

ISSN 0280-5316
ISRN LUTFD2/TFRT--5885--SE

Haptic Interface for a Contact Force Controlled Gantry-Tau robot

Iacopo Finocchi

Department of Automatic Control
Lund University
July 2011

| | | | |
|---|-------------------------------------|---|-------------|
| Lund University Department of Automatic Control Box 118 SE-221 00 Lund Sweden | | <i>Document name</i> MASTER THESIS | |
| | | <i>Date of issue</i> July 2011 | |
| | | <i>Document Number</i> ISRN LUTFD2/TFRT--5885--SE | |
| <i>Author(s)</i> Iacopo Finocchi | | <i>Supervisor</i> Anders Robertsson Automatic Control Lund, Sweden Rolf Johansson Automatic Control Lund, Sweden (Examiner) | |
| | | <i>Sponsoring organization</i> | |
| <i>Title and subtitle</i> Haptic Interface for a Contact Force Controlled Gantry-Tau robot. (Haptiskt gränssnitt för kontaktkraftreglerade robotar) | | | |
| <i>Abstract</i> <p>Robotic science points to integrate the robot in the normal life and operations, with the scope to help humans and improve their performance. Haptic is the field that tries to insert the sense of touch in the computer applications. It is easy to understand the modern interest to mix the robotic with haptic. One possible problem with this configuration could be the delay in the communication between the operator (master) and the manipulator (slave). This thesis studies the application of a haptic device to control a new prototype of industrial parallel robot, using a mathematical method (<i>Wave Variable Method</i>) to create a stable control, regardless the time delay. The model was tested with simulations and confirmed with practical experiments, utilizing also a normal serial robot. The versatility of haptic device has been used to create different control modes, selectable during functioning of the robot.</p> | | | |
| <i>Keywords</i> | | | |
| <i>Classification system and/or index terms (if any)</i> | | | |
| <i>Supplementary bibliographical information</i> | | | |
| <i>ISSN and key title</i> 0280-5316 | | | <i>ISBN</i> |
| <i>Language</i> English | <i>Number of pages</i> 89 | <i>Recipient's notes</i> | |
| <i>Security classification</i> | | | |

Contents

| | | |
|----------|---|-----------|
| 1 | Introduction | 7 |
| 1.1 | Previous work | 7 |
| 1.2 | Motivation of the work | 8 |
| 1.3 | Outline | 8 |
| 2 | Devices | 9 |
| 2.1 | Haptic | 9 |
| 2.1.1 | Phantom Premium A | 9 |
| 2.1.2 | OpenHaptics | 10 |
| 2.2 | Robotics | 10 |
| 2.3 | Gantry Tau PKM | 12 |
| 2.4 | Force Sensor | 13 |
| 2.4.1 | Gravity Compensation | 15 |
| 3 | Robot Theory | 17 |
| 3.1 | Kinematics | 17 |
| 3.2 | Denavit-Hartenberg Convention | 20 |
| 3.3 | Kinematics for Parallel Robots | 22 |
| 3.4 | Inverse kinematics | 24 |
| 3.5 | Calibration | 26 |
| 3.6 | Jacobian | 26 |
| 3.7 | Control theory | 29 |
| 3.7.1 | Impedance Control | 30 |
| 3.7.2 | Force control | 31 |
| 3.7.3 | Force/position parallel control | 32 |
| 3.7.4 | Hybrid control | 33 |
| 4 | Sensible Phantom Haptic Device | 35 |
| 4.1 | Mapping | 35 |

| | | |
|----------|--|-----------|
| 5 | Wave Variable Method | 39 |
| 5.1 | Basic Passivity Concepts | 39 |
| 5.2 | Wave Variables | 41 |
| 5.3 | Integrated Wave Variables | 42 |
| 5.4 | Single Channel | 43 |
| 5.5 | Multiple degrees of freedom system | 44 |
| 5.6 | Reflection | 45 |
| 5.7 | Wave Variable Simulation | 46 |
| 5.7.1 | Simple System | 46 |
| 5.7.2 | IRB140B and Gantry Tau | 47 |
| 6 | Analysis of softwares | 49 |
| 6.1 | ABB IRC5 Robot System | 49 |
| 6.2 | Opcom | 49 |
| 6.2.1 | Interface | 52 |
| 6.3 | Phantom Robot Connector | 52 |
| 6.3.1 | Startup | 52 |
| 6.3.2 | Digital Force Filter | 54 |
| 6.3.3 | A short description of the code structure | 54 |
| 6.4 | WinComm | 56 |
| 6.5 | Matlab and Simulink | 57 |
| 6.5.1 | Real Time Workshop | 57 |
| 6.6 | How to run | 57 |
| 7 | Simulink Model | 59 |
| 7.1 | Libraries | 59 |
| 7.2 | Choice of the Signals | 60 |
| 7.3 | Model | 61 |
| 7.3.1 | Force Reference (1) | 61 |
| 7.3.2 | TCP position and Velocity (2) | 61 |
| 7.3.3 | Wave Variables (3) | 63 |
| 7.3.4 | Reference Signals (4) | 64 |
| 7.3.5 | Impedance Parameters (5) | 65 |
| 7.3.6 | Delay (6) | 66 |
| 7.3.7 | Virtual Wall (7) | 66 |
| 7.3.8 | Velocity and Position Outputs (8) | 66 |
| 7.3.9 | Impedance Force controller (9) | 66 |
| 7.3.10 | Starting Parameters (10) | 67 |

| | |
|---|-----------|
| <i>CONTENTS</i> | 5 |
| 8 Simulations and Results | 69 |
| 8.1 Gantry Tau Robot | 69 |
| 8.1.1 Free Space | 70 |
| 8.1.2 Rigid Wall and Joystick Control | 70 |
| 8.2 IRB140B | 71 |
| 8.2.1 Free Space | 73 |
| 8.2.2 Flexible Wall | 73 |
| 8.2.3 Rigid Wall | 74 |
| 9 Final Remarks | 79 |
| A Simulink's function | 83 |

1. Introduction

This master thesis aims to develop a force control for a *Gantry Tau Parallel Kinematic Machine (PKM)*, working with the *Phantom[®] Premium A Haptic Device* [28] as external controller. It is important to clarify some terminology:

- *Parallel robots* are a particular type of manipulators, which obtain flexibility and fastness in the work, not using the common serial links construction. The negative aspect is the increasing of complexity for the kinematics calculation.
- *Haptics* is the science that tries to implement the sense of touch into computer applications, using the force or tactile feedback. This can be obtained by an haptic devices (e.g. Phantom Premium A). This science is gaining more attention in both industrial and biomedical areas, where the sense of touch for the operator is importance.

The PKM *Gantry Tau* robot is a prototype, created by a collaboration between the Automatic Control Department of the LTH University of Lund [7] and ABB [1], while the haptic device is furnished by a third part [6]. Various aspects have been considered and studied, for example the communication delay between the two devices, which leads the system to instability.

1.1 Previous work

Many articles have been written on the theory of parallel robots, master and slave robot communication and force control robot [15], [23], [25]. In this work, there is an attempt to mix all these aspects. A master thesis has already been conducted the Automatic Control Department on a haptic control for a serial robot [17]. Another master thesis has been written about the control of the Gantry Tau parallel robot [18] and was based on preliminars studies of Isolde Dressler [16]. Results obtained in these two works, have been particularly useful for this thesis. They created and developed libraries for Matlab & Simulink for parallel and serial robots, that allow to simplify calculations and models (e.g. kinematics, Jacobian, matrix inversion, etc).

1.2 Motivation of the work

Although there is a precedent work on teleoperation control with serial robot, there are no studies on the application of a Phantom haptic device on the Gantry-Tau parallel robot. Differently from other structures, the PKM manipulator offers higher mechanical qualities, a larger workspace zone and an incredible versatility. These represent interesting aspects for the industrial field, that will see in the next year an increment of the presence of parallel manipulators.// The implementation of a new "haptic sense" on an innovative structure as Gantry-Tau Manipulator, can create a very powerful structure and, hopefully, this thesis will try to lay the foundations for a new application field of haptic device.

1.3 Outline

Chapter 2 describes the devices used during this work, while chapter 3 presents the basics of the robotic's theory, with a description of the principal force controls. The connection modality between haptic device and real robot has a particular relevance and chapter 4 describes the mapping method. The wave variable method is the main part of this work and it is explained and validated with some simulations in chapter 5.// Working with many different software, it is necessary to give a description of their connections and their features. That is done is given in chapter 6. In chapter 7 the designed Simulink model is described, while chapter 8 shows the results of tests and simulation conducted with Gantry Tau and IRB140B robots. Finally, chapter 9 reports final considerations and conclusions of this thesis.

2. Devices

Since many different types of machines have been used a parallel (*Gantry Tau*) and a serial robot (*IRB140B*), an haptic device (Phantom Premium A) and a force sensor (*JR3100M40*), a general description of their characteristics is necessary. This chapter provides an overview of the main features of each device.

2.1 Haptic

Haptics is the science that tries to implement the sense of touch into computer applications, using the force or tactile feedback. This science is becoming increasingly popular, especially in the industrial and medical field, where the sense of touch for the operator is really important.

2.1.1 Phantom Premium A

The Phantom Premium A is a six degrees of freedom robot. It is composed of three links and at the end of the last, a pendant is installed (figure 2.1), to be used by the operator to position the *Tool Center Point* (TCP).



Figure 2.1: Phantom Premium A with pendant.

Only 3 of the 6 degrees of freedom can receive a force feedback, since only those 3 have a motor on their axis. In particular, the axes 1,2 and 3 could have a force control, while axes 4,5 and 6 use only a rotation sensor. The Phantom Premium A is connected to a machine, through a 110V amplifier with a PCI-card. The software *OpenHaptics*, provided with the device, gives the possibility to check, calibrate and control the robot.

2.1.2 OpenHaptics

In order to work with Phantom Premium A (as well as for any other device), it is necessary to choose an Application Programming Interface (API). In this work we used the one provided by *SensAble Technologies, Inc.* [28], the *OpenHaptics*. Being this interface made expressly for *SensAble* devices, it is not possible to work with robots of other brands. Other APIs could be chosen (e.g. third parts APIs, Open Source, etc.), but there are no motivations to change this interface, which was used in the previous projects developed in this department.

2.2 Robotics

The integration of the automatic devices into our life is becoming more evident. In modern homes, cars, trains, there are a lot of "*automatic*" components and objects, which help each of us in the normal tasks of each day. In the industrial field the use of robot to help the workers during their jobs, to increase and improve the production or to replace the human operator in dangerous situations, is present from many years. Today, with the development of control systems, robots have assumed a fundamental role in the production, especially serial robots (figure 2.2).

In the last years, nevertheless, parallel robots have gained crescent interest than in the past, thanks especially to the development of the control systems and of the construction materials. The main strength of parallel robots points are the bigger workspace (than a common serial robot) and their versatility, that makes this manipulator interesting also for small and medium industries.

One of the most sold parallel robots is *ABB IRB 360 FlexPicker* (figure 2.3) constructed with Delta configuration and used in food's industries to pick up and move objects from a conveyor belt. Its power is its velocity, impossible to obtain with a common serial robot. The actuator of this robot has a fixed orientation, so that is possible to increase the working speed. This means that this kind of manipulator has only 3 degrees of freedom. For all these types of robots, the acting force on TCP is transmitted along



Figure 2.2: ABB IRB 4400, an example of ABB serial robot.



Figure 2.3: ABB IRB 360 FlexPicker. It is the most sold PKM in the world today.

the arm as a pure axial force. This gives the possibility to use extremely light material, as, for example, carbon fiber. It is easy to understand that the possibility to act with high force, having low masses in movement are the best characteristics that a machine could ever have.// The researchers have worked trying to obtain a parallel kinematic robot that was possible to reconfigure with a large workspace and a prototype of the most promising non-symmetrical parallel kinematic robots was in the Automatic Control Department of Lund's University, so called *Gantry-Tau*.

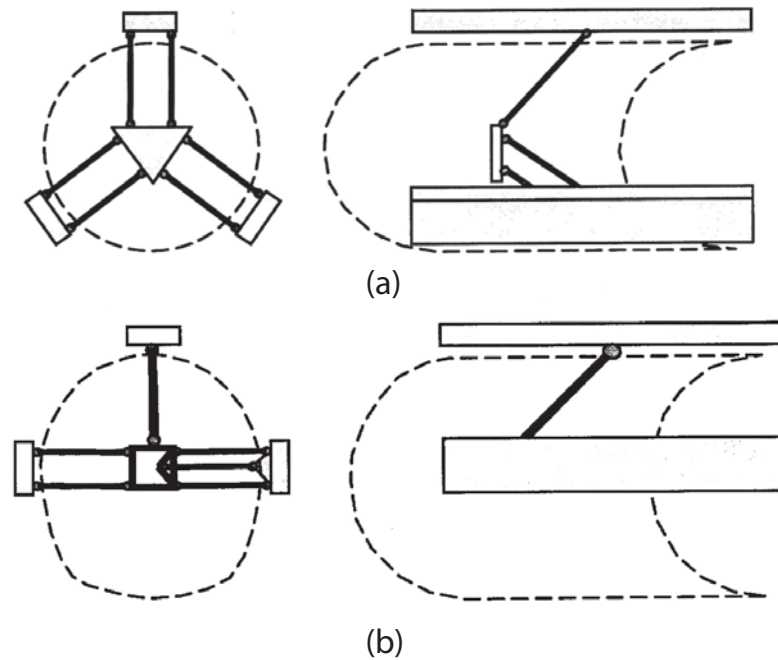


Figure 2.4: In (a), front and lateral view of workspace reachable with a PKM robot with a symmetrical Delta link configuration and linear actuators laid horizontally. In (b), Front and lateral view of a PKM robot with a non-symmetrical Delta link configuration and linear actuators laid horizontally. The workspace is more bigger than the symmetrical case.

2.3 Gantry Tau PKM

The robot with a non-symmetrical structure could seem worse with respect to others with a symmetrical disposition, since the isotropic structure is usually considered the optimal choice. In reality, the analysis of kinematics for a Delta structure shows that not all positions can be reached with a symmetrical disposition of links and joints. Hence a non-isotropic configuration is not only better, but also necessary.// The Gauntry Tau robot uses linear actuators instead of actuated arms of Delta robot, if high stiffness is required by the operations (i.e. drilling, cutting, milling , etc). It could be not obvious that, also in this case, the non-symmetrical configuration is better than the symmetrical one. In order to understand that, it is possible to do a kinematic study, but this result appears clearly observing figure 2.4

The Delta link structure, symmetrical and non-symmetrical, can be mounted

on three horizontal guides, as shown in figures 2.4(a) and 2.4(b), respectively. It is clear that the non-symmetrical configuration has a larger workspace, but both systems have a "hole" on the rightmost part. This is the consequence of a non-reconfigurable robot. It could be convenient to evaluate the possibility to construct a reconfigurable PKM, in order to obtain the same shape of the workspace on both sides (figure 2.5)

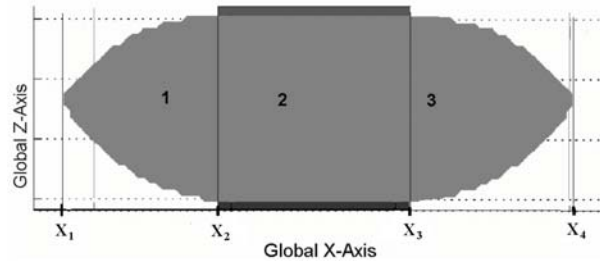


Figure 2.5: Cross-sectional workspace along YZ and XZ plane. When the robot is able to be reconfigured, it can reach an equal workspace in the left and right side.

There are different type of links configuration, considering the position of the three linear guides and the number of links connected with the manipulated platform. A lot of study has been carried out in order to obtain the best configuration and the right dimensioning of robot parts, considering stiffness and collision free workspace [21], [15]. The final result is shown in figure 2.6. This structure is a new version of the prototype present in the old robot laboratory, utilized for the previous works in the Automatic Control Department.

2.4 Force Sensor

The sense of touch is given to the robot by the force sensor. As explained in the next chapters, the robot is highly affected by the force measurement and then, the signals have to be transformed before being used in the controller.

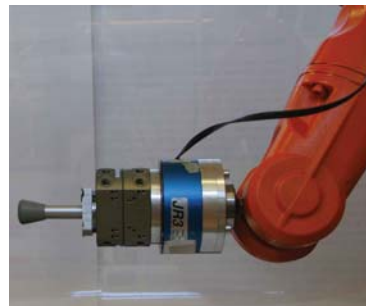
In this work the sensor *100M40* of *JR3* company [2] is used (figure $\text{\ref{Force:Sensor}(A)}$). It is connected to the mounting plate, before the end effector (figure $\text{\ref{Force:Sensor}(B)}$) and, hence, this latter is able to "read" information from the environment. However, the reading includes force contributions due to external factors (e.g. *gravity*, *centrifugal force*) and the sensor needs a "reset" operations, before starting to recognize different force signals.



Figure 2.6: The new prototype constructed in the Automatic Control Department and used for this work.



(a)



(b)

Figure 2.7: In (A), the force sensor *100M40* of *JR3* company, used in this work. In (b), an example of mounting of force sensor on serial robot.

2.4.1 Gravity Compensation

In order to avoid the disturbances due to gravity in the force signals, a Simulink program has been created at Automatic Control Department by the Phd student, *Andreas Stolt*. This program calculates a compensation for the weight of the sensor (or a generic end effector connected to it) during the normal functioning. The Simulink block is shown in figure 2.8

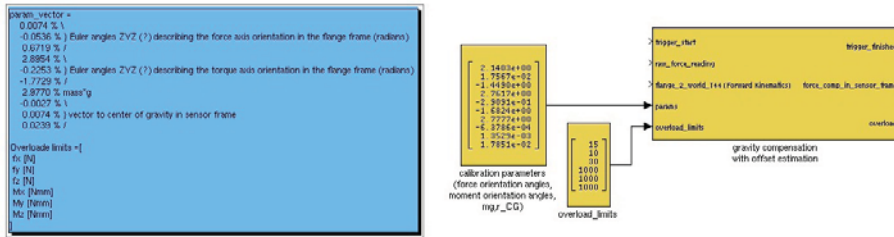


Figure 2.8: The Simulink block for the gravity compensation.

The huge amount of parameters necessary to work with this block requires a long initial procedure. The robot is moved in certain positions, where the values are read and saved. At the end, the output feeds the gravity compensator and, hence, the effect of the weight is removed from the Simulink controller.

3. Robot Theory

Robotics has the purpose of replacing workers during physical activities, helping them in dangerous operations and in their decisions. The last century has seen a big development of the robotic's theory and many books have been written on this subject (see [27]). Main aspects are reported in this chapter.

