Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione
Corso di Laurea in Ingegneria dell'Automazione

Sensorless Haptic Force Feedback for Telemanipulation using two identical Delta Robots.

Laureanda:
Roberta Lazzari
Matricola:
1107501

Relatore:
Ch.mo Prof. Ruggero Carli
Ch.mo Prof. Einar Nielsen

Anno accademico 2016/2017

Tù ne quaèsierìs, scìre nefàs, quèm mihi, quèm tibi
fìnem dì dederìnt, Lèuconoè, nèc Babylònios
tèmptarìs numeròs. Ùt meliùs, quìdquid erìt, pati,
sèu plùris hiemès sèu tribuìt Iùppiter ùltimam,
quaè nunc òppositìs dèbilitàt pùmicibùs mare
Tyrrhenùm: sapiàs, vìna liquès èt spatiò brevi
spèm longàm resecès. Dùm loquimùr, fùgerit ìnvida
aètas: càrpe dièm, quàm minimùm crèdula pòstero.
Orazio, Odi, I,11.

Abstract

Bilateral teleoperation systems with haptic feedback are becoming one of the key technology of the future. This technology allows human users to interact with objects or perform complex tasks in remote or inaccessible environments by providing the operator with haptic feedback. The ideal bilateral control of the haptic system should achieve transparency i.e. the correspondence of positions and forces between the master and the slave; but communication delays endangers system stability leading to degraded system performance and poor user experience. In this work two different control scheme have been developed. A first bilateral position-position control scheme with PD controller with gravity and passivity compensation has been implemented but the results were not satisfactory. Then a bilateral force sensorless acceleration control implemented with Kalman filters and disturbance observers has been developed. The estimate of position, velocity and external torque applied to the end effector and the acceleration control lead to a faster and ready to move system. Thus high transparency and good perception of the environmental stiffness was achieved. Both methods have been tested in two identical DELTA robots completely developed at the NTB, University of Applied Sciences in Technology, Buchs and St. Gallen, Switzerland.

Contents

List of Figures

Chapter 1

Introduction

"I, on the other hand, am a finished product. I absorb electrical energy directly and utilize it with an almost one hundred percent efficiency. I am composed of strong metal, am continuously conscious, and can stand extremes of environment easily. These are facts which, with the self-evident proposition that no being can create another being superior to itself, smashes your silly hypothesis to nothing."
Isaac Asimov; I, Robot.

The word robot was coined by Czechoslovakian fiction author Karel Čapek. He introduced the word in a play called R.U.R, Rossum's Universal Robots, which debuted in January of 1921. While writing this play, he struggled to come up with a word to name the robots, initially settling on "laboři", from the Latin "labor" and then "roboti", which derives from the Old Church Slavanic "rabota", meaning "servitude". Even if in the play the robots were not mechanical but biological beings more or less indistinguishable from humans in appearance, created to do work for humans and even with full cognitive abilities; the word "Robot" become current to indicate something different. There are many definitions for it but none of those is sufficiently accurate. Peter Corke refers to it as "a goal oriented machine that can sense, plan and act". In particular a robot senses its environment and uses that information, together with a goal, to plan some action.

Since in the last years robots have been used widely in many different applications (industry, medicine, rehabilitation and so on) a general desire of providing robotic systems with enhanced sensory capabilities had spread. It's important to underline that adopting a purely motion control strategy in order to manage the interaction of a robot with the environment turns out to be inadequate. The unavoidable modeling errors and uncertainties may cause a rise of the contact forces ultimately leading to an unstable behavior during the interaction. In this way force feedback has a main role in assist human operator while handling objects in the remote way with a slave manipulator. The force control problem is tackled as the natural evolution of the motion control problem, where feedback of the contact force as provided by a force/torque sensor or estimated from the position's measurements, is used to menage the interaction of a robot manipulator with a scarcely structured environment. Control of interaction between a robot manipulator and the environment is crucial for successful execution of a number of practical tasks where the robot's end effector has to manipulate an object or perform some operation such as exploration. Successful execution of an interaction task with the environment by using motion control could be obtained only if the task was accurately planned. This would in turn require an accurate model of both the robot manipulator (kinematics and dynamics) and the environment (geometry and mechanical features). Manipulator modeling can be known with enough precision, but a detailed description of the environment is difficult to obtain. As long as the tasks are characterized by very structured and repeated motion sequences in well known environments, fully automated robots can be used to interact with the environment in a pre-programmed manner. But if the task requires to operate in variable, unstructured, unknown, and dynamic working environments, the robot has to adapt to sudden changes by taking decisions and adapting plans appropriately. In this way for that applications in which it is more important to precisely control the force applied by the end-effector rather than controlling the robots positioning it is necessary to search a control law that gives back a sense of touch and a tactile perception. On this account, so called telepresence and teleaction systems have been developed, which combine skills as human adaptability and decision-making ability with the advantages of robotic manipulation. Using such a teleoperation system the human operator is not any more in direct contact with the environment, but interacts with it by means of technical systems. In doing so, the human operator uses a human-system interface, which allows her/him to control a robot, the teleoperator, that interacts in her/his place with the environment.
As shown in figure (1.1), a telemanipulator is a complex electro-mechanical system consisting of a master (or local) and a slave (or remote) device, interconnected by a communication channel. The overall system is interfaced on one side (the master) with a human operator, and on the other (the slave) with the environment. Hereby, the human-system interface provides multi-modal feedback in form of visual, auditory, and haptic

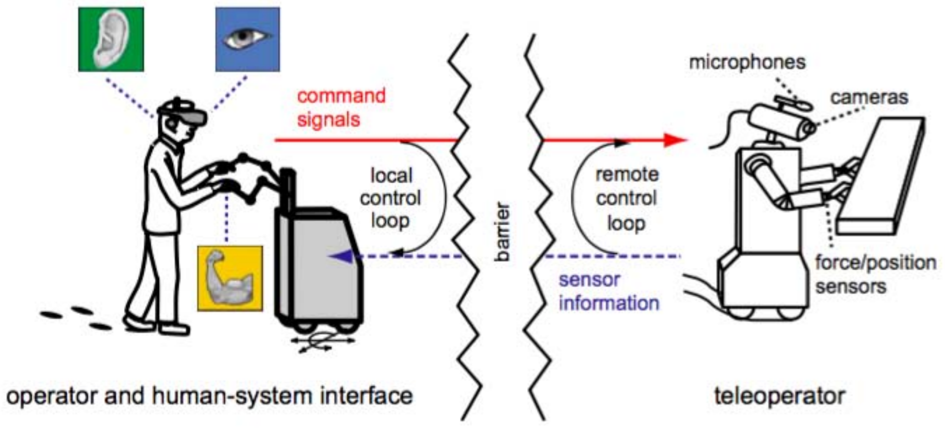


Figure 1.1: A bilateral telemanipulation system.

information and is used as control input for the teleoperator, which executes the commands. All command signals and sensory information is hereby transmitted over an appropriate communication channel. Ideally, the human operator should not perceive any difference between direct interaction with the remote environment and interaction via a teleoperation system. In this ideal case, the system is called transparent and the corresponding measure transparency [77, 139]. If the system is transparent and thus the human operator is not restricted by the telerobotic system in any sense, an intuitive interaction with the remote environment is possible. In order to achieve such a high-quality teleoperation, providing multi-modal feedback is hereby of special importance. While the visual and auditory modality are already rather advanced and several high quality devices are available on the market, the haptic modality needs further attention.
There can be two different type of telemanipulation, unilateral or bilateral respectively, depending on the number of channel in the communication line. The former employs a one way channel which connects an "active" slave and a "passive" master. Since the master is not provided by actuators but encoders, it is only able to send position data to the slave which will reproduce the same motion. However a limit can be placed on the absence of feedbacks from the environment where the slave is working leading to possible damages on the manipulator. In order to solve the problem an addictional communication channel from the slave to the master is added and bilateral telemanipulation is achieved. In this case the interaction between the slave and the unknown environment is sent to the master providing in this way a sense of touch to the operator which can simplify the achievement of the task. So in the latter case both master and server have actuators and encoders and the communication is bilateral. This can be obtained through force sensors attached to the slave or sensorless estimating the velocity and the force. Current examples of teleoperation span from space applications to rehabilitation, to surgery and to Unmanned Air/Ground Vehicles (UAV, UGV), to hazardous environment to underwater.

Robots are used in space for exploration and scientific experiments and the main reasons of using them are related to high costs of human operator and the hostile and dangerous environment for human beings. As shown in figure (1.2) main directions of current research activity are the development of:

1. arms for intra-vehicular and extra-vehicular activities (ESA, NASA,...),
2. free flying platforms,
3. planetary rovers, space exploration vehicles designed to move across the surface of a planet or other celestial body;
4. robonauts, humanoid robots which can function as an equivalent to humans during extra-vehicular activity (space walks) and exploration.

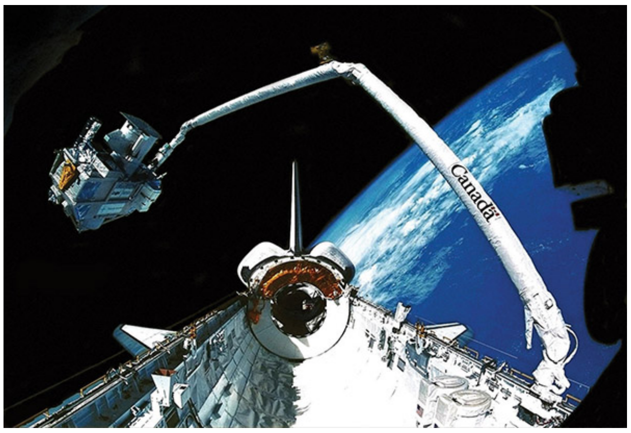
Due to important technical problem peculiar to water environment such as high pressures, poor visibility and communication difficulties, it is convenient to use telepresence underwater. Main use are:

communication and oil industries which require underwater pipes and cable;

scientific community which uses this technology for marine, biological, geological, and archeological mission;

military which use telerobotics in many savage operation.

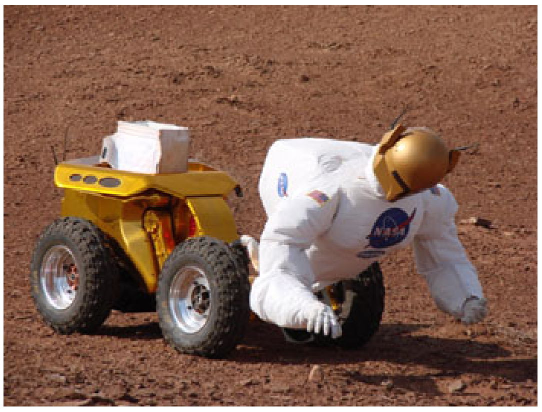
It is clear that telerobotics device for underwater are more affordable and more safe in many application.



(a) *Canadian Shuttle Remote Manipulator System: arm installed on the US space-shuttle to deploy, maneuver and capture payloads.*

(b) *Opportunity, a robotic rover on the planet Mars, active since 2004.*

(c) *Rollin' Justin, autonomous and programmable humanoid robot with two arms developed by the German Aerospace Center (DLR).*



(d) *Robonaut 1, developed by NASA, uses telepresence and has hands with fourteen degrees of freedom and use touch sensors at the tips of its fingers.*

Figure 1.2: Some space applications.

Another important branch of telepresence applications is medicine with telesurgery. Main application of robot manipulation in the medical field are:

- help to impaired people,
- surgery operations (surgery where small precise movement are needed, minimal invasive procedures, remote surgery),
- diagnose illness or injures when the patient is far away (for example militar soldiers),
- training of specialised personnel.

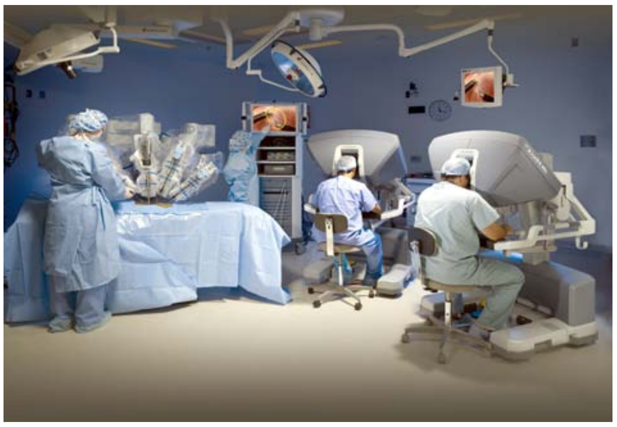


Figure 1.3: The da Vinci Robot, the most commonly used instrument for telesurgery.

