Characterization and evaluation of a bilateral command architecture for a tele-operated system.

Master thesis in automation engineering

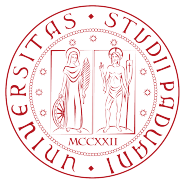
Student: Francesca Bristot

Supervisors : Prof. Roberto Oboe
Prof. Pierre Vieyres

Internship February-June 2011

Università degli studi di Padova

Université d'Orléans, Laboratoire Prisme



Contents

Introduction

The objective of the stage was the evaluation of a bilateral teleoperation benchmark for a tele-echography system and the final goal was to test the effectiveness of the wave variables formulation on this architecture.

With the term tele-robotics we generally describe a situation in which a human operator controls remotely the motion of a robot that is somewhat separated from the user. This separation may be a large distance, with the operator in one location and the robot in a different one, or for example due to scaling like in micro-surgery; teleoperation is involved in a number of different situations such as space robotics or dealing with hazardous environments, as well as medical applications. Telerobotics dates back to the 40s-50s when Raymond C. Goertz started researching on the topic, in particular creating a device to handle radioactive material behind shielded walls. Since then there has been an expansion in different fields and extensive research to obtain better and better results.

The term *bilateral* refers in particular to the fact that there is some sort of force feedback from the robot in contact with the environment (slave) to the device used by the operator to control it (master), in this situation the slave robot functions also as a sensor and the master robot as a display device. The human operator will control the robot in contact with the environment using different instruments: sensors can be used to provide a visual, acoustic, tactile and haptic display. Depending on the application the amount of information that can be obtained from the robot to provide an effective control may vary and be more or less complete, ultimately the goal is to achieve *telepresence*: the human user can manipulate the remote environment and perceive it as if he encountered it directly. The control performed by the operator can be direct, for example using a joystick to control the motion of the slave, or in some cases it may be more of a supervisory nature, with high-level commands sent by the operator autonomously refined by the robot.

In a bilateral architecture with force feedback we need to consider for the control not only the user's action and contact with the environment, but also the feedback loops that form and the two robots internal closed loop. Also when communicating over a channel a time delay may be inserted in the communication, this specific problem (*time delay*) has been a topic of interest since the 1960s and

since the internet has become a possible communication medium also the time-varying delay problem is a topic of research.

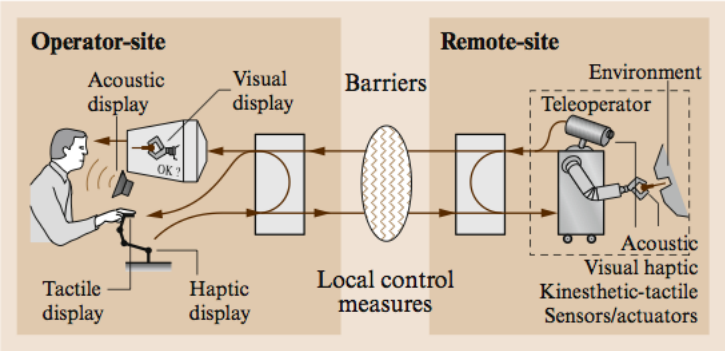


Figure 1: Teleoperation system (from [7])

With tele-echography architecture we describe a system used to perform an echography on a patient that is in a different location from the doctor performing it. The system is composed of two sides: we'll indicate with master the side where the doctor is, controlling the motion and receiving the results, and with slave the side of the patient. This architecture is conceived to enable an expert operator to remotely perform echographic examinations as if he were on site. The idea is to be able to perform an echography without local expert, of to let hospitals easily share their specialists for diagnoses and also improve the emergency capabilities.

Ideally the doctor would be able to perform the echography as if he were in the same room as the patient: with the aid of a visual feedback he can control the position of the probe through a joystick device and adjust the motion of the robot that moves the actual probe on the patient location. In order to obtain a more realistic experience and more complete information from the examination it would be useful to be able to apply more or less pressure on the probe and receive a feeling of the response of the body to such pressure, in other words we wish to realize a force feedback from the slave (patient) to the master (doctor).

To realize these operations we'll need to transmit some information on a communication channel, such as the motion of the probe and the force applied. One of the main problems that concerns all kind of tele-operations is that the channel can introduce a time delay, that may vary over time, and cause some instability in the system composed of master-channel-slave. Over the years researchers proposed a lot of different techniques to control the motion of the robots and to prevent instability when dealing with time delays. After studying the literature on these different algorithms we chose one among them that appeared to be the most suitable for our needs: the wave variables formulation. This technique seemed to

offer the most advantages and was reasonably easy to implement in its form for constant time delays, and it's also adaptable for variable time delays even though there was no time to implement or test this modification.

After studying the theory behind the problem I analyzed the hardware that was going to be used and in particular I focused on the end effector of the slave side and on how to realize the actual feedback of how the force was felt when some pressure was applied to the probe. The slave robot in this specific case will be just an actuator moving in one degree of freedom vertically and the force feedback will be realized with the aid of a sensor, the practical part of this thesis consists of the implementation of the control algorithms for the motor and of the program to communicate with the master, this includes the codification of the transmitted signals according to the wave variables formulation to address the issue of instability due to time delay. We were finally able to test the effectiveness of the control and of this particular technique when using a satellite communication with non-negligeable time delay.

The outline of the thesis is the following:

- In the first chapter we'll present the theory necessary to understand a tele-operation system and work with it, its mathematical representation and the concepts of passivity and transparency.
- In the second chapter we'll discuss some of the main control techniques developed to control a telerobotic system when dealing with time delay and explain why we chose one of them over the others.
- In the third chapter we present the wave variables formulation, the technique we chose to control the transmission of signals in the tele-echography system.
- The forth chapter contains the description of the architecture we used: the hardware and software we worked with.
- Chapter five presents the results and considerations from the simulations of the system.
- In chapter six we describe the practical implementation of the control and communication.
- The seventh chapter presents the experimental results and some consideration about them.
- In the last chapter we'll draw our conclusions and discuss possible future developments.

Chapter 1

Theory

1.1 Bilateral teleoperation

In this chapter we'll present some important theory basis and concepts that will be used for the thesis.

With the term teleoperation we indicate the ability of a human operator to manipulate objects remotely under conditions that replicate those at the remote location. This can be useful when dealing with hazardous environments, to operate in areas non easily accessible by humans, in operations that require a higher level of precision and control that a robot can provide (like micro surgery).

We can make a first distinction in telerobotic systems based on the control architecture and the connection that there is between operator and robot, as explained in [7] the three main categories of control architecture are:

1. Supervisory
2. Shared
3. Direct

The distinction is made based on the amount of intelligence or autonomy of the system: 'direct control' means that the slave motion is directly controlled by the user while in the supervisory control the user gives high level commands to the slave which has a high level of intelligence or autonomy, shared control is in between the other two situations: some degree of autonomy or automated help is available to assist the user.

1: Supervisory control was first proposed by Ferell and Sheridan in 1967, motivated by the time delay in the communication channel that made direct control not always efficient. In this approach the master gives high level commands and receives information from the slave that has autonomous control loops. A

particular implementation of this type of control is the telesensor programming approach: we have shared autonomy with local sensory feedback loops used by the robot system, while global task-level jobs are specified interactively by the human operator. If we can obtain sufficient information about the environment from sensors, the slave can execute partial tasks autonomously. A simulated environment on the master site is especially useful in case of time delays (they're not duplicated in the simulation, visual feedback would not be efficient) and to use internal variables which cannot be observed in the real system. This kind of control approach is especially useful when the time delay is large and a visual feedback in not sufficient to allow the human operator a good degree of control over the robot.

2: In long distance or risky applications using shared control can guarantuee more safety and enable telepresence at the same time. Shared means that gross commands from the master are refined autonomously by the slave using some kind of sensory intelligence. In operations with time delays the control can be distributed between human and robot assigning each different subtasks. Virtual fixtures that guide or forbid movement in certain areas are also considered shared control and can achieve safer and faster operations.

3: In direct control the user uses some kind of device, for example a joystick, to manually specify the motion of the slave robot; this type of control avoids any difficulty in creating local autonomy. In this particular type of control we can make a further distinction between **unilateral** and **bilateral** control: in the first we can say that the information only flows from master to slave, while in the second the master device not only is used to control the motion of the robot but also serves to display some sort of haptic feedback. To quote Passenberg, Peer and Buss [9] "Bilateral haptic teleoperation systems allow humans to perform complex tasks in a remote or inaccessible environment, while providing haptic feedback to the human operator".

In unilateral control the most common techniques are acceleration or rate (velocity) control and position control. Both acceleration and rate control can make it difficult for the operator to reach and hold a target position: he has to control dynamic systems of first or second order. Position control is more intuitive and easy, the user specifies a target position and the slave takes care of the dynamic part of the control.

When the slave is under position control it's kinematically coupled with the master, and we need to remember that they move in two different workspaces with possibly different obstacles to consider. Also the master and slave robots can be kinematically similar, usually connected at joint level, or kinematically dissimilar, for example when the master is a joystick and the slave a robot in a very different shape, and connected at the tips; in this second case scaling is also sometimes necessary.

When using *bilateral* control there are two basic architectures that take into

account the force feedback, position-position and position-force [7]. In the first the two robots are instructed to track each other and both implement a tracking controller. This works if position and velocity gains are the same but if the robots are substantially different it leads to scaled or distorted forces, also the user will feel the inertia of the slave, which should be avoided.

Remembering that the human arm is usually represented as a mass-spring-damper system, the commands that we wish to fulfill with the position/position architecture are the following:

$$F_m = -K_m(x_m - x_{md}) - B_m(\dot{x}_m - \dot{x}_{md}) \tag{1.1}$$
$$F_s = -K_s(x_s - x_{sd}) - B_s(\dot{x}_s - \dot{x}_{sd}) \tag{1.2}$$

where $x_m - x_{md}$ is the difference between the actual position and the desired one. If the K and B gains are the same for master and slave the forces are the same. One problem with this architecture is that the user will feel the slave's controllers forces and the slave inertia, instead of just the environment forces.

The position/force architecture solves the problem of force distortion by using a force sensor on the slave robot's tip.

$$F_m = F_{sensor} \tag{1.3}$$
$$F_s = -K_s(x_s - x_{sd}) - B_s(\dot{x}_s - \dot{x}_{sd}) \tag{1.4}$$

The advantage is that the user only feels the external forces and can relay on tactile senses and not only on visual senses, enhancing the task performance. On the other hand this feedback may also cause instability in the system if the communication channel introduces time delay, and this has been one of the main challenges for researchers in the past.

As explained in [4] there are two theoretical goals for a teleoperation system: *stability* of the closed-loop system regardless of the behavior of the operator and of the environment, *telepresence* which is transparency of the system between the environment and the operator to provide the latter with a a sense of presence in the remote location; these two tasks are usually in conflict and a trade-off must be found to obtain the best results. The communication medium also contributes to the complexity of the system, introducing for example delays and distortions, and it must be taken into account when studying a teleoperation architecture.

1.2 Representation of a teleoperation system

In this section we'll explain how a teleoperation architecture can be represented mathematically, in particular we'll focus on how it can be modeled as a two-port

network [4].

A bilateral teleoperation system is comprised of a slave robot that interacts with a usually unknown environment, the slave robot is connected to the master robot and the latter is controlled by a human operator that closes the loop, the two systems (master and slave) exchange signals like position, velocity and force. We can describe with a linear model the master and slave system separately as follows:

$$M_m\ddot{x}_m + B_m\dot{x}_m = f_m + f_h \tag{1.5}$$
$$M_s\ddot{x}_s + B_s\dot{x}_s = f_s + f_e \tag{1.6}$$

with $x_{m,s}, f_{m,s} \in \mathbb{R}^n$ are the generalized coordinated and input forces respectively, M_* is a positive inertia matrix, B_* is a damping matrix and f_h, f_e are external forces (operator, environment). We can also write a nonlinear model as follows:

$$M_m(x_m)\ddot{x}_m + C_m(x_m, \dot{x}_m)\dot{x}_m = f_m + f_h \tag{1.7}$$
$$M_s(x_s)\ddot{x}_s + C_s(x_s, \dot{x}_s)\dot{x}_s = f_s + f_e \tag{1.8}$$

where now C represents Coriolis and centrifugal terms and we have the two following properties:

(PD): $M_* = M_*^T$ is positive definite,
(SS): $\dot{M}_* - 2C_*$ is skew simmetric.

Studying the stability of the whole telerobotic architecture with the traditional tools can be difficult: it depends non only on the robots but also on the environment and human operator which are often unknown or difficult to capture in a mathematical description. A tool that is commonly used to avoid these problems is the concept of **passivity**, that provides a sufficient (but not necessary) condition for the stability of the system.

1.2.1 Passivity

The concept of passivity is linked to the energy exchange between interconnected systems. We know that a system is passive if and only if it cannot produce energy, that the combination of two passive systems is passive and also the feedback connection of two passive systems is passive. We usually assume that the slave robot will only interact with passive environments and if we use the common model

mass-spring-damper for the human operator the worst case scenario is when the operator is not holding the haptic device ($\mathbf{F_h} = \mathbf{0}$). We can define passivity as follows (see [4]):

Definition : A dynamical system given by

$$\dot{x} = f(x, u) \tag{1.9}$$
$$y = h(x, u) \tag{1.10}$$

is considered passive if there exist a continuously differentiable semidefinite scalar function $V(x) : \mathbb{R}^n \rightarrow \mathbb{R}$ such that:

$$\dot{V} \leq u^T y \rightarrow \int_0^t u^T(\eta) y(\eta) d\eta \leq V(t) - V(0) \tag{1.11}$$

and is *lossless* if

$$\dot{V} = u^T y \rightarrow \int_0^t u^T(\eta) y(\eta) d\eta = V(t) - V(0) \tag{1.12}$$

So as we said a system is passive if it doesn't produce energy.