3.1 Kinematics

A general manipulator can be seen as a chain of links, connected each other by joints (that represents the degrees of freedom (DOF) of the robot), in such a way to form a kinematic link, which can be *closed* or *open*. Joints can be of two types: *prismatic* and *revolute*. The difference is the type of movement that they permit.// A rigid body (as a robot) is completely described in the space by a *position* and an *orientation*. Looking at the figure 3.1, let $O-xyz$ the reference frame, with x, y, z the unit vectors of the axes of the frame. The position of a O' point of the rigid body, expressed with respect to the reference frame is expressed by

$$o' = o'_x x + o'_y y + o'_z z; \quad (3.1)$$

where o'_x, o'_y, o'_z are the components of O' on the axes of the frame. In order to simplify the description of the orientation, it is useful to fix an orthonormal frame to the rigid body and describe the unit vectors of this new frame with respect to the reference frame. Let be $O'-x'y'z'$ the rigid connected frame, with x', y', z' the unit vectors. They are described, with respect to the reference frame by the following relationships:

$$\begin{aligned} x' &= x'_x x + x'_y y + x'_z z; \\ y' &= y'_x x + y'_y y + y'_z z; \\ z' &= z'_x x + z'_y y + z'_z z; \end{aligned} \quad (3.2)$$

Components of each unit vector are the *direction cosines* of the three orthonormal axes $O'-x'y'z'$ with respect to the reference frame $O-xyz$. Using

a more convenient notation, the three unit vectors of (3.2) can be written in the $[3 \times 3]$ matrix R , the, so called, *Rotation Matrix*:

$$R = \begin{bmatrix} x'^T x & y'^T x & z'^T x \\ x'^T y & y'^T y & z'^T y \\ x'^T z & y'^T z & z'^T z \end{bmatrix} \quad (3.3)$$

It's important to note the following properties, that are valid for each pair of unit vectors

$$(x')^T y' = 0 \quad x'^T x' = 1$$

Consequently, the R matrix turns out to be *orthogonal*

$$R^T R = R R^T = I \quad (3.4)$$

where I is the identity matrix of $[3 \times 3]$ dimension. Right multiplying both terms of (3.4), another important result is obtained

$$R^T = R^{-1} \quad (3.5)$$

The rotation matrix can be used to define the relative orientation of two frames, but also to describe the different representations of an element (i.e. a vector) with respect to different frames or as the matrix operator, which permits to rotate a vector by a fixed angle around an axis. Obviously, the rotation of an element can be done in more steps. The final orientation is obtained by the composition of rotation matrices and that is a very important property

$$R_2^0 = R_1^0 R_2^1 \quad (3.6)$$

where it is assumed as rule that the rotation matrix R_i^j describes the rotation of the i frame with respect to the j frame.// As said at the beginning of this section, a rigid body is completely described in the space by its position (3.1) and its orientation (3.2). Look at figure 3.1, P is an arbitrary point in the space, whom coordinates with respect to a reference frame $O_o - X_o Y_o Z_o$ are expressed by the vector p^0 and the coordinates with respect to another frame $O_1 - X_1 Y_1 Z_1$ are expressed by the vector p^1 .

Observing the figure it is possible to obtain the position of the point P as

$$p^0 = o_1^0 + R_1^0 p^1 \quad (3.7)$$

where the rotation matrix R_1^0 expresses the orientation of frame 1 with respect to frame 0 .// In order to obtain a compact representation of the

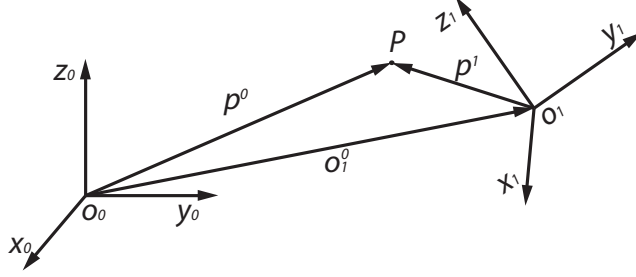


Figure 3.1: Representation of a vector in two different frames

existing links between the various coordinate of a point express with respect to different frames, it is useful to introduce the *homogeneous representation* of a generic vector p , of dimension $[3 \times 1]$, as the vector \tilde{p} , constructed adding a fourth unitary component

$$\tilde{p} = \begin{bmatrix} p \\ 1 \end{bmatrix} \quad (3.8)$$

Using this representation for vectors in (3.7), the coordinate transformation is describe by the $[4 \times 4]$ matrix

$$T_1^0 = \begin{bmatrix} R_1^0 & o_1^0 \\ 0^T & 1 \end{bmatrix} \quad (3.9)$$

which is called the *Homogeneous Transformation Matrix*. The transformation of a vector from the frame 0 to the frame 1 is described by one matrix, containing the rotation matrix of the frame 1 to the frame 0 and the translation vector from the origin of the frame 0 to the frame 1 . So the equation (3.7) becomes

$$\tilde{p}^0 = T_1^0 \tilde{p}^1 \quad (3.10)$$

For this work it is assumed as convention that the generic T_j^i matrix describes the transformation from the frame j to the frame i .

Considering the figure 3.2, the position and the orientation of the TCP is obtained by the T44 matrix

$$T_e^b(q) = \begin{bmatrix} n_e^b(q) & s_e^b(q) & a_e^b(q) & p_e^b(q) \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.11)$$

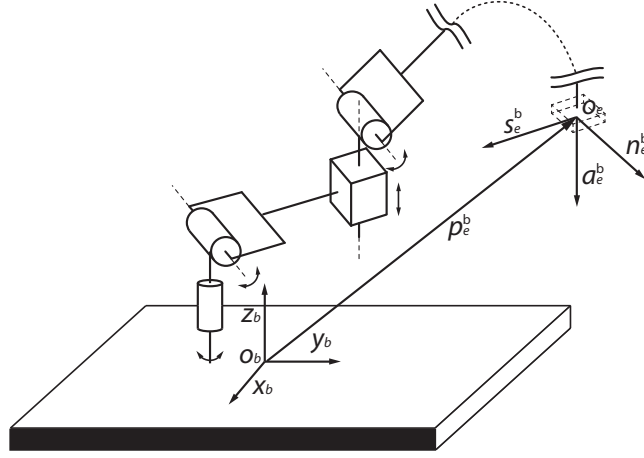


Figure 3.2: Position and orientation of the TCP

where q is a vector of $[nx1]$ dimension of the joint's variables; n_e, s_e, a_e are the unit vectors of the last frame of the manipulator and p_e^b is the position vector of the origin of the last frame with respect to the b frame (reference frame). For a generic robot, composed of n -links, the global transformation of coordinates (3.11) (fixing a frame on each link) from the frame b (assumed as the reference) to the last frame n is given by the recursive expression

$$T_n^0(q) = T_0^b(q_0)T_1^0(q_1)T_2^1(q_2) \dots T_n^{n-1}(q_n) \quad (3.12)$$

3.2 Denavit-Hartenberg Convention

In order to calculate the direct kinematics for an open chain manipulator, according to (3.12), it is necessary to decide a simple and unique method to define position and relative orientation between two consecutive links. That means that it is necessary to choose two frames, one for each link, and other to find the transformation of coordinates from one frame to the other. Considering the figure 3.2, illustrating two generic arms, i and $i - 1$ united by a joint i (joint of the arm i), it is possible to define the frame i using the so called *Denavit-Hartenberg Convention*:

1. the z_i axis is chosen along the axis of the joint $i + 1$;
2. the origin of the frame i , o_i , and of the frame $i - 1$ are detected by the intersection of, respectively, the z_i axis and z_{i-1} axis with the normal between z_i and z_{i-1} ;

3. the x_i axis is chosen along the normal between z_i and z_{i-1} , and direction "positive", from frame i to frame $i + 1$;
4. the y_i is taken on to complete a right-handed coordinate system.

When all the frames are chosen, the position and the orientation of the frame i with respect to to the frame $i - 1$, result completely defined by the following parameters:

- a_i distance of o_i from $o_{i'}$;
- d_i coordinate on z_{i-1} of $o_{i'}$;
- α_i angle, around x_i axis, between z_{i-1} and z_i , positive if counterclockwise;
- ϑ_i angle, around z_{i-1} axis, between x_{i-1} and x_i , positive if counterclockwise.

Two of the four parameters, a_i and α_i are constants, depending only on the geometry of the manipulator. Only one of the others two parameters is variable, depending on the joint used to connect link $i - 1$ with the link i :

- a revolute joint has ϑ_i as variable
- a prismatic joint has d_i as variable

Finally, the coordinate transformation from frame i to $s - 1$ can be obtained by the following steps, represented in figure 3.2:

- It is chosen a frame coincident with frame $i-1$, and it is moved of d_i along z_i , rotating ϑ_i , around z_{i-1} ;
- Now the frame is superimposed on the frame i' . It is moved a_i along $x_{i'}$, rotating α_i , around $x_{i'}$;
- The global homogeneous transformation, from i to $i - 1$, results:

$$T_i^{i-1}(q_i) = \begin{bmatrix} c_{\vartheta_i} & -s_{\vartheta_i}c_{\alpha_i} & s_{\vartheta_i}s_{\alpha_i} & a_i c_{\vartheta_i} \\ s_{\vartheta_i} & c_{\vartheta_i}c_{\alpha_i} & -c_{\vartheta_i}s_{\alpha_i} & a_i s_{\vartheta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.13)$$

where the values $c_{\vartheta_i}, s_{\vartheta_i}$ are the compact form of respectively, cosine and sine.

3.3 Kinematics for Parallel Robots

The relation (3.13) is only valid for a serial open-chain robot, hence it is applicable only for the last two frames of Gantry Tau (i.e. for the 4th and 5th DOF). The forward kinematics problem can be divided in two steps:

1. Find the relation between the orientation and the position of the mounting plate and the position of the three charts;
2. Find the relation between the mounting plate and the *Flange*.

For the second point, Denavit-Hartenberg convention can be applied, in order to obtain the homogeneous matrix between flange and mounting plate. For the first point, instead, the calculation is not so easy. In fact the forward kinematics for a parallel robot is more complex than for a serial robot, whereas the opposite is valid for the inverse kinematics (*duality* between parallel and serial robot). In figure 3.3 it is shown that the calculation of the position of the end effector of Gantry Tau is not univocal and depends on the type of structure adopted for the robot.

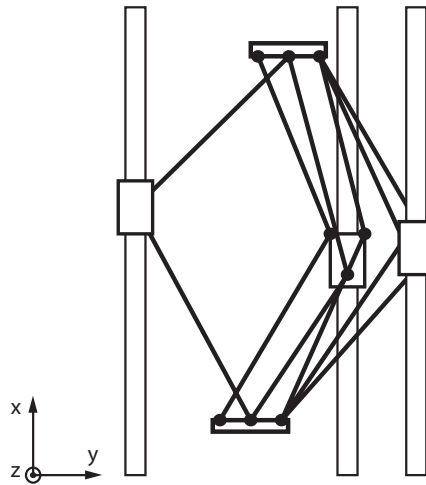


Figure 3.3: Top view of the Gantry-Tau robot. Two Different kinematics solutions for the same forward kinematics problem.

As previously stated, this work is based on Gantry Tau robot model (figure 2.6), which has three prismatic joints, mounted on three linear carts and connected to the mounting plate by three different link clusters, containing respectively 1, 2 and 3 links (see figure 3.4).

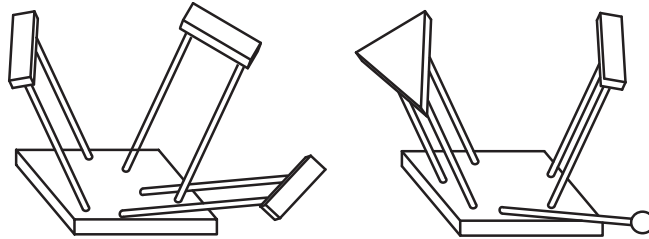


Figure 3.4: The mounting plate with a symmetrical, on the left, and a symmetrical, on the right, configuration of links.

A solution for forward kinematics was found in [15]. Referring to figure 3.5, the first step is the choice of two coordinate frames, the global frame and the mounting plate frame. This latter appears only translated, with respect to the global frame; in fact, no rotation is possible for a 3DOF PKM. Let L_x represent the length of an arm and q_x the position, with respect to the global frame of a cart.

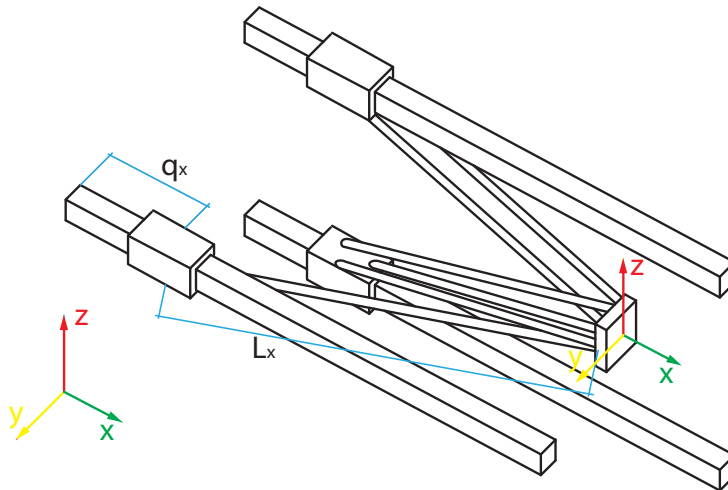


Figure 3.5: The base frame and TCP Frame of the Gantry-Tau robot. No reorientation is allowed for the TCP

3.4 Inverse kinematics

As said above, the inverse kinematics of the Gantry-Tau with one triangular link are relatively simple to find analytically. Considering the kinematic configuration of figure 3.5, the coordinates Y_1, Y_2, Y_3 and Z_1, Z_2, Z_3 of the linear charts are fixed, while their positions q_1, q_2, q_3 in the x direction (with respect to the global frame) are variable. Let x, y, z be the coordinates of the TCP and let (x_a, y_a, z_a) , (x_d, y_d, z_d) and (x_f, y_f, z_f) the coordinates of the platform points A, D and F respectively, as defined in figure 3.6. Those coordinate express the position of the mounting plate relative to the TCP origin, but in the global frame, using the angle α .

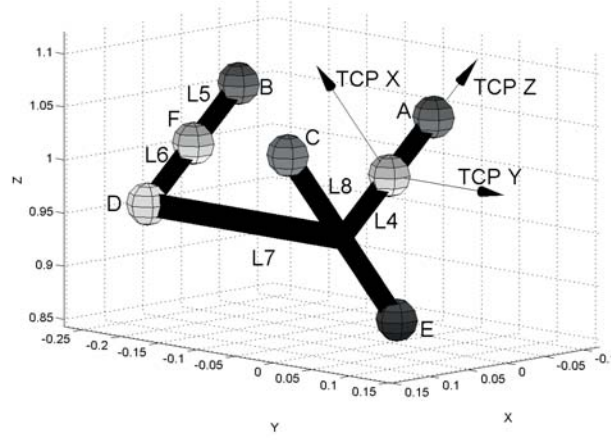


Figure 3.6: The Manipulated platform. The last mounting plate of Gantry-Tau is different, but this is the first configuration used for calculations.

If the coordinates of the points A, D and F are known, then the solutions for the inverse kinematics are

$$\begin{aligned}
 q_1 &= x + x_f \pm \sqrt{L_1^2 - (y_1 - (y + y_f))^2 - (z_1 - (z + z_f))^2} \\
 q_2 &= x + x_a \pm \sqrt{L_2^2 - (y_2 + y_{offs} - (y + y_a))^2 - (z_2 - (z + z_a))^2} \\
 q_3 &= x + x_d + S \sqrt{L_{3d}^2 - (y_3 - y_{offs} - (y + y_d))^2 - (z_1 - (z + z_d))^2}
 \end{aligned} \tag{3.14}$$

where: the variable S can be ± 1 and is introduced in the calculation of α ; L_{3d} is the length of the link in parallel with the triangular pair; y_{offs} is an offset to consider the width of the carts in the y direction. In order to solve the equations (3.14), α is calculated first by

$$\cos\alpha = \frac{-z_d}{L_4} \quad (3.15)$$

where L_4 is the z coordinate distance between the TCP and the points C, D and E, in the coordinate frame. A corresponding function can be found for the global frame

$$\cos\alpha = \frac{z + z_d - z_3}{\text{sqrt}(x + x_d - q_3)^2 + (z + z_d - z_3)^2} \quad (3.16)$$

Observing that

$$L_{3d}^2 = (x + x_d - q_3)^2 + (y + y_d - (y_3 - y_{offs}))^2 + (z + z_d - z_3)^2 \quad (3.17)$$

and combining equation(3.17)-(3.15), it is possible to obtain a solution for z_d depending only on know variables, i.e.

$$z_d = \frac{L_4(z_3 - z)}{L_4\sqrt{L_{3d}^2 - (y + y_d - (y_3 - y_{offs}))^2}} \quad (3.18)$$

Once calculated z_d , the α is obtained by equation(3.15)

$$\alpha = -\text{Scos}^{-1}(-z_d/l_4)$$

The inverse kinematics are given by the equation(3.14), after a α rotation of the mounting plate around y axis

$$R_y = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix} \quad (3.19)$$

Finally, the coordinates of the platform points in the global frame are given by

$$\begin{aligned} [x_a, y_a, z_a] &= R_y[0, 0, L_5 - L_4] \\ [x_b, y_b, z_b] &= R_y[0, -L_7, L_5 - L_4] \\ [x_c, y_c, z_c] &= R_y[L_8, 0, -L_4] \\ [x_d, y_d, z_d] &= R_y[0, -L_7, -L_4] \\ [x_e, y_e, z_e] &= R_y[-L_8, 0, -L_4] \\ [x_f, y_f, z_f] &= R_y[0, -L_7, L_6 - L_4] \end{aligned} \quad (3.20)$$

3.5 Calibration

Another important issue is the calibration of the robot. In fact, the kinematic model presented before assumes that all the linear actuators are mounted perfectly in parallel. This is almost impossible to achieve, if it is not use an expensive calibration equipment. Therefore, a more interesting way is consider the non-perfect montage during the calibration, modifying the model's parameters. Many studies were conducted on this argument and it can be solved in several manners (e.g. applying constrain on the end effector or on the actuators [13], inserting extra sensor on the robot [20]). A new innovative method, working with a vision based system, is presented in [12]. Differently of most of the robot calibration procedures, it does not need any priori knowledge on the robot+sensor system, providing a closed-form solution.

3.6 Jacobian

The relation equation(3.9) gives the position and orientation of the end effector of a generic manipulator of n degrees of freedom, as a function of $q = [q_1, ,q_n]$, the vector of joint variables. Varying q , the position and the orientation are modify. It is important to be able to valuate the relation between the velocity of the end effector (linear and rotational) and the velocity of the joint variables equation(3.21), in order to calculate the behavior of the robot.

$$\dot{p} = J_q \dot{q} \dot{\omega} = J_o \dot{q} \quad (3.21)$$

or in a more compact form

$$v = \begin{bmatrix} \dot{p} \\ \omega \end{bmatrix} = J(q) \dot{q} \quad (3.22)$$

that is the equation of *differential kinematics* of a manipulator. The J matrix is called *geometric Jacobian* and it has dimension $[6 \times n]$, where n are the degrees of freedom of the robot.

$$J = \begin{bmatrix} J_p \\ J_o \end{bmatrix} \quad (3.23)$$

The geometric Jacobian is function of the kind of joints used in the robot and it can be calculated easily, referring to fig. and using this result

$$\begin{bmatrix} J_p \\ J_o \end{bmatrix} = \begin{cases} \begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix} & \text{for a prismatic joint} \\ \begin{bmatrix} z_{i-1}x(p - p_{i-1}) \\ 1 \end{bmatrix} & \text{for a revolute joint} \end{cases} \quad (3.24)$$

If the position and orientation of the end effector are expressed with respect to a minimal number of parameters, it could be useful to calculate the Jacobian with a derivative operation of the direct kinematic functions of the joint variables. The end effector velocity can be describe as the sum of two contributions, the translation velocity and the rotation velocity. The first one is expressed as the time derivative of p , the vector that describe the distance of the frame with respect to the reference (base) frame. For the second velocity it is necessary to choose a minimal representation (i.e. Euler angles 3.7), in terms of three variables ϕ , that is function of the joint variables q . The time derivative of ϕ , in general, it is different with respect to the angular velocity defined in (3.21) and its calculation is not easy

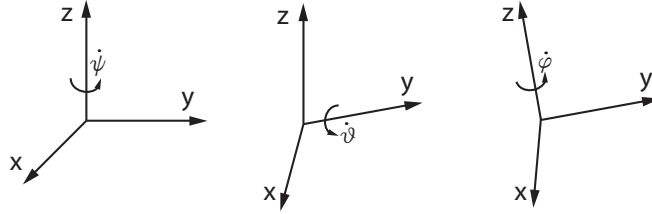


Figure 3.7: Rotational velocity of Euler angles ZYZ in current frame

Finally, it is possible to define the following differential kinematic expression

$$\dot{x} = \begin{bmatrix} \dot{p} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} J_P(q) \\ J_\phi(q) \end{bmatrix} = J_A(q)\dot{q} \quad (3.25)$$

where the $J_A(q)$ is the *analytic Jacobian*

$$J_A(q) = \frac{\partial k(q)}{\partial q} \quad (3.26)$$

Generally, the J and the J_A are different and, as a general law, the geometric Jacobian will be used when it is necessary to refer to quantities of

evident physical meaning, while the analytic Jacobian will be prefer where it is necessary to work with differential quantities of variables defined in the operational space.