Summarizing, bilateral teleoperation is a key technology to allow humans to interact with remote environments by providing the operator with haptic feedback. Haptic feedback improves human perception and therefore the quality of the human-robot interaction, but on the other hand it can tamper with the stability of the system when the communication between the master side and the slave side is not instantaneous but affected by delay and packet loss (drops).

In this work haptic sensorless force feedback has been studied and applied on two telemanipulated interacting DELTA robots.The aim of the work is to find a control law that implicates stability and transparency between the two devices. To implement the control an open source software framework for robotic EEROS has been used. EEROS which is in its early stages, was developed at NTB University of Technology, Buchs & St. Gallen, Switzerland where this thesis has been pursued. As a first step a simple bilateral position-position control scheme with PD controller with gravity compensation has been implemented. In these architecture the path of the master robot, which is established by the operator, is used as reference trajectory for the slave robot. This has been achieved by sending the position reached by the master directly to the server in a reversible way such that both robots can be the master. Force reflection is obtained as a result of the actuation produced by the controller when tracking error grows due to the interaction between the slave robot and the contacted environment. With this first application a teleoperation system bilaterally controlled has been studied, but several problems due to time delay and stability have been reached. As a consequence of this problem a passivity based teleoperation has been achieved; in these, the stability of a position-position bilateral teleoperation is guaranteed by adding a passive term to the PD controller at each side of the architecture. Even though a good stability level could be reached thank to the passivity term introduced, the time delay was still a reason of low performance by making the end effector stiff to move. Finally another solution has been developed: a force sensorless control implemented with Kalman filter based state observer (KFSO). The latter can be applied in order to estimate the position, the velocity and the external torque applied to the end effector. Compared to the previous control scheme, the bilateral control system based on KFSO offers the advantage of wider bandwidth and a faster response free from sensor noise. Thus it is possible to achieve high transparency and good perception of the environmental stiffness even with high time delays. This thesis is organized in nine different sections. In section 2 there is a survey about haptic interfaces and in section 3 the instrumentation used. In the fourth a modeling of 3D parallel mechanism delta robot is described in order to evaluate the kinematics and the dynamics of the device. In section 5 the software used is presented and in section 6 the basic software components are described i.e. the parts of the control scheme which are common to both the application analyzed. Finally in section 7 the first architectural position position control scheme is presented and in section 8 it is described the bilateral acceleration control using Kalman filters and disturbance observer. Section 9 present the experimental results of both control scheme in comparison and the last section 10 is about conclusion and future work.

Chapter 2

Haptic devices and state of the art

As already seen in the previous section telerobotic systems usually involve a slave robot, which interacts with the remote environment, and a master console, operated by a human operator. The slave robot reproduces the movements of the operator, who in turn needs to observe the remote environment with which the robot is interacting. The latter can be achieved by a combination of visual and haptic cues that flow from the environment to the operator. Visual feedback is already available in several popular telerobotic systems (e.g., the Intuitive Surgical da Vinci Si and the Space Shuttle Canadarm figured above), but current teleoperated systems have very limited haptic feedback. This omission is related to many different factors. One of the most relevant is the negative effect that haptic feedback has on the stability of teleoperation systems. Haptic force feedback can in fact lead to undesired oscillations of the system, which interfere with the operation and may be dangerous for both the environment and the human operator.

Haptic comes from the Greek haptesthai which means "to touch". The sense of touch is one of the most informative senses that human posses. Mechanical interaction with a given environment is vital when a sense of presence is desired, or when a user wishes to manipulate objects within a remote or virtual environment with manual dexterity.[1] Haptics is the science of applying tactile sensation to human interaction with computers. A haptic device is one that involves physical contact between the computer and the user, usually through an input/output device, such as a joystick or data gloves, that senses the body's movements. With the incorporation of haptic feedback into virtual or remote environments, users have the ability to push, pull, feel,and manipulate objects in virtual space rather than just seeing a representation on a video screen. In this way the operator can not only feed information to the computer but can receive information from the computer in the form of a felt sensation on some part of the body. The haptic force reflecting interface is the robotic device that allows the user to interact with a virtual environment or teleoperated remote system. The haptic interface consists of a real-time display of a virtual or remote environment and a manipulator, which serves as the interface between the human operator and the simulation. The user moves within the virtual or remote environment by moving the robotic device. Haptic feedback, which is essentially force or touch feedback in a man-machine interface, allows computer simulations of various tasks to relay realistic-tangible sensations to a user.[1] The typical "Haptic devices" as they are sold commercially provide a mechanical I/O device with which a user interacts. The device will track one or more end effectors in physical space and provide force and/or torque feedback (a bidirectional channel of interaction between a virtual environment and user). The reproduction of the haptic information does not permit the time delay. Furthermore, to reproduce vivid sensation, the haptic information should have wide bandwidth. In order to reproduce the haptic information, synchronism and interactivity are very important. [21] For this reason it can be said that the haptic information is a kind of bilateral environmental information. [21] There is much research about bilateral control in order to attain the tactile feedback from the remote environment. One of the major objectives in designing bilateral control systems is achieving transparency which is defined as a correspondence of positions and forces between master and slave (Yokokohji and Yoshikawa, 1994), or a match between the impedance perceived by the operator and the environment impedance (Lawrence, 1993). It has been pointed out that bilateral controllers can not preserve transparency and stability simultaneously because of the presence of uncertainties in the system and the environment. It has been considered that there is a compromise between these two goals (Hashtrudi-Zaad and Salcudean, 2002a,b). In the conventional bilateral control, much research has paid attention to develop novel control architectures, and implemented force sensors to detect external force. Since a force sensor uses a strain gauge, it has a soft portion on its structure (Li and Chen, 1998). Narrow bandwidth of reaction force sensing has a big influence not only on the force transmission but also on the stability of whole bilateral control system (Katsura, Matsumoto, and Ohnishi, 2003). Generally, in order to solve the instability in force control systems, a viscous term has been enlarged (Chiaverini, Siciliano, and Villani, 1999). However, robot's response becomes slow due to large viscosity. Therefore, since a system becomes unstable with small viscous term and robot's response becomes slow with large one, there is a trade-off between the stability and the transparency.[21]

Hannaford express ideal condition of bilateral control by using hybrid matrix (Hannaford,1989). Yokokohji et al. proposed control system to ensure transparency in the bilateral control (Y. Yokokohji and T. Yoshikawa,1993). Bilateral control makes reproduction of "law of action and reaction" between environments and operators possible artificially. Katsura et al. proposed bilateral control system based on acceleration control by using modal transformation to common and differential mode (S. Katsura, W. Iida and K. Ohnishi,2005). Because operators can manipulate objects with force feedback from remote environments, bilateral control is expected as a promising technology for safe tele-manipulation in telemedicine systems or hazardous environments where human beings cannot enter.[21]
Haptics has any applications but the following are the most important for future research work.

- A haptic display can help a blind computer user interact with graphics-based operating systems. Computer interface for blind users:
  - Programmable Braille;
  - Access to GUIs (Graphical user interfaces);
- Entertainment:
  - Arcade (steering wheels);
  - Home (game controllers);
- Visual display alone is not sufficient for certain types of virtual environments. To learn physical skills, such as using complicated hand tools, haptic information is a requirement. Training:
  - Medical Procedures (rehabilitation, teleoperation, telemedicine);
  - Astronauts;
- Automotive:
  - BMW "iDrive";
  - Haptic Touchscreens;
- Education;
- Mobile Phones: Immersion "Vibetonz";
- Virtual Prototyping;
- Animation/Modeling;
- Art;
- Material Handling: Virtual Surfaces.

Finally, looking across applications, it is possible to find common motivations: [2]

Haptics is required to solve a problem (e.g.,Interfaces for the blind),

Haptics improves realism and sense of immersion (Entertainment),

Haptics provides constraint (Prototyping).

Chapter 3

Instrumentation

The overall equipment is constituted by a computer and two DELTA robots.

The **computer** that has been used was running a Virtual Machine (Oracle Virtual Box) for Windows. The VM allowed to run Ubuntu 14 as operating system and Kdevelop has been used as Development Environment. In the VM the framework EEROS has been downloaded from the website (www.eeros.org) and installed. EEROS, as will be described better in the following section, is the Easy, Elegant, Reliable, Open and Safe Real-Time Robotics Software Framework that allows real-time application in the robotic field without losses in safety. PuTTY a free open source terminal emulator, serial console and network file transfer application, supports several network protocol such as SSH that has been used in order to have a serial connection between the computer and the deltas. SSH is a cryptographic network protocol for operating network services securely over an unsecured network. More in detail SSH provides a secure channel over an unsecured network in a client-server architecture, connecting an SSH client application (delta) with an SSH server (laptop). To connect the Deltas with the Virtual Machine on a console the code `ssh rootthe.ip.of.delta` has to be typed with the ip address of the robot.

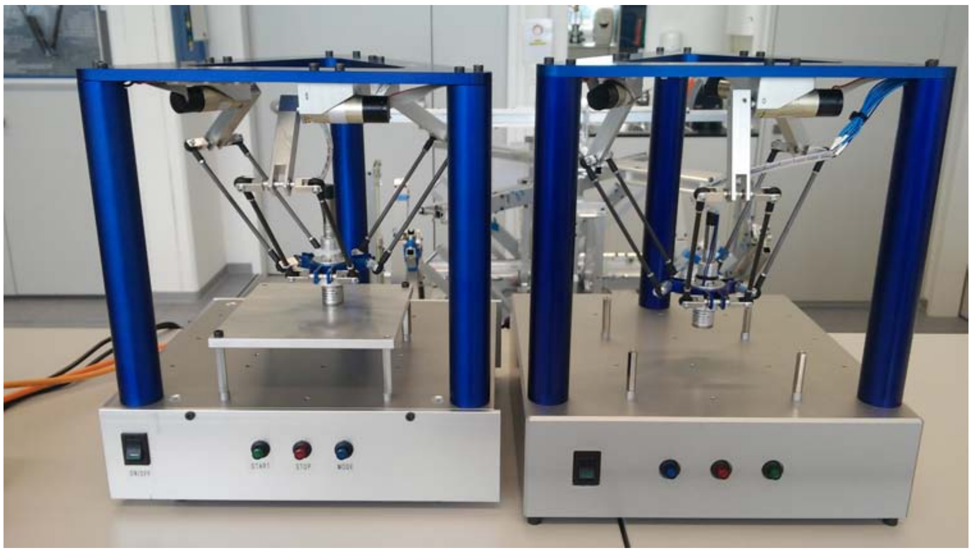


Figure 3.1: Two DELTA robots connected by socket application.

The two **Delta Robots** used were completely built at NTB. Delta robots are widely used in the industry with relevance for high-speed applications. Thanks to their special structural and kinematic properties, Delta robots offer a more advantageous dynamics than SCARA robots or conventional articulated arms. They are therefore made for pick-and-place applications and high-speed applications.
The delta robots consist of a fixed base which is secured to the frame and connected in turn to the traveling plate by three arm systems (with lower and upper arms) displaced by 120°. The upper arms are actuated by the motors mounted on the fixed base. To measure each motor shaft angle a encoder is used. Each of the three lower robot arms consists of two parallel bars, which connects the upper arm with the traveling plate on which the effector is also mounted. The latter can move products in a three dimensional Cartesian coordinate system. The combination of the constrained motion of the three arms connecting the traveling plate to the base plate ensues in a resulting 3 translatory degrees of freedom (DOF). Finally, thanks to the rotating axis at the Tool Center Point (TCP), 4 DOF are possible. The actuator for this axis is then mounted on the upper side of the

robot base plate. The bar is connected directly to the tool and ensures for an additional rotation motion.