Proposition Assuming that the human and environment are passive ($\int_0^t [f_h^T(\eta)\dot{x}_m(\eta) - f_e^T(\eta)\dot{x}_s(\eta)]d\eta \geq 0$) the system 1.8 is passive with respect to the storage function

$$V = \frac{1}{2} \begin{bmatrix} \dot{x}_m \\ \dot{x}_s \end{bmatrix}^T \begin{bmatrix} M_m & 0 \\ 0 & M_s \end{bmatrix} \begin{bmatrix} \dot{x}_m \\ \dot{x}_s \end{bmatrix} \tag{1.13}$$

So in the non-linear system we can take forces as inputs and velocities as outputs and study the energy exchanged.These results will be particularly useful when considering the following information:

- A series cascade of passive two-ports is passive.
- Passivity leads to establishing the stability of the overall system by taking as a Lyapunov function the sum of storage functions of all constituent blocks.

A connection of a number of passive subsystem is passive, but often it creates an overdamping, combining active and passive subsystems can create a system stable and with less dissipation. We have therefore established the importance of

knowing if a teleoperator system is passive or not in connection with its stability. Unfortunately **passivity doesn't hold when we have time delays** introduced by the channel and in a separate chapter we'll address this issue in connection with some of the control algorithms proposed in the literature.

1.2.2 Two-port network

We have already seen how we can represent the master and the slave subsystems using traditional differential equations with inertia, damping and forces involved, now we'll see how we can represent the whole teleoperation system as a 2-port network (see figure 1.1), allowing for an easier analysis of its properties. We can consider this a mechanical/electrical analogy with the external signals being efforts and flows; for a mechanical system the efforts are the forces applied by the operator and the flows are for example the velocities, for an electrical system the efforts are the voltages and the flows are the currents. We'll also choose a sign convention: the power will be positive if flowing for example from master to slave and negative in the opposite direction.

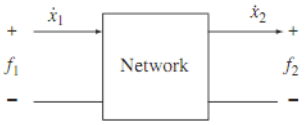


Figure 1.1: 2-port network

When the slave is in contact with the environment its forces and velocities are related by the impedance that characterizes the environment [5]:

$$F_s = Z_e V_s \tag{1.14}$$

A similar relation should link the forces and velocities of the master:

$$F_h = Z_t V_h \tag{1.15}$$

and ideally we want that if the forces are the same $F_s = F_h$ also the corresponding motions are equals, so that we can achieve **transparency**:

$$Z_e = Z_t \tag{1.16}$$

where of course this is just the ideal behavior, not possible in practice. We can describe the whole teleoperator as a two port network using a hybrid formulation with a so called hybrid matrix:

$$\begin{pmatrix} f_h(s) \\ \dot{x}_m(s) \end{pmatrix} = \underbrace{\begin{pmatrix} h_{11}(s) & h_{12}(s) \\ h_{21}(s) & h_{22}(s) \end{pmatrix}}_{H(s)} \begin{pmatrix} \dot{x}_s(s) \\ -f_e(s) \end{pmatrix} \tag{1.17}$$

All the H_{ij} parameters could be affected by the mechanical dynamics of master and slave and by the control architecture, so it's necessary to choose specific values for them, not arbitrary ones. To the hybrid matrix we can associate a physical interpretation and an ideal form that gives perfect transparency if we define it based on the perceived impedance: $Z_t = Z_e$ the impedance at the master equals the environment one.

$$H(s) = \begin{pmatrix} Z_{in} & \text{ForceScaling} \\ \text{VelocityScaling} & Z_{out}^{-1} \end{pmatrix} \quad H_{ideal}(s) = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \tag{1.18}$$

in fact $H_{ideal}(s)$ assures that the perceived impedance is equal to the environment impedance, which is the **transparency condition**:

$$Z_t = \frac{f_h(s)}{\dot{x}_s(s)} = (h_{11} - h_{12}Z_e)(h_{21} - h_{22}Z_e)^{-1} \tag{1.19}$$

The complete representation of the teleoperation system as a series of 2-port networks is presented in figure 1.2 and the arrows can be seen as the power flow.

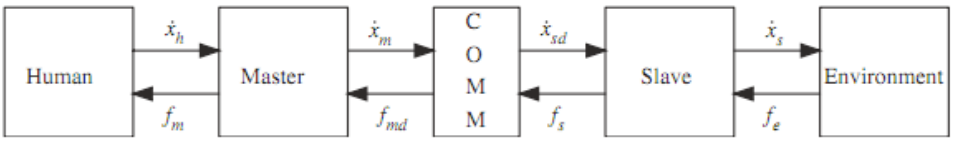


Figure 1.2: Complete 2-port network representation

1.2.3 Transparency

We'll add in this section a few more considerations on the concept of transparency, already introduced above. This property refers to the fact that the technical medium between the operator and the environment is not felt [9] and referring to the signals generally exchanged, forces and velocity, we can define it as follows:

$$f_h = f_e \qquad \dot{x}_m = \dot{x}_s \tag{1.20}$$

Now let's assume that we know the impedance mapping from velocities to forces in the frequency domain:

$$F_h(\omega) = Z_t(\dot{X}_m(\omega), \omega) \tag{1.21}$$
$$F_e(\omega) = Z_e(\dot{X}_s(\omega), \omega) \tag{1.22}$$

we can define transparency based on the two following conditions:

$$Z_t = Z_e \qquad \dot{X}_m = \dot{X}_s \tag{1.23}$$

So basically transparency means that when moving in free space external dynamics won't be felt, while when making contact with an object the latter is felt exactly at the master site.

In many architectures transparency and stability are often conflicting objectives, like for example in the 4-channel architecture (see chapter 2), so it's necessary to find a trade-off between the two.

Chapter 2

Control algorithms and time delay

One of the issues when dealing with a teleoperation system is the presence of a time delay introduced by the communication channel. This creates limitations on the actual performances of the system, in particular we can see that when we send a command from the master the corresponding action at the slave takes place after at least the one way delay T, and of course the reaction to the environment contact can only take place after a complete RTT of 2T. So in fact the closed loop bandwidth is limited by the same time constant [8].

So for example in the basic position-position architecture and in the position-force architecture the information that flows between the two sides is:

$$x_{md} = e^{-sT} x_m \qquad x_{sd} = e^{-sT} x_s \tag{2.1}$$

$$x_{md} = e^{-sT} x_m \qquad F_{sd} = e^{-sT} F_{sc} \tag{2.2}$$

so during the communication energy may be generated and this could be cause of instability.

There are a number of different approaches to operate with a time delay, some of them more efficient than others. In order to determine the best solution for our case we studied the literature on this topic and ultimately decide upon one specific control algorithm the seemed to be the best solution. We'll present in this chapter some of this control techniques and some considerations about them to explain our choice.

In the past numerous techniques to deal with this issued have been developed, and we can summarize the main ones and their characteristics as follows [1]:

1. Position-force

2. Position-position

3. Shared Compliance Control: a compliance term is inserted in the slave controller to modify the desired displacement received from the master according to the interaction with the environment.
4. Force reflection with passivity
5. Intrinsically passive controller (*wave variables*).
6. 4-channnels
7. Adaptive motion/force control
8. Sliding mode controller
9. Predictive control
10. Predictive control with passivity

The first two schemes are the classical ones that we presented in chapter one, none of this two schemes is intrinsically stable with time delay and in particular the position-force scheme can present stability problems even without time delay: if the loop gain is too high a small motion command can turn into a large force if the slave is pressing against a stiff environment. Depending on the system parameters we'll find a maximum time delay that preserves stability: this means that we cannot choose arbitrary values for the controllers of master and slave if we want the whole system to remain stable and the results we can achieve in terms of performances are less then ideal.

The *force reflection with passivity* modifies the classical position force architecture to achieve passivity by adding a damping injection term in the force feedback:

$$F_m = G_c F_{sensor} + B_i v_m \tag{2.3}$$

clearly this damping injection will affect the inertia perceived from the operator, creating stability at the expense of transparency. As explained in [1] there are also very poor tracking performances suggesting that this method is not efficient for what we need to achieve.

In scheme number **3 (Force reflection with passivity)** a compliance term is inserted in the slave controller to modify the commands received from the master accordingly to the interaction with the environment. The control equations are:

$$F_{mc} = G_c F_{sd} \tag{2.4}$$

$$F_{sc} = K_c(x_{md} - x_s + G_f(s)F_e) \tag{2.5}$$

with G_c and K_c control parameters and $G_f(s)$ a low pass filter. The first thing we can say is that using a local compliance control alters the perceived stiffness and this can cause a position drift, also the tracking error will depend on the time delay. It's not possible to ensure stability for this scheme regardless of how we choose the control parameters, in fact if the time delay is too high we can only obtain good tracking (using visual feedback) by turning off the force feedback. Clearly this is not what we want for our tele-echography system, so this control scheme is not suitable.

The **intrinsically passive controller** comes from concepts of line theory applied to a teleoperation architecture. The signals that we want to transmit are encoded as wave or scattering variables in order to achieve a formulation that as the name says is in itself passive, without requiring a particular tuning of the control parameters. In fact in this formulation there is only one main parameter called *wave impedance* that will be used to tune the system and obtain better performances. In fact tracking error and stiffness perception depend both on the time delay and the tuning parameter, we can adjust our system to fit the time delay of the communication channel.

We'll discuss the **4-channel architecture** in a little more detail in the following section to explain why it's not the ideal choice for what we need.

The **adaptive control** implements separately on both master and slave adaptive controllers for position and force. Once again this scheme can only achieve perfect transparency and stability as long as there is no time delay. If this is not the case it's necessary to find a trade-off between the ideal values of the parameters and the limitations that we have if we want to keep the system stable.

Scheme number **8: sliding mode controllers** can be used at the slave to achieve perfect tracking of the delayed master position, but it only works under specific conditions and depending on the control parameters there is a maximum time delay admissible to preserve stability.

The **predictive control** is similar to a position-force architecture, but we consider the remote dynamics into the local controller in order to predict the slave behavior, for example using a Smith filter to anticipate computation of the delayed information from the slave. In particular the Smith predictor computes the force feedback in advance using the current master position. For this scheme as well stability can be preserved only as long as the time delay is under a certain T_{max} and only with an appropriate choice of the control parameters.

To summarize intrinsic stability is achieved only by the architectures that take explicitly into account passivity: 4, 5 and 10. The other schemes are possibly stable, they're stable for any T but only for some choices of the controller's parameters, or for $T \leq T_{max}$. The 4C scheme is possibly stable but choosing some of the parameters properly it's intrinsically stable with respect to the other parameters.

The adaptive control (7) and the sliding modes (8) are strongly dependent on the external environment and the predictive control's stability is related to a good knowledge of the slave manipulator and time delay because we use a Smith predictor. The force reflection scheme achieves passivity through a damping injection and is far from ideal. Let's see in the next section how the 4-channel architecture works.

The 4-channel architecture

In the early 90s Lawrence realized that in order to achieve transparency ($Z_t = Z_e$) it was necessary to have a feedback of both forces and velocities and use it in the control laws for master and slave. The 4-channel architecture [5] as conceived by Lawrence is composed by the master and slave dynamics and by four channels for the bilateral communication used to transmit the necessary variables: $\dot{x}_m$ and f_m in one direction and $\dot{x}_s$ and f_s in the other. We will see how this architectures can offer only a trade-off between stability and transparency, especially in the presence of time delay.

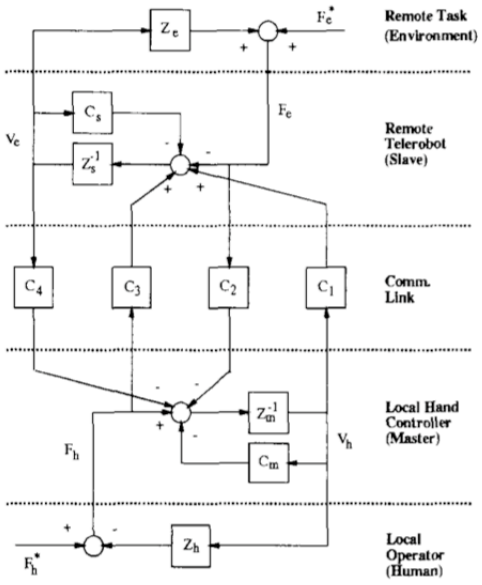


Figure 2.1: 4-channel basic architecture

Considering the hybrid formulation for a 2-port network we have:

$$\begin{bmatrix} F_h(s) \\ V_h(s) \end{bmatrix} = \begin{bmatrix} H_{11}(s) & H_{12}(s) \\ H_{21}(s) & H_{22}(s) \end{bmatrix} \begin{bmatrix} V_e \\ -F_e \end{bmatrix}$$

and considering $F_e(s) = Z_e(s)V_e(s)$ and $F_h = Z_t V_h$ we can write the expression for the transmitted impedance (dropping the Laplace s):

$$Z_t = (H_{11} - H_{12}Z_e)(H_{21} - H_{22}Z_e)^{-1} \tag{2.6}$$

from which we can derive conditions on the H_{ij} parameters to match $Z_e = Z_t$. Furthermore in reference to the system of figure 2.1 we can obtain the following expressions:

$$\begin{aligned} H_{11} &= (Z_m + C_m)D(Z_s + C_s - C_3C_4) + C_4 \\ H_{12} &= -(Z_m + C_m)D(I - C_3C_2) - C_2 \\ H_{21} &= D(Z_s + C_s - C_3C_4) \\ H_{22} &= -D(I - C - 3C_2) \\ D &= (C_1 + C_3Z_m + C_3C_m)^{-1} \end{aligned}$$

Using appropriate values for the controllers C_1, C_2, C_3, C_4 one can achieve transparency; the first thing is to eliminate Z_e from the denominator of 2.6:

$$C_3C_2 = I$$

then we can also make Z_t a linear function of Z_e with:

$$C_4 = -(Z_m + C_m) \quad ; \quad C_1 = (Z_s + C_s)$$

This leads to the final formula

$$Z_t = H_{12}Z_eH_{21}^{-1} = C_2Z_e$$

which suggest to put $C_2 = I$.

Regarding the stability of this architecture one can obtain sufficient conditions based on passivity arguments; it's easy to verify that these conditions are violated as soon as we don't have at least one of the subsystems strictly passive and when a time delay is introduced. As stated in [5]:

Lawrence proposes a method to obtain a passive communication link using appropriate filters, unfortunately this filters compensate a specific time delay and a specific wave attenuation factor. It's not clear if they're able to compensate a variable time delay and retain passivity.