It is shown that Jacobian refer the velocity of the joints to the velocity of the end effector. During real process it is common to use the *inverse* of the Jacobian, in order to obtain the joint velocities for a certain velocity of the end effector. The inversion of a matrix could be a problem if the matrix is not *square* and with a *full rank*. That happens when the manipulator is *redundant* and when it is at a singular configuration (i.e. J contains linearly dependent equations). In those cases the solution is not easy and new methods must be used (e.g. *damped least-squares inverse*, *pseudo inverse Jacobian*). Solutions for the inversion of the Jacobian are proposed in the theory ([27]).

The Gantry Tau has *5DOF* and it is possible to calculate the velocity of the flange with respect to the flange frame, using 6 variables, three for the translation velocity (v_x, v_y, v_z) and three for the angular velocity ($\omega_x, \omega_y, \omega_z$)

$$\begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} \dot{j}_{11} & \dot{j}_{12} & \dot{j}_{13} & \dot{j}_{14} & \dot{j}_{15} \\ \dot{j}_{21} & \dot{j}_{22} & \dot{j}_{23} & \dot{j}_{24} & \dot{j}_{25} \\ \dot{j}_{31} & \dot{j}_{32} & \dot{j}_{33} & \dot{j}_{34} & \dot{j}_{35} \\ \dot{j}_{41} & \dot{j}_{42} & \dot{j}_{43} & \dot{j}_{44} & \dot{j}_{45} \\ \dot{j}_{51} & \dot{j}_{52} & \dot{j}_{53} & \dot{j}_{54} & \dot{j}_{55} \\ \dot{j}_{61} & \dot{j}_{62} & \dot{j}_{63} & \dot{j}_{64} & \dot{j}_{65} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \\ \dot{q}_5 \end{bmatrix} \quad (3.27)$$

where j_{ij} is the ij component of the J matrix. It is evident that Jacobian is not square and hence it is not invertible. However, since it has full rank, a pseudo inverse of $J(q)$ could be achievable. Anyway, it is not useful do this calculation. In fact the communication libraries for the Gantry-Tau, developed by I.Dressler, are based on the ABB IRC5 Robot System, that is built for 6DOF ABB serial robots. In order to implement kinematics of the parallel robot, the addition of one 6th imaginary joint is necessary. The result is an invertible $[6 \times 6]$ jacobian matrix

$$\begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} \dot{j}_{11} & \dot{j}_{12} & \dot{j}_{13} & \dot{j}_{14} & \dot{j}_{15} & \dot{j}_{16} \\ \dot{j}_{21} & \dot{j}_{22} & \dot{j}_{23} & \dot{j}_{24} & \dot{j}_{25} & \dot{j}_{26} \\ \dot{j}_{31} & \dot{j}_{32} & \dot{j}_{33} & \dot{j}_{34} & \dot{j}_{35} & \dot{j}_{36} \\ \dot{j}_{41} & \dot{j}_{42} & \dot{j}_{43} & \dot{j}_{44} & \dot{j}_{45} & \dot{j}_{46} \\ \dot{j}_{51} & \dot{j}_{52} & \dot{j}_{53} & \dot{j}_{54} & \dot{j}_{55} & \dot{j}_{56} \\ \dot{j}_{61} & \dot{j}_{62} & \dot{j}_{63} & \dot{j}_{64} & \dot{j}_{65} & \dot{j}_{66} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \\ \dot{q}_5 \\ \dot{q}_6 \end{bmatrix} \quad (3.28)$$

This new imaginary joint gives the possibility to invert the Jacobian, but it also takes new problems and it necessary to pay attention to its reorientation. However, considering the particular configuration of Gantry-Tau (the

impossibility of the flange to rotate with respect to the base) allows to simplify the expression of J and, consequently, the inversion problem. In fact, the Jacobian matrix can be "divided" in two parts: one for the translation part and one to consider the rotational part. Since there is no reorientation, this second part shall be zero:

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} j_{44} & j_{45} & j_{46} \\ j_{54} & j_{55} & j_{56} \\ j_{64} & j_{65} & j_{66} \end{bmatrix} \begin{bmatrix} \dot{q}_4 \\ \dot{q}_5 \\ \dot{q}_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (3.29)$$

3.7 Control theory

In the precedent sections have been considered problems connected with description and modeling of manipulators. Now the focus shifts on the control theory, and, particularly, on the different kind of control's laws available for robots. The control problem could be divided in two parts, considering the type of the work: the control of the movement in a **free workspace** and in presence of **interaction with the environment**. In the joint space, the equation of the dynamic model of a manipulator is

$$B(q)\ddot{q} + C(q, \dot{q})\dot{q} + F_v\dot{q} + F_s \text{sgn}(\dot{q}) + gq = \tau - J^t(q)h \quad (3.30)$$

where q is the $[nx1]$ vector of joint's variables, $B(q)$ is the $[n \times n]$ inertia matrix, $C(q, \dot{q})$ is a $[n \times n]$ matrix of the centrifugal and Coriolis terms, F_v is a diagonal $[n \times n]$ matrix of the viscous friction, $f_s = F_s \text{sgn}(\dot{q})$ is the simplified model (Coulomb friction) of static friction (F_s is the diagonal matrix $[n \times n]$ and $\text{sgn}(\dot{q})$ is a $[nx1]$ vector), gq is $[nx1]$ vector relative of the contribute of gravity force, τ contains the torque of each actuators and, finally, $J^T(q)h$ shows the torque due to contact force (J^T is the Jacobian and h represents the vector of forces and momentums applied from he robot to the environment). In the absence of external forces on the end-effector and, for simplicity, not considering the static friction, the equation (3.30) becomes

$$B(q)\ddot{q} + C(q, \dot{q})\dot{q} + F_v\dot{q} + gq = \tau \quad (3.31)$$

Naturally, for the control in a free workspace (i.e. PD control with compensation of gravity, inverse dynamic control, robust control), it will use the equation (3.31) [27], while to consider the external forces will be necessary the equation (3.30). In this work, proofs have been conducted using an *Impedance control*, a *Force Control* and a *Hybrid control*.

3.7.1 Impedance Control

Impedance control is one of the most used control, since it is extremely versatile and usable for different typologies of manipulators [19]. It is an inverse dynamic control on the operative space . Considering the equation (3.30), it is applied the control law

$$u = B(q)y + n(q, \dot{q}) + J^T(q)h \quad (3.32)$$

where u is

$$n(q, \dot{q}) = C(q, \dot{q})\dot{q} + F_v\dot{q} + gq \quad (3.33)$$

Under the action of equation (3.32), the robot is described by

$$\ddot{q} = y - B^{-1}(q)J^T(q)h \quad (3.34)$$

that shows a non-linear term due to the contact force. The y terms is chosen as

$$y = J_A^{-1}(q)M_d^{-1}(M_d\ddot{x}_d + K_D\dot{\tilde{x}} + K_P\tilde{x} - M_d\dot{J}_A(q, \dot{q})\dot{q} - h_A) \quad (3.35)$$

where M_d, K_D and K_P are definite-positive diagonal matrices, $\tilde{x} = x - x_d$ ($x_d = \text{desired position of end-effector}$) and J_A represents the analytic jacobian. Substituting equation (3.35) in equation (3.34), remembering

$$\ddot{x} = J_A(q)\ddot{q} + \dot{J}_A(q, \dot{q})\dot{q} \quad (3.36)$$

and under the conditions of a not redundant manipulator in a not singular position, it is possible to get

$$M_d\ddot{\tilde{x}}_d + K_D\dot{\tilde{x}} + K_P\tilde{x} = h_A \quad (3.37)$$

The term h_A in equation (3.37) is the equivalent force vector defined in the operative space. It is clear that for this kind of control an reliable measure of external forces is necessary, hence an accurate sensor on the end effector shall be mounted. In this work it is be a sensor from *JR3, Inc.* [2]. Impedance control analyzes the system and its interaction with the environment, as *Spring-Mass-Damper* problem.

Referring to figure 3.8, the general equation for this kind of system is

$$\ddot{x} = M\ddot{x} + D(x - \dot{x}_d) + K(x - x_d) \quad (3.38)$$

The matrices M, D and K (corresponding to M_d, K_D and K_P in equation 3.35 characterize the system completely. Their choice is the task to solve

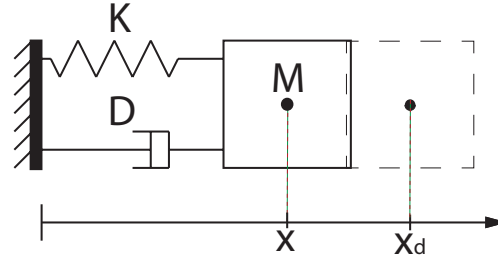


Figure 3.8: Impedance Control Scheme

the impedance control problem, but that is not a trivial task. In fact, it is not simple to choose the right value for each parameter, connecting that with the real behavior of the system. An useful consideration can be made, comparing the characteristic polynomial of impedance control, $M\ddot{x}(t) + D(\dot{x}(t)) + K(x(t))$, with the generic polynomial of a second order system

$$s^2 + 2\zeta\omega s + \omega^2 \quad (3.39)$$

where ω is the natural frequency of the spring-mass-damper system and ζ is the damping coefficient. Using this formulation the K and D parameters (and hence K_D and K_P) can be chosen as

$$K = \omega^2 M \quad (3.40)$$

$$D = 2M\zeta\omega \quad (3.41)$$

3.7.2 Force control

In the impedance control (and also in admittance control), the interaction force is controlled indirectly, acting on a reference position x_d . If it is necessary to be sure about the contact force generated by the robot, a "direct technique" may be used. It is opportune to note that this control's family doesn't accept force inputs throughout the operative space, but the directions, along which the end effector is bounded, should be properly defined. Those dimensions compose the $R(K_A)$ space. As for the position control, a PD control, acting on the force error, would be necessary to ensure the right following of the reference signal. Unfortunately that is not advisable, since the measure from the force sensor is always affected by noise, and the using of the derivative part would disturb the behavior of the manipulator. The

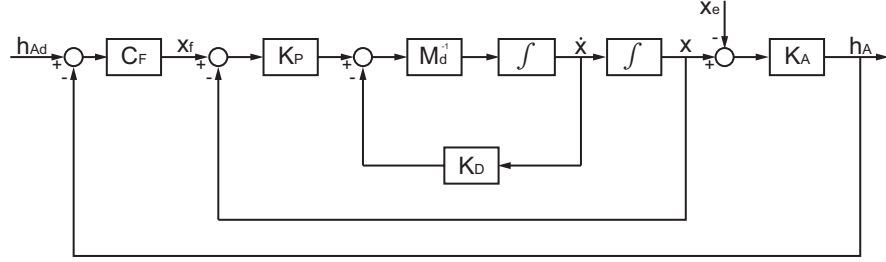


Figure 3.9: Force control scheme with position inner loop

stabilizing effect has to be obtained by damping contributions related to the velocity and the position of the robot. An example of force control with a position inner loop is shown on figure 3.9

Observing figure 3.9, the dynamic of the system is described by equation (3.42)

$$M_d \ddot{x}_d + K_D \dot{x} + K_P (I + C_F K_A) x = K_P C_F (K_A x_e + h_{Ad}) \quad (3.42)$$

where the environment is considered elastically soft and x_e is the equilibrium position of the undeformed workspace, whose contribution affected the interaction force. The block C_F determines the type of the action on the force error: if it has only a pure proportional part, the h_{Ad} will never be equal to h_A . Adding an integral part (obtaining a *PI* action), both targets ($h_A = h_{Ad}$ and rejection of x_e effects) can be completed. Since the stiffness of the environment is often very high, the coefficients of C_F should be chosen very low.

3.7.3 Force/position parallel control

A different type of force control is the force/position parallel control [25]. This system offers the possibility to feed the system with a force input signal, together with a position reference signal and hence, to reach a desired position. In fact, the precedent controls are not able to control the position of the end-effector, even when the input force belongs to the $R(K_A)$ [27]. The block diagram is shown in figure 3.10.

The relative dynamic equation is

$$M_d \ddot{\tilde{x}} + K_D \dot{\tilde{x}} + K_P \tilde{x} = K_P C_F (h_{Ad} - h_A) \quad (3.43)$$

where \tilde{x} is equal to $(x - x_d)$. At the equilibrium equation (3.43) becomes

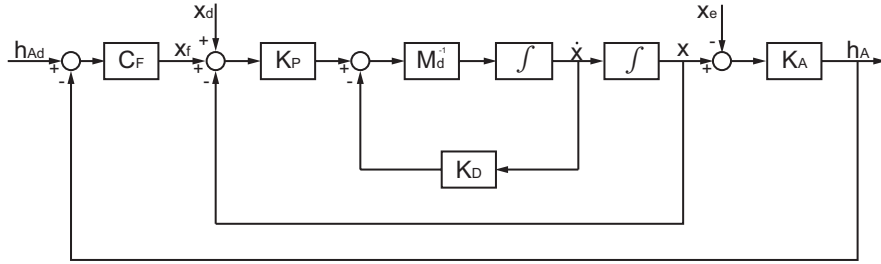


Figure 3.10: Force/Position parallel control scheme

$$x = x_d + C_F(K_A(x_e - x) + h_{Ad}) \quad (3.44)$$

and it shows that along the directions external to $R(K_A)$, the free motion is allowed and the desired position x_d is reached; while along the directions belonging to $R(K_A)$, the position input is seen as a noise and the integral part in C_F ensure achieving h_{Ad} (at the expense of a position error).

3.7.4 Hybrid control

As one can guess from the precedent sections, no force algorithm, no matter how sophisticated is it, can command a force with the end effector in all the free space, nor command a motion while the end effector is held fast: the environment has to be modeled. It could be useful to have a control that can get this target. The *Hybrid control* accomplishes this task ([11]). It represents the combination of force and position controls, but they are applied along different directions, which are selected by considering the work required to the robot. A block diagram of this control is shown on figure 3.11

In order to work with this controller, it is necessary to define *virtual* and *real* constrains, considering the type of operation to deal with the robot [27]. The \sum matrix is called *selection matrix* and it is a diagonal matrix composed by 0 or 1 components, depending on the direction to select.

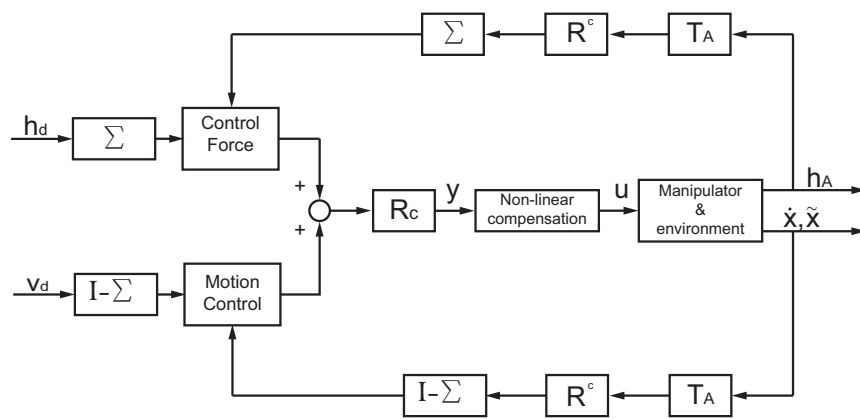


Figure 3.11: Hybrid control scheme. The T_A matrix makes the signal \dot{x} and \tilde{x} homogeneous with artificial constrains, while R_C is a rotational matrix, to convert dimensions from base frame to constrain frame.

4. Sensable Phantom Haptic Device

The control of a real robot, through an external device (e.g. a haptic device), is not a trivial task. There are some important factors that have to be considered. First of all, each signals have to be mapped with respect to the relative frame of each robot. This aspect becomes more important complicating the control strategy and, hence, a mapping strategy has to be defined. Another important aspect is the delay in the communication. The delay's problem becomes critical working with a force closed loop system, where even a small delay could degrade the system's performance, causing instability.

4.1 Mapping

The Phantom device and the Gantry Tau robot have different coordinate's frames. When they are connected together, it is necessary to define rules, to avoid errors and weird behaviors. There are two different frames for the haptic robot and two for the parallel robot, as shown in figures 4.1, 4.2 and 4.3

where the h and R indicate the relation with haptic and real robot respectively, while the $base$ and TCP indices describe the belonging of the frame to the base or to the Tool Center Point of the manipulator. In particular, they have been chosen, as shown in figure 4.1

- h_{TCP} is assumed centered on the tip of the third joint and it is placed with the same origin of the R_{TCP} , hence their relation remains constant.
- h_{Base} is fixed in the center of the workspace of the haptic device in order to maximize the possibility of the movement in all the direction.
- R_{TCP} has origin on the manipulated platform with z-axis directs perpendicularly to it.
- R_{base} is considered at the same level of the lower binaries (fig.4.2).

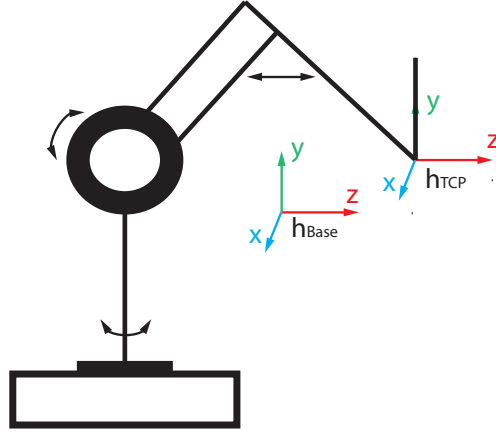


Figure 4.1: The h_{base} frame is positioned in the middle of the operative space, while the h_{TCP} is fixed at the end of the third joint, on the tip of the pendant.

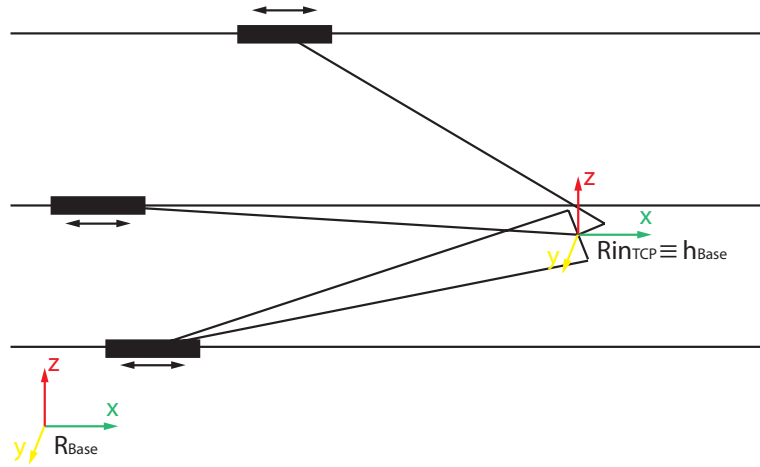


Figure 4.2: Starting robot's frame. The Rin_{TCP} frame is coincident with h_{base} at the initial time. Their relation remains constant for all the time.

