of delay could still be possible, since the predictor shows a good behaviour also in presence of noise in the two previous experiments.

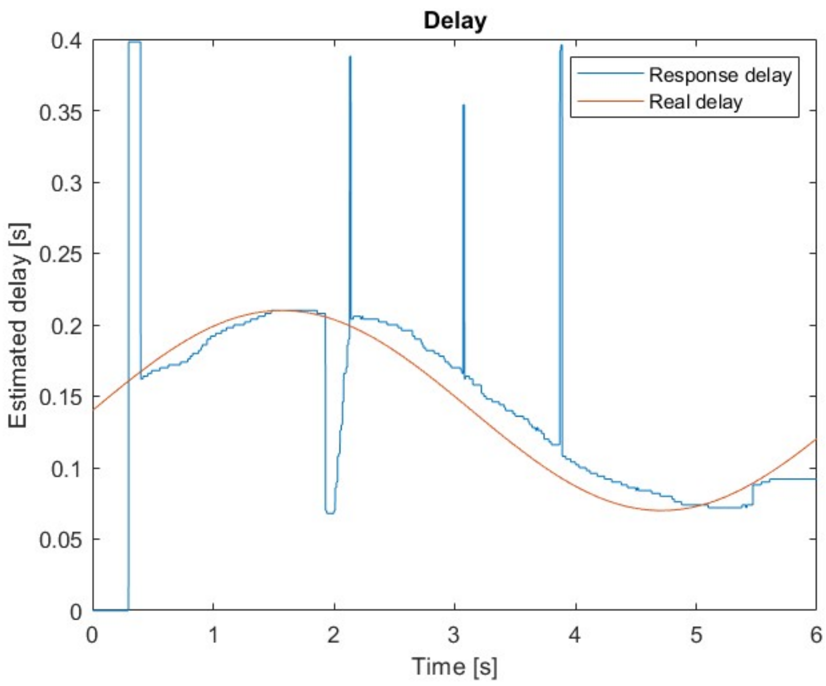

Figure 6.13: Delay response for experiment with variable delay

Looking at the delay graph, it is possible to see that the behaviour in the first second is like the previous experiment, and that the error could be the same as before. As in the previous scenario indeed, after the first 0.3s of the time window, the value of estimated delay is much higher than the real one for almost 0.1s, and after this it looks following the right estimation. So, the problem of error in first second is due to problem in delay estimation and the code should be corrected following this analysis. As in figure5.13), there is a large error in delay estimation also after the top value of the delay. Probably this is a critical point for the estimator algorithm, and it should be studied better. Averagely, the compensation in position worked well, so the next step is to test it with a contact with an object, like it has been done in the simulations.

6.5 Contact experiment

In the last experiment I am testing the behaviour of the control system if there is an object, that generates a force due to viscosity and elastic properties, that could influence the position response in a worst way. This time the setup is the

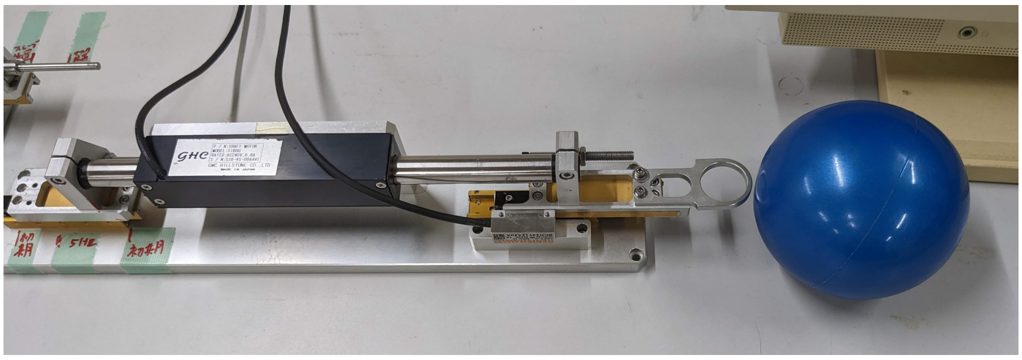

Figure 6.14: Setup for experiment of contact with environment

same, while the object that is used is a ball filled with sand. This could be seen as a response of a general object in a real world. The target position was the same of previous scenarios, and the response, considering an object, should have been the same of figure5.2. Since I didn't know the mechanical properties of this ball, and that the estimation would have been difficult, no simulation of target position response has been conducted. Nevertheless, I know the response that it should follow, and if the response it like what the simulation suggests, the experiment could be consider positive. I have already explained the presence of a force due to the permanent magnet, but it is almost constant, and it doesn't reflect a real environment. This new external force can cause different problems in delay estimation since they already happened in previous case. In previous cases the starting error generated a position that overcome the position where it is now the object. So the scope is to understand if the system is stable or unstable. The delay is set constant at 0.2s, but it is wrong in Smith's predictor initially and it must self-correct. From position response it is clear that the system is working, despite also here with some errors: in the first second, as in the previous experiments, the robot fails to follow the target, and it move further the target. The robot should find the object and stop or be decelerated by it, but a problem