Another paper by Hashtrudi-Zaad and Salcudean [3] proposes a slightly different architecture with two additional controllers for local force-feedback. In the presence of time delay they find conditions using the Nyquist criterion to ensure the passivity of the system for any time delay T and therefore stability. This conditions put limitations on the values of mass-spring-damping parameters of the

systems and work well when the environment is heavier, has more damping and is stiffer than the operator's arm. When this conditions are not met, to preserve stability, the transparency has to be compromised.

Chapter 3

Wave variables and scattering formulation

Both the wave variables formulation and the scattering formulation are based on the interpretation of the signals involved in the teleoperation system as incoming and reflected waves. These formulation have different properties and characteristics, in the following we'll underline the main advantages or flows of both, showing how the wave variables formulation is more suitable for the task at hand.

3.1 Scattering formulation

The idea to apply the notion of scattering variables (used in line theory) to the problem of bilateral teleoperation was first introduced by Anderson and Spong.

Let's consider a bilateral teleoperator viewed as a series cascade of 1 and 2-port networks with an effort-flow pair being exchanged at each port. The scattering operator is defined as follows:

Def. Given the incident wave $(f(t)+\dot{x}(t))$ and the reflected wave $(f(t)-\dot{x}(t))$ the *scattering operator* links them as $(f(t) - \dot{x}(t)) = S(f(t) + \dot{x}(t))$.

Let's consider the case of a 2-port network represented through the hybrid matrix:

$$\begin{pmatrix} f_1(s) \\ -\dot{x}_2(s) \end{pmatrix} = \begin{pmatrix} h_{11}(s) & h_{12}(s) \\ h_{21}(s) & h_{22}(s) \end{pmatrix} \begin{pmatrix} \dot{x}_1(s) \\ f_2(s) \end{pmatrix}$$

in this case we can define the scattering operator in the frequency domain as follows in terms of the hybrid matrix:

$$S(s) = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} (H(s) - I)(H(s) + I)^{-1}$$

To ensure that the scattered wave will not have energetic content greater than the incident wave, and therefore have passivity, we need the following condition:

Theorem A n-port system is passive if and only if $||S(j\omega)||_\infty \leq 1$ of the corresponding scattering matrix.

If we consider the input and output power we can see where this condition comes from.

$$\Delta P = P_{in} - P_{out} = f^T \dot{x} = \left(\frac{f+\dot{x}}{2}\right)^T \left(\frac{f+\dot{x}}{2}\right) - \left(\frac{f-\dot{x}}{2}\right)^T \left(\frac{f-\dot{x}}{2}\right)$$

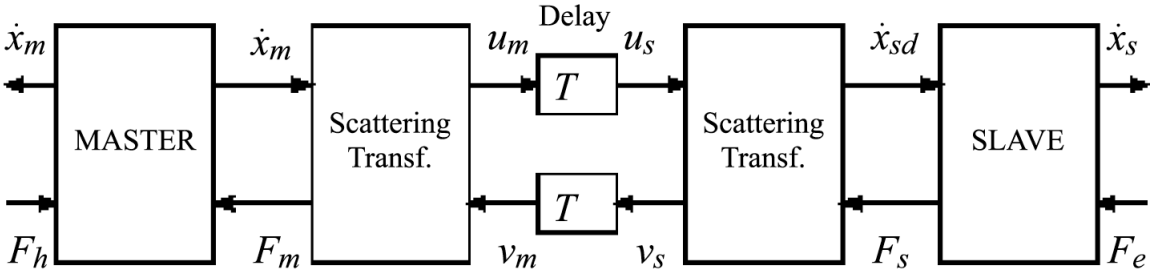
we can then define $s_+ = \left(\frac{f+\dot{x}}{2}\right)$ and $s_- = \left(\frac{f-\dot{x}}{2}\right)$ and calculate ΔP:

$$\Delta P = s_+^T s_+ - s_-^T s_- = s_+^T (I - S^T S) s_+$$

and imposing that it's non-negative finally have the condition

$$||S||_\infty = \bar{\sigma}(S^T(j\omega)S(j\omega)) \leq 1$$

This condition unfortunately doesn't hold when there's a time delay, in fact in that case we have $||S||_\infty = \infty$, to achieve passivity under a constant time delay it's necessary to adopt a scattering formulation that passifies the communication, leading to a poorer performance.



Source: Niemeyer (1996)

Figure 3.1: Scattering transformation layout

3.2 Wave variables

The wave variables formulation was first proposed by Niemeyer and Slotine and is conceptually close to the scattering formulation. They're based only on the concepts of power and energy, they're applicable to nonlinear systems and can handle

unknown models and large uncertainties [8]. We can see why this formulation can be highly appealing and work really well when properly used.

To be able to deal with the time delay in the communication we need both inertial and compliance characteristics: the inertial part captures the fact that the slave robot cannot accelerate or decelerate to execute a command until after the time delay T, the compliance represent the limitation given by the fact that the master receives a contact force after a time delay T, by that time the slave could not be in the same position anymore.

The magnitude of the inertial and compliance properties should be proportional to the time delay T, this adjustment to the time delay happens automatically in the wave variables formulation: the natural frequency of this formulation is the square root of stiffness over inertia and equals 1/T. The tuning parameter to find a tradeoff between inertia and compliance is the wave impedance *b* that is accessible online to be adjusted to the current task.

Encoding

Given a standard pair of power variables, such as for example (F, $\dot{\mathbf{x}}$) force and velocity, we can calculate the corresponding pair of wave variables (**u,v**) as follows:

$$\mathbf{u} = \frac{b\dot{\mathbf{x}} + \mathbf{F}}{\sqrt{2b}} \qquad \mathbf{v} = \frac{b\dot{\mathbf{x}} - \mathbf{F}}{\sqrt{2b}} \tag{3.1}$$

where **u** indicates the forward wave and **v** the backward wave. The parameter *b* is called **wave impedance** and will be used to tune the control. From this basic equations we can easily obtain formulas to calculate a velocity command and the force feedback:

$$\dot{\mathbf{x}} = \frac{1}{\sqrt{2b}}(\mathbf{u} + \mathbf{v}) \qquad \mathbf{F} = \sqrt{\frac{b}{2}}(\mathbf{u} + \mathbf{v}) \tag{3.2}$$

$$\mathbf{u} = -\mathbf{v} + \sqrt{2b}\dot{\mathbf{x}} \tag{3.3}$$

$$\mathbf{F} = b\dot{\mathbf{x}} - \sqrt{2b}\mathbf{v} \tag{3.4}$$

The power flow as well can be redefined using the wave variables:

$$P = \dot{\mathbf{x}}^{\mathbf{T}}\mathbf{F} = \frac{1}{2}\mathbf{u}^{\mathbf{T}}\mathbf{u} - \frac{1}{2}\mathbf{v}^{\mathbf{T}}\mathbf{v} \tag{3.5}$$

where $\frac{1}{2}\mathbf{u}^{\mathbf{T}}\mathbf{u}$ represents the flow along the main direction while $\frac{1}{2}\mathbf{v}^{\mathbf{T}}\mathbf{v}$ is against it.

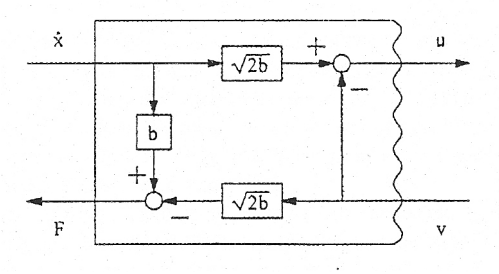


Figure 3.2: Wave variables transformation

An important result that shows how useful the wave variables formulation can be concerns the passivity of the system. Keeping in mind that a system is passive if it doesn't produce energy, and assuming an initial energy of zero we have [6]:

$$\begin{aligned}\mathbf{E} &= \int_0^t P_{in}(\tau)d\tau = \int_0^t (\dot{x}_m(\tau)F_m(\tau) - \dot{x}_s(\tau)F_s(\tau))d\tau \\ &= \frac{1}{2}\int_0^t u_m^T(\tau)u_m(\tau) - v_m^T(\tau)v_m(\tau) + v_s^T(\tau)v_s(\tau) - u_s^T(\tau)u_s(\tau)d\tau \\ &= \frac{1}{2}\int_{t-T}^t (u_m^T(\tau)u_m(\tau) + v_s^T(\tau)v_s(\tau))d\tau \geq 0 \end{aligned} \tag{3.6}$$

where the relations $u_s(t) = u_m(t-T)$ and $v_m(t) = v_m(t-T)$ have been used. We can see that the system is passive **regardless of the time delay T**, even if this results only holds for constant T; unlike many other control techniques discussed in the previous chapter we don't have a maximum time delay that still preserves stability, we can tune our control parameters regardless of the magnitude of the delay. We'll briefly discuss the case of a *variable time delay* in the next section, even though there was no time to implement it. The control equations are the following:

$$\mathbf{u}_s(t) = \mathbf{u}_m(t-T) \tag{3.7}$$

$$\mathbf{v}_m(t) = \mathbf{v}_s(t-T) \tag{3.8}$$

$$\mathbf{u}_m(t) = \frac{b\dot{\mathbf{x}}_m(t) + \mathbf{F}_m(t)}{\sqrt{2b}} = \sqrt{2b}\dot{\mathbf{x}}_m - \mathbf{v}_m \tag{3.9}$$

$$\mathbf{v}_s(t) = \frac{b\dot{\mathbf{x}}_s(t) + \mathbf{F}_s(t)}{\sqrt{2b}} = \mathbf{u}_s - \sqrt{\frac{2}{b}}\mathbf{F}_s \tag{3.10}$$

Other advantages in the use of wave variables include their symmetric formulation that makes them easy to understand and implement and the hybrid encoding, this in particular means that the wave itself does not distinguish the kind of command that it carries, so any element may interpret the incoming wave as best suited to its current needs. For example a robot in contact with the environment will use it to generate a force, while a robot in free space will generate a motion providing a great flexibility when dealing with unknown environments. The wave commands are generally indicated as "move or push" instructions and the sign determines the direction as forward or backward compared to the main power flow direction. The master command and the slave command are shown in the following equations:

$$\mathbf{F}_m = b\dot{\mathbf{x}}_m - \sqrt{2b}\mathbf{v}_m \tag{3.11}$$

$$\dot{\mathbf{x}}_s(t) = \sqrt{\frac{2}{b}}\mathbf{u}_s(t) - \frac{1}{b}\mathbf{F}_s(t) \tag{3.12}$$

In the basic layout for wave communication (fig 3.3) we can identify three different feedback paths created by the wave transformation itself that need to be taken into consideration to obtain good results. First we have a damping created in the wave transformation that is immediate and doesn't go through the delay. This first feedback creates an appropriate damping and doesn't need to be eliminated.

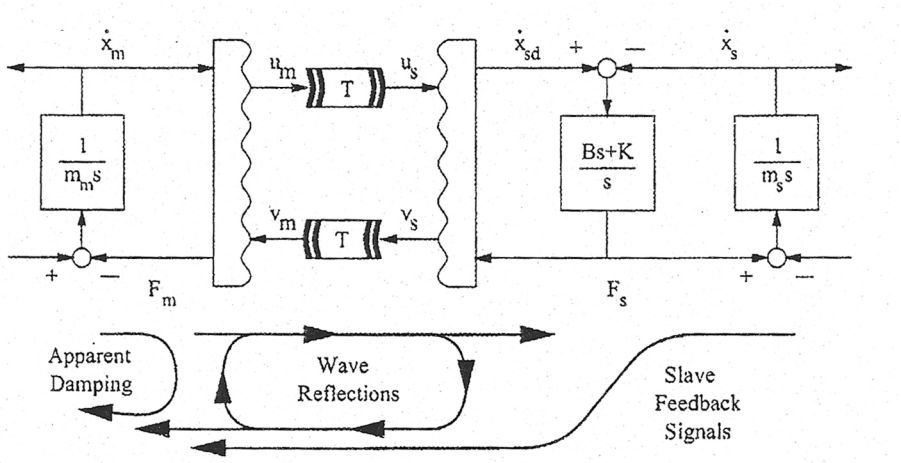


Figure 3.3: Loops in the wave variables layout

The second is generated by wave reflections at both sides: when a wave reaches a side, part of it gets reflected back, and again when it reaches the other side there is a reflection and so on, as we can see from the equations (3.9) and

(3.10). This reflected waves contain no useful information and can last for several cycles before dying out, therefore disturbing the communication of the actual information, we can also interpret this as a resonance of the wave at its natural frequency 1/T. The third path is the one that we actually need: the feedback from the slave to the master that carries useful information.

Tuning of wave impedance and wave filters

Wave reflections appear when a wave signal hits an element with an impedance different from its own, so ideally we would know the impedance of the environment and choose a value for the parameter *b* that matches it. Unfortunately we often deal with unknown environments, or environments with impedance that varies over time. What we can do in this case is choose *b* close to the estimated value of the environment impedance and then apply *wave filters* to smooth the system behavior. These **wave filters** are inserted in the transmission path (figure 3.4) and they don't compromise passivity because wave variables are constructed to be unaffected by delays or phase lag; the only downside is that they could reduce tracking performances so they must be used carefully. Their gain should be limited below unity and the constants should be chosen such that the actual bandwidth is close to the actual time delay. For example:

$$\frac{d}{dt}\mathbf{u}_s(t) + \lambda \mathbf{u}_s(t) = \lambda \mathbf{u}_m(t-T)$$
$$\frac{d}{dt}\mathbf{v}_m(t) + \lambda \mathbf{v}_m(t) = \lambda \mathbf{v}_s(t-T)$$

with $\lambda = \mathbf{1/T}$. The filters create a smoother behavior by eliminating the high-frequency components often seen in the wave reflection, high frequency events are for example impact or stick/slip. The use of wave filters may reduce performances because it increases inertia and reduces stiffness, they reduce some of the feedback signals, so it's important to carefully tune both the wave impedance and the cut frequency of the filters to still have good performances.

3.3 Wave variables with time varying delay

The wave variables formulation as proposed by Niemeyer and Slotine has been extended to the case of a variable time delay by Lonzano Chopra and Spong [6].