The other two frames are hin_{TCP} and Rin_{TCP} and represent the initial version (at the startup) of h_{TCP} and R_{TCP} . As explained in section 3, the homogeneous transformation 4.1 $T_{R_{TCP}}^{R_{Base}}$ shows the relation between the TCP's robot frame and base's robot frame, in position and orientation, and it is

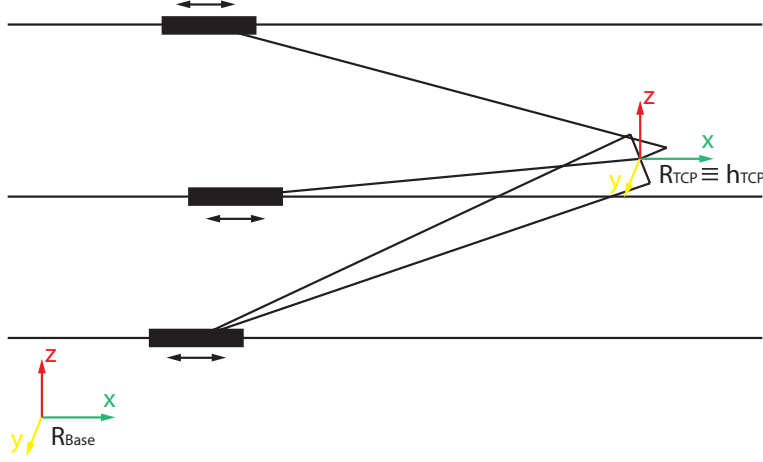


Figure 4.3: After movement robot's frame. This time, the R_{TCP} has the same application point of h_{base} and their relation is constant during the functioning

necessary for a correct mapping between the two devices

$$T_{RTCP}^{R_{Base}} = T_{h_{Base}}^{R_{Base}} T_{h_{TCP}}^{h_{Base}} T_{RTCP}^{h_{TCP}} \quad (4.1)$$

Analyzing the terms of this relations, one can note that: $T_{RTCP}^{h_{TCP}}$ is constant, $T_{h_{TCP}}^{h_{Base}}$ if furnished by the haptic device and $T_{h_{Base}}^{R_{Base}}$ can be estimated, using

$$T_{h_{Base}}^{R_{Base}} = T_{RTCP}^{R_{Base}} T_{h_{TCP}}^{RTCP} T_{h_{Base}}^{h_{TCP}} \quad (4.2)$$

At the startup it becomes

$$T_{h_{Base}}^{R_{Base}} = T_{Rin_{TCP}}^{R_{Base}} T_{hin_{TCP}}^{Rin_{TCP}} T_{h_{Base}}^{hin_{TCP}} = T_{Rin_{TCP}}^{R_{Base}} T_{h_{TCP}}^{RTCP} \quad (4.3)$$

considering that $T_{h_{Base}}^{hin_{TCP}}$ is an identity.

This control strategy imagines the TCP position of the Phantom robot as the "real" TCP of Gantry Tau. Moving the pendant of haptic device, the operator would move the real robot. In this case, Phantom Robot Connector, should convert the information read from haptic world to the real robot world, using the kinematic equations of parallel robot to achieve the right configuration.

Even if that is valid in general for different kind of robots, this solution requires a different implementation for each manipulator, since the kinematic

functions would not be the same. The purpose of this work was to use the haptic device to control a general slave robot, regardless their typology, and hence another strategy has been adopted. In order to obtain position and velocity directly (without any other kinematics calculation), some dedicated functions are used, available in the Phantom libraries. The position and velocity signals are converted into *rBase* frame before being sent, and hence they can be used by Simulink controller as reference signals. With this modality the slave robot is able to move only along x, y and z in *rBase* frame, but it is enough for our target. In order to create a "position zero" starting point, avoiding an undesirable "jump" a reset of haptic device is necessary before running the wave mode and the *Reset Mode* on PRC has precisely this purpose.

5. Wave Variable Method

The *Phantom-Gantry Tau* system can be considered as a *Master-Slave* structure. Many studies have been conducted on this theme [23] and some solutions are proposed for stabilization and control, as the *wave variable method* [9], [8], the one analyzed in this thesis. The wave variable method is based on the *Theory of Passivity* [14], [22], which creates robustness to arbitrary time delays. The main idea is to convert all the information before and after the transmission, from and to slave and master robots, into a wave variable form, which is a particular combination of velocity and force signals. As explained in this chapters, it is possible to use other types of signals, but velocity and force are the most used for robot systems.

5.1 Basic Passivity Concepts

The passivity formalism represents a mathematical description of the physical concepts of power and energy. It furnishes a simple and stable instrument to study the stability of all kind of systems, without restrictions on their connections.

Intuitively, a system is *passive* if it absorbs more energy than it produces. Considering figure 5.1, define the power " P_{in} " entering in a system, as the

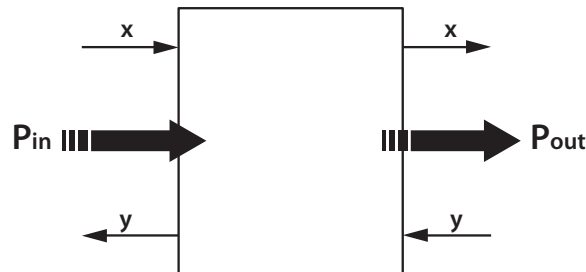


Figure 5.1: The incoming and outgoing "power" of a general system.

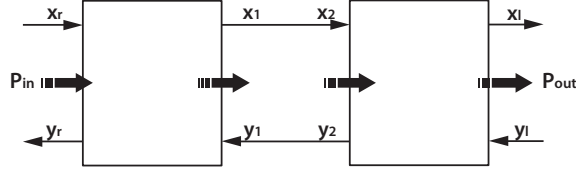


Figure 5.2: A general two-ports elements in cascading connection. Power is considered positive when flow from left to right side, along the major direction.

product between the input (x) and the output signal (y). Note that "energy" and "power" terms does not necessarily correspond to any physical meaning. A system is said passive if it obeys to 5.3

$$P_{in} = x^T y = \frac{dE}{dt} + P_{diss} \quad (5.1)$$

where E is the "energy storage" function, while P_{diss} is the nonnegative "power dissipation" function. The total flowing energy is hence limited by the initial store energy

$$\int_0^t P d\tau = \int_0^t x^T y d\tau = E(t) - E(0) + \int_0^t P_{diss} d\tau \geq -E(0) = \text{constant} \quad , t \geq 0 \quad (5.2)$$

This "power" concept can be applied to a multi-degree and multi-port system and, in order to facilitate the introduction of wave variable method, let consider two-ports elements of figure 5.2.

According with equation 5.3, the total power input is given by

$$P_{in} = x_l^T y_l - x_r^T y_r \quad (5.3)$$

that can be translated in the form

$$P_{in} = x_l^T y_l - x_r^T y_r = \frac{1}{2} u_l^T u_l - \frac{1}{2} v_l^T v_l + \frac{1}{2} u_r^T u_r - \frac{1}{2} v_r^T v_r \quad (5.4)$$

where subscripts r and l indicate the left and right signals. Formally treated in the scattering theory ([14]), the u and v variable represent the *wave variable* and the are interpreted, respectively, as the *input* and *output* waves. Associating *velocity* and *force* to the general x and y signals of the two-ports system, a conversion law from power variables to wave variables is given and it will be detailed in the next sections.

5.2 Wave Variables

As already said, using the "wave" data in the transmission, in place of the standard quantities, the system is stabilized regardless of the delay. An example of a scheme is shown in figure 5.3

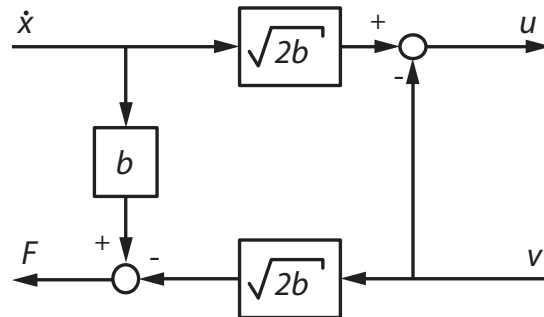


Figure 5.3: An example of wave variables scheme. This basic transformation relates velocity \dot{x} and force F (on the left) with the moving wave u and v , on the right.

It is evident that the strength of this system is the simplicity of the conversion method, since it is composed by algebraic matrices easy to implement. The b coefficient represents the wave impedance and it may be a constant or a symmetric positive definite matrix, depending on the dimension of the signals. It represent the "weight" of the communication channel, and affects the signals, operating as a scaling factor. Its values has also impact on the operator behavior, modulating the feedback signal from the master. The force F and the velocity \dot{x} can be replaced by others combination of signals, it depends on which information is needed for the control system. The u and v are, respectively, the *forward wave* and the *backward wave*, whose give also an indication on the *outgoing energy* and the *incoming energy*. Referring to velocity $\dot{x}(t)$ and force $F(t)$, for a single degree of freedom system, the wave variables (u, v) can be computed by

$$\begin{aligned}
u_m(t) &= \frac{bx_m(t) + F_m(t)}{\sqrt{2b}} \\
v_m(t) &= \frac{bx_m(t) - F_m(t)}{\sqrt{2b}} \\
u_s(t) &= \frac{bx_s(t) + F_s(t)}{\sqrt{2b}} \\
v_s(t) &= \frac{bx_s(t) - F_s(t)}{\sqrt{2b}}
\end{aligned} \tag{5.5}$$

where s and m subscript refers to master and slave side.

One of the main reasons for dealing with wave variables is their effect on the condition for passivity [14]. In fact, if a system was passive in the normal notations, it remains passive after transformation into wave variables, but, more important, a time delay is now modeled as a passive element. So a system expressed in wave variable form is robust to delay, regardless the amount of it.

5.3 Integrated Wave Variables

So far, it was used only force and velocity as signals for the transmission and no position feedback was inserted in the system. This type of configuration may be affected by a slow position drift between master and slave, if numerical errors or data loss occur in the functioning. Instead of using the velocity integral to calculate the position, it is possible to send the *wave integral*. In fact, as the wave signals encode velocity and force, their integral bring information on position and momentum.

With respect to equation 5.5, the integrated wave variables can be evaluated by

$$U(t) = \int_0^t u d\tau = \frac{bx(t) + p}{\sqrt{2b}} \quad V(t) = \int_0^t v d\tau = \frac{bx(t) - p}{\sqrt{2b}} \tag{5.6}$$

where p is the integral of the force

$$U(t) = \int_0^t F d\tau \tag{5.7}$$

If the position can be measured directly without problems, the computation of the momentum could be a trivial task and it could bring errors

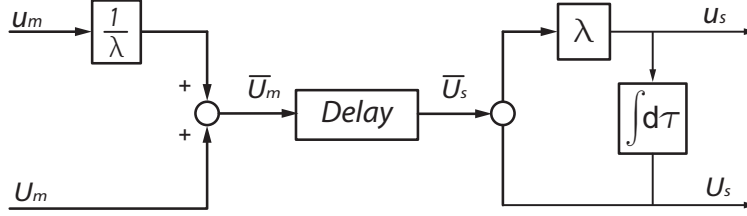


Figure 5.4: An example of the combination between wave and integrated wave signals.

and inaccuracy (the same problems that we had before with the velocity and force). However, the main information is the position and many times p is not relevant for the control. Those new variables are sent together with the precedent, and hence the system has to be extended with those new equations for the slave side

$$p_s = \int_0^t F_s d\tau x_{sd} = \frac{\sqrt{2b} - p_s}{b} V_s = \frac{bx_{sd} - p_s}{\sqrt{2b}} \quad (5.8)$$

and for the master side

$$U_m = \sqrt{2b}x_m - V_m \quad (5.9)$$

Using both wave and integrated wave signals, it is avoided the position drift between slave and master. Other solutions have been proposed [22] or [24].

5.4 Single Channel

This new configuration seems to need more than one channel for the communication. In reality this is not true, in fact, the two type of waves can be combined together into a single quantity before the transmission and separated after the arrival using and stable first order filter, as shows in figure 5.4

The term λ represents the bandwidth of the filter; it can be choose arbitrary and it may be used as a scale factor, but, however, it should be less than the sampling rate of the digital implementation (i.e. $\sim 250Hz$). The new term \bar{U} contains position, velocity information, necessary for the control system.

5.5 Multiple degrees of freedom system

For a multiple degrees of freedom system the equations presented in the precedent section, have to be modified. In [26] and [8] it is presented a more generic formulation for the transformation equations (5.5)

$$\begin{aligned}
 u_m(t) &= A_w \dot{x}_m(t) + B_w F_m(t) \\
 v_s(t) &= C_w \dot{x}_s(t) - D_w F_s(t) \\
 v_m(t) &= C_w \dot{x}_m(t) - D_w F_m(t) \\
 u_s(t) &= A_w \dot{x}_s(t) + B_w F_s(t)
 \end{aligned} \tag{5.10}$$

where A_w , B_w , C_w , and D_w are $n \times n$ wave variable scaling matrices and n is the number of DOFs of the teleoperation system. It is clear that this matrices can not be chosen freely, since the passivity's condition has to be ensured [22].

In [10], it has found four rules to identify a new stable and large family of all consistent scaling matrices, that can be used for working with wave variable method in a multiple degrees of freedom system:

1. A_w must be non-singular;
2. $B_w = \frac{1}{2}(I + S_w)A_w^{-T}$, where S_w is a $[n \times n]$ skew-symmetric matrix (i.e. $S_w = -S_w^T$).
3. $C_w = QA_w$, where Q is an orthogonal matrix of $[n \times n]$ dimension;
4. $D_w = \frac{1}{2}Q(I - S_w)A_w^{-T}$

Note that choosing $A_w = \sqrt{\frac{b}{2}}$ and $Q = 1$ for the scalar case, one obtains the same solution as in 5.5. The choice of Q does not influence the stability of the system, since it applies a orthogonal transformation to the wave variables, without affects the power of the signals. It is convenient to choose Q as an identity matrix.

If for the single degree of freedom case, could be simple to understand the role of the wave impedance b , in the multi-degrees case is not so obvious. A_w matrix can considered as the damping factor of the communication, the increment of the dimension of the signal, involves an increment of variables and, looking at 5.10, it is easy to understand that a modification of one wave matrix parameter has effect on all the others matrices (i.e. changing the first row of A_w affects not only the first element of signal). If the meaning of A_w can be simple to understand, the weight of S_w is less clear. An increasing

in its norm comports a bigger overshoot and longer settling time for master response. It is evident that the choice of the scaling matrices is not a trivial task, and the parameters has to be tuned with respect to both robot and controller parameters and considering the desired goal. After many proofs the matrices used for this work are for Gantry Tau and IRB140B respectively

$$A_w = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix} \quad S_w = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (5.11)$$

$$A_w = \begin{bmatrix} 14 & 0 & 0 \\ 0 & 14 & 0 \\ 0 & 0 & 14 \end{bmatrix} \quad S_w = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (5.12)$$

Those are the values that better accomplish to the double target: offer small reflection on operator side, avoid reflection (and hence control error) on the robot side. The using of a S_w different from zero would improve the reflection decreasing, but at the same time, would increase the reflection error on the master side.

5.6 Reflection

Using the wave solution for the communication creates several internal loops, which in turn creates some reflection effects. As shown in figure 5.5, there are three principal paths. The first one appears when the haptic device is moved and it is due to the impedance value. The path number 2 is created by both master and slave wave transformation. In fact, part of signals continue to cycle, even if the two manipulators have finished their task. Those disturbances are drying out by the system, but they can be persistent for several cycles. The last path (3) is the most important, since it is the real force feedback from robot to haptic and should be lead to the operator as clear as possible.

All these reflection phenomenons represent a real problem during the communication and the multi-variable case is more affected by them than the single one. The choice of a correct wave impedance (or the A_w and S_w matrices) is the principal way to decrease the reflections, even if some solution are proposed in [22].

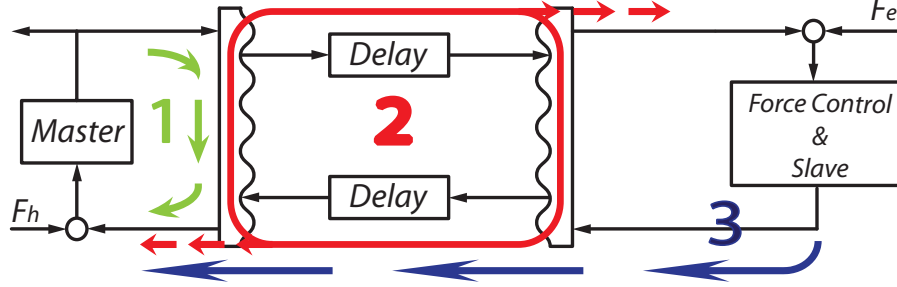


Figure 5.5: The reflection problem during the communication. It can distract and confuse the operator and, hence, it should be avoided.

5.7 Wave Variable Simulation

Since during this thesis the *GT* robot has been longer not available, a lot of studies were conducted on the serial ABB robot *IRB140B*. This has allowed to develop two models together, one for the serial and one for the parallel manipulator, obtaining the possibility to compare the result and the behavior on two different manipulators. After many modifications, it has obtained a unique model for both robots (the only difference are the kinematics blocks, which are different for each robot). Here are presented the results of some simulations.

5.7.1 Simple System

A simple simulation has been conducted on a 3DOF Master/Slave Simulink model (whose scheme is reported in figure 5.6), in order to verify the general validity of the wave method.

The two manipulators are both described by the simple linear models:

$$\tau_m = J_m \ddot{\theta}_m + B_m \dot{\theta}_m \tau_s = J_s \ddot{\theta}_s + B_s \dot{\theta}_s \quad (5.13)$$

where τ_m, τ_s are the input torques, J_m, J_s are the inertias, while B_m, B_s are the damping terms. The A_w and S_w scaling matrix are

$$A_w = \begin{bmatrix} 5 & 1 & 1 \\ 1 & 5 & 1 \\ 1 & 1 & 5 \end{bmatrix} \quad S_w = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (5.14)$$

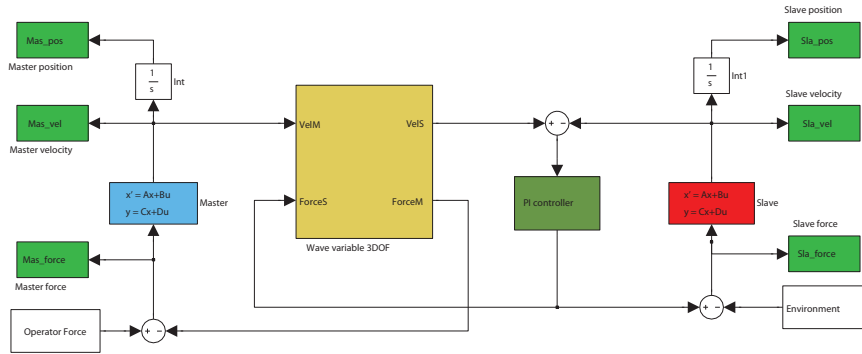
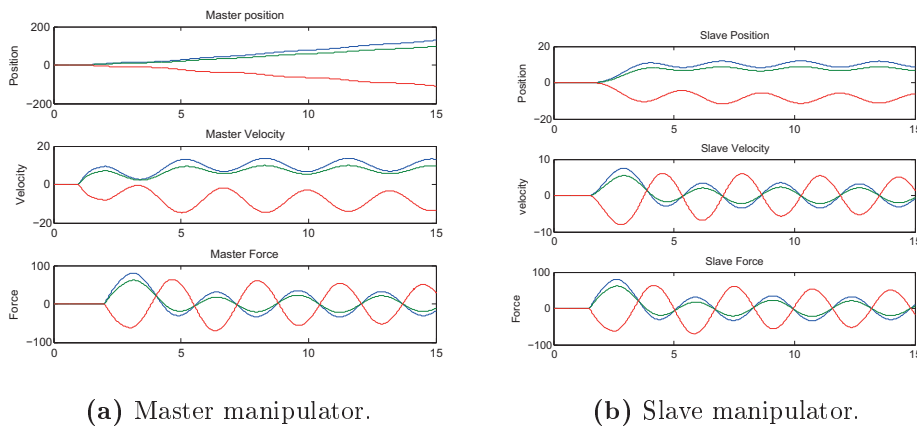


Figure 5.6: The simulink model of the Master/Slave system used for the simulation.