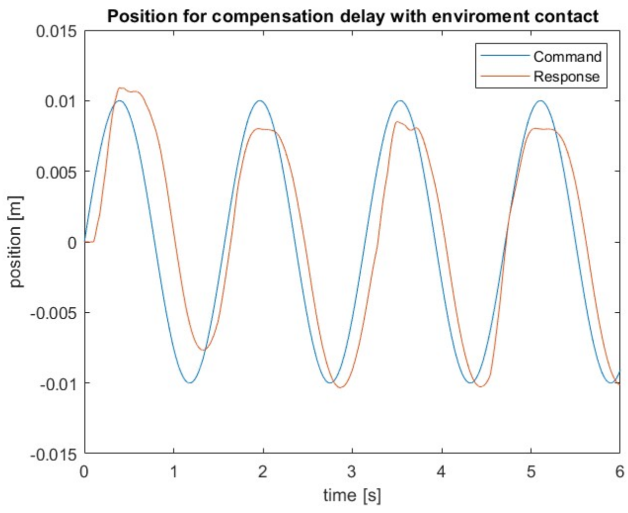

Figure 6.15: Position result for contact experiment

occurred: the force generated by the robot for follow its path is too strong and it moves the object, before obtaining a stability, due to the smaller error. After this first second, the other interactions with the object follow the expected trajectory, and the stability of the system is overall satisfactory.

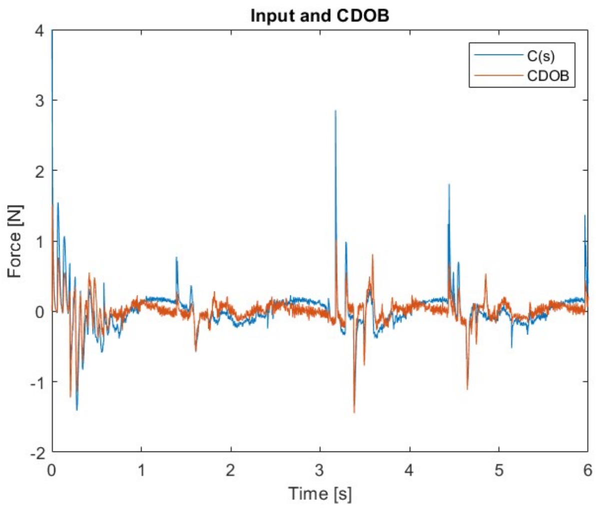

Figure 6.16: Results of input and CDOB values for contact experiment

Looking at the graph of signals, we can see that the first second has the same aspect of the other experiments, and it suggests that the error is generated by the same source. The point when the signals don't follow the standard path, are the point where there are contacts with the object. In this case it is still possible to identify the delay for the system, despite the fact that there is no regularity.

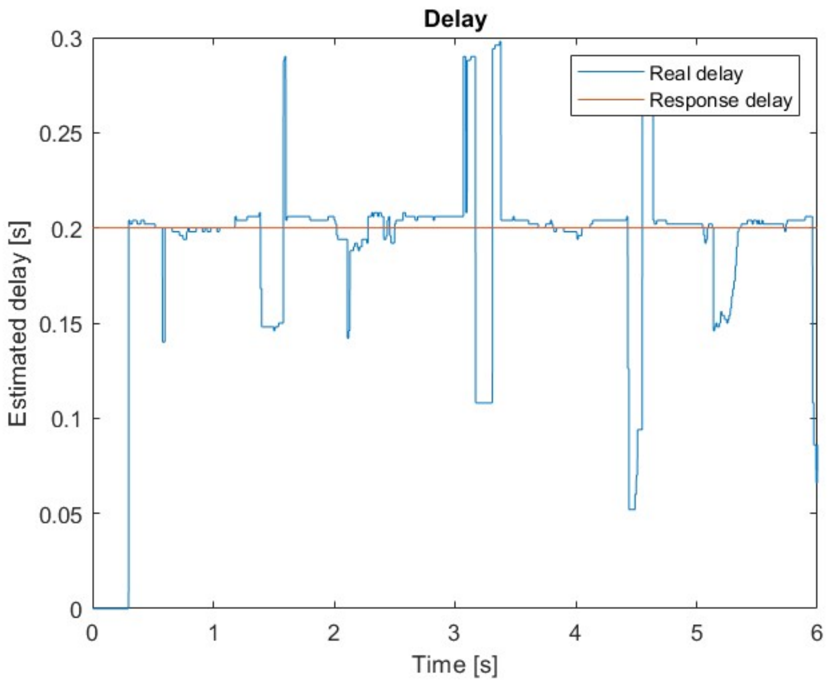

Figure 6.17: Delay response for contact experiment

Looking at the delay estimation, also here the response in the first 0.3s is zero due to the dimension of time windows. After this point the estimation looks working satisfactorily, with a small exception for some point during the contact. Overly the delay estimation is not compromised by the presence of the object, and this allows to control the system in a satisfying way.