Ever since the internet become a possible communication medium for teleoperation the time delays became variable due to factors such as congestion, bandwidth, packet loss ect.. To address this problem the idea is to start from the basic

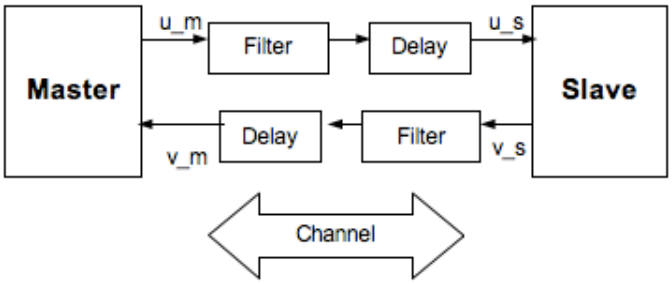


Figure 3.4: Wave filters in the communication block with a channel that introduces time delay

wave variables layout and add a time varying gain in the communication block, this ensures passivity for arbitrary time varying delays as long as we have a known bound on the rate of change of the time delay.

We'll consider again the equation for the total stored energy (see 3.6) and use a time delay $T = T(t)$ that varies over time.

The transmission equations will present two different time delays, one for the forward wave and one for the feedback wave:

$$\mathbf{u}_s(t) = \mathbf{u}_m(t - T_1(t)) \tag{3.13}$$
$$\mathbf{v}_m(t) = \mathbf{v}_s(t - T_2(t)) \tag{3.14}$$

in this case the equation for the stored energy becomes:

$$\begin{aligned}\mathbf{E} &= \int_0^t P_{in}(\tau)d\tau = \frac{1}{2}\int_0^t u_m^T(\tau)u_m(\tau) - v_m^T(\tau)v_m(\tau) + v_s^T(\tau)v_s(\tau) - u_s^T(\tau)u_s(\tau)d\tau \\ &= \frac{1}{2}\{\int_{t-T_1(t)}^t (u_m^T(\tau)u_m(\tau)d\tau + \int_{t-T_2(t)}^t v_s^T(\tau)v_s(\tau))d\tau \\ &+ \int_0^{t-T_1(t)} u_m^T(\tau)u_m(\tau)d\tau + \int_0^{t-T_2(t)} v_s^T(\tau)v_s(\tau)d\tau \\ &- \int_0^t u_m^T(\tau - T_1(t))u_m(\tau - T_1(t)) + v_s^T(\tau - T_2(t))v_s(\tau - T_2(t))d\tau\}\end{aligned} \tag{3.15}$$

change of variables: $\sigma = \tau - T_i(\tau) := g_i(\tau)$ and impose the condition of causality to the change:

$$g_i' = 1 - \frac{dT_i}{d\tau} \geq 0, \quad i = 1, 2 \tag{3.16}$$

and after some calculations obtain:

$$\int_0^t P_{in}(\tau)d\tau = \frac{1}{2}\{\int_{t-T_1(t)}^{t} (u_m^T(\tau)u_m(\tau)d\tau + \int_{t-T_2(t)}^{t} v_s^T(\tau)v_s(\tau))d\tau$$
$$- \int_0^{t-T_1(t)} \frac{T_1'(\sigma)}{1-T_1'(\sigma)} u_m^T(\sigma)u_m(\sigma)d\sigma - \int_0^{t-T_2(t)} \frac{T_2'(\sigma)}{1-T_2'(\sigma)} v_s^T(\sigma)v_s(\sigma)d\sigma\} \tag{3.17}$$

We can see that the last two terms are are non-positive whenever the delay is increasing and determine the energy produced by the communications due to the increasing delay. Therefore, the system is, in general, not passive due to the time varying delay [6].

The proposed architecture inserts a time varying gain in the communication block as seen in figure 3.5 leading to the transmission equations:

$$\mathbf{u}_s(t) = f_1(t)\mathbf{u}_m(t - T_1(t)) \tag{3.18}$$
$$\mathbf{v}_m(t) = f_2(t)\mathbf{v}_s(t - T_2(t)) \tag{3.19}$$

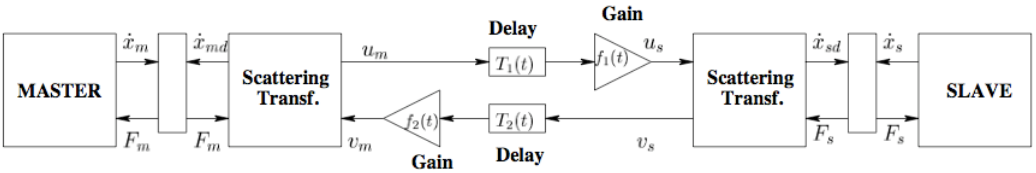


Figure 3.5: Wave variables with time varying gain in the communication block

With this modification the equation for the energy becomes:

$$\int_0^t P_{in}(\tau)d\tau = \frac{1}{2}\{\int_{t-T_1(t)}^{t} (u_m^T(\tau)u_m(\tau)d\tau + \int_{t-T_2(t)}^{t} v_s^T(\tau)v_s(\tau))d\tau$$
$$+ \int_0^{t-T_1(t)} \frac{1-T_1'-f_1^2}{1-T_1'} u_m^T(\sigma)u_m(\sigma)d\sigma + \int_0^{t-T_2(t)} \frac{1-T_2'-f_2^2}{1-T_2'} v_s^T(\sigma)v_s(\sigma)d\sigma\} \tag{3.20}$$

and choosing $f_i^2 = 1 - T_i'$ would ensure passivity. We can see that a less strict condition still preserves passivity, in fact it's enough to have:

$$f_i^2 \leq 1 - \frac{dT_i}{d\tau} \tag{3.21}$$

We can see that to apply this method we need an estimation of the value $\frac{dT_i}{d\tau}$ and that of course the performances will depend not only on the tuning of the wave impedance *b* but also on the value of the gains.

In the paper by Lonzano Chopra and Spong [6] it's noted that if the time delays are large the tracking performances are not satisfactory, so Chopra and Spong in [2] proposed a modified control configuration that included an additional feedforward control. Their method was able to obtain better tracking performances and still preserve the stability of the system.

Chapter 4

Hardware and software description

After analyzing the problem in theory we want to apply it to an actual robots for tele-echography and see how the control algorithm that we chose to use works in this particular situation.

As previously said the purpose of this thesis was the study of an actuator used for the end effector of a tele-echography architecture at the slave side, so in this chapter we will present the hardware and software used, the general architecture of the robot and discuss the various aspects we dealt with while working on it.

4.1 Hardware

First we'll focus on the hardware, presenting the whole system and than focusing on the motor object of the study. In figure 4.1 we can see the prototype of the robot to be used at the slave to perform the echography, the linear actuator will be placed where the orange piece is and will allow vertical movement to improve the precision of the task, this robot is used to test the algorithms to control the position of the probe on the body of the patient. To understand the proportions of the photos we can say that the length of the linear actuator of the probe is about 10cm, and the same actuator in the probe of figure 4.2 is used in the end effector of figure 4.1 and is the one in figure 4.3.

The probe used to control the slave is shown in figure 4.2, and it consists of a linear actuator like the one used for the slave side that compresses 2 springs with **elastic constant of $\mathbf{K_s = 440N/m}$** on its left and right to push the tip. Finally the linear actuator studied is in figure 4.3. The actuator can only move 2cm downwards with respect to the zero position and 2cm upwards. As we can see it pushes on a spring also with elastic constant $K_s = 440$N/m used to simulate the patient and the spring is placed on the force sensor that we'll need to realize the force feeback. The robot in figure 4.1 won't actually be used to test the force feedback

Figure 4.1: Tele-echography robot, slave side

in this thesis, so for us the slave will be just the linear actuator-sensor system.

Figure 4.2: Tele-echography robot, master side

In the beginning there where actually two different options for the actuator, a standard DC motor (fig 4.4) and a linear actuator, in the following we'll explain the reasons for choosing the linear one, but basically the other actuator was part

Figure 4.3: Tele-echography robot, linear actuator used for the end-effector.

of a first prototype and the linear actuator was chosen as a possible improvement, as we'll see it actually works better.

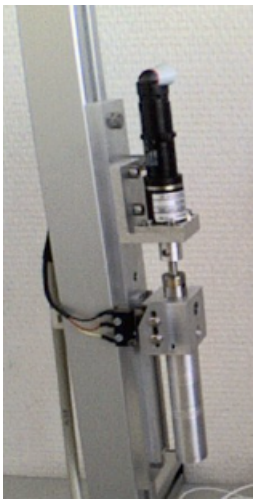


Figure 4.4: Tele-echography robot, standard DCactuator used for the end-effector.

The movements of both actuators could be controlled using a control module shown in figure 4.5 and either directly with a software provided by the producers

of programming a microprocessor in C to communicate with these control modules and execute more complex tasks.

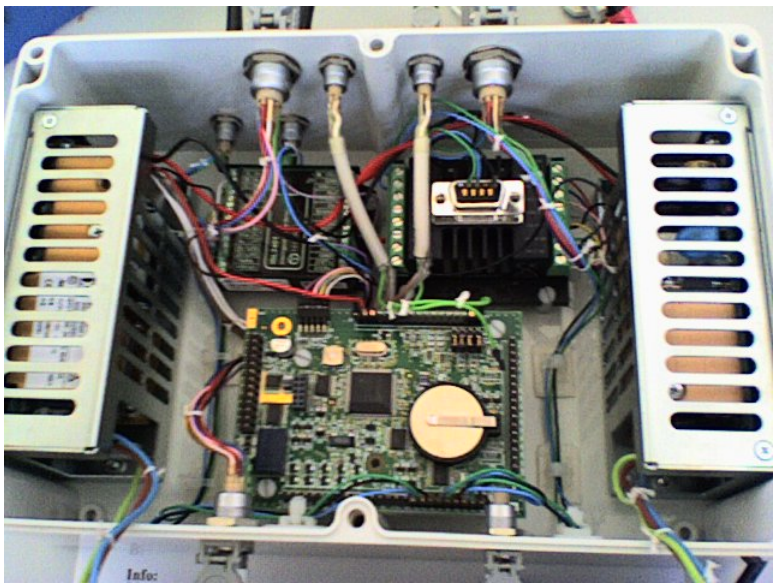


Figure 4.5: Control module.

General architecture

The complete structure of the architecture is composed of *master and slave computers* that are used to program the *master and slave control modules* to which they're connected via a *serial link*.

The master control module is connected to the probe in figure 4.2 that is used to set the pressure we want to apply on the "patient": it measures the vertical displacement when we push the probe and sends the information to the master computer, this information will be communicated to the slave computer. It also presents the user with a feeling of the pressure felt at the slave side: the more we push down the more we feel a force that pushes back.

On the slave side the control module is connected to the actuator to send it the vertical displacement that the master set. It's also connected to the force sensor and to the sensor that measures the position of the actuator (to verify that we are in fact following the commands received).

The master and slave computer communicate using a UDP protocol and a satellite connection.

4.1.1 Actuators

The benchmark studied consists of 2 actuators, a standard DC motor and a linear actuator. The DC actuator had been used in a previous prototype of the end effector of the probe and the goal was to compare its performances with the linear actuator to determine if it was more suitable for the task or not.

The first thing we did was to translate the performances that we expected in mathematical terms in order to have the instruments to analyze the actuators:

Table 4.1: Goals

Rising time	$T_r \leq 0.5s$
Settling time	$T_s \leq 1s$
Overshoot	$S \approx 0$

Table 4.2: Actuators

Standard DC actuator	Harmonic drive, pma-5a-100-01-e5-12ml
Linear actuator	Faulhaber, lm 2070-040-01

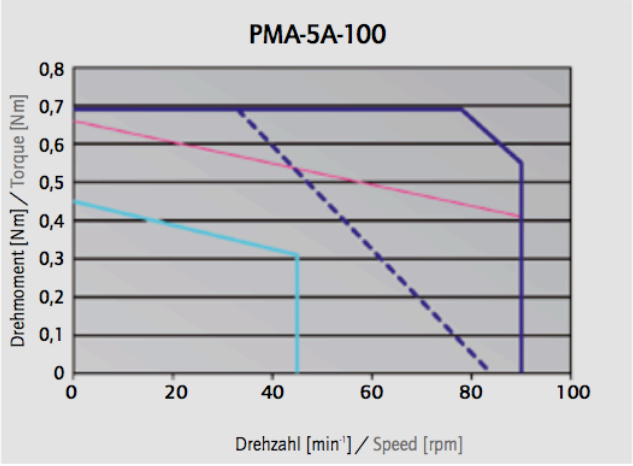


Figure 4.6: Harmonic drive actuator, performance curves.

Antrieb Actuator	Einheit Unit	PMA-5A		
Untersetzung Ratio		50	80	100
Maximales Drehmoment Maximum output torque	Nm	0,39	0,59	0,69
Maximale Drehzahl Maximum output speed	min^{-1} rpm	180	113	90

Figure 4.7: Data sheet information on maximum output torque and speed of the Harmonic Drive actuator

As we can see from figure 4.6 and also from the datasheet the DC actuator can reach a maximum output speed of 90rpm. We also know that to it takes 10 turns to move the actuator of 1cm.

90rpm $= 1.5$rps $\rightarrow$ 10turns $= 1$ cm $\rightarrow$ 1cm in 6.6s

We can see that it takes a relatively long time to perform a movement of 1cm and this will definitely disrupt the feeling of presence. Since we cannot obtain the specifics that we wish to achieve, we decided to focus on the linear actuator for the implementation part. The actuator is studied for 1DOF (degree of freedom) and moves vertically, as it moves downwards it pushes a spring that represents the environment impedance, the patient in our case, and has an **elastic constant of $\mathbf{K_s = 440N/m}$**. The vertical sliding movement object of this study is performed by the tip of the end-effector of the slave, the probe for the ecography (fig. 4.1).

The technical data for the linear actuator can be found in table 4.3; as we can see it appears to offer much better performances, in particular is has good values for speed and acceleration, good precision and the stroke length works for the application we want to implement. Figure 4.8 from the data sheet shows the dimensions of the actuator.