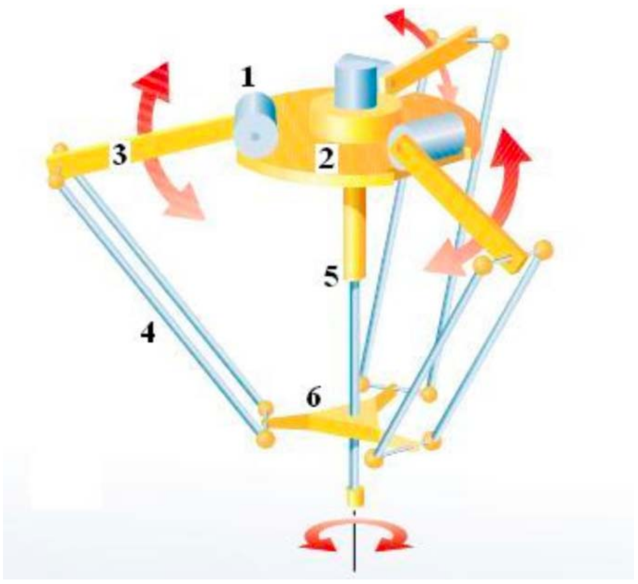


Figure 3.2: Scheme of the Delta[8].

The Robot consists of, consider numbers in figure (3.2):

1. Actuators (3 motors and 3 encoders)
2. Base plate
3. Upper robot arm
4. Lower robot arm (Forearm)
5. Rotation arm (4-DOF)
6. Travelling plate, TCP

The NTB Delta robots are able to control contact forces in a coordinate system that accompanies their movements individually and independently for each direction without using additional sensors, thank to the use of high-resolution encoders and gears without backlash. In this way it is possible to perform even complex operation in a simple and economic way. However, it involves high requirements for the drives which constitute the central physical element of the robot. Their clearance must be as low as possible to ensure accurate position control and good reproducibility. The drive force of the four axes is provided by Faulhaber 1524 012 series, combined with a clearance-free reducer and a magnetic encoder.
The delta robot is controlled by a *BeagleBone Black*. The importance of controlling the robot through a embedded microcontrolller depends on its structure. A microcontroller is a complete system which integrate in the same chip:

- a microprocessor;
- RAM random access memory;
- non-volatile memory: ROM, EEPROM, FLASH...
- I/O interface.

A microcontroller in order to be embedded must:

1. provide real-time (predictable, though not necessarily fast) response to events in the embedded system they are controlling;
2. have saved the program in a non-volatile memory;
3. be transparent to the final user (must work as a dedicated-hardware).

This *BeagleBone Black* features a powerful TI Sitara^{TM}ARM Cortex^{TM}-A8 processor which runs at 1 GHz. A 2 GB on-board flash memory acts as the "hard drive" for the board to host a Linux operating system and other software development tools. The combination of powerful computing power and small physical size allows them to be used in wide variety of projects. Through the board it is possible to prototype electronic systems that interface with real-world applications. The board provides more complicated interfaces including C/C++ functions to access digital and analog pins aboard the ARM Cortex A8 microprocessor. The full power and capability of the BBB board can be programmed in the underlying onboard Linux operating system, such as Angstrom or Ubuntu. An acronym highlights the important Beagle features: **B**ring your own peripherals, **E**ntry level costs, ARM Cortex-A8 Superscalar processor, **G**raphics accelerated, **L**inux and open source community, **E**nvironment for innovators and a features summary can be seen in Table (3.1). Since the software has been developed thought the PC, the executable has to be first cross-compiled and then copied in the microcontroller in order to perform the task; this is possible through the command `copy2robot.sh`. Cross-compiling is necessary since the microcontroller on the Beaglebone has an arm architecture.

Processor	AM3358; 1GHz, USB powered or DC powered
Memory	512 MB DDR3 SDRAM; 2GB eMMC Flash
Power Options	USB connection, 5 VDC external jack
Board Features	HDMI with audio; USB, 10/100 Ethernet; serial debug via external header
Processor Subsystems	176K ROM; 64K RAM; 3D graphics engine; LCD and touchscreen controller; PRU-ICSS; Real Time Clock(RTC); USB ports (2); Ethernet; Controller Area Network; UART (2); McASPs(2); McSPI (2); I2C(2); Analog-to-digital converter; Enhanced Capture Module (3); Pulse width modulation (3); Crypto accelerator.

Table 3.1: Beagle Black Board features summary.

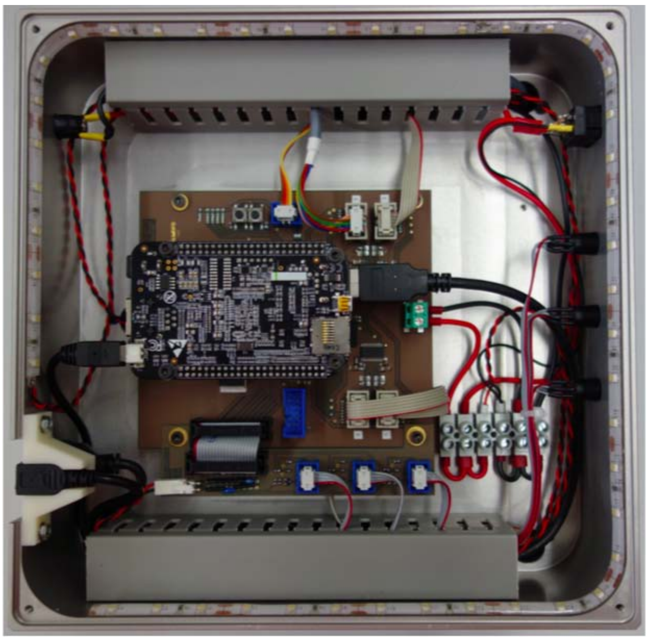


Figure 3.3: Base case with BeagleBone Black mounted.

In figure (3.3) in shown the base case of the Delta with the BeagleBone Black mounted. The board which hosts the BeagleBone was fully developed at the NTB. This board hosts also an FPGA (**F**ield **P**rogrammable **G**ate **A**rray) and some peripherals. From the figure (3.3) and (3.1) is possible to see the Human-Machine Interface (HMI)extension board. The latter connects three buttons with integrated LEDs to the FPGA on the main board. The LEDs have three different colors: green, blue and red respectively. These different button and leds can be programmed depending to the user and the application.

Chapter 4

Delta Robot

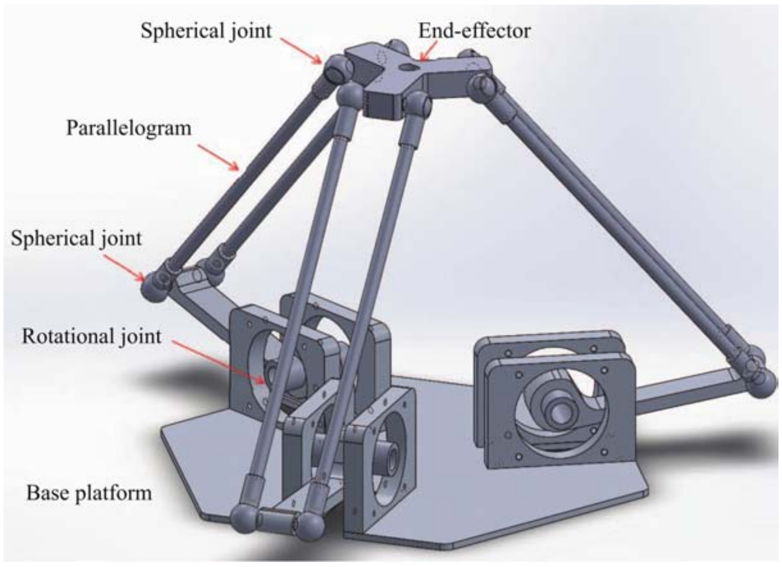
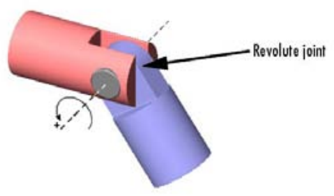
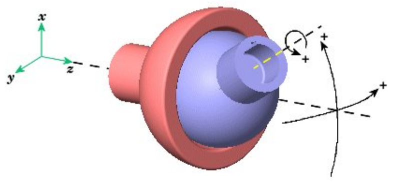


Figure 4.1: Delta parallel robot diagram.[31]

As shown in figure (4.1) a manipulator can be schematically represented from a mechanical viewpoint as a kinematic chain of rigid bodies *links* connected by means of revolute or prismatic *joints*. The start of the chain is constituted by a base platform, while an end-effector is mounted to the end of it. The links that compose the robotic mechanism are assumed to be perfectly rigid bodies having surfaces that are geometrically perfect in both position and shape. Accordingly, these rigid bodies are connected together at joints where their idealized surfaces are in ideal contact without any clearance between them. The respective geometries of these surfaces in contact determine the freedom of motion between the two links, or the joints kinematics. A kinematic joint is a connection between two bodies that constraints their relative motion. There can be two different type of kinematic joints, i.e. lower pair joints if contact occurs over surface or higher pair joints if contact occurs only at points or along lines. In the former case, which is the one of the Delta robot, there are only six possible forms: revolute (or rotational), prismatic, helical, cylindrical, spherical and planar. [10]



(a) *Revolute joint.*



(b) *Spherical joint.*

Figure 4.2: Delta's joint.

In the analyzed case the Delta is furnished with spherical and rotational joints. More in details, rotational joints which can also be represented as R-Joint, allow the joints to move in a rotary motion along the axis, which is

vertical to the arm axes. As a consequence a revolute joint permits only rotation of one of the bodies joined relative to the other and the joint has one degree of freedom. On the contrary the spherical joint is formed by contact of two congruent spherical surface. It permits rotation about any line through the center of the sphere. Thus, it permits independent rotation about axes in up to three different directions and has three degree of freedom.
As described in previous section a Delta robot is a type of parallel robot which consists of a moving platform connected to a fixed base through three parallel kinematic chains. Each chain contains a revolute joint actuated by a rotational motor which are mounted on the base platform. Movements are transmitted to the moving end-effect or platform through parallelograms formed by bars and spherical joints. A delta robot with three closed kinematic chains equal among themselves and separated by a 120° angle is shown in figure (4.1). Three AC servo motors are placed on the fixed platform and used to transfer motion from base platform to moving platform. The encoder attached to these Ac servo motors can measure the rotational position response. By fixing the position of the motors the inertial load is reduced. Thus, higher moving platform speed and acceleration can be obtained in the cartesian space. Cartesian space has been chosen in order to have the motion between two points at all time known and controllable.

4.1 Kinematics

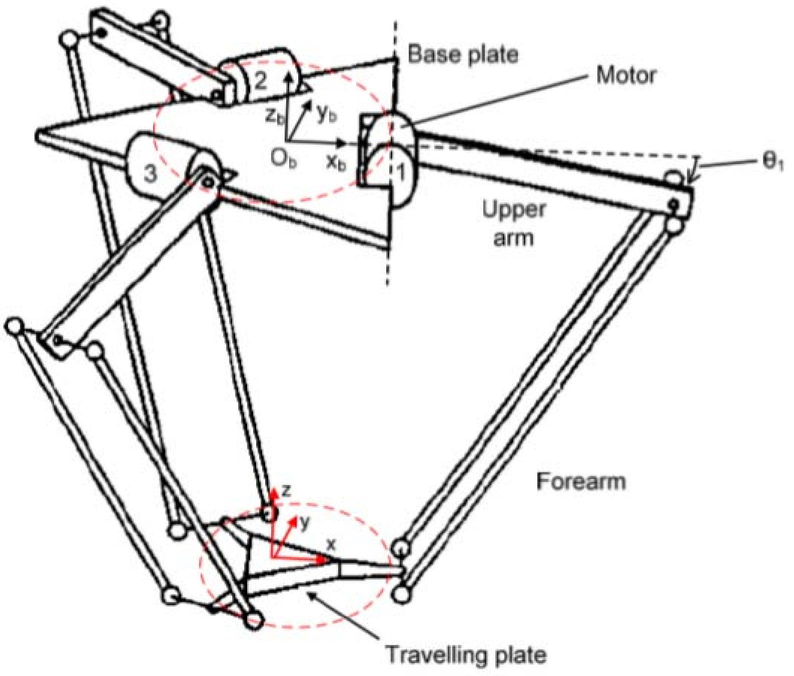


Figure 4.3: Delta robot where the travelling plate is parallel to the base plate. The frame at the base plate is the world frame[11].