(a) Master manipulator.

(b) Slave manipulator.

Figure 5.7: As simple example of a teleoperation system, where a time delay causes the instability.

Figures 5.7, 5.8 show the stabilizing effect of the wave variable method for a delay of 0.5s.

5.7.2 IRB140B and Gantry Tau

Confirmed the validity of variable method, simulations have been conducted to test the model of the IRB140B and Gantry tau robot during a contact with a virtual environment, schematized as a spring-damping element. The input signal is a step, acting only along X direction (in base frame). The results are shown in figures 5.9

In both proofs the stability is confirmed. A little oscillation is still present,

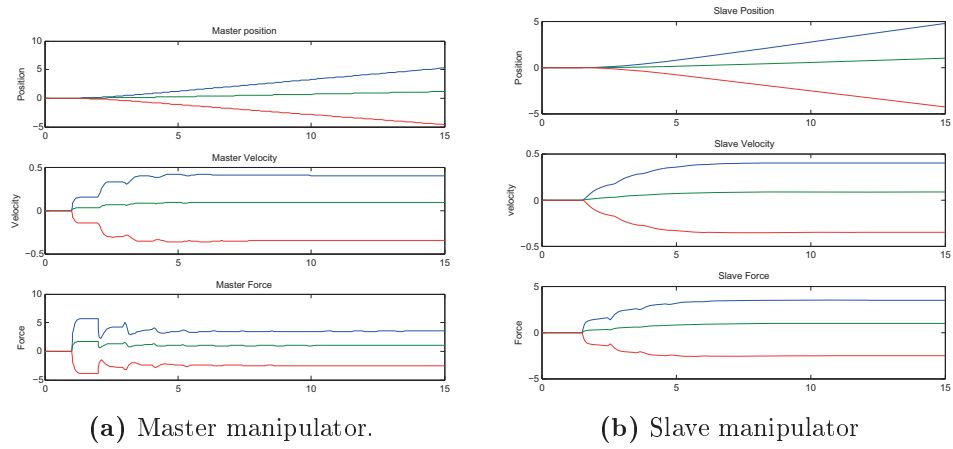


Figure 5.8: The same system is now stabilized using the wave variable method.

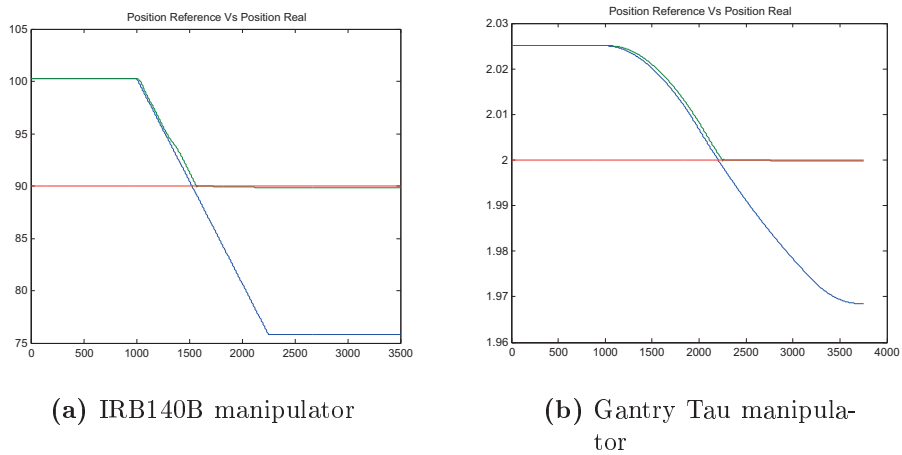


Figure 5.9: Both the model are feeded with a step, that cause the contact with a virtual wall. Small oscillations appear after the contact, since the parameters are not perfectly tuned.

but it could be adjusted varying the parameters of the virtual wall and the impedance controller, but it was non relevant for those first tests, which have the purpose of testing the validity of this "wave variable model".

6. Analysis of softwares

In this thesis many devices (e.g. *Phantom Premium A haptic device*, *Gantry-Tau Parallel robot*, *Force Sensor JR3*) and many different programs, (e.g. *Matlab and Simulink*, *Real Time Workshop*, *WinComm*, *Phantom Robot Connector*) have to work together, on the same work structure. It is easy to understand that this is not a trivial task and, hence, it is good to clarify how the communications have been realized. Figure 6.1 presents all the links between all the devices.

6.1 ABB IRC5 Robot System

The interface between the robot and *Simulink* is develop by ABB [1]. Such structure is the "brain" of the Gantry-Tau Robot and it is the device where the reference position and the reference velocity are calculated. The system is divided in two parts: the *Robot Axis Computer* and the *Robot Main Computers* (figure 6.2).

Angles, velocities and, force coming from the robot, are used in the *Robot Main Computers* to sent the reference signals (e.g. velocity or position) to the *Robot Axis Computer*, which computes the torque for the motors, using a separate internal control for each joint.

6.2 Opcom

OPCOM is the *GUI* (Graphical User Interface), created in the Automatic Control department, to work with IRC5 Robot System. The control, developed in Matlab/Simulink environment, is compiled into C code (using *Real Time Workshop*) and after it is loaded in the *Opcom* (it is done clicking the *load* button on the interface). An example of this interface is shown in figure 6.3

The controller in the *OPCOM* can be run in two ways:

- *Submit* mode: in this mode the signals are sent from the main computer (hence from the IRC5 Robot System) to the external controller, but

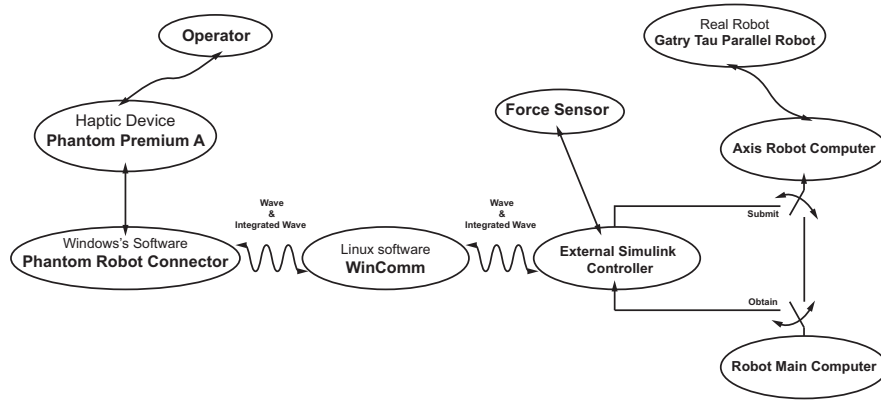


Figure 6.1: A schematic view of links between all the devices.



Figure 6.2: The IRC5. It is responsible for all the control calculation of the robot.

there are no signals back from the real robot. This mode is like a *safe* mode and it is an obligatory step before running the *Obtain* mode. It can be used each time is necessary to test a new controller or a new structure, avoiding risks of damage for the real robot or for objects present in the robot's workspace.

- *Obtain* mode: this second mode allows the exchange of signals from the main computer and the real robot, whom behavior is ruled by the external controller. Before using this modality it is necessary to be sure that every softwares have been activated correctly.

It is important to note that the Phantom Robot Connector needs a declaration of which ports and which signals will be used in the controller. To enable the communication a programm called *Labcomm* is used. This latter

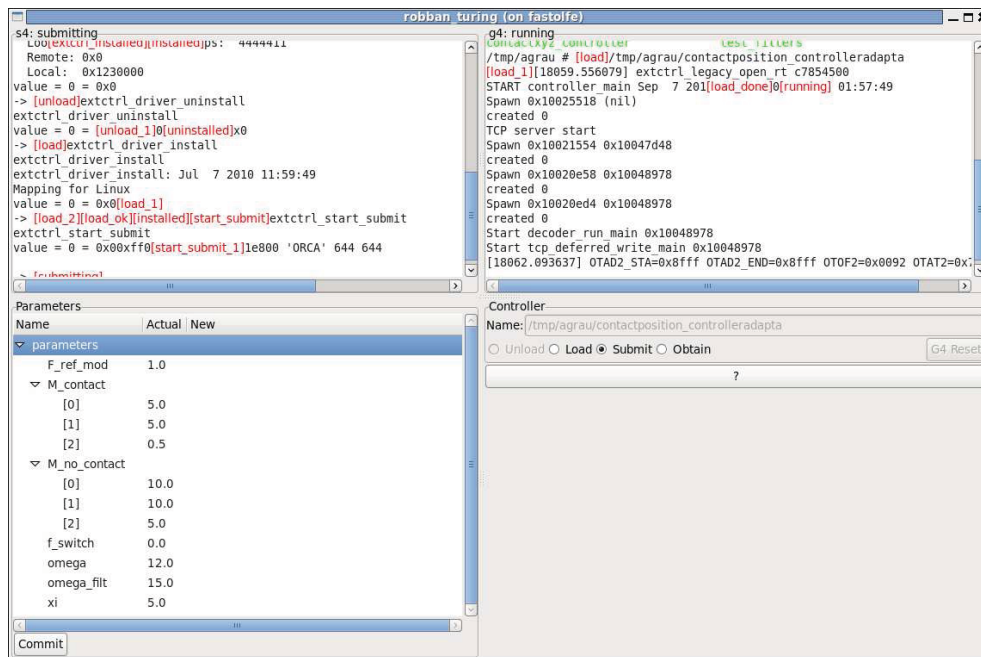


Figure 6.3: Opcom GUI. It is the graphical user interface and it is necessary to load and run the controller.

is a *Linux* application, that needs a *.lc* file, where the signals are declared. There is no implementation for a Windows structure. An example of the signals is:

- *sample float ph2RobAngles[3];*
- *sample float rob2PhAngles[3];*

After writing of this file, a *LabComm* command creates a *.c* file and a *.h*, with the same name of the *.lc* file. Inside these files there are the necessary function to exchange signals from/to haptic device, to/from the controller. The *[3]* number after the name indicates the dimension of the incoming or outgoing signals. Naturally, the *out* and *in* name, indicated in the *.lc* file, have to be inserted in the *Simulink* model, in order to be recognized and compiled by Real Time Workshop (*RTW* 6.5.1) and used for the communication. The compiled controller is loaded on *Opcom*, running on the machine connected with the *Robot Main Computer*, and, hence, the operator has to be connected with this computer before the loading of the controller.

6.2.1 Interface

The interface is very intuitive and simple 6.3. The *load*, *submit* and *obtain* are positioned on the right of the screen, while the *parameters* window is on the bottom left part, and it offers the possibilities to modify some parameters on-line, by inserting a new value on the relative variables and clicking *Commit* button. Not all variables are present on the interface, since they have to be declared on the Simulink controller, before compiling with RTW.

6.3 Phantom Robot Connector

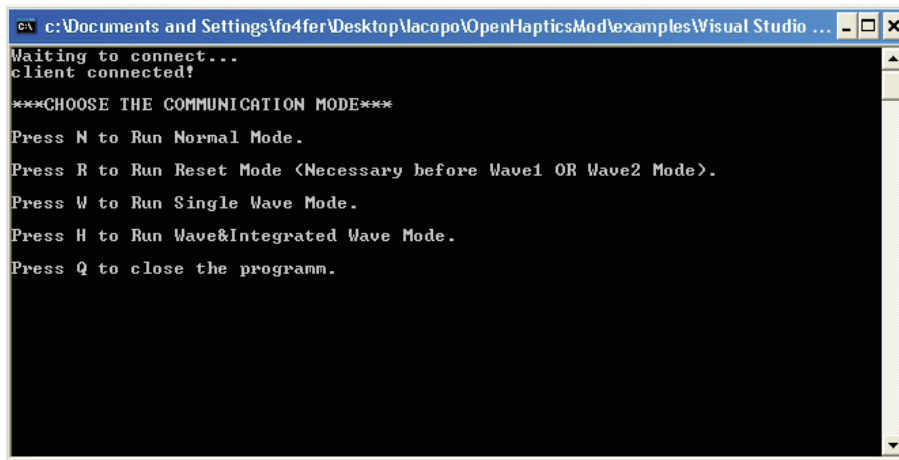
In order to allow the communication between Phantom Haptic Device (working on Windows machine) and the rest of the structure, a C++ program was made by Fredrik Eriksson and Marcus Welander, the so called *Phantom Robot Connector (PRC)*. In this work, PRC software has been modified, in order to be able to work with different kind of signals, compared to those used by Fredrik and Marcus. Next section will furnish a short description of the new version of PRC programm.

6.3.1 Startup

In the original software, the operator had to choose at the startup which typology of functioning was needed (figure6.4:

- *Phantom2Graphic*: the Phantom is connected with a virtual robot and the real robot is not controlled. This modality could be useful to test the behavior of the system
- *Robot2Graphic*: the virtual robot maps the real robot. It is the modality active when Phantom Robot Connectors starts.
- *Phantom2Robot*: the real robot is handled by the haptic device and the operator can control the system.

In the new version those choices are not available and the communication can only be between Phantom and robot. In fact, *Phantom2Graphic* and *Robot2Graphic* worked with a virtual model of a serial ABB robot, the IRB140B, but no virtual model are available for PKM Gantry Tau. Hence, when the software is started, a new interface appears. Before starting of *Phantom Robot Connector*, it is recommended to reset the encoders of the haptic device, using a dedicated software, created by *Sensable* [6]. The robot has to be positioned as in figure 6.5, in fact that is the optimal configuration



```

c:\Documents and Settings\fo4fer\Desktop\Iacopo\OpenHapticsMod\examples\Visual Studio ...
Waiting to connect...
client connected!

***CHOOSE THE COMMUNICATION MODE***

Press N to Run Normal Mode.
Press R to Run Reset Mode (Necessary before Wave1 OR Wave2 Mode).
Press W to Run Single Wave Mode.
Press H to Run Wave&Integrated Wave Mode.
Press Q to close the programm.

```

Figure 6.4: The Phantom Robot Connector interface with the three selectable choices.

to increase the workspace of the joystick. After the position is reached the *reset encoder* button has to be pressed. After that the application can be closed and the haptic device is ready to work.

In the new version of PRC software, a *Reset* option has been inserted. This is not a real reset of the encoders of Phantom device, but a necessary step before starting others modes. The last position of the haptic TCP is stored, in order to be used to create a "*starting zero position*". In fact instead sending the *position* value, read from haptic device, it is used the value "*Position-StartingZeroPosition*".

Starting the new Phantom Robot Connector, the interface offers different choices

- *Reset Mode*: The current position of Phantom device, is stored and it will be used in the others "*mode*", for the transmission of the signals.
- *Normal Mode*: Position is sent to the controller, without any transformation, while force is received from the robot.
- *Single Wave Mode*: Phantom velocity is combined with feedback wave signal *vim* (coming from robot) to obtain the outgoing wave *mu*, that has to be sent to the real robot. *vim* is converted to get the force feedback.
- *Wave&Integrated Wave Mode*: Wave and Integrated Wave signals are constructed using, respectively, a combination of Phantom velocity and ingoing wave signal *vim*, for the first and Phantom position and ingoing wave signal *VIM*, for the second. These signals have to be merged,



Figure 6.5: Haptic device at the startup. In order to increase the operative workspace, the Phantom has to be initialized with this position.

before the transmission, in order to use a single communication channel 5.4.

6.3.2 Digital Force Filter

The sample time of the haptic rendering is approximately 1000 HZ, while that of force sensor is 250 Hz (the same of the Simulink Controller). This condition could be critical for the stability of the haptic device, that should work with render frequency more high than 250 Hz. In order to avoid problems a stable digital force filter (6.1) is introduced in the Phantom Robot Connector. That filter smooths the force signals, mapping the high steps, with more gradual variations steps. The parameters are chosen so that the transfer function has poles inside the unitary circle.

$$y[k] = \beta_1 u[k] + \alpha_1 y[k - 1] + \alpha_2 y[k - 2] \quad (6.1)$$

6.3.3 A short description of the code structure

The program is constructed on three threads (even if the Robot Thread is not more longer used in this work):

- Haptic Thread
- Robot Thread
- Connection Thread

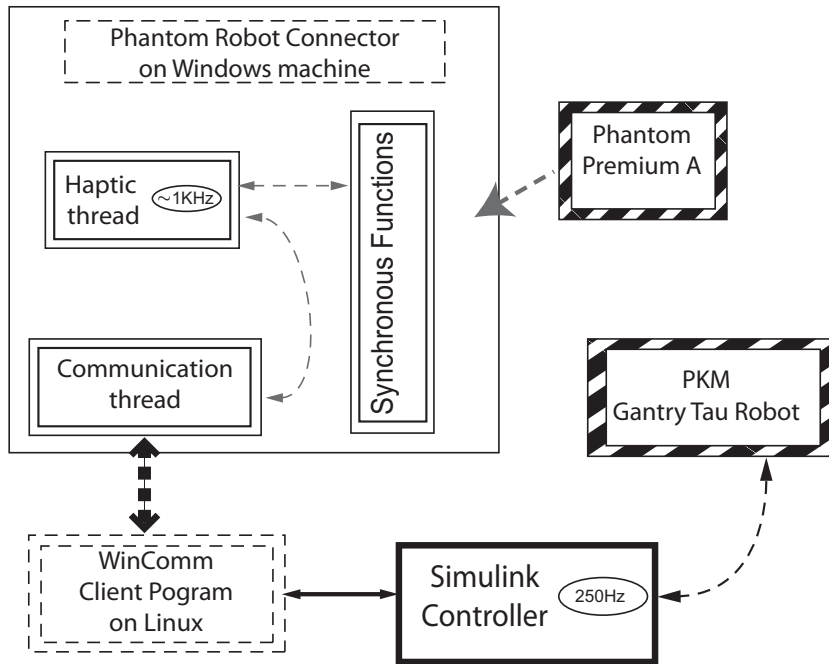


Figure 6.6: A schematic view of the program structures working in the system.

An exhaustive scheme is shown in figure 6.6.