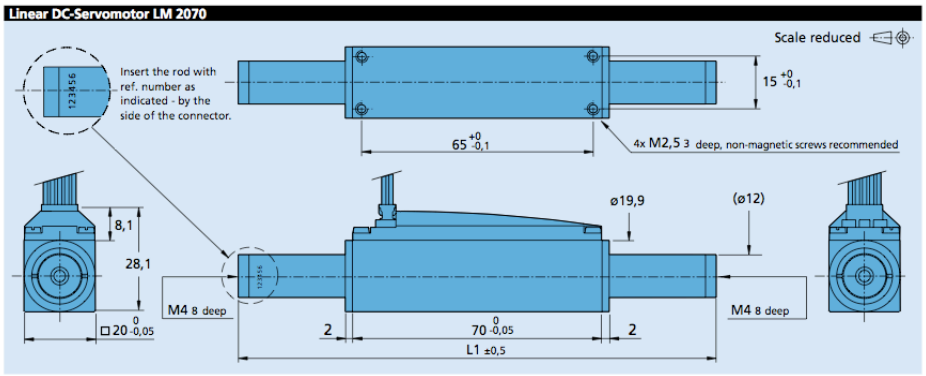


Figure 4.8: Data sheet of the linear actuator, dimensions

Table 4.3: Linear actuator data

Continuous force	9,2 N
Peak force	27,6 N
Continuous current	0,79 A
Peak current	2,37 A
Back-EMF constant	$K_e = 9,5$ V/m/s
Force constant	$K_f = 11,64$ N/A
Terminal resistance, phase-phase	R=10,83 Ω
Terminal inductance, phase-phase	L=1,125 μH
Stroke length	40mm $(+20, -20)$
Repeatability	60 μm
Precision	200μm
Acceleration	93,9 m/s^2
Speed	1,9 m/s
Rod weight	98g
Total weight	236g
Magnetic pitch	24mm

The position of the actuator is measured with Hall sensors.

4.1.2 Force sensor

To realize the force feedback we need to measure the force felt by the patient, in order to do so in this simulation we used a force sensor placed under the spring that's supposed to represent the human body. The benchmark was already equipped with such force sensor connected to an analog input of the control module, so at first the measurements where made using readings from this analog input of the sensor. The values obtained are in the range $0 \div 2047$ and they can be converted in Volts according to the gaincode used in the function provided to read the analog input. Using this method we noticed some imprecisions in the values, therefore I run some tests to understand what exactly was the problem. Applying a specific voltage from 0V to 2V (the gaincode chosen) at fixed steps I measured the output value (the one between 0 and 2047) and than converted it in volts using

$$\text{volts} = \frac{\text{output} \cdot 2}{2047}$$

now if we compare the input value and the output value converted in volt we can see that the conversion in not perfectly precise (figure 4.9).

Table 4.4: Sensor test

Input [V]	Output [V]
0	0.04
0.1	0.13
0.2	0.18
0.3	0.31
0.4	0.4
0.5	0.49
...	...
1.6	1.48
1.7	1.52
1.8	1.65
1.9	1.75
2	1.86

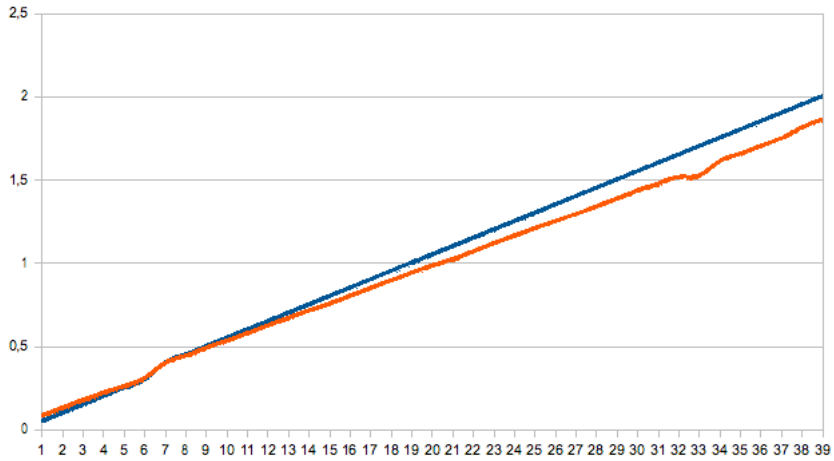


Figure 4.9: In blue the input value (Voltage), in red the output converted in volts.

We have a gap between the input value and the output that becomes bigger as we input bigger voltages, as we can see in figure 4.10 where we can evaluate how big is the error in function of the voltage (force) applied.

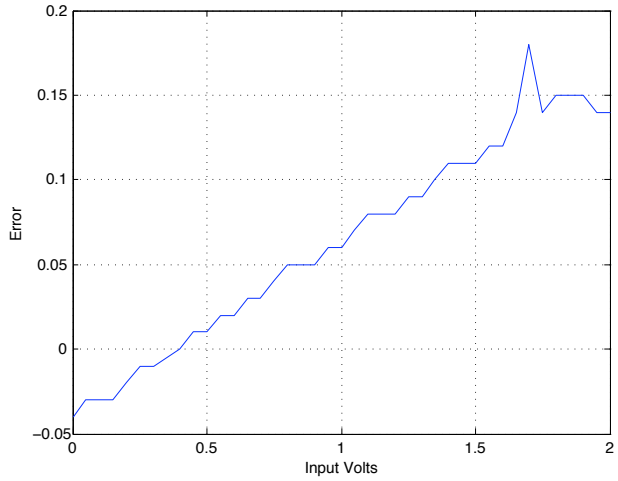


Figure 4.10: Plot of the error (difference) between the input value (the voltage we applied) and the output converted in volts and the input voltage we applied (on the x-axis)

Now the question is how this error in the values we'll be able to measure will affect our system. If we measure the position, multiply it by the elastic constant and we plot the resulting force in function of the measured voltages we find a linear response (figure 4.11) and most importantly we can see that the **maximum voltage reached is under 0.5V** since we have movement limitations. Comparing figures 4.11 and 4.10 we can deduce that the error in the conversion from the value between 0 and 2047 that we read and its corresponding actual voltage related to the force applied is not very significant.

We can also easily derive a linear function that links the voltages we measure to the force applied, in fact plotting volts as a function of the position I found the linear function that fits the data:

$$\begin{aligned} V &= A \cdot pos + B \\ &= A\frac{F}{4.4 \cdot 10^{-2}} + B \end{aligned} \tag{4.1}$$

still from our tests one problem emerged: **the parameters A and B of this function are not fixed**. In fact due to how the sensor is attached to the structure that holds the actuator its calibration changes slightly every time we run the

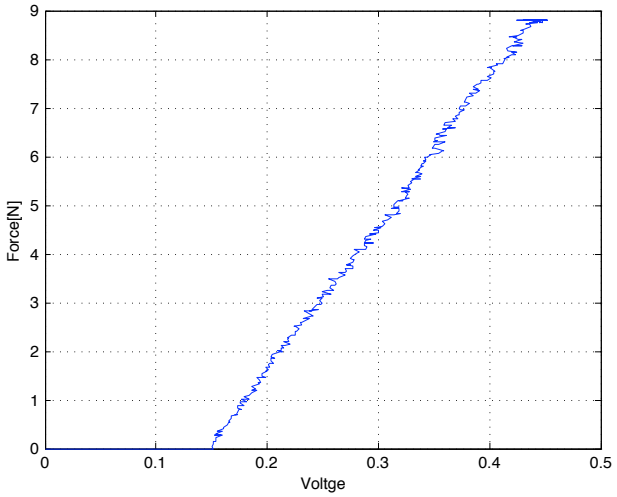


Figure 4.11: Force in function of voltage

program. This forced us to implement a manual calibration to calculate the parameters of the linear function every time we run the program, , so B it's calculated as the average of the values of tension measured when the actuator is just enabled and at the 0 position, and for A we calculate the average of the tensions measured when the actuator holds the position 2[cm] for a few seconds, furthermore we know the relation between position and force since we know the elastic constant of the spring ($K_s = 4.4N/cm$) and we can derive the formula:

$$A = \frac{V(max) - V(0)}{200} \tag{4.2}$$

we use 200 (the max position) in tenth of millimeter because that's how we receive it from the actuator and then make appropriate conversions when calculating the force.

$$F = \frac{V \cdot 4.4 \cdot 10^{-2}}{A} - B \tag{4.3}$$

There was a possible alternative way to obtain the force feedback is to send the command "get current consumption" to the actuator and then use this value to calculate the force feedback, multiplying the current by the force constant provided in the datasheet:

$$F = I \cdot 11.64[\frac{N}{A}]$$

but after running some tests with this method we can say that it's not reliable and that the current consumption does not vary linearly with the position.

4.2 Control module and software

The motion controller for the linear actuator is a faulhaber MCLM 3006s and it provides a PI controller to regulate the linear speed and a PD controller for the position. It's also possible to configure a current limitation. The linear actuator is also equipped with three analog Hall signals to measure position and speed. This module can be programmed directly (through the program provided by the producer for example) or using a C-programmable single board computer, in our case a Rabbit LP3500 (actually programmed using a slight variation of C called Dynamic C, designed by Rabbit). This last option employs a serial link communication with the computer, that in turn will allow the management of the communication with the master.

The use of a programmable board allowed us to perform a complex set of operation, to save the data and implement the force feedback, which required a number of operations due to technical difficulties that we are going to further explain in a specific section.

The LP3500 is a low-power single-board computer with built-in analog and digital I/O. As we can read in the data sheet it's "ideal for monitoring equipment or processes that are far-removed from a power supply, remote telemetry (RTUs), pipeline control and monitoring, well-head monitoring and use on mobile equipment such as refrigeration trucks."

Characteristics:

- Rabbit 3000 microprocessor operating at up to 7.4 MHz.
- 512K/128K static RAM and 512K/256K flash memory options.
- 26 digital I/O: 16 protected digital inputs and 10 high-current digital outputs provide sinking and sourcing outputs.
- 8 single-ended or 4 differential analog channels with Vcc monitoring option: 11-bit single-ended or 12-bit differential channels: the single 8-channel A/D converter chip has a resolution of 12 bits for differential measurements or 11 bits for single-ended measurements.
- 3 PWM outputs: the D/A conversion outputs are pulse-width modulated and scaled to provide an output from 0 V to Vcc (approx. 2.8 V).

- Six serial ports: all six serial ports operate in an asynchronous mode. An asynchronous port can handle 7 or 8 data bits. A 9th bit address scheme, where an additional bit is sent to mark the first byte of a message, is also supported.

  1 RS-485

  3 RS-232 (one 5-wire and one 3-wire or three 3-wire), jumper option for logic-level outputs; Serial Port E has a “listen” and “wake-up” capability

  1 logic-level serial interface for optional add-ons

  1 asynchronous clocked serial port dedicated for programming, Dynamic C uses the programming port to download and debug programs. The programming cable is used to connect the LP3500’s serial programming port to a PC serial COM port. The programming cable converts the RS-232 voltage levels used by the PC serial port to the CMOS voltage levels used by the Rabbit 3000.

- Battery-backed real-time clock.

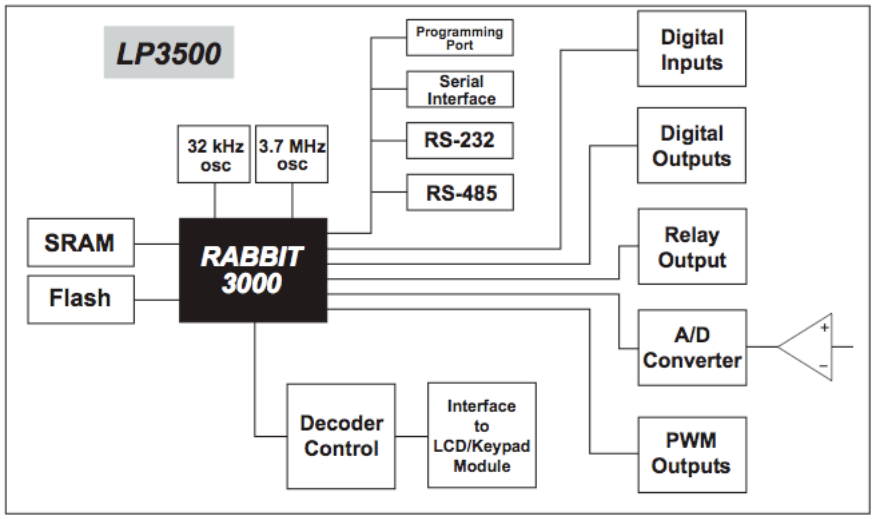


Figure 4.12: LP3500 subsystems

Chapter 5

Simulation

Before starting to actually implement the control algorithms on the architecture we decided to simulate them using simulink.

All the parameters from the linear actuator are in table 4.3, and we used them in both the master side (the probe) and the slave (the actuator-spring system). Another parameter we needed was the mass of both the probe and the actuator and to calculate this we used the displacement of the springs due to the weight of the object.

$$F = m \cdot g = k \cdot \Delta x \tag{5.1}$$

The elastic constant is $K_s = 440N/m$ for the slave (where we only have 1 spring) and $K_s \cdot 2 = 880N/m$ for the master probe where we have 2 springs. The two masses we calculated are:

- M_m= 0.22 Kg Master
- M_s= 0.154 Kg Slave

For the slave side we used a detailed model of the actuator derived from the classical equations of the mechanical and electrical parts, we used this detailed model also to test the performances with the two controllers (PD and PI). The to simulate the wave variables we used for the master the transfer function of the system with the parameters above for mass and spring. One problem we had for this simulation was that the damping parameter was unknown for both systems.

We attempted to do an identification with a classical grey-box approach, but because the data has quite a high sample time (as we see in the next chapter) it didn't work properly, so we chose a reasonable value to use in the simulation: $\mathbf{d} = 35$.