Kinematics is the science of motion that treats motion without regard to the forces which cause it. Within the science of kinematics, one studies position, velocity, acceleration, and all higher order derivatives of the position variables with respect to time. Hence, the study of the kinematics of manipulators refers to all the geometrical and time-based properties of the motion.[9] A kinematic problem can be of two different types: forward or inverse. A forward kinematics problem aims to determine the position of the end effector (e.g. to make some corrections of its current position) when the joint angles (through motor encoders) are known. This solution is needed for control purposes, calibration and motion planning. The inverse kinematic problem consists of determining the corresponding angles of each of three arms (joint angles) to set motors once the desired pose of the end effector is known.
Since the Delta is a parallel robot, it is a system of rigid bodies in which there are two members that are connected together by multiple joints (i.e. each joint is often itself a serial chain). More in details a parallel manipulator can be defined as a closed-loop mechanism composed of an end-effector having n degrees of freedom and a fixed based, linked together by at least two independent kinematic chain.[9] The resulting motion of the structure is obtained by composition of the elementary motions of each link with respect to the previous one.
Delta robot is a moving end-effector that is connected to the base by three legs leading to a complex kinematics

and position interference between legs. Delta robot provide three dimensional space with rigid body and fast movement that can be accurately controlled by appropriate kinematic solutions. However, direct kinematics of Delta robot have around 40 solutions due to the complexity of pair of spherical joint that may present difficulty in design and manufacturing. The kinematics of delta robot can be analyzed in order to solve the inverse kinematic problem and the forward kinematic problem.
In order to calculate the kinematics two assumption can be made in order to simplify the model and to reduce the number of parameters[8]:

1. The Travel plate always stays parallel to the base plate and its orientation around the axis perpendicular to the base plate is constantly zero. Thus, the parallelogram type joints (forearm) can be substituted by simple rods without changing the robot kinematic behavior.

2. The revolute joints (between the base plate and the upper arms and between the forearms and the travelling plate) are identically placed on a circle, see Figure (4.3). Thus, the travelling plate can be replaced by a point P which the three forearms are connected to.

Before processing solution to kinematics it is necessary to describe the end-effector position and orientation in order to manipulate an object in space, since a rigid body is completely described in space by its pose with respect to a reference frame.

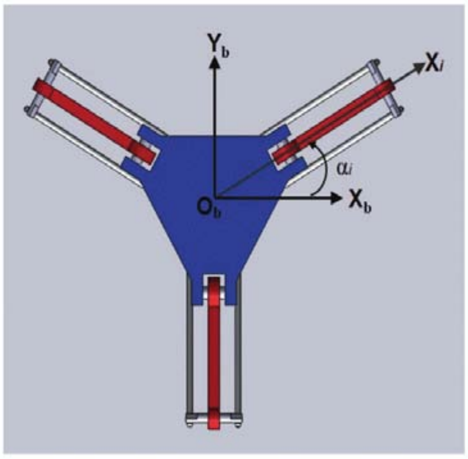


Figure 4.4: Top view of a Delta robot.[31]

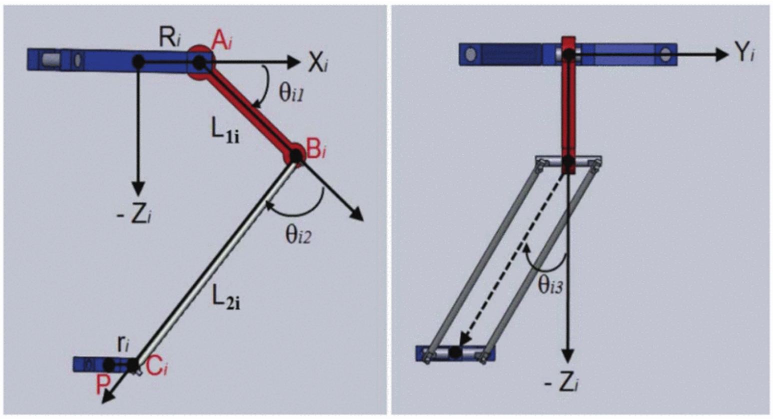


Figure 4.5: Side and front views of Delta robot.[31]

The reference frame $\{R\} = (x_b, y_b, z_b)$ is chosen as shown in figure (4.3) at the centre of the circle with z_b pointing upwards and x_b perpendicular to the axis of motor 1. Because of the Delta robots symmetry of the arms, each arm can be treated separately. The new model for one of the arms is shown in figure (4.5). The index i, $(i = 1, 2, 3)$, is used to identify the three arms. Each arm is separated by an angle of 120°. For each arm, a corresponding frame (x_i, y_i, z_i) is chosen, located at the same place as the frame $\{R\}$ but rotated $\alpha_1 = 0°$, $\alpha_2 = 120°$ and $\alpha_3 = 240°$ for the three arms respectively, figure (4.4). The different frames $\{R_i\}$ can be described by a rotation matrix around z axis of the reference frame $\{R\}$.

The rotation matrix ${}^{i}R_z(\alpha_i)$ is given by:

$$
{}^{i}R_z(\alpha_i) = \begin{bmatrix} \cos(\alpha_i) & -\sin(\alpha_i) & 0 \\ \sin(\alpha_i) & \cos(\alpha_i) & 0 \\ 0 & 0 & 1 \end{bmatrix}.
$$

As shown in figure(4.5) each upper arm measures l_1 and is attached to the based at radius R from center of base, while each fore arm measures l_2 and is attached to the traveling plate at a radius from center of it of r. The assumption 2 made above, saying that the three forearms can be connected in one point P, allows to consider the distance from the reference frame R to one of the motors as $e = R - r$. Each angle of each of the three upper arms has its initial value, 0°, parallel with the x axis of the frame $\{R_i\}$. The angle θ_i (i.e. $\theta_i 1$ in figure(4.5)) is increasing when the upper arm is moving downwards and decreasing when it is moving upwards. Secondary angle θ_{i2} and third angle θ_{i3} of fore arm can be computed directly from the measure θ_i of the encoders and is not necessary their computation in order to calculate forward and inverse kinematic.

4.1.1 Forward Kinematics.

The solution of the forward kinematic problem consists in determining the pose (x, y, z) of the traveling plate by reference to base-frame $\{R\}$ given the configuration $q = [\theta_1 \quad \theta_2 \quad \theta_3]^T$ referring to each angle θ_i of the actuated revolute joints which are known thank to the encoders. In order to solve the problem the main idea is to find the intersection of three sphere that have origin point at the elbow of each robot arm chain and forearms length l_2 as radius. This three point (J_1, J_2, J_3), figure (4.6), can be obtained knowing the position of vector q. Joints J_1E_1,

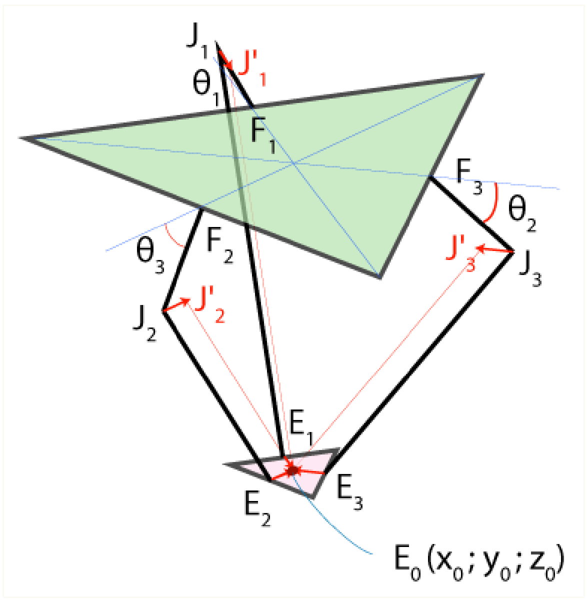


Figure 4.6: Model for computing forward kinematic.[8]

J_2E_2 and J_3E_3 can freely rotate around points J_1, J_2 and J_3 respectively, forming three spheres with radius l_2. This three spheres will intersect in two different points: one has positive z coordinate and the other has negative z coordinate instead. Since the base frame $\{R\}$ has positive z-axis the desired point which represent the end effector's position will be the one with negative z-axis. The three spheres are shown in figure(4.8). By observing figures (4.5) and (4.6) it is easy to find out the elbow coordinates for each of the three arms as:

$$
\begin{aligned}
B_i = J_i &= \begin{bmatrix} R - r + l_1\cos(\theta_i) & 0 & l_1\sin(\theta_i) \end{bmatrix} \\
&= \begin{bmatrix} e + l_1\cos(\theta_i) & 0 & l_1\sin(\theta_i) \end{bmatrix}.
\end{aligned}
$$

To describe all the three points J_i by reference to reference frame $\{R\}$ it is necessary to multiply all of them by the rotational matrix ${}^{i}R_z(\alpha_i)$ explained previously. The result is the matrix:

$$
\begin{aligned}
J = J_i \cdot {}^{i}R_z(\alpha_i) &= \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix} \\
&= \begin{bmatrix} \cos(\alpha_1)(e + l_1\cos(\theta_1)) & -\sin(\alpha_1)(e + l_1\cos(\theta_1)) & -l_1\sin(\theta_1) \\ \cos(\alpha_2)(e + l_1\cos(\theta_2)) & -\sin(\alpha_2)(e + l_1\cos(\theta_2)) & -l_1\sin(\theta_2) \\ \cos(\alpha_3)(e + l_1\cos(\theta_3)) & -\sin(\alpha_3)(e + l_1\cos(\theta_3)) & -l_1\sin(\theta_3) \end{bmatrix}
\end{aligned}
$$

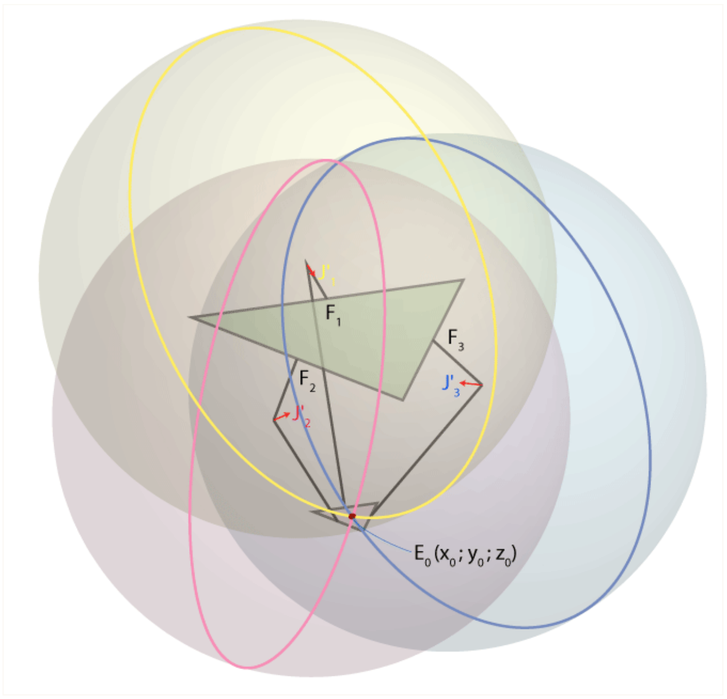


Figure 4.7: The three spheres whose intersection gives the TCP position.[8]

Since the equation for a sphere is $(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2 = r^2$ where (x_i, y_i, z_i) are the coordinates of the center and r is the radius it is possible to write the following system of equations:

$$\begin{cases} (x - [\cos(\alpha_1)(e + l_1\cos(\theta_1))])^2 + (y - [-\sin(\alpha_1)(e + l_1\cos(\theta_1))])^2 + (z - [-l_1\sin(\theta_1)])^2 = l_2^2 \\ (x - [\cos(\alpha_2)(e + l_1\cos(\theta_2))])^2 + (y - [-\sin(\alpha_2)(e + l_1\cos(\theta_2))])^2 + (z - [-l_1\sin(\theta_2)])^2 = l_2^2 \\ (x - [\cos(\alpha_3)(e + l_1\cos(\theta_3))])^2 + (y - [-\sin(\alpha_3)(e + l_1\cos(\theta_3))])^2 + (z - [-l_1\sin(\theta_3)])^2 = l_2^2 \end{cases} \tag{4.1}$$

which knowing that $\alpha_1 = 0°$, $\alpha_2 = 120°$ and $\alpha_3 = 240°$ becomes:

$$\begin{cases} (x - [e + l_1\cos(\theta_1)])^2 + y^2 + (z - [-l_1\sin(\theta_1)])^2 = l_2^2 \\ (x + \frac{1}{2}[e + l_1\cos(\theta_2)])^2 + (y + \frac{\sqrt{3}}{2}[e + l_1\cos(\theta_2)])^2 + (z - [-l_1\sin(\theta_2)])^2 = l_2^2 \\ (x + \frac{1}{2}[e + l_1\cos(\theta_3)])^2 + (y - \frac{\sqrt{3}}{2}[e + l_1\cos(\theta_3)])^2 + (z - [-l_1\sin(\theta_3)])^2 = l_2^2 \end{cases}$$

in the unknown variable (x,y,z). Once this equation have been resolved the solution (x, y, z) with the lower value of z will be kept. Otherwise if the system is without a valid solution an error occurs and the delta stop its movement. The code and the numerical method for solving (4.1) can be found in appendix (A).