Those threads work together, exchanging information and signals. Inside the code there are many classes, one for each task requested. Here are reported the principals:

- *RobotHaptics*: in this class are calculated all the homogeneous matrices necessary for the transformations of the signals (e.g. T_{hBase}^{hTCP}). Using the Phantom functions ([30], [29]) and kinematics functions, all the Phantom signals can be calculated.
- *Communicator*: as already explained, the exchanging of information is not a trivial task and, hence, this class is of a great importance. A signal has to be transmitted from a Windows machine to the robot main computer, through a Linux machine, where a dedicated program is running. The communicator class offers 4 different types of functions: *SendToRobot*, *ReceiveFromRobot*, *SendDelayToRobot* and receive *DelayFromRobot*. Inside *SendToRobot* and *ReceiveFromRobot* have to be declared the names and the dimensions of the vectors exchanged.

| | |
|-------------------|--|
| copyRobotAngles | Copies the content of "angles" signals. |
| setPhantomForce | Set the force between Phantom and the operator. |
| copyhTCP2hBase | Copies the current homogeneous transformation matrix of Phantom. |
| copyUserMode | Copies the active functioning mode. |
| setUserMode | Change the user mode. |
| copyStartPosition | Copies the starting position of Phantom's TCP. |
| copyUserMode | Copies the current homogeneous transformation matrix of Phantom. |

Figure 6.7: Synchronous functions utilized inside Phantom Robot Connector programm.

In order to work with Phantom, it is possible to choose between two libraries: *HLAPI* (Haptic Library API, designed for high-level programming) and *HDAPI* (Haptic Device API, suitable for a low-level programming) and Phantom Robot Connector utilizes the latter. *HDAPI* controls all the communication, setting the schedule, through the use of one of those functions: *hdScheduleSynchronous* or *hdScheduleAsynchronous*, where *Synchronous* means that the scheduler will execute (if possible) the function declared inside in the arguments. *HDAPI* controls a lot of functions (??), but the most important is the *altrigenderism* (scheduled as asynchronous and with high priority), which elaborates all the signals to control the robot.

6.4 WinComm

WinComm is a client programm to exchange information between the main computer and haptic device running on a Windows machine. It was created by Fredrik Eriksson and Marcus Welander and in this thesis it has been modified, inserting the new IP address of the windows machine and the new *Labcomm* file, containing names and typologies of the signals used, which, obviously, have to be same as those present inside of *SendToRobot* and *ReceiveFromRobot* functions of *Communicator* class in Phantom Robot Connector program. This software has the role of linker the *Windows* machine and main Robot computer. In fact, *labcomm* functions are available for Linux system only and hence, it was necessary to create an apposite program to permits exchanging of information.

6.5 Matlab and Simulink

Matlab [3] is an high-level language and interactive environment, that permits to join the programming language (as *C*, *C++* or *Fortran*), with the mathematical world. It is a very powerful tool and today it is a fundamental component in the engineering. The other application used together with Matlab is Simulink [4], which is a tool for modeling, simulating and analyzing multi domain dynamic system. It is composed by an interactive graphical interface, where the operator can add customizable set of block libraries, that gives the opportunity to design and test a lot of time-varying systems. As it is done in this work, it is possible to insert external components, as, for example, signals or functions.

6.5.1 Real Time Workshop

Real Time Workshop [5] is an application, that creates and executes a stand-alone C-code, starting from a Simulink and Embedded Matlab developed model. This tool package permits to create a real time program, executable externally of the Simulink or Matlab environment. In this work the compiled C-code is loaded in the *Robot Main Computer*.

6.6 How to run

All this programs have to be start in the right order to work correctly. The first software is *Phantom Robot Connector*. At the startup a black window appears and the system attends for the connection with the Linux machine, through *WinComm*. Now it is necessary to activate the *WinComm*, but before it would be able to run, the controller has to be loaded on *Opcom*, running on a machine called *Calvin*. When all those steps have been made, the communications are enabled and on the *Windows* computer is now possible to choose one of the five options (*Reset Mode*, *Normal Mode*, *Single Wave Mode*, *Wave&Integrated Wave Mode* and *Quit*). If one of the first four mode is chosen, a noise comes from the Phantom device, indicating that its motors are now in torque and the device is ready to work. The last step is to enter in *Submit* or *Obtain* mode on *Opcom*, thus starting the exchange of information. The parameters, selected inside the Simulink controller, show up on the *Opcom* GUI.

7. Simulink Model

As introduced in the previous chapters, *Simulink* has been used to create a model for the force controller and this section points to describe its principal parts. A particular attention is given to the external libraries utilized, where some functions are from precedent works, while others have been created expressly for this thesis. The template model (available at the automatic control department) shows all input/output channels, utilizable for the communication with the robot. The external signals (force sensor, Phantom signals) have to be declared in the *.lc* file, before using in the model.

7.1 Libraries

The model is constructed utilizing the normal Simulink blocks and other two libraries: one contains the kinematic blocks created specifically for Gantry tau parallel robot (figure 7.1) by Isolde Dressler and improved by Johan Friman [18], and the other is the *ExtCtrl Library 7.2*. It is available in the automatic control department and it furnishes kinematic blocks for serial robots (i.e. forward and inverse kinematics, jacobian, etc.).

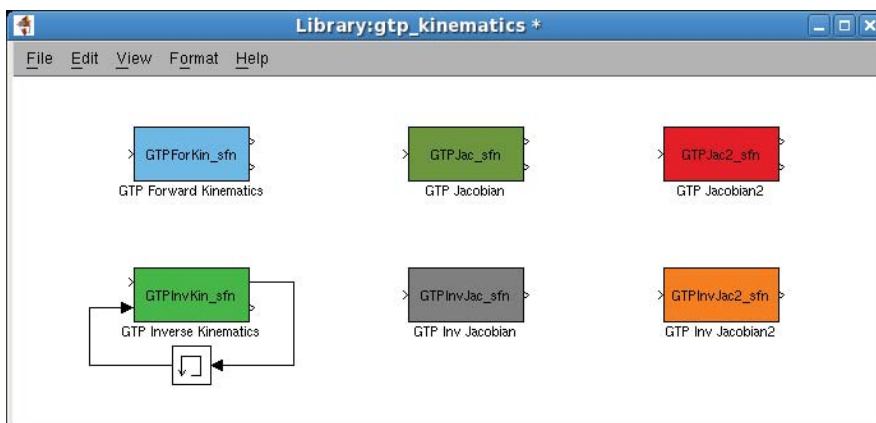


Figure 7.1: The Gantry Tau Simulink library

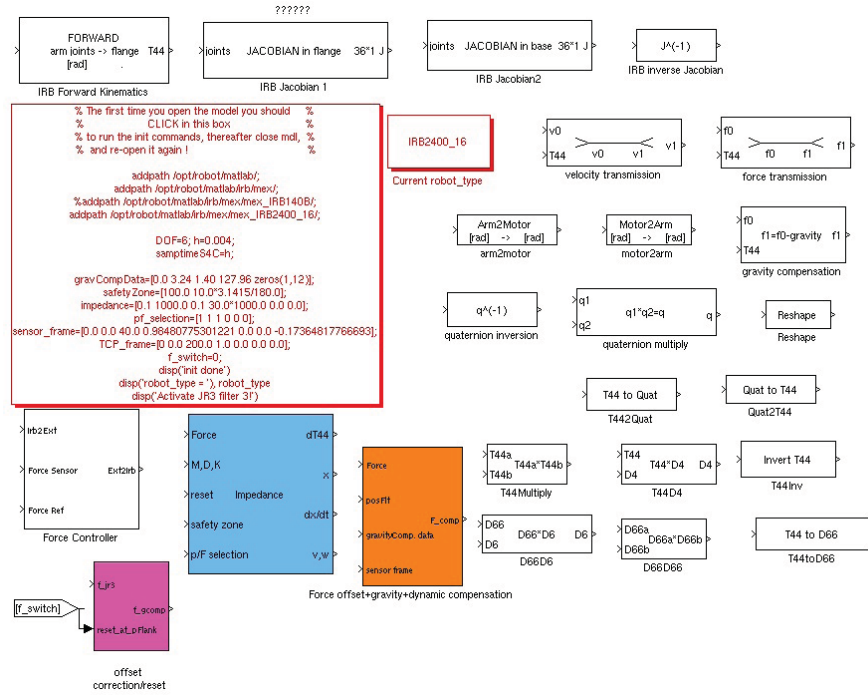


Figure 7.2: The ExtCtrl Simulink library

The dedicated blocks permit to consider Gantry Tau robot as a normal serial robot, allowing the using of normal kinematics functions. This operation would be not possible because its parallel configuration.

7.2 Choice of the Signals

As already mentioned, the template Simulink model offers different signals to read information from the real robot. Considering, for example, the position, it is possible to use $irb2ext.robot[0].joint[i].posFlt$, $irb2ext.robot[0].joint[i].posRawAbs$ and $irb2ext.robot[0].joint[i].posRef$, where the names furnish an indication on the characteristic of the signals. The choice of one signal, instead of another one, depends on which behavior is desired. The one used in this model is the $irb2ext.robot[0].joint[i].posRef$. It has not a true physical meaning, in fact it is a pure mathematical quantity, calculated by ABB Robot Main Computer. It changes only when the robot is joined manually. This aspect does not effect the robot during normal control with Phantom device, but it could be dangerous when the communication is stopped, since the reference position would return the starting one, causing an undesirable jump of Gantry Tau. However, the presence of a constant ref-

erence signal could also be a positive aspect, if all the dimensions are referred to it.

7.3 Model

Since the two models developed are equivalent, only the one of Gantry Tau is presented. In the next subsections will be described various blocks, highlighted with numbers in figure 7.3

7.3.1 Force Reference (1)

Since no force sensor were installed on Gantry Tau robot at the beginning of this thesis, a virtual wall has been implemented, in order to simulate an interaction force. The position of the environment can be set on Opcom and the force is generated as a spring damping element. The resulting contribute is added to the values coming from the force sensor (when present), feeding the impedance controller. As already explained in section 2.4, the force sensor needs a reset procedure before using on the control, and that can be done by GUI, switching *ResetForceSensor* from 0 to 1, when the force sensor is active. The force reads from the sensor is expressed with respect to *Sensor frame*, while the contact is on TCP. Some transformations are necessary to convert the force in *rBase frame*, as reported in equation 7.1

$$f_{base} = T_{TCP}^{base} (T_{TCP}^{flange})^{-1} T_{sensor}^{flange} f_{sensor} \quad (7.1)$$

where T_{TCP}^{flange} and T_{sensor}^{flange} depending on the physical dimension of the sensor and the tool. This conversion is implemented inside the force block (figure 7.4).

7.3.2 TCP position and Velocity (2)

The real current position and real current velocity of the robot can be evaluated through the direct kinematic. The first is obtained adding the position output of impedance control to the starting position (7.5), calculated from *irb2ext.robot[0].joint[i].posRef* signal. The result can be compared with a real measurement of the position, read from *irb2ext.robot[0].joint[i].posFlt* or *irb2ext.robot[0].joint[i].RawFb* input ports. The velocity is calculated using the current angles and the angle's velocity *irb2ext.robot[0].joint[i].velRef* (7.6). However, the real velocity is no more used in the controller.

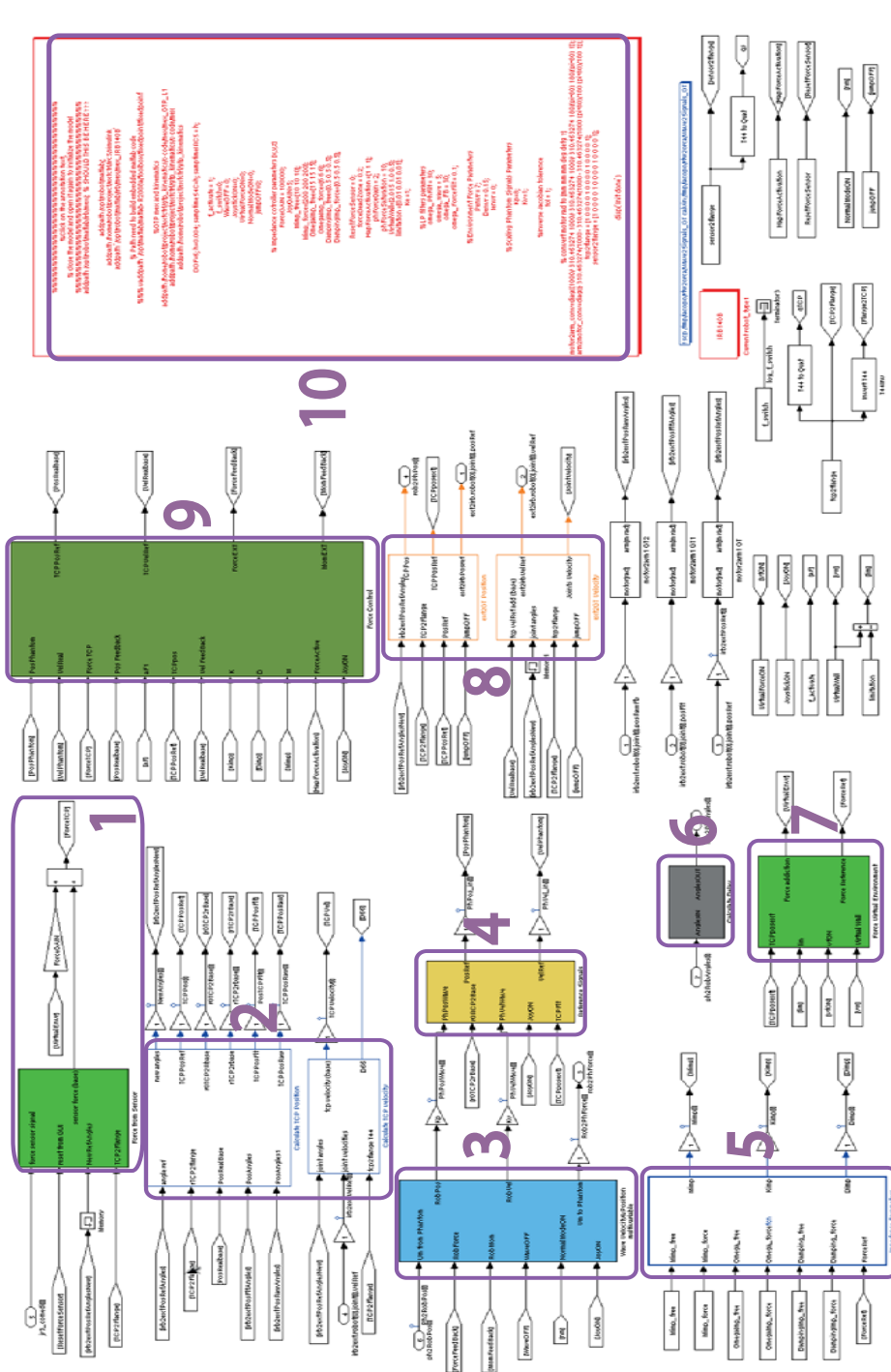


Figure 7.3: The Simulink Model for Gantry Tau, where the most significant blocks are marked with a number.

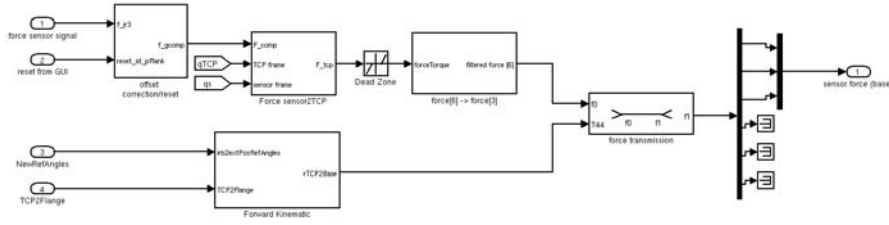


Figure 7.4: After a reset of the sensor, the force can be translated in robot base frame and sent to the impedance controller.

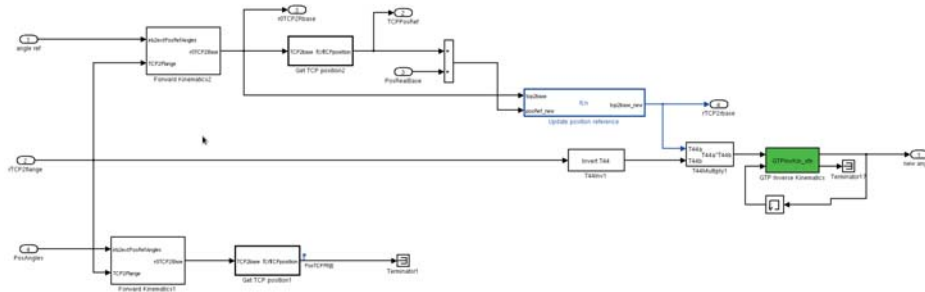


Figure 7.5: The position output from impedance controller, update the starting position, furnishing the current position of the robot. The result can be compared with a real measurement of the position from posFlt or RawFb inputs.

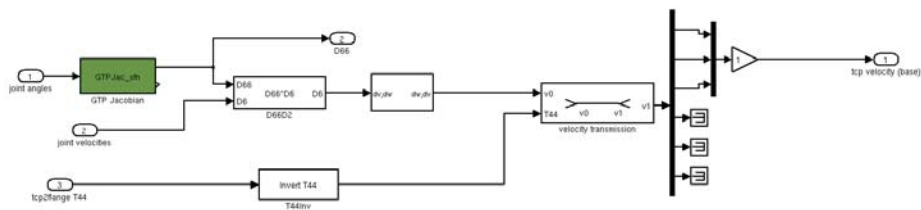


Figure 7.6: The current angles, together with the current angles's velocity, calculate the current robot velocity.

7.3.3 Wave Variables (3)

The single command signal (created in Phantom Robot Connector) is decoded in *"Wave&Integrated Wave"* block, obtaining position and velocity references (see figure 7.7). Those signals have to be converted into robot base frame before feeding the controller and it is done in *"Reference Signals"*

(number 4) block. A low pass filter is used to reduce the reflections that appears in the communication, due to wave variable. Its transfer function is expressed by 7.2

$$G_{LP}(z) = \frac{1 - e^{-\omega * h}}{z - e^{-\omega * h}} \quad (7.2)$$

where h is the discretization time (0.004s), ω is the bandwidth of the filter, equal to 10rad/s. This value can be changed by Opcom interface and, as general rule of wave transformation, it should be $\geq (1/T)$, where T is the communication delay.

Inside this block is also created the signal for Phantom device, containing force feedback information. Since several modalities of work are available, some switches are inserted to cut or modify the incoming or outgoing signals.

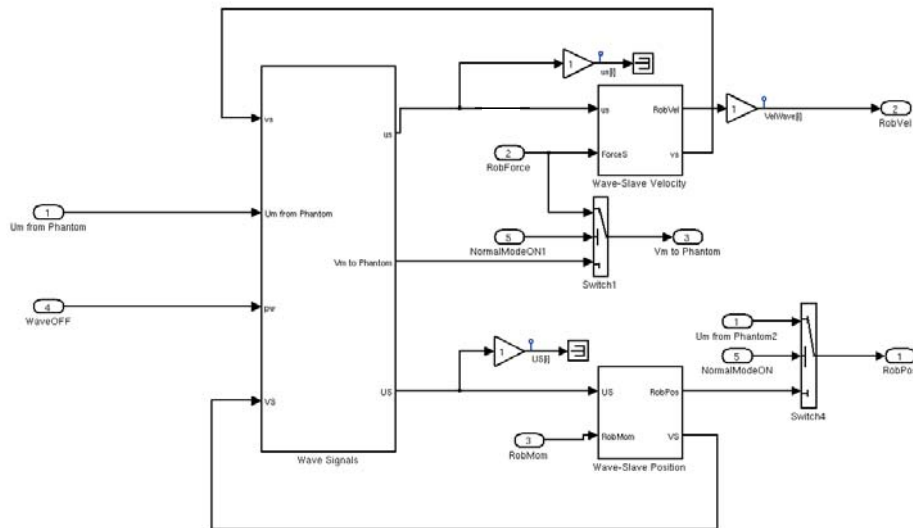


Figure 7.7: Integrated variable and variable wave are obtained from the incoming signal, and converted into position and velocity reference. Low pass filter helps the reduction of noise and reflections.