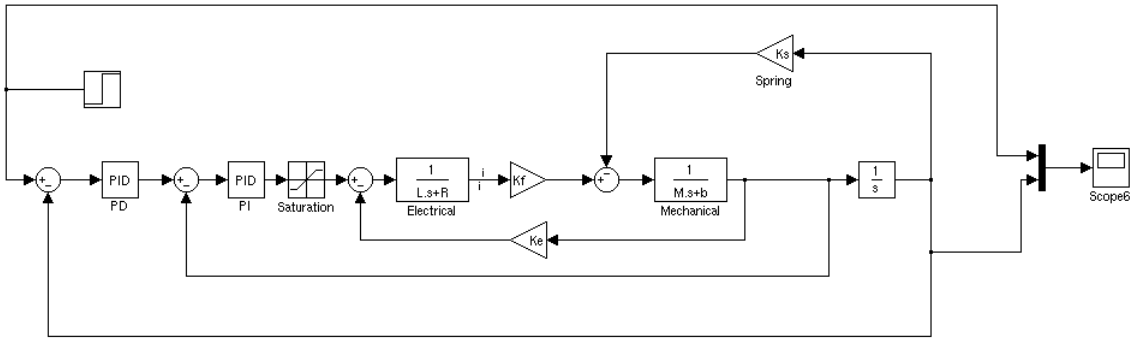


Figure 5.1: Simulink model of the linear actuator

The transfer function of the master will take force as an input and have position as the output:

$$W_m(z) = \frac{1}{M \cdot z^2 + d \cdot z + 2K_s} \tag{5.2}$$

and the poles of this transfer function are:

$$\begin{aligned} p_1 &= -198.48 \\ p_2 &= -28.79 \end{aligned} \tag{5.3}$$

First we can run a simulation of the step response of the slave system alone to verify its performances, we tuned the two controller to achieve better performances, even though in the real system the parameters are already set to an optimal value.

As we can see from figure 5.2 the performances are very good in simulation: we have almost no overshoot and the rising and settling time are under the limits that we wanted to achieve (see table 4.1).

We can now simulate the control architecture for the bilateral system, using for the master the transfer function in 5.2 and for the slave the scheme in figure 5.1 that takes position as its input and has again position in output. Then we multiply this output for the elastic constant of the spring in order to obtain the force. We'll insert in the communication channel a

- Constant Time delay T=0.5s

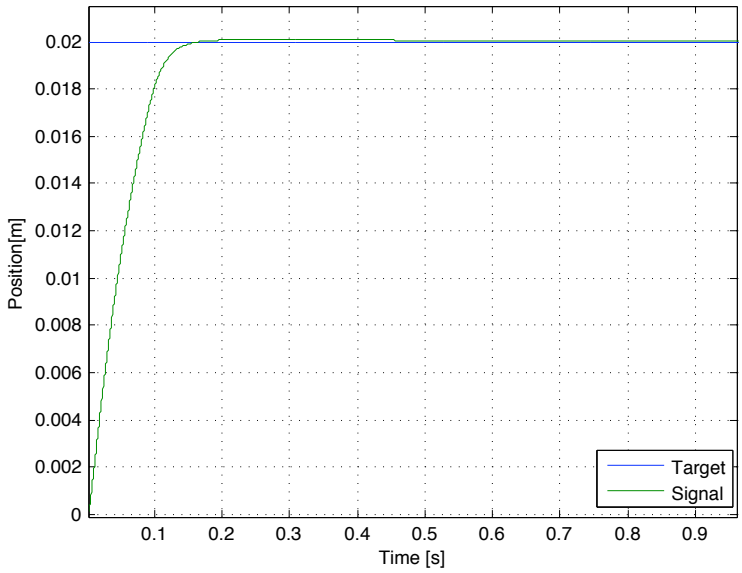


Figure 5.2: Position tracking simulation for the linear actuator

and also in the the communication block we'll have the two low-pass filters used as wave filters to reduce the wave reflection in the communication. We also added the option of having a slightly variable impedance for the environment by using a variable gain with small variance around the value of the elastic constant of the spring at the slave.

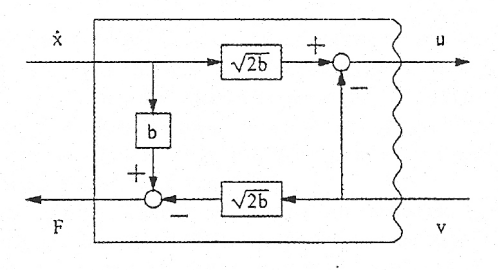


Figure 5.3: Signals transformation with the wave formulation

The structure of the wave variables encoding is represented using a simulink scheme in figure 5.4, where the second figure shows the "inside" of the pink box of the first figure. The block marked "wave filter" in fig. 5.5 contains also the time delay block. Figure 5.5 performs the encoding that we presented in the theory chapter about wave variables and corresponds to fig. 5.3.

For the simulation we tuned the *wave impedance b* and the *cut frequency* of the filters. For the wave impedance we tried to match the impedance of the environment (the spring) and tuned it around the value **b=440** adjusting it to obtain good performances. The cut frequency of the filters was chosen close to the inverse of the value of the time delay and then adjusted as well.

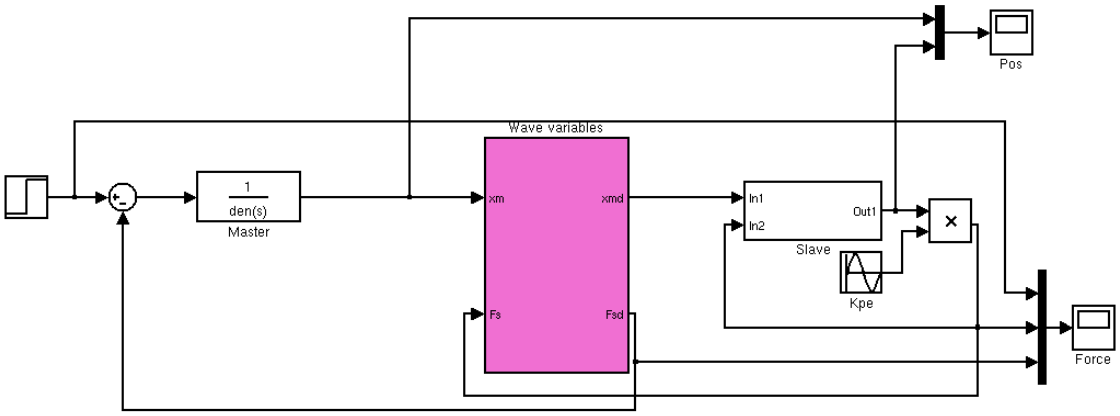


Figure 5.4: Simulink scheme of the wave variables encoding

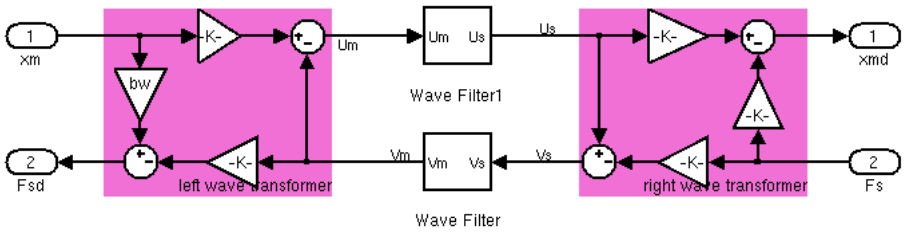
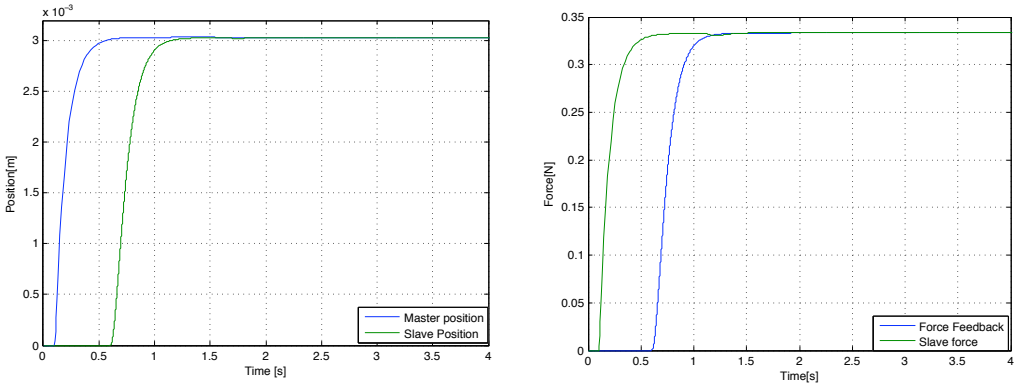


Figure 5.5: Simulink scheme of the wave variables encoding

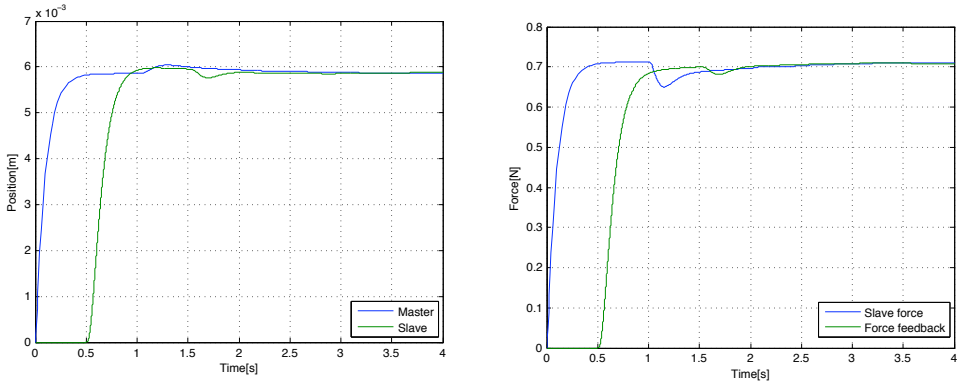
The force used as input signal of the system is a step of 1N, as we can see the position tracking gives really good performances with a time delay that is the sum of the one introduced by the channel and the rising time of the slave actuator. The force feedback tracks correctly the force felt by the slave, which is damped with respect to the input force.

We can also test with a variable environment impedance with average **b=440**, variance 0.1 and frequency of variation 0.5s. In this case we have small oscillations in both the position and force tracking that can be adjusted even further with a careful tuning of the control parameters to obtain a smoother behavior.



(a) Position (b) Force

Figure 5.6: Position and force tracking simulation for bilateral architecture with wave variables and constant environment impedance.



(a) Position (b) Force

Figure 5.7: Position and force tracking, variable environment impedance.

Chapter 6

Control/Communication

In this chapter we'll describe how we developed the software to actually realize a bilateral architecture: the control of the actuator, the force feedback, the communication with the master computer.

First of all we can make a distinction between the programs written in **Dynamic C** and the ones written in **C++**. Dynamic C is used to program the Rabbit processor, its functions are used to communicate with the motion controller of the actuator on one side and with the slave computer on the other. C++ manages the communication with the Rabbit processor and with the master computer.

The whole communication process can be summarized in the following steps:

1. The master "sets" a target position (for example using the probe).
2. The target position is sent to the "slave" computer through a local network or satellite link.
3. The "slave" computer receives a target position and sends it to the single board computer.
4. The single board computer sends the "go to target position" command to the control module.
5. The control module executes the commands and sends the required information to the single board computer, including the reading from the analog input of the sensor.
6. Following the previous path this information (properly converted) is carried back to the master computer to be elaborated.

6.1 Dynamic C functions

The functions that we load on the Rabbit processor handle the communication with both the actuator and the computer, the functions that manage the communication with the actuator are labeled **MotT** (MOTor Translation) and the ones that manage the communication with the computer are called **TcTm** (Tele Command TeleMetry) and everything is managed by a main function that calls the functions at the right moment. We'll relay on a buffer structure used to send data or save the data we receive that also contains informations on the size of the buffer and other useful information to decode it.

The **TcTm** receive function receives a buffer from the computer containing the target position that has to be passed to the actuator. After checking the integrity of the data by verifying the checksum it simply saves the target position in an appropriate variable and saves in another variable a number that we use to identify the kind of operation we want the actuator to perform (stop, run). The send function creates the buffer that we'll send to the computer, complete with start-end flags and checksum; the information we send is the actual position of the actuator and the reading from the analog input of the force sensor. We have specific functions to read the sensor and convert the value we obtain in volts according to the gaincode we chose to adopt.

For the **MotT** send and receive functions we need to consider the format of buffer required by the MCLM3006s controller. The list of commands that we will send and the relative response is the following:

Table 6.1: Commands list

Action	Command	Response
Go to position 0	HO	none
Enable	EN	none
Go to position x	LAx	none
Apply target position	M	none
Read position	POS	actuator position
Disable	DI	none

every send command has to be followed by the "CR" (carriage return) character and the response (when there is one) is always followed by "CR" and "LF" (Line Feed) characters. In the *send function* using a switch command we can create different buffers according to the specific operation we want to perform, which are enabling the actuator, moving it and reading the position at each sample time and finally disabling it. In order to send the target position in a format suitable for

the control module the function *itoa* is used to copy all the digits and the sign of the target position into string characters, the format required by the controller.

```
Buffer[Index ++] = 'L';
Buffer[Index ++] = 'A';

    itoa(Position, Buffer + Index);

    Bytes = 0; // number of bytes for position

    while(Buffer[Index] != 0x00)
    {
       Bytes++;
       Index ++;
    }
  Buffer[Index ++] = 0x0D;    //CR
```

In the *receive function* we receive from the actuator a buffer with the structure

```
position [CR][LF]
```

and to decode it we identify the cells before the [CR][LF] sequence containing the position and then use the *atoi function* to convert the string of the position into the number.

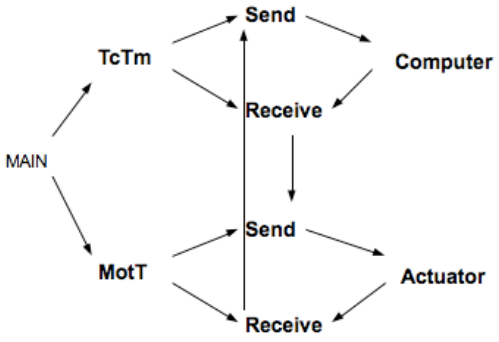


Figure 6.1: Dynamic C functions diagram

We verified that the rate (**sample time**) at which we are able to send command at the actuator is about $\mathbf{T_s} = 16[ms]$ with sometimes small differences of $\pm 1[ms]$.