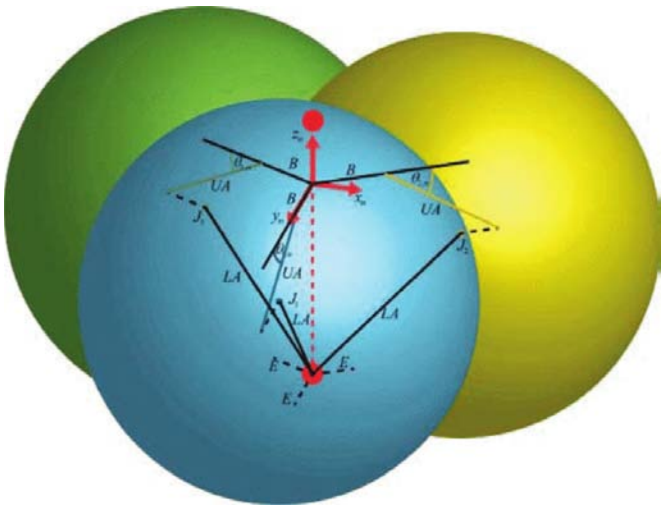


Figure 4.8: Intersection positions between the spheres are illustrated by red dot.

4.1.2 Inverse Kinematics.

The inverse kinematics of a parallel manipulator determines the θ_i angle of each actuated revolute joint given the (x, y, z) position of the travel plate in base-frame $\{R\}$. This mean that inverse kinematics transform the robots TCP position (x, y, z) into the three upper arm angle position $(\theta_1, \theta_2, \theta_3)$. As shown in figure(4.9) the inverse kinematics gives multiple solutions of the $\boldsymbol{\theta} = [\theta_1 \quad \theta_2 \quad \theta_3]^T$ vector with the three revolute joint angles that all satisfies the specific travel plate position. Since the system has to be able to choose only one configuration a reasonable choice would be the closest solution where manipulator moves the upper arm as little as possible. Corresponded angular position of each motors that provide specific endeffector's position in delta robot are

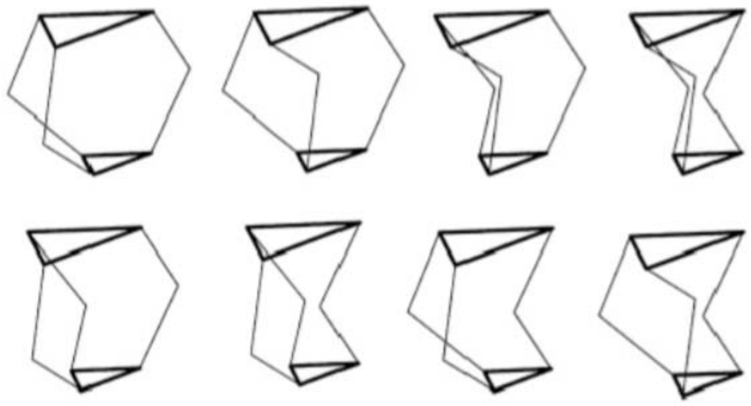


Figure 4.9: The eight solutions of the inverse kinematics.[8]

calculated separately with position of endeffector and using two circle equations to find the intersect position $J_i = [x_j^i, y_j^i, z_j^i]$ which is a joint that indicate the angular position of motor as shown in figure(4.10). In the case that intersection have more than one point, as explained before, the point with higher x-axis value is selected. The origin point of the first circle with radius l_1 is calculated by distance between center of motor and center of base B. The origin point of the second circle with radius l_2 is calculated by distance between center of endeffector T and the position of endeffector (x, y, z).

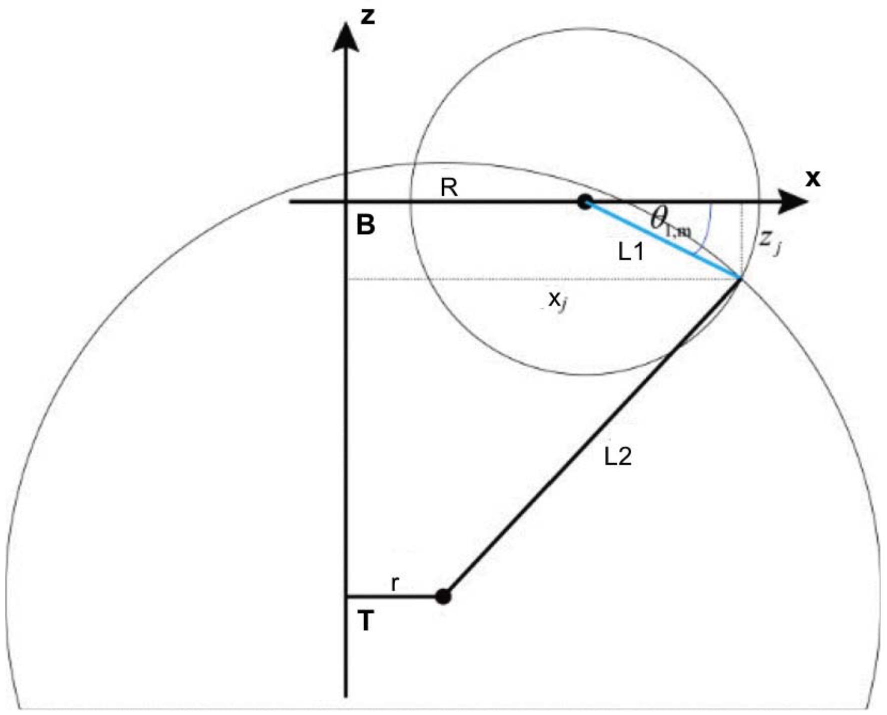


Figure 4.10: Inverse kinematic method for delta robot.

As a result, the equation for i-joint position in xz plane is represented by the following:

$$\begin{cases} (x_{ji} - (R - r))^2 + z_{ji}^2 = l_1^2, \\ (x_{ji} - x_i)^2 + y_{i^2} + (z_{ji} - z_i)^2 = l_2^2. \end{cases} \tag{4.2}$$

in the variable (x_{ji}, y_{ji}, z_{ji}). Once this equation have been resolved the solution (x_{ji}, y_{ji}, z_{ji}) with the higher value of x_j will be kept. Otherwise if the system is without a valid solution an error occurs and the delta stop its movement. The code and the numerical method for solving (4.2) can be found in appendix (A).

4.2 Jacobian matrix.

In the previous section, direct and inverse kinematics equations establishing the relationship between the joint variables and the end effector pose were derived. Now turns out to be important to find a relationship between the joint velocities and the corresponding end-effector linear and angular velocities i.e. between motion in joint space and motion in cartesian space. The linear mapping between the two velocities is provided by the Jacobian matrix $\mathbf{J}$:

$$\mathbf{v_e} = \begin{bmatrix} \dot{\mathbf{p}}_\mathbf{e} \\ \omega_e \end{bmatrix} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}; \tag{4.3}$$

where $\dot{\mathbf{q}} = [\dot{\theta_1} \quad \dot{\theta_2} \quad \dot{\theta_3}]^T$ stays for the actuated joint velocity, $\mathbf{v_e}$ stays for translational and angular velocities of the traveling platform and $\mathbf{J(q)}$ is the (6×6) end-effector geometric jacobian matrix. The jacobian can be partitioned as:

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_p \\ \mathbf{J}_o \end{bmatrix} \tag{4.4}$$

to separate the contributions of the joint velocities to the linear and the angular velocities in $\mathbf{v_e} = \begin{bmatrix} \dot{\mathbf{p}}_\mathbf{e} \\ \omega_e \end{bmatrix}$.

The Jacobian determines a linearized matrix of the first derivatives. To calculate the Jacobian matrix for the Delta robot a set of constraint equations, linking the Cartesian space variables to the joint space variables, can be used [12]. The three constraints equations for the Delta robot can be chosen taking advantage to the constant length of the forearms. This lead to three equations (see figure (4.11)):

$$\|P_{ci}P_{bi}\|_2^2 - l_2^2 = 0 \qquad i = 1, 2, 3. \tag{4.5}$$

Let s_i be the vector $P_{ci}P_{bi}$, then equation (4.5) can be written as:

$$s_i^T \cdot s_i - l_2^2 = 0 \qquad i = 1, 2, 3. \tag{4.6}$$

The vector s_i can be written as:

$$\begin{aligned} s_i &= O_iP_{ci} - (O_iP_{ai} + P_{ai}P_{bi}) \\ &= \begin{bmatrix} x \\ y \\ z \end{bmatrix} -^i R_z(\alpha_i) \left(\begin{bmatrix} R \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} l_1 \cos(\theta_i) \\ 0 \\ l_1 \sin(\theta_i) \end{bmatrix} \right) \end{aligned} \qquad i = 1, 2, 3. \tag{4.7}$$

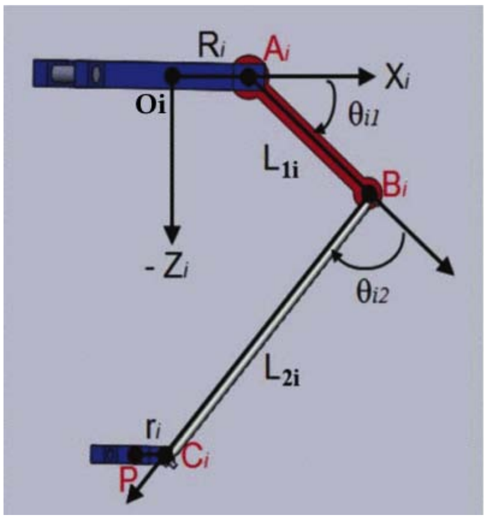


Figure 4.11: Model for delta robot.[31]

The first step for finding a maps between the first time derivative in joint space and in cartesian space is deriving equation (4.6):

$$\dot{s_i}^T s_i + s_i^T \dot{s_i} = 0 \qquad i = 1, 2, 3. \tag{4.8}$$

Due to the commutativity property of the product the latter can be rewritten as:

$$s_i^T \dot{s_i} = 0 \qquad i = 1, 2, 3. \tag{4.9}$$

where the time derivative of (4.7) is given by:

$$\begin{aligned} \dot{s_i} &= \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} +^i R_z(\alpha_i) \begin{bmatrix} l_1 \sin(\theta_i) \\ 0 \\ -l_1 \cos(\theta_i) \end{bmatrix} \dot{\theta_i} \\ &= v_e - b_i \dot{q} \end{aligned} \qquad i = 1, 2, 3; \tag{4.10}$$

where

$$b_i = {}^i R_z(\alpha_i) \begin{bmatrix} l_1 \sin(\theta_i) \\ 0 \\ -l_1 \cos(\theta_i) \end{bmatrix}. \tag{4.11}$$

Thank to equation (4.10) now it is possible to rewrite equation (4.9) for each robotic arm as:

$$s_i^T \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} - s_i^T b_i \dot{\theta}_i = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \qquad i = 1, 2, 3 \tag{4.12}$$

and for the whole system as:

$$\begin{bmatrix} s_1^T \\ s_2^T \\ s_3^T \end{bmatrix} v_e - \begin{bmatrix} s_1^T b_1 & 0 & 0 \\ 0 & s_2^T b_2 & 0 \\ 0 & 0 & s_3^T b_3 \end{bmatrix} \dot{q} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.13}$$

which can be summarized in:

$$v_e = J\dot{q} \tag{4.14}$$

where

$$J = \begin{bmatrix} s_1^T \\ s_2^T \\ s_3^T \end{bmatrix}^{-1} \begin{bmatrix} s_1^T b_1 & 0 & 0 \\ 0 & s_2^T b_2 & 0 \\ 0 & 0 & s_3^T b_3 \end{bmatrix}. \tag{4.15}$$

It is important to underline that in the application of Delta robots the matrix J is not only depending of q but is also a function of the TCP position (x, y, z), which can be computed thanks to forward kinematics. The code for computing the jacobian in `C++` can be found in appendix (B.1).