7.3.4 Reference Signals (4)

Figure 7.8 shows how the translation of incoming signals into robot base frame is made. Position and velocity, that are already rotated with respect to

the robot base frame in Phantom Robot Connector, are added respectively, to the current position and the current velocity of the robot.

When the "Joystick" mode is selected, incoming position does not feed the controller directly, but it is combined to calculate the reference velocity, as in a common joystick. Since the position reference remains stuck at the last start position, before the "Joystick" mode was activated, an apposite function has been created, in order to avoid a jump, when the normal functioning of haptic device is reactivated.

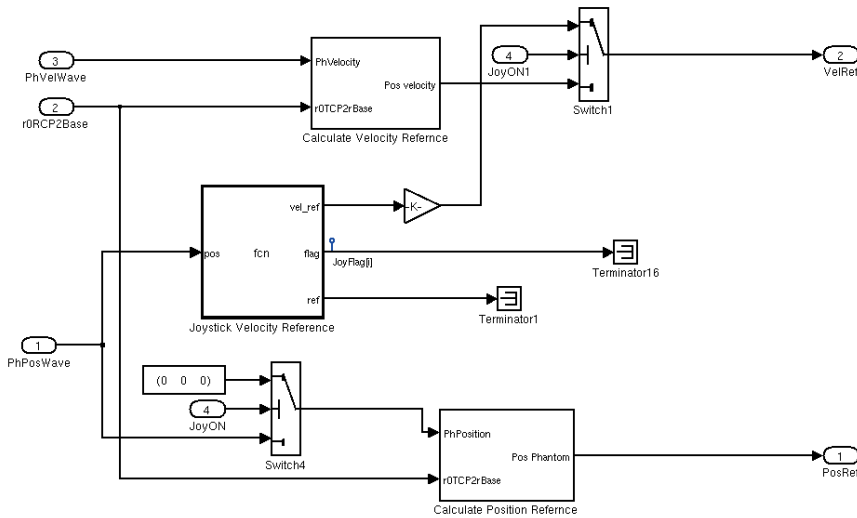


Figure 7.8: The conversion of position and velocity signal. The "Joystick Velocity Reference" uses the incoming position to impose the velocity to the robot.

7.3.5 Impedance Parameters (5)

The impedance force controller needs some parameters to work, as described in 3.7.1. Since they are responsible of the behavior of the robot during both interaction with environment and movement in the free space, it would be perfect to have different values, depending on the type of task executed:

- High mass and low stiffness, along the directions where it is expected the contact (low forces).
- Low mass and high stiffness, along the direction where the motion is free (good trajectory tracking).

The block number **5** implements this characteristic when it is simulate the interaction with a virtual wall. The parameters are set for "free-space", until the robot reaches the proximity of the wall, when the switch to "force". This is not so simple in the real case, where the position of the wall is unknown. In fact, a change of impedance values during interaction could bring the robot to the instability.

7.3.6 Delay (6)

In order to obtain an information about the communication delay, a input is added in a free channel of six dimensions port *rob2hangles*, that is not used in the controller. The time necessary to get back the input from *ph2robangles* represents the delay.

7.3.7 Virtual Wall (7)

This is the function implemented to simulate the interaction with the environment. Looking at figure 7.9, when the robot reaches the position of the virtual wall (editable by GUI) a force is produced proportionally to the depth of penetration. A derivative part is inserted to avoid oscillations. The gain of the virtual force is proportionally linked with stiffness of the surface (rigid wall=high gain; flexible wall=low gain). When the robot's TCP reaches a *limit zone* near the wall, signal *ForceRef* changes from 0 to 1 and goes to feed the *Impedance Parameters* block (number **(9)**), that switches the mass to the high value. Also the *limit zone* can be tuned from the GUI.

7.3.8 Velocity and Position Outputs (8)

Those two blocks calculate angles and angles velocities (q and \dot{q}) to feed the robot and product the real movement. They use the normal kinematics to translate the information from operative space into joint space, and hence need homogeneous transformation matrices, which have been got from "*Calculate TCP Position*" block. The switching of *jumpOFF* parameter, in GUI interface, from zero to one ensures to avoid undesired jump after using of *joystick mode*.

7.3.9 Impedance Force controller (9)

This is the main part of the controller, where the position, velocity and force information are mixed, together with *Mimp*, *Kimp* and *Dimp* constants

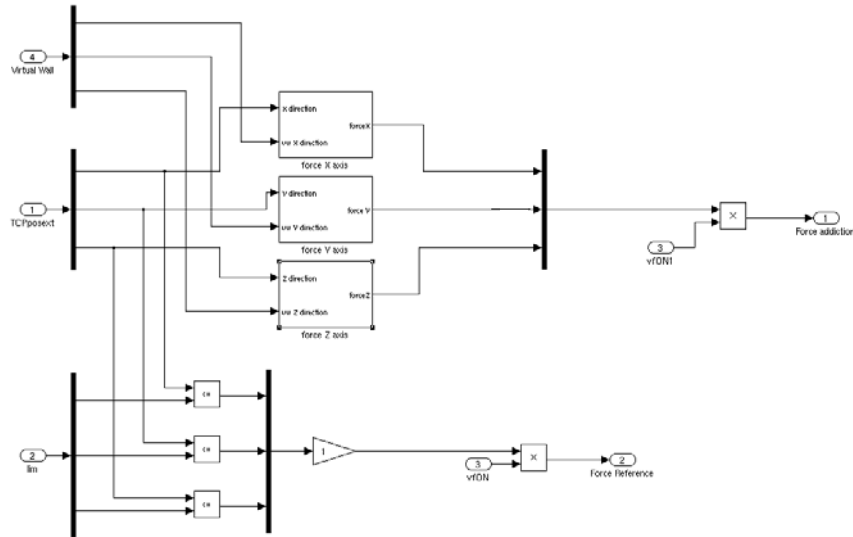


Figure 7.9: The interaction force is directly proportional with the depth of penetration. When the TCP reaches the *limit zone* mass switches to the high value.

(calculated with the apposite functions). The result is the acceleration of the robot and, after integrations, the position and the velocity (figure 7.10). The feedback force and feedback momentum returns to "*Wave Variables*" block to construct the wave and integrated wave outgoing variables. The complete code of the Impedance Controller function is available in the appendix.

7.3.10 Starting Parameters (10)

Before starting to work, it is necessary to declare values and parameters present in the model. Inside the red square, all the variables are defined and some paths (necessary for using the external library and for building the model with Real Time Workshop) are added to Matlab paths. The block has to be click once at the beginning and after the model has to be closed and reopened, in order to make the declarations effective.

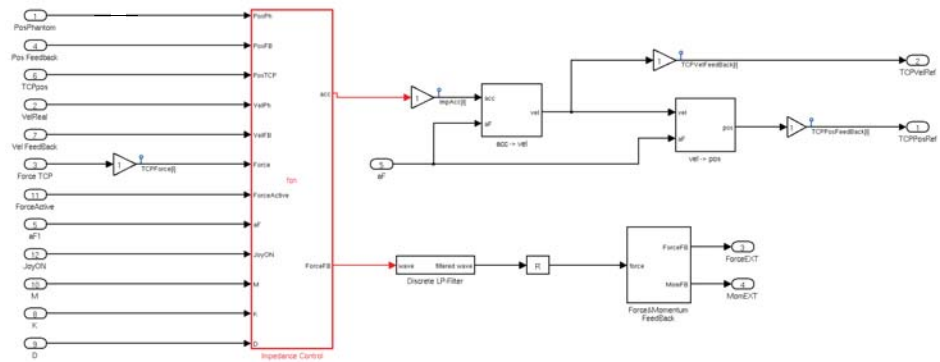


Figure 7.10: The Impedance force Controller is the main block of the model, where the acceleration, velocity and position of the robot are calculated.

8. Simulations and Results

In this chapter models and functioning are tested. Some results are obtained on IRB140B robot, since at the time of this work no force sensor was implemented on Gantry Tau Robot and it was important to do some tests with a real force measurement. In order to simulate the force interaction for the parallel robot, a virtual wall has been inserted. The joystick control has been tested only on Gantry Tau, best suited for this type of control (large movements are possible along X direction).

If the Simulink model of IRB140B is almost the same of the GT one, the other programs have to be adapted to the type of robot chosen. In particular, it is necessary to modify:

- WinComm, since the two manipulators work on different machine with different address.
- Phantom Robot Connector, where homogeneous transformation matrices, scaling matrices, position and velocity, are particular for each robot.
- Opcom, that obviously is relative on which ABB Main Computer is running.

For both manipulators, a lots of scaling matrices were tested, starting from a matrices with low norm value, up to high norm value, in order to reach the best feeling between the operator and the machine. After that, it has been analyzed the impedance controller, trying to match the best parameters. Remembering 3.39, a higher value for ω means an higher eigenfrequency, hence a faster system, but at risk of oscillations, while increasing the value of the damping ζ the robot decreases oscillations faster.

8.1 Gantry Tau Robot

All the tests report a good feeling between haptic and real robot. After many attempts, a good tuning of scaling matrices and others variables, avoid

| $Mimp_{free}$ | $Mimp_{force}$ | Ω_{free} | Ω_{force} | $Damping_{free}$ | $Damping_{force}$ |
|---------------|----------------|-----------------|------------------|------------------|-------------------|
| 10 | 200 | 15 | 6 | 0.5 | 0.5 |

Table 8.1: Parameters for Gantry Tau Robot

reflection and weird behavior of the manipulator. Those are the parameters utilized:

$$A_w = \begin{bmatrix} 14 & 0 & 0 \\ 0 & 14 & 0 \\ 0 & 0 & 14 \end{bmatrix} \quad S_w = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (8.1)$$

8.1.1 Free Space

In the free space, the manipulator follows well the reference position reference and no reflection appears during the working. For a better reading of plots, only the X -axis is controlled in all the simulations. However it is possible to implement the other axes without problems and figure 8.1 shows an example of movement in a free space along an almost circular trajectory on YZ plane.

The communication delay is approximately $0.004s$ and it has no effect during this operation, since there is no interaction with the environment.

8.1.2 Rigid Wall and Joystick Control

The behavior of the robot in presence of a force interaction is studied implementing a rigid wall along X direction. When Opcom is started the operator can choose *Joystick mode*, controlling the velocity of the robot with the Phantom device, without considering the position. This modality could be useful to move the robot from the "starting" position to the "working" zone, where the joystick mode would be deactivated. In this case, the robot interacts with the environment, and the result is presented in figure 8.2.

The blue line represents the input position signal, coming from the Phantom device. In this case, the *position reference signal* has to be updated, in order to avoid a jump during switching to *normal mode*.

At the end of the test, it is necessary to impost value 1 on GUI to *jumpOFF* variable before release the dead-man's switch. otherwise the position reference would return to the first one, causing a unwanted jump.

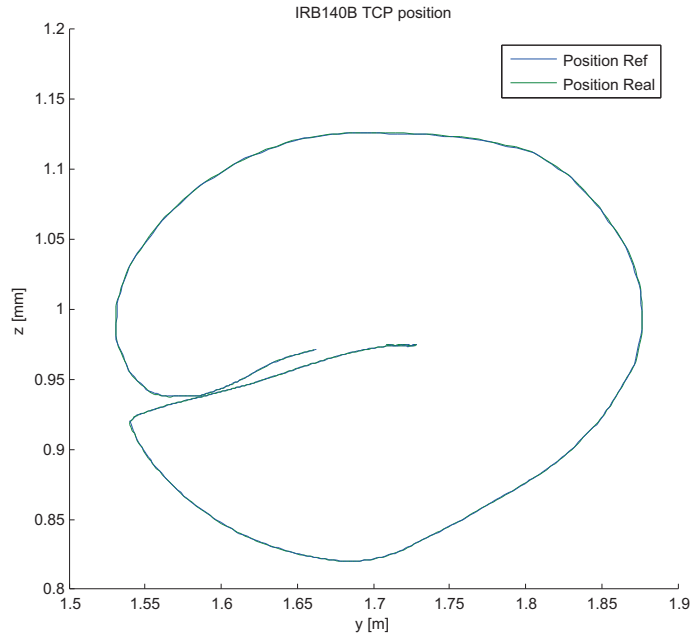


Figure 8.1: An almost circular trajectory on YZ plane.

The interaction with the virtual wall is without oscillations (figure 8.3 (a)). The operator feels clearly the interaction force with the environment and can adjust the desired trajectory. In this proofs the switch between a low and a high mass value during the contact is done, increasing the stability and the sensitivity.

8.2 IRB140B

Since this work was focus on Gantry Tau robot, not a perfect tuning of wave variables has been obtained for the serial robot. Small reflections still appear at the operator side, disturbing the feeling with the IRB140B robot. Many attempts have been done, trying to find the best parameters for impedance controller, with the target to improve both stability of the system and the "haptic sense". Increasing the force's gain, the robot will have more difficult to "penetrate" the surface, but it will be more easily subject to instability.

As was done in the last section, here the values utilized are reported. For the force controller two kind of sets are proposed: one with a low value and

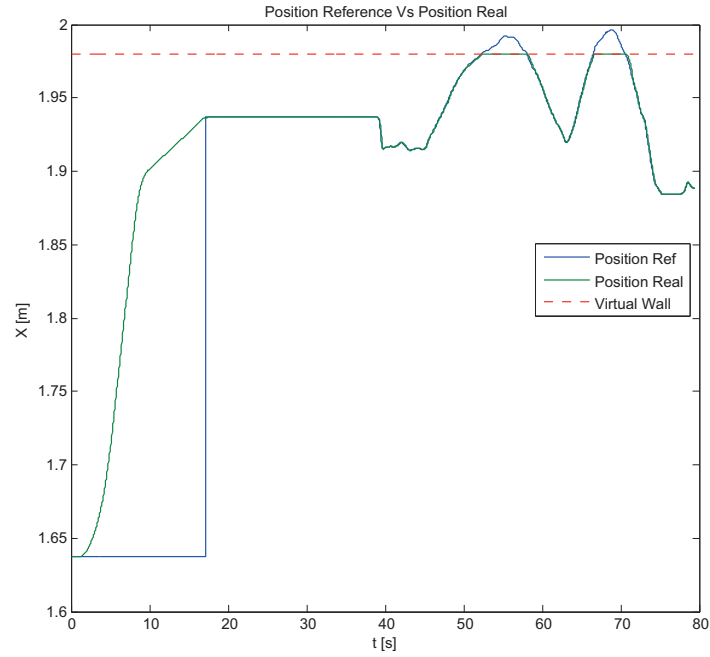


Figure 8.2: Gantry Tau is moved using joystick mode, until the "working" zone is reached, where the robot starts to work normally, interacting with a virtual surface.

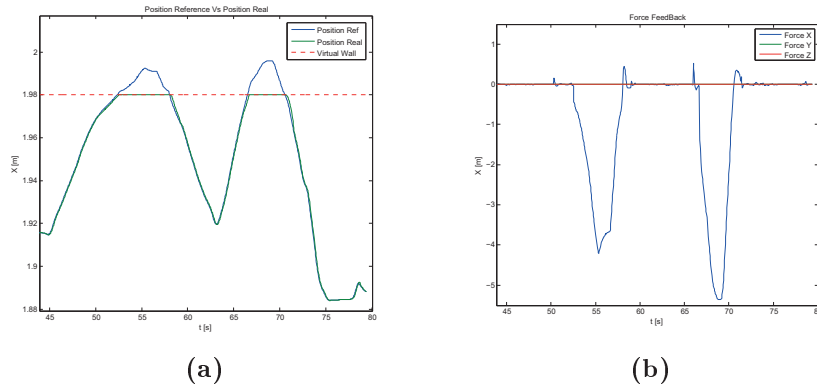


Figure 8.3: In (A), a zoom of the preceding plot shows the interaction with a virtual wall. The equilibrium on the surface is reached without oscillations and with a feedback force presents in (b).

one with an high value for force gain.

$$A_w = \begin{bmatrix} 1.5 & 0 & 0 \\ 0 & 1.5 & 0 \\ 0 & 0 & 1.5 \end{bmatrix} \quad S_w = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (8.2)$$

| $F_{gain}=30$ | | | $F_{gain}=200$ | | |
|---------------|-------|---------|----------------|-------|---------|
| Mimp | omega | damping | Mimp | omega | damping |
| 24 | 9 | 0.6 | 110 | 9 | 0.9 |

Table 8.2: Parameters for Gantry Tau Robot

In this case there is no differentiation for impedance parameters between "free" and "force" movement, since it is supposed that the operator has no information on the position of the environment.

In this case, a force sensor and a tool are installed on the flange of the robot and, hence, they modify the TCP frame. Homogeneous transformation matrices T_{sensor}^{flange} and T_{TCP}^{flange} have been determined. The result is

$$T_{sensor}^{flange} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 104.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_{TCP}^{flange} = \begin{bmatrix} 1 & 0 & 0 & 9.5 \\ 0 & 1 & 0 & 1.5 \\ 0 & 0 & 1 & 216 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (8.3)$$

Two different types of tools have been mounted on IRB140B: the first (figure 8.4(a)) presents a rubber cup on the top and it is used for static contact tests; the second (figure 8.4(b)) is a ball transfer unit and, hence, it has been utilized during rolling tests.

8.2.1 Free Space

Also in this case, a perfect tracking is obtained during free space movement 8.5. The manipulator follows well an almost circular trajectory on YZ plane, imposed by Phantom.

The communication delay is not affected by the type of manipulator selected.

8.2.2 Flexible Wall

In order to test the model on a flexible (but not too soft) wall, a carton box has been equipped with a metal sheet, as shown in figure 8.6. It creates a good compromise between flexibility and resistance.

Due to the elasticity of the wall, the gain of the force sensor has to be increased (from the GUI), in such a way to obtain a clear feedback of the force on the Phantom.

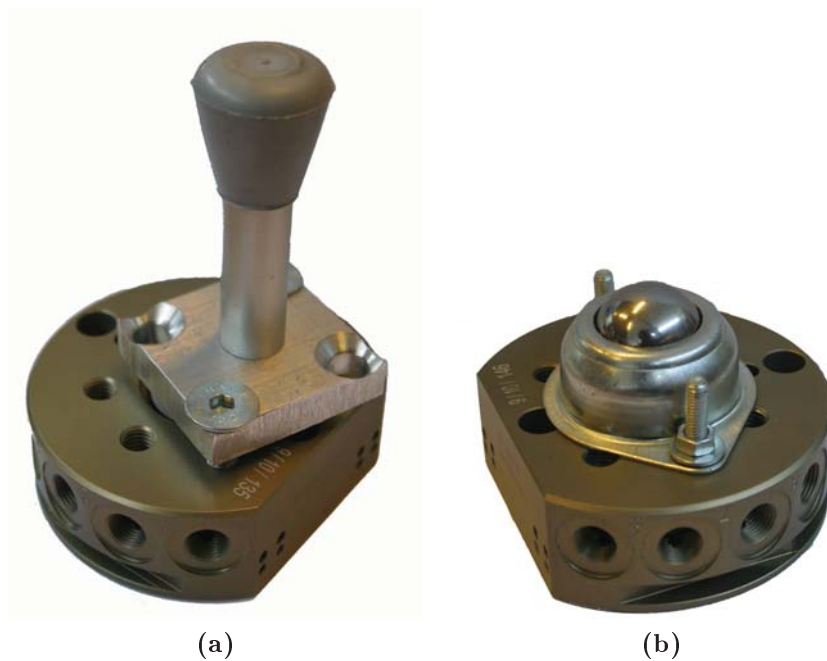


Figure 8.4: On the left, the tool mounted on IRB140B during the static contact tests, while on the right, the tool mounted on IRB140B during rolling tests.