6.2 C++ functions

The computer at the slave site has to manage the communication based on a UDP protocol with the master over the satellite channel and the serial communication to communicate with the control module. For this we have a series of functions all called by the main function in the proper order:

1. Initialize
2. UDP Receive
3. UDP Decode
4. Serial Receive
5. Serial Decode
6. Wave variable codification done directly in the main
7. Serial + UDP Code
8. Serial + UDP Send

In 1) we initialize the buffer to save the received data, initialize a data structure to save the information received and call a function that creates the UDP connection using the Winsock library.

Then before we start receiving data from the master we calculate the parameters to calibrate the force sensor as we explained in section 4.1.2, the actuator is enabled and stays at the 0 position for a couple of seconds, then moves at the position 2[cm] and holds it for another couple of seconds before returning to 0. This allows us to calculate the coefficients we'll use to convert volts in force.

After this we can start receiving data from the UDP connection with the master: the *receive* function reads the socket and saves the buffer in the appropriate structure we created, the *decode* function takes this buffer and after verifying the checksum reads the **incoming wave variable** and saves it in the corresponding variable.

Similarly the *serial* receive and decode functions will read the data coming from the control module via the serial link and decode the buffer to obtain the position of the actuator and the reading of the sensor.

At this point we can convert Volts→Force, apply the **wave filter** to the incoming wave, decode the wave variable to obtain the target position and encode the force feedback in a wave variable. Each operation takes a number of lines, we show for example the conversion of the target position (saved in the variable named si16_pu_Serial_Cmd_Pos) starting from the incoming wave named r2_piApMain_Us (r2_bcAux is a temporary variable):

```
// Target Position
r2_bcAux  = sqrt(2. / r2_bcWaveImpedance);
r2_bcAux *= r2_piApMain_Us;
r2_bcAux -= r2_bcFe / r2_bcWaveImpedance;

r2_bcAux  = (r2_bcAux < 0) ? 0 : r2_bcAux;

si16_pu_Serial_Cmd_Pos = - (INT16)(r2_bcAux * 1e4);
```

Finally in 7) we create the buffer that encodes the feedback information in a way compatible with the master (CRC16 checksum, flags...) and the buffer that we send to the control module with the target position with the suitable format for the TcTm function. We then send the buffers on the satellite link and serial link respectively.

The last thing we do is close all the connections (satellite and serial) and save our data in a file.

The scheme of the main function as described above is shown in the following lines of code:

```
int main()
{
  ... (constants definition)

   void_Init("COM1", 1234);
   printf("Connections opened !\n\n");
  do
  {
//UDP data receive
         void_UDP_Receive();

         //UDP data decoding
         void_UDP_Decoding(&Buffer_UDP);

   // Serial Data Receive
void_Serial_Receive();

// Serial Data Decoding
void_Serial_Decoding(&Buffer_Serial);

... (wave variables decoding and encoding)
```

```
// Serial and UDP Buffer Coding
void_Coding();

// Buffer Sending
void_Send();

  } while(!_kbhit());

  void_Close();
}
```

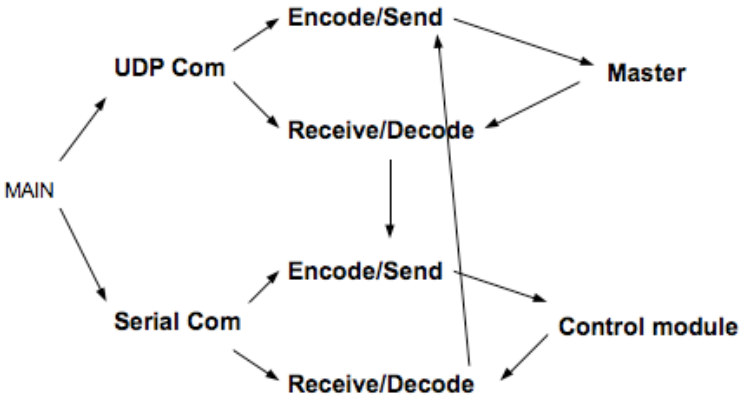


Figure 6.2: C++ functions diagram

Chapter 7

Experimental results

To test the wave variables control architecture we decided to position the master probe on a fixed support and program it to autonomously perform a specific movement. Instead of pushing and releasing the probe manually (the movement we want to apply is only in 1 DOF vertically) we decided to program the linear actuator in the probe to move up and down in a regular motion so that it should take a constant time to complete a movement up-down-up. This kind of movement when plotted over time corresponds to a cosine signal, instead if during the movement we change the period it takes to complete an up-down-up motion to go from a slower movement to a faster one, we will obtain a chirp signal. From now on we'll refer to the time that it takes to perform the up-down-up motion with the term *period* and we'll test different ones.

Our communication channel will be a satellite connection with a non-negligable time delay (the RTT is up to 1.5 seconds), a condition that is necessary to test the effectiveness of the wave variables control architecture. The parameters that we can vary in order to improve the results are the wave impedance and the gain of the wave filters.

We'll divide the tests in two sections: first with a *constant wave impedance* we'll try different values of the filters' gains in combination with both the cosine signal (and different periods) and the chirp signal. We used a wave impedance of **b=440**, this matches the value of the spring at the slave side, since in theory the best value is the one that matches the environment impedance. In the second section we'll consider just the cosine signal with the parameters that will give us the best results and *vary the wave impedance* to see how it affects the results.

7.1 Cosine signal

In this section we show the results obtained applying a cosine signal of 2cm of amplitude with different periods and for each one of this periods we tested the response with different frequencies of the wave filters. In table 7.1 are listed the different values combined in the tests.

Table 7.1: Filters frequency, cosine period

Cut Frequency	Period
0.5 Hz	1s
1Hz	2s
2 Hz	5s

First of all let's see the results with a small period (1s) and a gain for the wave filters of 1Hz:

As we can see the results are not ideal, the tracking is not perfect and also the force feedback presents some non-negligable errors and of course a time delay. We can deduce that the gain of the filters must be adjusted either with a smaller value or a bigger one, in fact the other two values (0.5Hz and 2Hz) were chosen to conduct the experiments after testing with other values, both smaller and bigger and seeing that to show significant results we just needed 0.5, 1 and 2Hz.

We can start by considering the difference between 0.5Hz and 2Hz with a period of 2s and see that the second frequency gives worse results than 0.5Hz, the signal oscillates more. We can also consider the smallest and the biggest periods tested to see how the velocity of the movement influences the performances. The figures show the results for the position tracking.

From this comparisons we can deduce that for the filters it's better to choose a smaller gain and that a slower motion improves the response of the slave. Let's see in more detail the results for a gain of 0.5Hz and a period of 2 seconds. We can see how the oscillations are limited in amplitude to about 2[mm] and the tracking of the trajectory is good. This results can be improved by an appropriate tuning of the wave impedance parameter as we'll see in a following section.

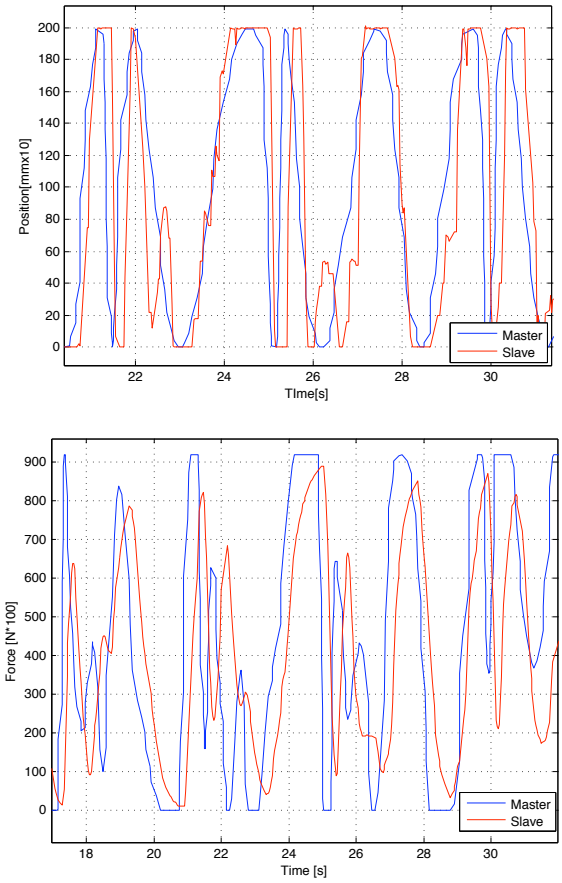
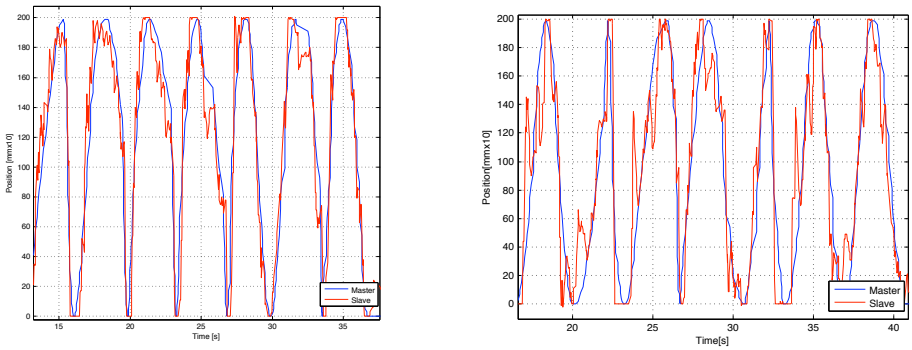


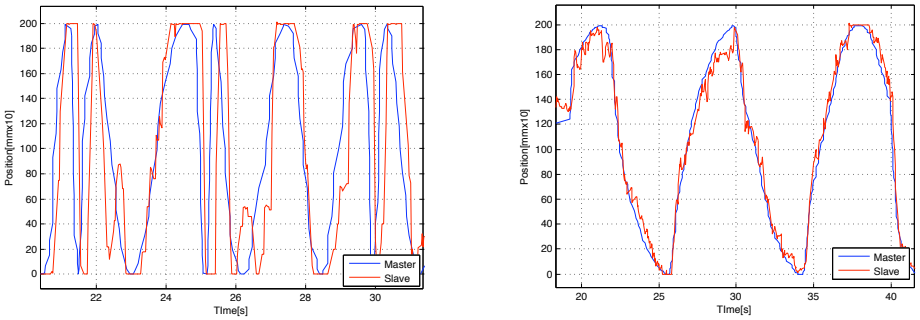
Figure 7.1: Position and force tracking for filter's gain 1Hz and cosine period 1s



(a) 0.5 Hz, period 2s

(b) 2 Hz, period 2s

Figure 7.2: Comparison 0.5Hz and 2Hz at 2s



(a) 1 Hz, period 1s (b) 1 Hz, period 5s

Figure 7.3: Comparison 1s and 5s at 1Hz

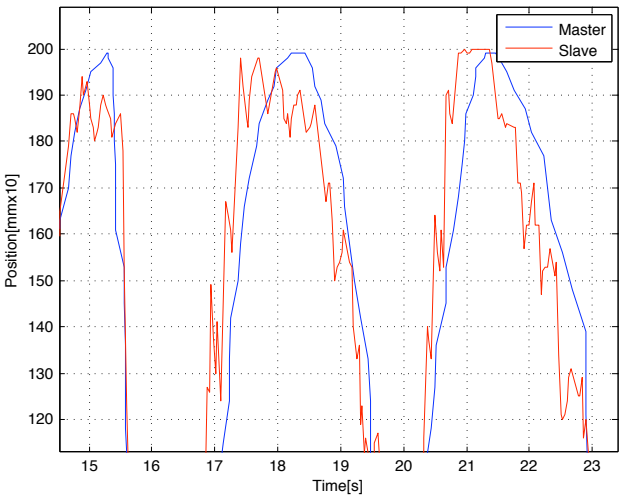
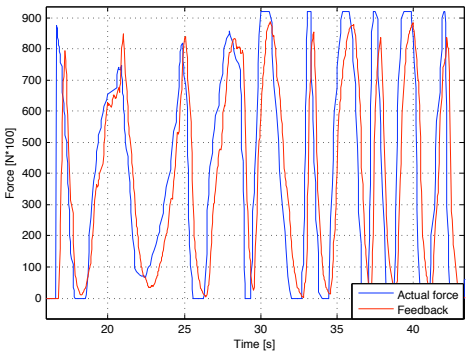


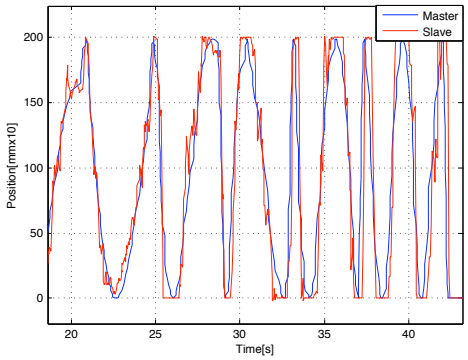
Figure 7.4: 0.5Hz gain and 2s period, detail

7.2 Chirp signal

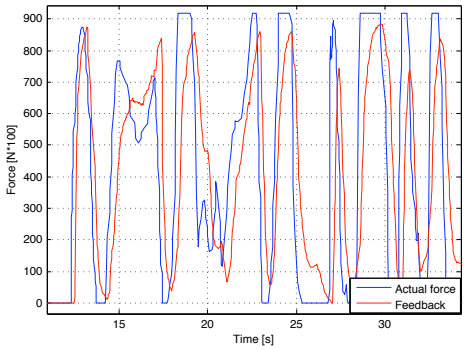
To verify the effects of a dynamic change of the period we applied a chirp signal and tested different gains in the wave filters. Also with this signal we can notice how the wave filters work better with a gain of 0.5Hz than with 2Hz, we still have oscillation and the result can possibly improve with a more accurate matching of the wave impedance.