4.3 Dynamic model.

Dynamic modeling of a robot manipulator consists of finding the relationship between the forces exerted on the structure and the joint positions, velocities and accelerations i.e. between the generalized forces acting on the robot and the robot motion.[13] The dynamic model can be derived using the Lagrange formulation. Since the joint variables θ_i constitute a set of generalized coordinates for the mechanical system, the Lagrange equations can be written as:

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\theta}_i} - \frac{\partial \mathcal{L}}{\partial \theta_i} = \tau_i \qquad i = 1, 2, 3; \tag{4.16}$$

where

$$\mathcal{L} = \mathcal{T} - \mathcal{U} \tag{4.17}$$

is the Lagrangian given by the difference between the kinetic energy and potential energy and τ_i is the generalized torque at joint i. The contributions to the generalized torque are given by nonconservative torque, i.e. the joint actuator torques, the joint friction torques, as well as the joint torques induced by end effector forces at the contact with the environment. The kinetic energy is a quadratic form of the joint velocities,

$$\mathcal{T} = \frac{1}{2}\dot{\boldsymbol{\theta}}^T \mathbf{B}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}} \tag{4.18}$$

where the (3×3) matrix $\mathbf{B}(\boldsymbol{\theta})$ is the inertia matrix of the robot manipulator which is symmetric and positive definite. Substituting (4.18) in (4.17) and taking the derivatives needed by (4.16) leads to the equations of motion:

$$\mathbf{B}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} + \mathbf{g}(\boldsymbol{\theta}) = \boldsymbol{\tau} \tag{4.19}$$

where $\boldsymbol{\tau}$ is the (3×1) vector of joint torques, $\mathbf{g}(\boldsymbol{\theta})$ is the (3×1) vector of gravity torques and $\mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}}$ is the 3×1 vector of Coriolis and centrifugal torques. This lead to a inverse dynamic problem i.e. calculate the torques that a robot's motors must deliver to make the robot's end-effector move in the way prescribed by its current task. The inverse dynamic model of a parallel robot can be computed with different methods (Lagrange or Newton-Euler) but some simplifying hypotheses have to be used since a complete model, taking into account the masses and inertias of all the links leads to very complicated solutions.[12] For the Delta robot, the complexity of the model arises mainly due to the movement of the forearms. This problem can be simplified if their rotational inertias are neglected. This assumption is not very restricting due to use of carbon fibres in their construction. Thus, the force between the traveling plate and the arm is in a direction given by the orientation of the forearm.

In this work the model for the Delta has been calculated with Newton-Euler method with the following simplifying hypothesis:

1. the rotational inertias of forearms are neglected;
2. for analytical purposes the masses of forearms are optimally separated into two portions and placed at their extremities, i.e. $\frac{2}{3}$ at its upper extremity (elbow) and $\frac{1}{3}$ at its lower extremity (traveling plate);
3. friction effects and elasticity are neglected.

In order to find the dynamics of the Delta some parameters are required. Thank to hypothesis number 2 it is possible to compute the new value for the mass of the traveling plate i.e. the traveling plate mass m_n, the mass of the payload $m_{payload}$ that in this application is equal to zero and the 3 reported masses of each of the forearms m_{ab} multiplied by a constant.

$$m_{nt} = m_n + m_{payload} + 3(1-r)m_{ab}. \tag{4.20}$$

The parameter $r \in [0,1]$ is the ratio of the mass of the forearms that is located at their upper extremities and as explained before in this work has the value of $\frac{2}{3}$. The position of the centre of mass for each of the upper arms is calculated as the average of a systems particles position r_i, weighted by their masses m_i:

$$r_{Gb} = l_1 \frac{\frac{1}{2}m_{br} + m_c + rm_{ab}}{m_b} \tag{4.21}$$

with $m_b = m_{br} + m_c + rm_{ab}$ where m_{br} is the mass of the upper arm, m_c the mass of the elbow, m_{ab} the mass of the forearm, and $r = \frac{2}{3}$ the portion of m_{ab} that is placed at the elbow. The inertia contribution of each upper arm I_{bi} is the sum of the motor inertia I_m multiplied by the quadratic of the gear constant and the arm inertia I_{br}:

$$I_{bi} = k^2 I_m + I_{br} \tag{4.22}$$

where the arm inertia is given by the sum of the inertia created from the upper arm mass and the one from the end point mass of the upper arm i.e.

$$I_{br} = l_1^2 \left(\frac{m_{br}}{3} + m_c + rm_{ab}\right). \tag{4.23}$$

Once this parameters are known the use of the jacobian matrix (4.15) of the robot leads to a simplified model for the delta. Indeed, thank to the virtual work principle the contributions of the forces acting on the cartesian space can be computed in the joint space. In fact, virtual work on a system is the work resulting from either virtual forces acting through a real displacement or real forces acting through a virtual displacement. If a rigid body is in equilibrium total virtual work of external forces acting on the body is zero for any virtual displacement of the body. The term displacement may refer to a translation or to a rotation and the term force to force or a moment. Since the delta can be seen as a system of connected rigid bodies which remains connected during the displacement , only the work of the external forces need to be considered given that work done by internal forces (equal, opposite and collinear) cancels each other. Since any generalized force system can be used, the equality of the virtual works associated to the joint and cartesian space yields:

$$\tau^T \cdot \delta\theta = \tau_n^T \cdot \delta X_n \tag{4.24}$$

where τ is the force/torque vector corresponding to virtual displacement $\delta\theta$ in joint space, and τ_n is the force/torque acting on the traveling plate corresponding to the virtual displacement δX_n in cartesian space. Thank to relation $\dot{X}_n = J\dot{\theta}$ equations (4.24) becomes $\tau^T = \tau_n^T \cdot J$ which gives back the value of the generalized force τ:

$$\tau = J^T \cdot \tau_n. \tag{4.25}$$

Thank to the simplifying hypothesis the robot can be reduced to 4 bodies only: the traveling plate and the 3 upper arms. Then the calculation of the contribution of torque/forces acting on the traveling plate can be transferred to joint space through the use of the jacobian matrix. Two kinds of forces act on the traveling plate; the gravity force G_n and the inertial force F_n. They are respectively given by:

$$G_n = m_{nt}\,[0 \quad 0 \quad -g]^T \quad \text{and} \quad F_n = m_{nt}\ddot{X}_n. \tag{4.26}$$

The contribution of these two forces to each motor in joint space can be then calculated by multiplying it with the transpose of the jacobian matrix (equation(4.25)).

$$\tau_{Gn} = J^T G_n = J^T m_{nt}\,[0 \quad 0 \quad -g]^T \tag{4.27}$$

$$\tau_n = J^T F_n = J^T m_{nt}\ddot{X}_n. \tag{4.28}$$

From equation $\dot{X}_n = J\dot{\theta}$ the value of $\ddot{X}_n$ can be found as $\ddot{X}_n = J\ddot{\theta} + \dot{J}\dot{\theta}$ and equation (4.28) becomes:

$$\tau_n = m_{nt} J^T (J\ddot{\theta} + \dot{J}\dot{\theta}). \tag{4.29}$$

Two kinds of force are acting on the actuated joints from each arm: the torque τ_{Gb} produced by the gravitational force of each upper arm and the torque τ_b produced from the inertial force acting on each upper arm. Since the magnitude of torque depends on the force applied, the length of the lever arm connecting the axis to the point of force application, and the angle between the force vector and the lever arm, the gravitational torque is given by:

$$\begin{aligned} \tau_{Gb} &= r_{Gb} \times G_b \left[\cos(\theta_1) \quad \cos(\theta_2) \quad \cos(\theta_3)\right]^T \\ &= r_{Gb} m_b g \left[\cos(\theta_1) \quad \cos(\theta_2) \quad \cos(\theta_3)\right]^T ; \end{aligned} \tag{4.30}$$

where, as shown in figure (4.12), G_b is the gravitational force acting on the mass centre point of each upper arm. The torque contribution from each upper arm to the actuated joints can be expressed by:

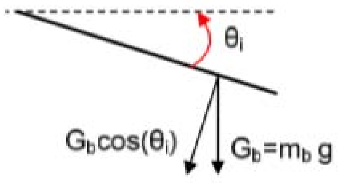


Figure 4.12: Gravitational force acting on upper arm.

$$\tau_b = I_b \ddot{\theta} \tag{4.31}$$

where I_b is the inertia matrix of the arms in joint space and is given by:

$$I_b = \begin{bmatrix} I_{b1} & 0 & 0 \\ 0 & I_{b2} & 0 \\ 0 & 0 & I_{b3} \end{bmatrix}. \tag{4.32}$$

According to the virtual work principle, the contribution of all non-inertial forces must equal the contribution of all inertial forces. This applied at the joint level leads to:

$$\tau + \tau_{Gn} + \tau_{Gb} = \tau_b + \tau_n \tag{4.33}$$

where τ is the vector of torques that have to be applied to the three motors. Substituting equations (4.27),(4.30),(4.31) and (4.29) into (4.33) the final equation of the dynamics can be found:

$$\begin{aligned} \tau &= (I_b + m_{nt} J^T J)\ddot{\theta} + (m_{nt} J^T \dot{J})\dot{\theta} - (\tau_{Gn} + \tau_{Gb}) \\ &= B(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} + g(\theta). \end{aligned} \tag{4.34}$$

where $B(\theta) = (I_b + m_{nt} J^T J)$ is the inertia matrix, $C(\theta) = (m_{nt} J^T \dot{J})$ describes the accounting of the centrifugal and Coriolis forces and $g(\theta) = -(\tau_{Gn} + \tau_{Gb})$ contains the gravity forces acting on the manipulator. In this thesis the contributions of centrifugal and Coriolis forces have been neglected. The inertia matrix could be computed from an easier way considering only the total kinetic energy of the robot, but the establishment of the dynamic model is fundamental in order to compute the gravity contribution. The code in `C++` can be found in appendix (B.2).

Chapter 5

Software.

5.1 Framework EEROS

EEROS is the Easy, Elegant, Reliable, Open and Safe Real-Time Robotics Software Framework. The open source model of EEROS complements its unique architecture, making it universally viable for robots in education and industry. EEROS provides a lightweight layer of software abstraction, allowing code to be reused on multiple platforms and reducing development costs. Since robots have different architecture and different task to do, being able to reuse established processes and algorithms is really powerful.[30] EEROS is a robotic middleware (i.e. collection of software frameworks for robot software development), based on Linux Operating System. A robotic middleware is designed to manage the complexity and heterogeneity of the hardware and applications, promote the integration of new technologies, simplify software design, hide the complexity of low-level communication and the sensor heterogeneity of the sensors, improve software quality, reuse robotic software infrastructure across multiple research efforts, and to reduce production costs.[7] Moreover EEROS uses a real-time patch of Linux. Thanks to its reliability and robustness, the Linux kernel is now widely used in control, safety and mission critical systems.[5] EEROS also offers real-time support. As already underlined before in these contexts, time is extremely important: these systems must meet their temporal requirements in order to reach their goal. A system where the correctness of an operation depends upon the time in which the operation is completed is called a real-time system.[5] The first efforts to adapt the Linux kernel to real-time requirements began about ten years ago. Over the years some different solution have been obtained that can be real time even if were based on linux, which is non real time. Also EEROS achieved this important goal but the explanation of the way it resolved the problem is not the subject of this thesis. Some other framework like ROS (which is not real time at the moment) are now attempting to become real time but research is still ongoing. Finally EEROS also prioritizes safety, which is important for industrial applications.[30] EEROS is developed at NTB University of Technology, Buchs & St. Gallen, Switzerland, where students and professors frequently build new robots. For this reason EEROS is supporting the philosophy that the framework needs to be flexible enough to handle new areas of robotics that have yet to be created. EEROS is the right candidate for educational purposes for the following reasons[30]:

- **flexibility**: allows students to implement and use varying control structures, all while ensuring safety;
- **pre-defined structure**: the modularity of EEROS allows creation and use of libraries with universal components that can be used in different contexts and applications;
- **software consistency** Blocks in EEROS are provided with information about signals flowing through them, which guarantees consistency in the development of control systems;
- **real-time capability**: allows easier control of robotic arms and manipulators;
- **open source**: fully available to the robotics community;
- **object-oriented programming**: allows students to learn how to program in `C++`;
- **built-in robotics blocks**: pre-defined blocks can be used and understand easily by students.