6.6 Experiment conclusion

The experiments brought generally satisfactory results since it has been possible to estimate the delay and to control the robot without large problems. During the experiments several conditions have been tested and in all these the position response was positive after a first second of assessment, when the absolute and relative position error was too large than simulations. I believe there is still some room for improvement this starting error, but the lack of time didn't allow me to test further improvement.

Conclusions

The research brought good results but there is still some room for improvement. After the first attempt to find a way to estimate the delay, that didn't generate any results and was a time-consuming activity, another and different approach has been found, mainly based on a statistical way to estimate the delay. The first simulations, that has been done to check if the estimation of delay was or was not possible using a method based on statistic, and that return results according to the theory it was time of simulations. The simulation, conducted under conditions through just on a computer, suggested that the method was working also for self-correction, not just for the estimation of the delay. The simulations suggested a high stability of the system and that it have the flexibility to work also if the delay estimation is not perfect. Indeed, we saw in all simulations some point when the delay estimation failed, but a small error didn't affect the response in a dramatically way, as it could be possible. Good results from simulation allowed the next step, so trying to implement the system to control a real robot in a real environment simulating the delay in an appropriate way. The results of the experiments suggest that there is a small problem in the control algorithm for the first second, so when the estimation is not possible due to the dimension of time window used. Despite also in the simulation this problem should have happened, it appeared only when I did the experiments. The main reason could be an error in the control code, but due to the short time I was not able to put it right. The experiments overall confirm the stability of the control algorithm also when the delay is not correctly estimated, or when the contact with an object bring another disturbance to the system. Although there are some errors, I believe the general result is positive and the idea behind the system is working satisfactorily. Despite almost good results some improvement can be done, considering the control algorithm in the first part, where the error is too large, but also to achieve a better result in delay estimation that it is not so accurate in every step. Future works that can start for this research are mainly focused on trying to apply the same algorithm in different control method. First of all, the method could be applied to a force control system, where the control could be done using Smith's predictor and the CDOB estimates the delay. This could be an interesting improvement to force control under variable time delay. Another field of research could be the haptic system. In conclusion the research generated some good results, but further improvement could be done, and this algorithm should be test also for different types of control, different from the simple PD control.

Bibliography

[1] S. Chiaverini and L. Sciavicco, "The parallel approach to force/position control of robotic manipulators," *IEEE Transactions on Robotics and Automation*, vol. 9, no. 4, pp. 361–373, 1993.

[2] R. O. E. Sariyildiz and K. Ohnishi, "Disturbance observer-based robust control and its applications: 35th anniversary overview," *IEEE Transactions on Industrial Electronics*, vol. 67 no. 3, pp. 2042–2053, 2020.

[3] E. Sariyildiz and K. Ohnishi, "A Guide to Design Disturbance Observer," *Journal of Dynamic Systems, Measurement, and Control*, vol. 136, no. 2, 12 2013, 021011. [Online]. Available: https://doi.org/10.1115/1.4025801

[4] Y. Yokokohji and T. Yoshikawa, "Bilateral control of master-slave manipulators for ideal kinesthetic coupling-formulation and experiment," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 5, pp. 605–620, 1994.

[5] O. Smith, "A controller to overcome dead time," *ISA Journal*, vol. 6 No.2, pp. 28–33, 1959.

[6] K. Natori, R. Oboe, and K. Ohnishi, "Robustness on model error of time delayed control systems with communication disturbance observer-verification on an example constructed by double integration controlled object and pd controller," *IEEJ Transactions on Industry Applications*, vol. 128, no. 6, 2008.

[7] A. A. Rahman and K. Ohnishi, "Robust time delayed control system based on communication disturbance observer with inner loop input," in *IECON*

2010 - 36th Annual Conference on IEEE Industrial Electronics Society, 2010, pp. 1621–1626.

[8] K. Natori, T. Tsuji, K. Ohnishi, A. Hace, and K. Jezernik, "Time-delay compensation by communication disturbance observer for bilateral teleoperation under time-varying delay," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 3, pp. 1050–1062, 2009.

[9] R. O. K. Natori and K. Ohnishi, "Analysis and design of time delayed control systems with communication disturbance observer," in *2007 IEEE International Symposium on Industrial Electronics*, 2007, pp. 3132–3137.

[10] C. O. Saglam, E. A. Baran, A. O. Nergiz, and A. Sabanovic, "Model following control with discrete time smc for time-delayed bilateral control systems," in *2011 IEEE International Conference on Mechatronics*, 2011, pp. 997–1002.

[11] A. H. Zamanian and E. Richer, "Identification and compensation of cogging and friction forces in tubular permanent magnet linear motors," 10 2017.