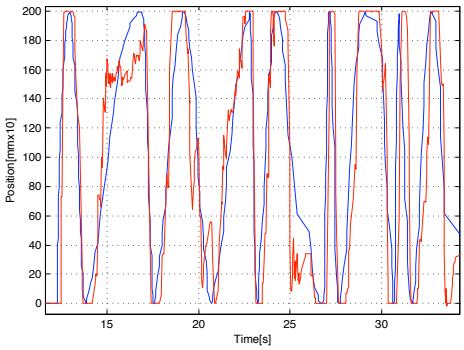
(a) Force



(b) Position



(c) Force



(d) Position

Figure 7.5: Force and position 2Hz chirp signal

7.3 Wave impedance variation

To test the adaptation of the wave impedance parameter we used the cosine signal with a period of 2s and the gain of the wave filters 0.5Hz, the value that appeared to give the best results. Since we already tested with b=440 we tested smaller and bigger values to try and improve the results.

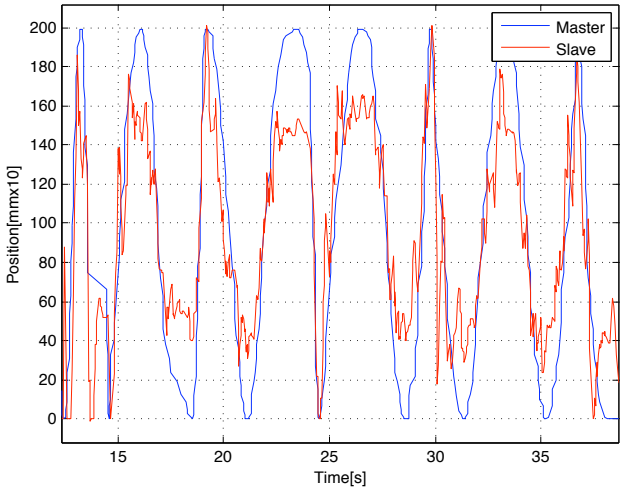


Figure 7.6: Position tracking with b=220

We can see in figure 7.6 that a wave impedance smaller than 440 gives worse results with bigger oscillation as we can see from figure 7.2(a). On the other hand a bigger value, b=550 or b=660, gives much better results. We tested with even bigger values of the wave impedance parameter, but there were no significant improvements and with much bigger values the results worsened. As we can see there are still some undesirable oscillation like the one at the 19th second in figure 7.8 that are probably due to an increased delay in the transmission and consequent misreading of the buffers, we'll discuss this in a bit more depth later.

We can see how also the force feedback provides good results being close to the applied force in figure 7.11.

Notice how for small values of b (like 220) not only the force feedback is strongly oscillating, but also amplified with respect to the actual force, which also presents strong oscillations (highly undesirable). On the other hand for b=660 we can see that the master is presented with a force feedback very close to the actual force, only slightly damped and delayed.

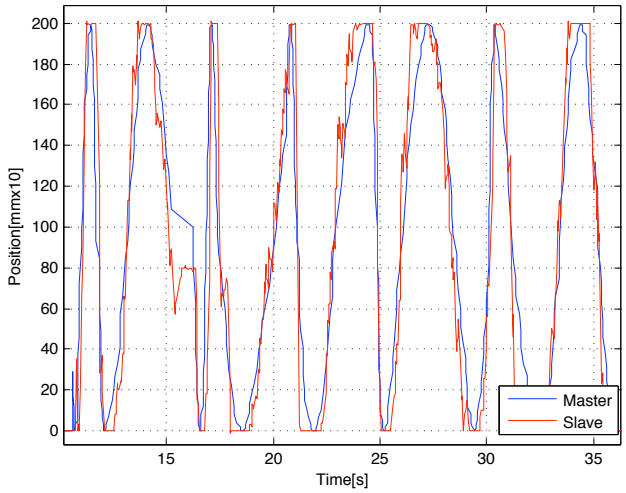


Figure 7.7: Position tracking with b=550

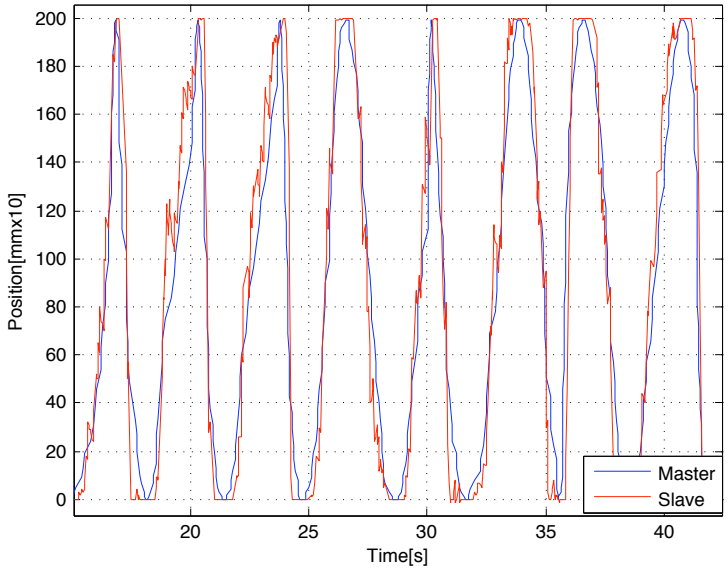


Figure 7.8: Position tracking with b=660

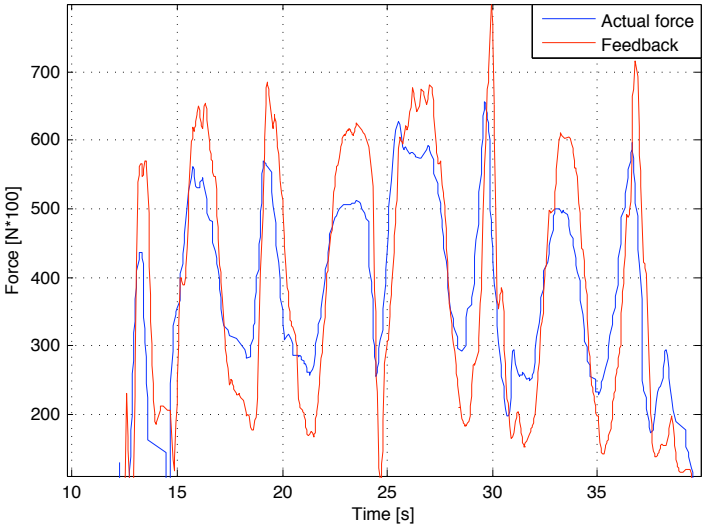


Figure 7.9: Force feedback with b=220

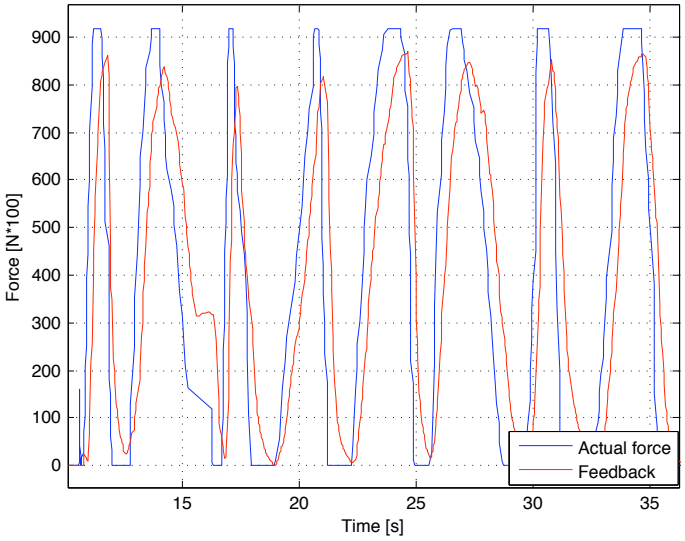


Figure 7.10: Force feedback with b=550

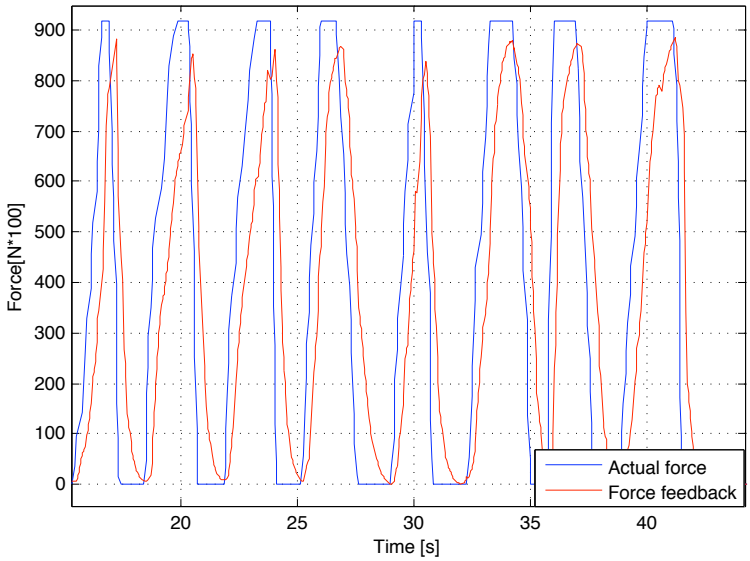


Figure 7.11: Force feedback with b=660

We want to take a moment to discuss the problem of imprecisions in the position tracking as seen in figure 7.8. This is evidently undesirable, and the probable cause is that some of the packets suffer from a higher time delay than others.

When the master probe performs the movement $\mathbf{1} \rightarrow \mathbf{5}$ we collect each sample time (each runtime of the program) a buffer containing an intermediate position and these buffers are stored in packets and sent to the slave over the communication channel. In figure 7.12 we represented in a simple way this process, with positions 1 to 5 and the relative packets named A, B, C, D, E.

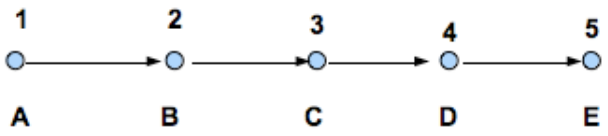


Figure 7.12: Positions 1,2,3,4,5 and corresponding packets

At the slave we ideally receive the packets in the correct order and move smoothly from 1 to 5, but since the delay introduced by the channel is not perfectly constant we may receive the packets in the wrong order and for example have:

$$A \rightarrow B \rightarrow D \rightarrow C \rightarrow E$$
$$1 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 5$$

evidently causing an oscillation in the position tracking. Dealing with a **variable time delay** is one of the problems that should be addressed in a future development of this work.

Chapter 8

Conclusions and future work

In this thesis we presented the study of a bilateral teleoperation architecture for tele-echography focusing on the actuator used on the end effector of the probe at the slave and how to realize the force feedback to the master.

After studying the theory used for every teleoperation architecture we focused on the problem of stability when the communication channel introduces a time delay connected to our particular application of *tele-echography*. As we've seen the presence of a time delay could cause stability problems and over the years researchers proposed a number of different methods to control bilateral architectures under such a condition. We found that many of these techniques were not suitable for our needs because for example they ensured stability only when setting the control parameters in a certain way and offered bad performances in terms of either position tracking of feeling of presence. What we needed for the tele-echography architecture was not only stability but also a good tracking and a force feedback that provided the human operator with a good feeling of the actual force that was applied on the patient, also we needed to deal with uncertainties in the environment impedance.

The control method that we chose to use is the wave variables formulation, that uses concepts of line theory applied to a teleoperation architecture. It encodes the signals we want to transmit over the channel (position and force) in a particular way that ensures that the system remains passive (and therefore stable) even if the channel introduces a time delay.

We tested this control technique on a real system composed of a probe controlled by the human operator that was used to control the motion of a linear actuator in 1 degree of freedom. The actuator applied pressure on a spring to simulate the human body and we measured the force felt using a force sensor, this measure was replicated on the master probe to provide the user with a feeling of presence. The goal was to help the operator in the tele-echography by giving him/her a feeling of the pressure applied on the patient just like in a regular echography.

We tested different settings of the control parameters until we obtain satisfying results in terms of position tracking and fidelity of the force feedback. The wave variables provide a good mean of control and allowed us to obtain good performances with relatively easy implementation and very few parameters to tune. The tuning of these parameters can be done without worrying about the stability of the system and only to improve the performances.

One of the possible developments for this work is the study of the **variable time delay** issue and how it's possibly linked to the small oscillations and imprecisions in the position tracking: as we discussed in the previous chapter not all the packets arrive with the same time delay and this causes of course errors in the position tracking. One possible solution to this problem is to assign a serial number to the packets we send and discard the ones that arrive in the wrong order: as long as the number of packets we discard is not too big this could ensure no further time delay and the data loss would not affect the tracking significantly.

Another development is the application of the end-effector studied to the complete tele-echography robot, since in this thesis we limited the study to the linear actuator we want to apply to the echography probe on its own.

Bibliography

[1] P. Arcara and C. Melchiorri. Control schemes for teleoperation with time delay: a comparative study. *Robotics and autonomous systems*, 2001.

[2] N. Chopra and M. W. Spong. Bilateral teleoperation over the internet: the time varying delay problem. *Proceedings of the American Control Conference*, 2003.

[3] K. Hashtrudi-Zaad and S.E. Salcudean. On the use of local force feedback for transparent teleoperation. *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on Robotics and Automation*, 1999.

[4] P.F. Hokayem and M.W. Spong. Bilateral teleoperation: an historical survey. *Automatica*, Vol.42, 2006.

[5] Dale A. Lawrence. Stability and transparency in bilateral teleoperation. *IEEE Transactions on robotics and automtion*, Vol 9(Num 5), 1993.

[6] R. Lozano, N. Chopra, and M. W. Spong. Passivation of force reflecting bilateral teleoperators with time varying delay. *Proceedings of the 8. Mechatronics Forum*, pages 24–26, 2002.

[7] G. Niemeyer, C. Preusche, and G. Hirzinger. *Springer Handbook of robotics*. Springer, 2008.

[8] G. Niemeyer and J.E. Slotine. Telemanipulation with time delays. *The international journal of robotics research*, Vol. 23:pg(873–890), 2004.

[9] A. Passenberg, C. Peer and M. Buss. A survey of environment-, operator-, and task-adapted controllers for teleoperation systems. *Mechatronics*, 20(Issue 7):Pages: 787–801, 2010.