Thank to the particular attention to priorities such as real time and safety, EEROS is suitable also for industrial applications. Specifically, EEROS's industry-related strengths are[30]:

- **Real-time capability**;
- **Safety**: the unique structure of the safety system intrinsically gives the program safety;

- **Flexibility**: EEROS allows implementing varied control strategies, all while ensuring safety;
- **Open source**: EEROS can be freely downloaded with no licensing costs and it can be modified to fit different application;
- **Advanced control**.

As shown in figure (5.1) EEROS consists of three strictly connected systems: the Control System, Sequencer, and Safety System. These work together to make developing new robotics software quickly while minimizing potentially dangerous bugs.[30]

1. **Control System:** controls the movement in real-time;
2. **Safety System:** responsible of the security;
3. **Sequencer:** runtime software.

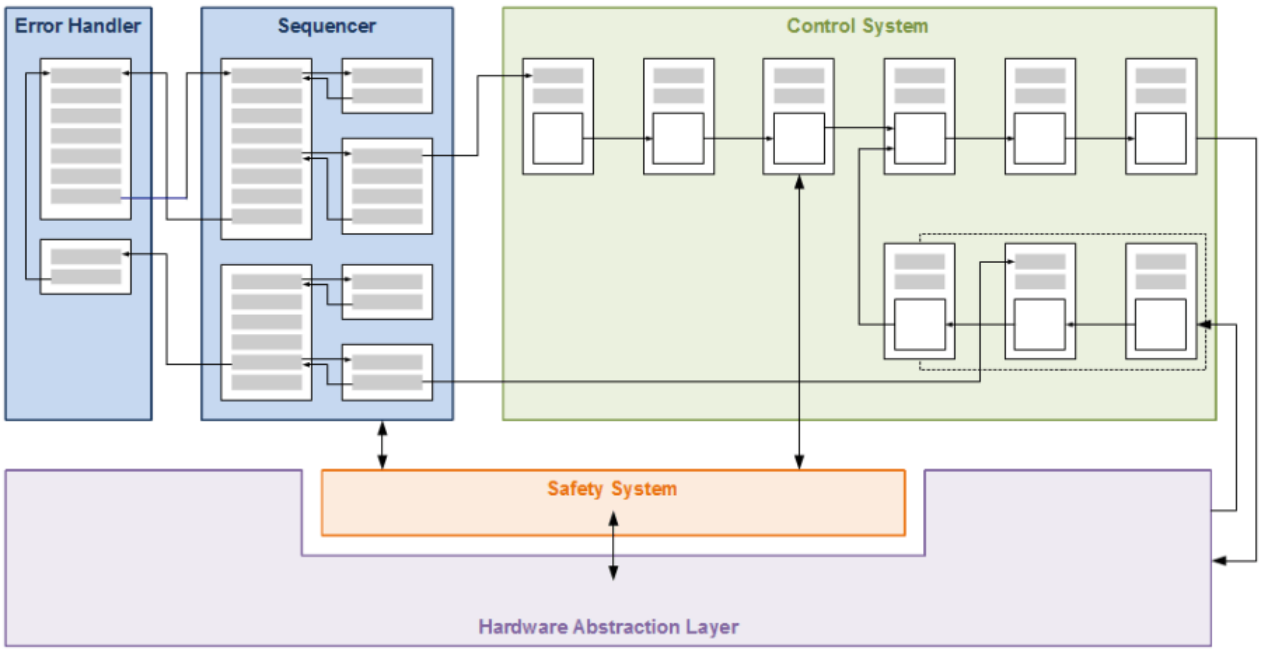


Figure 5.1: EEROS architecture.[30]

The **Control System** will run in real-time and allows the creation of control structures through three basic components: signals, blocks and time domains.

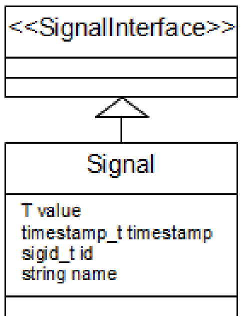


Figure 5.2: Signal type.[30]

A signal represents a succession of varying physical quantities over time as used in signal processing and control. As a consequence each signal brings as information a time stamp which represents the time of detection of the signal. Signals are used to connect different blocks in the control system and share information between the control system, the sequencer and the safety system. Since signals can be of different type T such as logic, arithmetic, vectors or even matrices it is important to specify it at the declaration of the signal together with a name and the mentioned time stamp. Blocks represent something that will alter a signal, or produce a new signal based on the input it receives. When all necessary blocks have been created the blocks must be wired together: this can be done through the call of the "connect" method of the inputs which doesn't have graphical support. Each block belongs to a certain time domain, which determines the execution frequency of its run-method. More precisely time domains comprise blocks which run with the same periodicity. A control system may have one or several time domains running with different periods and in a real-time mode or not thank to the presence of the executor.

The **Safety System** controls the entire system and each subcomponent in order to protect the developer, the end user and the robot against the consequences of any programming errors. This is achieved by introducing different safety levels organized in a predefined structure in such a way that the higher the number of the level is, the higher is the risk of an error. [30]

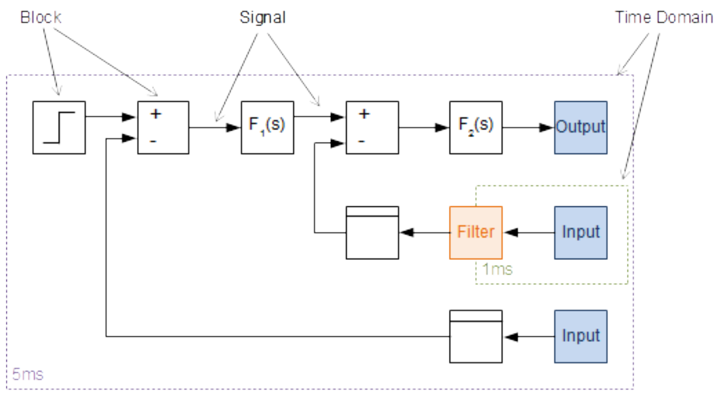
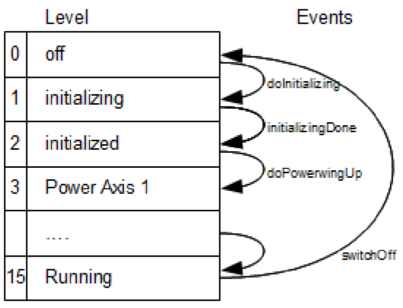


Figure 5.3: Blocks, signals and time domains are the building blocks of the control system.[30]

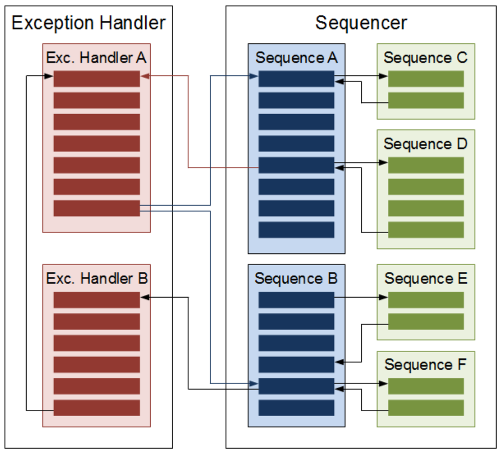


Each level has a unique name and a unique number and for each level, safety-relevant outputs (e.g., watchdog) as well as safety-related inputs (e.g., acknowledgment buttons) are defined. In addition, it is determined how the safety-related outputs are to be set if the safety-relevant inputs deviate from the setpoints. Each level reads the states from all safety-critical inputs and defines the states of all safety-critical outputs through the transition between three different steps: initializing, homing and running. By triggering events, it is possible to switch between the safety levels. This action can be done by[30]:

- the safety system itself (as a level action or by checking an critical input);
- the control system through a special `SignalChecker` block;
- the sequencer.

There are no actions allowed during a transition between two safety levels, since this would be a hidden transition state. The safety system is functional from the very beginning of the development of a new robot control system. This ensures that no dangerous condition can arise even when a system is set up.

The **Sequencer** ensures that a robots can execute a series of user-defined tasks since the control system alone makes it difficult to write a robot program that performs a task whose level is higher than a certain path.[30] Due to its sequential mode the sequencer runs in non-realtime. The robot's motions are defined as a sequence of steps. The sequencer has to be assigned to a sequence and each sequence can include many single steps which must run to completion (unless interrupted by the safety system). Additionally, several subsequences may be defined and added to the same sequencer. Such a subsequence can be called in a blocking or non-blocking way. Blocking means that the step waits (or blocks) until the subsequence has finished. Non-blocking means that subsequence and main sequence run concurrently. As soon as a sequence should run concurrently to another sequence a second sequencer has to be defined. In such a case the two sequencers each run in a separate thread of execution. If a certain step in a sequence cannot run due to some condition, an exception will be thrown and a special exception handler is executed. The latter determines whether the failed operation has to be repeated or left undone, or whether another process has to be executed.



Example of a Sequencer.[30]

5.2 Socket Communication

The **Communication Protocol** that has been used is Socket Communication which is an abstraction through which an application may send and receive data. A socket is a software endpoint that establishes bidirectional communication between a server program and one or more client programs. The socket associates the server program with a specific hardware port on the machine where it runs so any client program anywhere in the network with a socket associated with that same port can communicate with the server program. Computer ports provide the input and output interfaces the device needs to communicate with peripherals and computer networks.

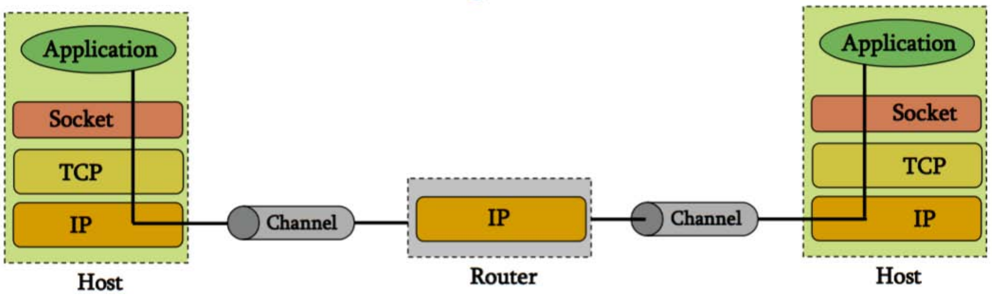


Figure 5.4: Data flow process-to-process.

A Socket is uniquely identified by:

1. an internet address;
2. an end-to-end protocol (e.g. TCP or UDP);
3. a port number.

Two processes can communicate with each other only if their sockets are of the same type and in the same domain. There are two widely used address domains, the unix domain, in which two processes which share a common file system communicate, and the Internet domain, in which two processes running on any two hosts on the Internet communicate. Each of these has its own address format. The address of a socket in the Unix domain is a character string which is basically an entry in the file system. The address of a socket in the Internet domain consists of the Internet address (IP address) of the host machine. In addition, each socket needs a port number on that host. Port numbers are 16 bit unsigned integers.
There are two widely used socket types, stream sockets, and datagram sockets. Stream sockets treat communications as a continuous stream of characters, while datagram sockets have to read entire messages at once. Each uses its own communciations protocol. Stream sockets use TCP (**T**ransmission **C**ontrol **P**rotocol), which is a reliable, stream oriented protocol, and datagram sockets use UDP (**U**nix **D**atagram **P**rotocol), which is unreliable and message oriented. More in details:

TCP:

- reliable byte-stream channel
  (in order, all arrive, no duplicates);
- flow control;
- connection-oriented;
- bidirectional.

UDP:

- no acknowledgements;
- no retransmissions;
- out of order, duplicates possible;
- connectionless, i.e., app indicates destination for each packet.

Summarizing, TCP is used for services with a large data capacity, and a persistent connection; UDP is more commonly used for quick lookups, and single use query-reply actions instead. For this reason, sockets in the Internet domain using the TCP protocol have been choosen for the application of this thesis (5.5). Moreover the Deltas were both connected to the network so TCP/IP protocol was the easiest way to implement the communication since it is via Ethernet.