8.2.3 Rigid Wall

The interaction with a rigid wall has been created using a steel column anchored to the ground. Two proofs have been conducted: the first one (8.7) is a pure static interaction with the wall, while in the second (8.8) the tool is moved along Z-direction during the contact.

The rubber cup on the TCP is subject to little deformations, when the robot exerts a force, but that has no effect on the stability.

The flexibility of the structure, backlashes of the robot, little deformability of the TCP and a not smooth sliding surface, create a no constant contact position, but, in any case, the system results stable in both cases. The feeling on the Phantom's pendant is good, and the operator is able to manage the interaction with the environment. The force feedback during the rolling is reported in figure 8.9. It is important to remember that the force is not read directly from the force sensor, but it is a combination of the position and the velocity (7.3.9) and hence, it is more affected by disturbances than a normal measurement.

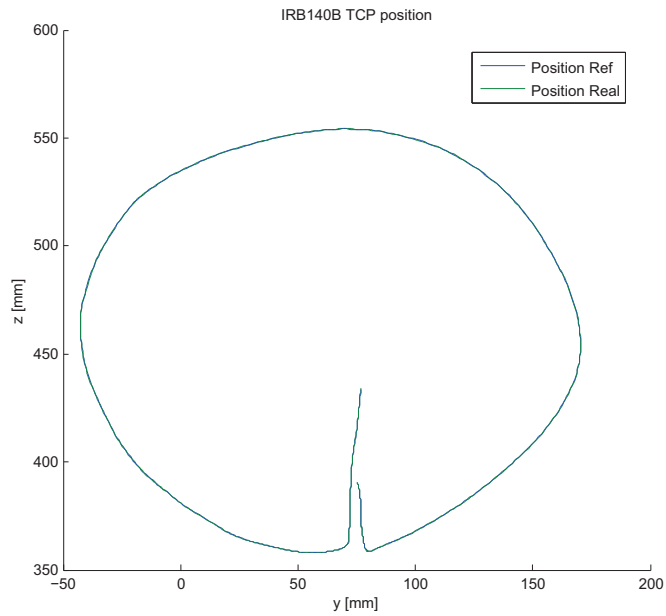
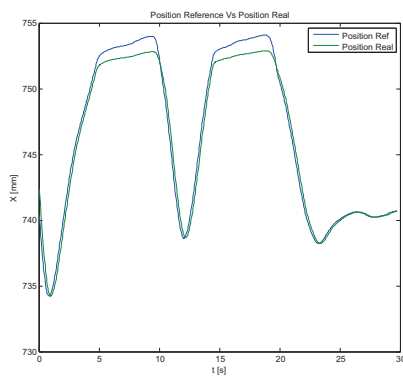


Figure 8.5: An almost circular trajectory on YZ plane is well tracked by IRB140B robot in a free space.

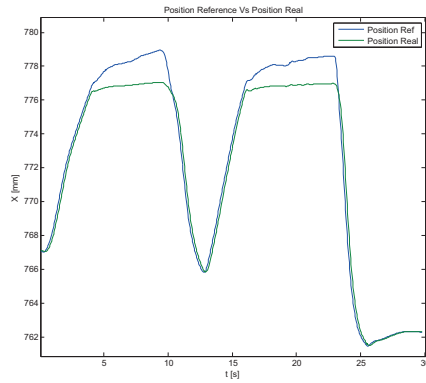


(a)



(b)

Figure 8.6: On the left the result of the proof conducted on a flexible surface, shows on the right. Due to the elasticity of the structure, the TCP deforms a little the structure, but the system results still stable.



(a)



(b)

Figure 8.7: The result (a) of the interaction with a rigid structure (b), confirms the stability of the controller.

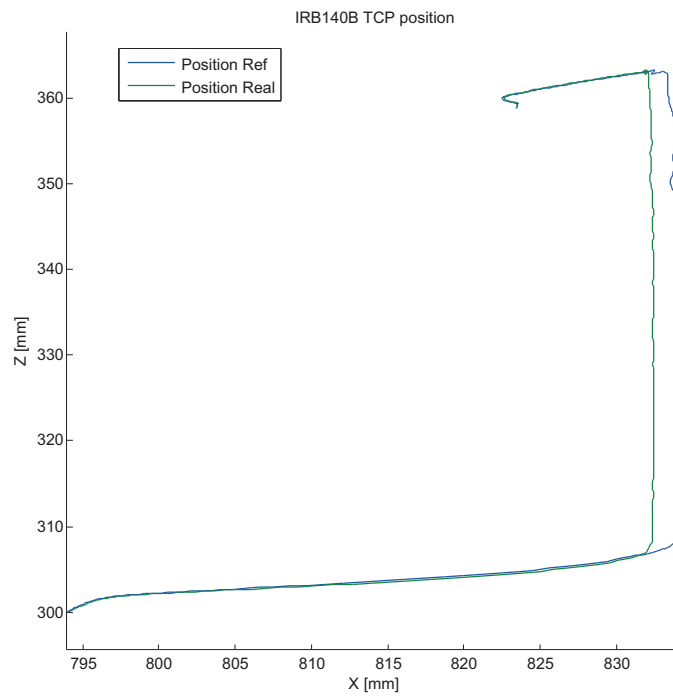


Figure 8.8: The robot is controlled on XZ plane. When the contact with the wall is reached, it is moved along Z -direction, rolling on the surface.

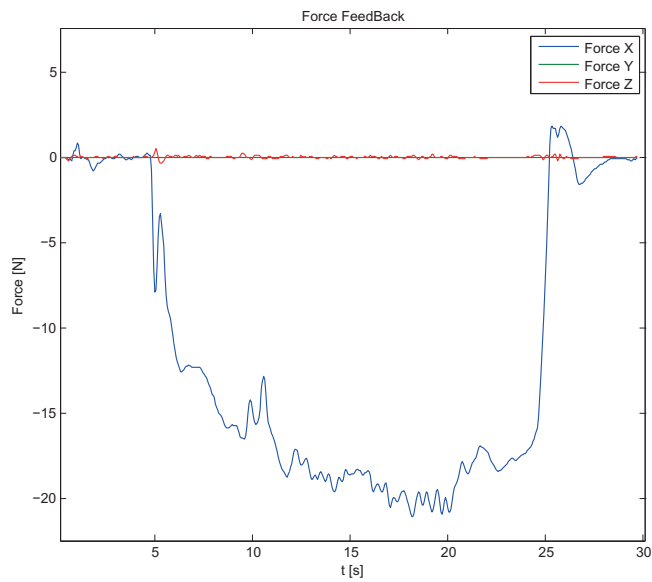


Figure 8.9: The feedback force is not a direct reading of the force sensor, but it is a combination of position and velocity.

9. Final Remarks

The aim of this thesis was to create an haptic interface for a force controlled parallel robot (*Gantry Tau*), using the wave variable method. However, Gantry Tau was not available for a long period, hence many studies have been carried out on a serial ABB robot (*IRB140B*). The controlled has been developed on a Matlab/Simulink platform, while an appropriate C++ application programme (*Phantom Robot Connector*) has been modified to allow the exchange of information between the haptic device and the real robot. A dedicated Linux programme (*WinComm*) creates a linker between Windows and the Simulink controller, that runs on the *Robot Main Computer*.

Results

After many simulations and real tests, it is possible to confirm the validity of the wave variable method. However, it is important to highlight that this method is affected by the type of manipulator being used as well as by the desired type of working. Even if this method can be applied to any type of teleoperation structure, it needs a lot of "calibrations" procedure before running (i.e. the choice of the scaling matrices). The impedance control turns out to be a good choice for the objective of this thesis, where no specific tasks had to be done, except force interactions.

Future work

Due to technical problems and to the limited time, many aspects are not developed in this work, even if a teleoperation system lends itself to a lot of applications. First of all, implementing a force sensor on Gantry Tau could allow a real interaction with the environment, that could be used for manipulation or machining tasks. In order to avoid reflection and disturbances, a new approach to wave variable could be handled, as proposed in [24], where an impedance matrix is used variable instead of a fixed one. In this case, a new version of Phantom Robot Connector would be necessary. As reported in chapter 4, the structure created in this work permits movements only

along the x, y and z axes (in robot base frame). It could be interesting to implement a control for all the degrees of freedom of Gantry Tau (5 DOFs). Finally, many other force controllers could be implemented, instead of the impedance controller, making possible, for example, the direct force control.

Discussion

I'm going to insert a self-evaluation of the results of this thesis. At the end of the work, the Wave Variable Method utilized has shown good performance with respect to the instability, especially to avoid oscillations. However this result was not easy to obtain. The choice of the scaling matrices was very stressful and slow. All the mathematical transformations, apart from creating three separate paths (through which signals may travel to the operator, disturbing the control), they also cause a separation between the input signal in the controller and the real behavior of the robot: the operator loose the direct contact with the reality and each modification to improve stability or to decrease reflection is not more so obvious. However, this method reaches good results, with a fast stabilization when the robot enters in contact with the environment, also with respect to precedent work [17], where oscillations were persistent. The last doubt, that for time's problem I was not able to dispel, is validation of the method with a real imposed communications delay. In fact, it could be interesting to understand the "stability limit" for the system, with a delay different with respect to the one present in the automatic control laboratory.

Acknowledgements

If I look back I understand how I have learnt and how I am changed during these years and, especially, after this months in Lund. This thesis represents the most rewarding result that I could imagine and it signs not only the end of my studies, but also of a part of my life.

Reached the end of my work, there are many people that I have to thank, first of all, my supervisor, Anders Robertsson, who gave me the possibility to conclude my academic career with a so great experience. He was always helpfully and ready to solve with me all the troubles encountered on the way, despite his busy days. A special thanks to all my family that always believed in me, remaining on my side for all the time. Even when everything went wrong, they were present, encouraging and supporting me. A particular though to my friends, who have supported me during bad times, giving me a smile when I needed it. Many thanks to Anders Blomdell, for the help with communications and compiling problems. He had always a precious solution. Thanks the whole automatic control department for the support and the structures offered during all this time. I want also to say thank you to my italian tutors, Prof.Michele Basso and Prof.Luigi Chisci. They have ensured me the more complete support for my thesis and they have offered their help to create this final work. All the people I encountered on the road helped me to become what I am today, and I want to say thanks to each of them.

A. Simulink's function

Impedance Parameters

```
function [Mimp,Kimp,Dimp]= fcn(Mimp_free,Mimp_force,Omega_free,
                               Omega_force,Damping_free, Damping_force,
                               ForceRef)

M = zeros(1,3);
Kimp = zeros(1,3);
Dimp = zeros(1,3);
Mimp = zeros(1,3);
omega = zeros(1,3);
damping = zeros(1,3);

for i=1:3
    if ForceRef(i)==1
        M(i) = Mimp_force(i);
        omega(i) = Omega_force(i);
        damping(i) = Damping_force(i);
    else
        M(i) = Mimp_free(i);
        omega(i) = Omega_free(i);
        damping(i) = Damping_free(i);
    end
    Mimp(i)=M(i);
    Kimp(i) = M(i)*omega(i)*omega(i);
    Dimp(i) = 2*M(i)*omega(i)*damping(i);
end
end
```

Impedance Force Control

```
function [acc,ForceFB] = fcn (PosPh,PosFB,PosTCP,VelPh,VelFB,Force,
                              ForceActive,aF,JoyON,M,K,D)
```

```

ForceFB=zeros(1,3);
DeltaPos=zeros(1,3);
DeltaVel=zeros(1,3);
acc=zeros(1,3);
PosOFF= not(JoyON);

for i=1:3
DeltaVel(i)= VelFB(i)-VelPh(i);
DeltaPos(i) = (PosTCP(i)+PosFB(i))-PosPh(i);
end

for i=1:3
    if aF==0
        acc(i)=0;
    else
        acc(i) = (Force(i)*ForceActive(i)-DeltaPos(i)*K(i)+
                *PosOFF-DeltaVel(i)*D(i))/M(i);
    end
    ForceFB(i)=(DeltaPos(i)*K(i)+DeltaVel(i)*D(i))*PosOFF;
end
end

```

Joystick

```

function [vel_ref,flag,ref] = fcn (pos)

vel_gtp=(0.000:0.001:0.01);
pos_hp=pos;
ref=abs(pos_hp)*1000;
vel_out=zeros(1,3);
flag=zeros(1,3);

for i = 1:3
    if (ref(i) < 10)
        vel_out(i)=sign(pos_hp(i))*vel_gtp(1);
        flag(i)=1;

    elseif (ref(i) >= 10) && (ref(i) < 20)
        vel_out(i)=sign(pos_hp(i))*vel_gtp(2);
        flag(i)=2;
    end
end

```

```
elseif (ref(i)>=20) && (ref(i)<30)
    vel_out(i)=sign(pos_hp(i))*vel_gtp(3);
    flag(i)=3;

elseif (ref(i) >= 30) && (ref(i) < 40)
    vel_out(i)=sign(pos_hp(i))*vel_gtp(4);
    flag(i)=4;

elseif (ref(i) >= 40) && ( ref(i) < 50)
    vel_out(i)=sign(pos_hp(i))*vel_gtp(5);
    flag(i)=5;

elseif (ref(i) >= 50) && ( ref(i) < 60)
    vel_out(i)=sign(pos_hp(i))*vel_gtp(6);
    flag(i)=6;

elseif (ref(i) >= 60) && ( ref(i) < 70)
    vel_out(i)=sign(pos_hp(i))*vel_gtp(7);
    flag(i)=7;

elseif (ref(i) >= 70) && ( ref(i) < 80)
    vel_out(i)=sign(pos_hp(i))*vel_gtp(8);
    flag(i)=8;

elseif (ref(i) >= 80) && ( ref(i) < 90)
    vel_out(i)=sign(pos_hp(i))*vel_gtp(9);
    flag(i)=9;

elseif (ref(i) >= 90) && ( ref(i) < 100)
    vel_out(i)=sign(pos_hp(i))*vel_gtp(10);
    flag(i)=10;
else
    vel_out(i)=sign(pos_hp(i))*vel_gtp(11);
    flag(i)=11;
end
end
vel_ref=vel_out;
end
```

Position Reference

```
function tcp2base_ref = fcn(tcp2base,posRef_new,UpPos)

flag=0;
tcp2base_ref = tcp2base;

for i=1:3
    if UpPos(i)~=0;
        flag =1;
    else
        flag=0;
    end
end

if flag==1
    tcp2base_ref(4) = posRef_new(1)+UpPos(1);
    tcp2base_ref(8) = posRef_new(2)+UpPos(2);
    tcp2base_ref(12) = posRef_new(3)+UpPos(3);
else
    tcp2base_ref(4) = posRef_new(1)+tcp2base(4);
    tcp2base_ref(8) = posRef_new(2)+tcp2base(8);
    tcp2base_ref(12) = posRef_new(3)+tcp2base(12);
end
```

Get TCP Position

```
function TCPposition = fcn(TCP2base)

TCPposition = [TCP2base(4) TCP2base(8) TCP2base(12)];

end
```


Bibliography

- [1] The abb group. <http://www.abb.com>.
- [2] The jr3 company. <http://www.jr3.com/>.
- [3] Matlab and simulink. <http://www.mathworks.com/products/matlab>.
- [4] Matlab and simulink. <http://www.mathworks.com/products/simulink>.
- [5] Real time workshop. <http://www.mathworks.com/products/rtw>.
- [6] The sensible company. <http://www.sensible.com>.
- [7] The the automatic control department of the lth university of lund. <http://www.control.lth.se/>.
- [8] M. Alise, R.G. Roberts, and D.W. Repperger. The wave variable method for multiple degree of freedom teleoperation systems with time delay. *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2908–2913, 2006.
- [9] M. Alise, R.G. Roberts, D.W. Repperger, and C.A. Moore. A haptic teleoperation study using wave variables and scaling matrices. *2009 IEEE International Conference on Systems, Man and Cybernetics*, pages 1–5, 2009.
- [10] Marc T. Alise. *Expansion and implementation of the wave variable method in multiple degree-of-freedom systems*. PhD thesis, Tallahassee, FL, USA, 2007. AAI3301518.
- [11] R. Anderson and M. Spong. Hybrid impedance control of robotic manipulators. *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, 4:1073–1080, 1987.

- [12] N. Andreff and I. Dressler. Closed-form calibration of the gantry-tau parallel robot. *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 993–998, 2008.
- [13] David Daney. Kinematic calibration of the gough platform. *Robotica*, 21(6):677–690, 2003.
- [14] C. A. Desoer and M. Vidyasagar. *Feedback systems: Input-output properties*. 1975.
- [15] I. Dressler, M. Haage, K. Nilsson, R. Johansson, A. Robertsson, and T. Brogardh. Configuration support and kinematics for a reconfigurable gantry-tau manipulator. *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 2957–2962, 2007.
- [16] Isolde Dressler. *Force control interface for ABB S4/IRC5*. LTH, 2009.
- [17] Fredrik Eriksson and Marcus Welander. Haptic interface for a contact force controlled robot. Master’s thesis, May 2009.
- [18] Johan Friman. Kinematics and force control for a gantry-tau robot. Master’s thesis, Department of Automatic Control, Lund’s University, February 2010.
- [19] N. Hogan. Impedance control: an approach to manipulation. *Journal of Dynamic Systems, Measurement, and Control*, 1985.
- [20] Baron L. and Angeles J. The direct kinematics of parallel manipulators under joint-sensor redundancy. *Robotics and Automation, IEEE Transactions on*, 16(1):12–19, 2000.
- [21] M. Murray, G. Hovland, and T. Brogardh. Collision-free workspace design of the 5-axis gantry-tau parallel kinematic machine. *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2150–2155, 2006.
- [22] Gunter Niemeyer and Jean-Jacques E. Slotine. Stable adaptive teleoperation. *IEEE Journal of oceanic Engineering*, 23(9):873–890, January 2004.
- [23] Anderson R.J. and Spong M.W. Bilateral control of teleoperators with time delay. *Decision and Control, 1988., Proceedings of the 27th IEEE Conference on*, pages 167–173 vol.1, 1988.

- [24] E.J. Rodriguez-Seda and M.W. Spong. A time-varying wave impedance approach for transparency compensation in bilateral teleoperation. *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4609–4615, 2009.
- [25] Chiaverini S., Siciliano B., and Villani L. Parallel force/position control with stiffness adaptation. *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, 2:1136–1141 vol.2, 1997.
- [26] Munir S. and Book W.J. Internet based teleoperation using wave variables with prediction. *Advanced Intelligent Mechatronics, 2001. Proceedings. 2001 IEEE/ASME International Conference on*, 1:43–50 vol.1, 2001.
- [27] Lorenzo Sciavicco and Bruno Siciliano. *Robotica Industriale*. McGraw-Hill, 2001.
- [28] SensAble Technologies, Inc., <http://www.sensable.com>. *OpenHaptics Toolkit*, 2008.
- [29] SensAble Technologies. *OpenHaptics® Toolkit, API Reference*, 2008. version 2.0.
- [30] SensAble Technologies. *OpenHaptics® Toolkit, Programmer's Guide*, 2008. version 3.0.