The socket communication uses the *client server model.* These terms refer to the two processes which will be communicating with each other. One of the two processes, the client, connects to the other process, the server, typically to make a request for information. The client needs to know of the existence of and the address of the server, but the server does not need to know the address of (or even the existence of) the client prior to the

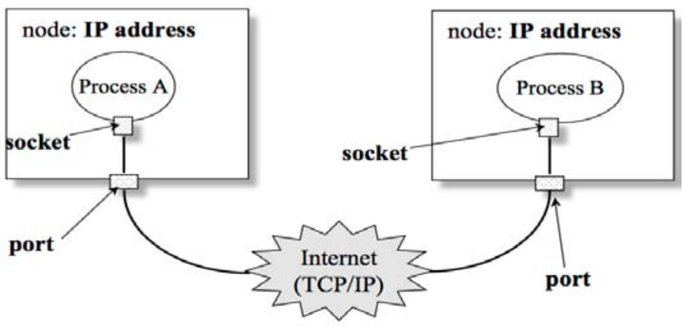


Figure 5.5: Sockets in the Internet domain using the TCP protocol.

connection being established. In this way the server is a passive socket which passively waits for and responds to clients; and the client is a active socket which initiates the communication and must know the address and the port of the server instead. Once a connection is established, both sides can send and receive information.

The steps involved in establishing a socket on the client side are as follows:

1. create a socket with the `socket()` system call;
2. connect the socket to the address of the server using the `connect()` system call;
3. send and receive data. There are a number of ways to do this, but the simplest is to use the `read()` and `write()` system calls.

The steps involved in establishing a socket on the server side are as follows:

1. create a socket with the `socket()` system call;
2. bind the socket to an address using the `bind()` system call. For a server socket on the Internet, an address consists of a port number on the host machine.
3. listen for connections with the `listen()` system call;
4. accept a connection with the `accept()` system call. This call typically blocks until a client connects with the server.
5. send and receive data.

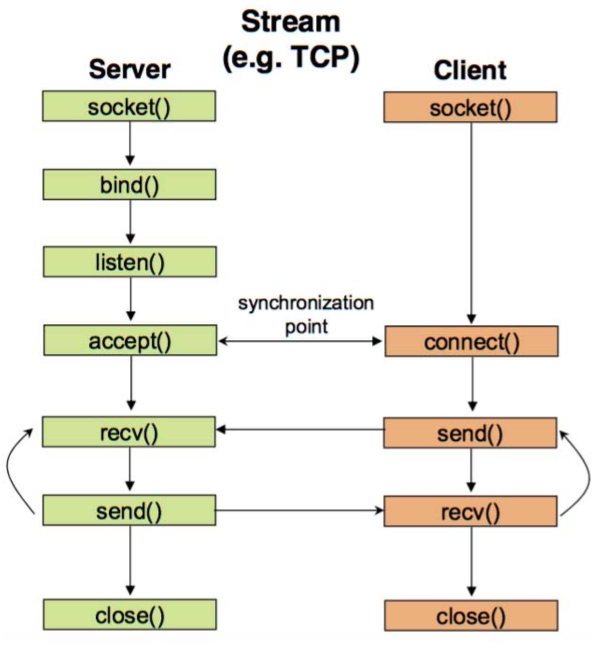


Figure 5.6: Client-Server communication.

Primitive	**Meaning**
Socket	Create a new communication endpoint.
Bind	Attach a local address to a socket.
Listen	Announce willingness to accept connections.
Accept	Block caller until a connection request arrives.
Connect	Actively attempt to establish a connection.
Send	Send some data over the connection.
Receive	Receive some data over the connection.
Close	Release the connection.

Table 5.1: Primitives in Client-Server communication.

Figure (5.6) and Table (5.1) describe the Client-Server communication.
These sockets use TCP (Transmission Control Protocol) for data transmission. TCP provides connection-oriented transport which requires proper control messages for connection establishment, control and release. The reliability of TCP is achieved by retransmitting data, which has been sent but not acknowledged by receiver within given time. Thus sending TCP must keep the sent data in memory until it has received the acknowledgements of sent data. TCP assumes that IP is inherently unreliable, so TCP adds services to ensure end-to-end delivery of data. TCP has very few expectations on the services provided by the networks and it thus can be run across a large variety of hardware. All that is required from lower level is unreliable datagram service. TCP assumes that the connection is full-duplex, that is, both connection endpoints need to exchange data and support through flow control for bidirectional communication must be provided. TCP includes the main feature of providing connection-oriented services over a connectionless IP network. To implement this feature, a key property of the TCP protocol is that every single byte exchanged through a TCP connection must be labeled with a 32-bit sequence number. Figure (5.7) shows the structure of the TCP packet, also called TCP segment. More in details, *source* and *destination* port identify through 16 bit the sending and receiving port respectively. The *sequence number* usually specifies the number assigned to the first byte of data in the current message. In the connection-establishment phase, this field also can be used to identify an initial sequence number to be used in an upcoming transmission. As TCP aims to guarantee a reliable transport layer, acknowledgements are required for the transmitted packets; packets that are transmitted and are not acknowledged must be retransmitted. As there are acknowledgements, there is also an *acknowledgement number* which indicate the sequence number of the next packet that the sender expects to receive. In this way, when a packet is lost or delayed, it will be continuously requested by the receiver. Then the number of 32-bit words in the *TCP header* indicates where the data begins. The length of the TCP header is always a multiple of 32 bits. Following the flag contains information like the SYN, ACK, and FIN bits used in connection establishment and teardown. In particular:

URG the urgent pointer field contains valid data;

ACK the acknowledgement number is valid;

PSH the receiver should pass this data to the application as soon as possible;

RST reset the connection;

SYN synchronize sequence numbers to initiate a connection

FIN the sender is finished sending data.

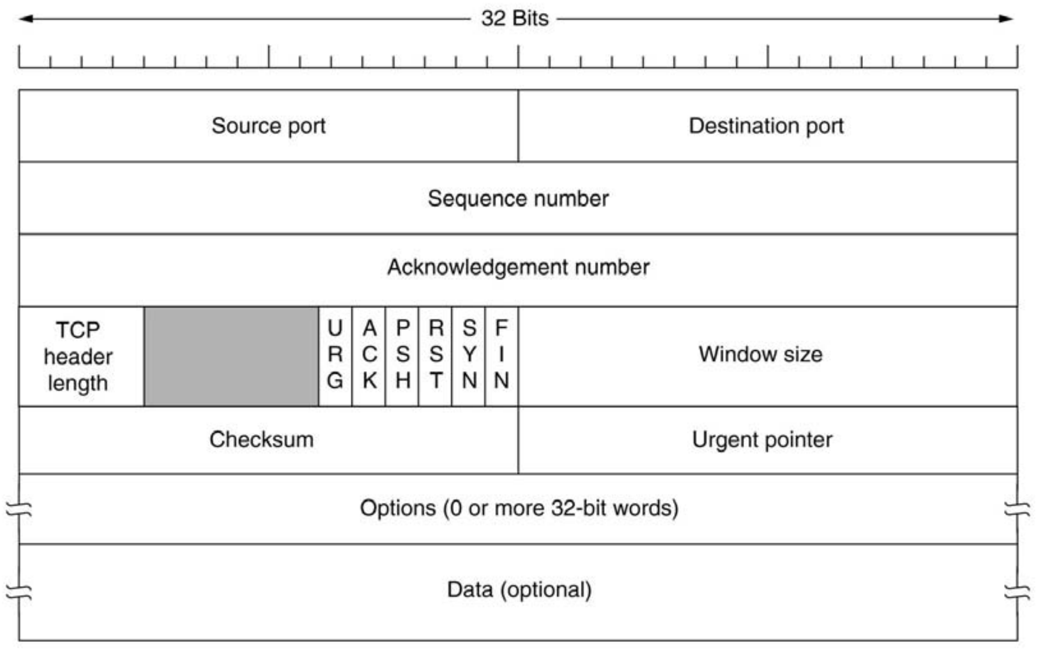


Figure 5.7: The packet structure of TCP.

Then the *Window size* specifies the size of the sending machine's receive window and the *Checksum* field implements a simple error-checking of the header and data. As TCP implements an ARQ mechanism, retransmission of lost data is therefore required. The *Urgent Pointer* denotes that certain data is urgent and may violate the basic flow control. Finally further option and the data are introduced.
The packet structure of TCP underline how this protocol has a high exchange of control message in order to have reliability. This lead to an higher bandwidth consumption, a higher CPU/ memory requirements and a higher delay. The time delay, as will be explained later, was a big problem for this thesis and several different strategies have been implemented in order to avoid instability which descend from it. Despite time delay, Transmission Control Protocol guarantees two goals i.e. flow control and congestion control which both assure reliability to the communication.[4]
TCP performs *flow control* which controls the amount of data send by the sender through a window-based approach. This is achieved by using a sliding window mechanism. The receiver TCP module sends back to the sender an acknowledgment that indicates a range of acceptable sequence numbers beyond the last successfully received segment. This range of acceptable sequence numbers is called a window. The basic reasons to do so are the limited capabilities available at the receiver's side. As the receiver has a finite buffer in TCP connections the receiver is required to advertise in advance its availability to receive data, and the transmitter is requested to comply with the advertised value. The transmitter can send data only through a window of finite size. Sending more data than the receiver can accept would result in losing some of it, so the TCP transmission window should not exceed the receiver advertised window.
At the same time TCP perform *congestion control*. The network between the two connection endpoints is a dynamic system, which does not comprise just the transmitter and the receiver. It is possible that, even though the receive buffer is not full, data is lost in transit through the network. There might be a given quantity of data going through, but the rest does not reach the destination leading to a congestion which drops the performance of the transport protocol. The way to reduce this loss of data is simply to send fewer packets since the more data are sent, the more difficult is to solve the congestion. This is the reason why TCP considers a congestion control strategy besides the flow control procedure based on a sliding window mechanism. The congestion control is a mechanism to regulate the rate of packets entering the network, keeping the data flow under the size that would trigger congestion. This means that the management of the window size in TCP depends on both the congestion window and sliding window; and the transmission window used by TCP is the minimum between the two. In spite of the reliability guaranteed the used communication protocol show up some disadvantage.[6]
Although TCP provides reliable transmission for applications, it does not guarantee the time within which segments are transmitted from the sender to the receiver. IP datagrams may also arrive in disorder into the receiver in TCP. In other words, TCP does not offer constant transmission delay and therefore it does not match up with multimedia, audio or real-time applications. Another issue is the following. End-to-end performance of TCP may suffer due to network asymmetry, especially in the context of wide-area wireless network, since full bandwidth may not be achieved in unidirectional transfers because of the slow arrival of acknowledgments.
Nevertheless TCP/IP was the natural and easiest choice in order to develop the connection between the robot.

Chapter 6

Basic Software Components

6.1 Motor model.

Once the kinematics, the dynamics and the controller have been developed it is necessary to find a suitable model for the DC motor in order to compute the voltage that has to be given to the real system. In this work the behavior of the motor has been approximated by a first order electric circuit shown in figure (6.1).

In table (6.1) the variables of the model are described. This variables are related through fundamental

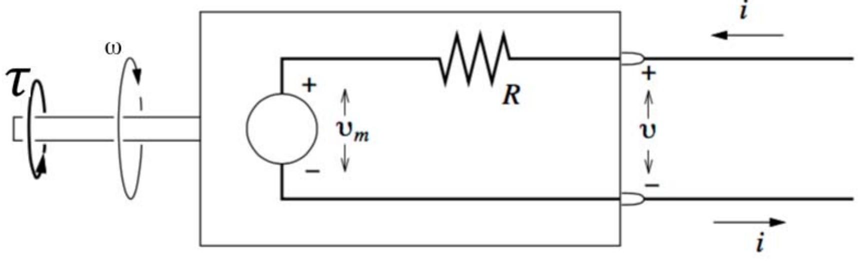


Figure 6.1: Equivalent circuit for a DC electric motor.

$\boldsymbol{\omega}$	Rotation rate.
$\boldsymbol{\tau}$	Shaft torque.
$\mathbf{k_v}$	Speed constant.
$\mathbf{k_t}$	Torque constant .
$\mathbf{v}$	Motor terminal voltage.
$\mathbf{R}$	Resistance of the motor, assumed constant.
$\mathbf{i}$	Current.
$\mathbf{v_m}$	Internal back electromotive force.

Table 6.1: Primitives in Client-Server communication.

equations. The shaft torque τ is assumed proportional to the current i via the torque constant k_t. The latter value can be found in the datasheet of the motor (appendix D). This lead to the equation:

$$\tau = ik_t. \tag{6.1}$$

Then the internal back-EMF v_m is assumed proportional to the rotation rate ω via the motor speed constant k_v:

$$v_m = \omega k_v. \tag{6.2}$$

The constant k_v which is usually expressed in RPM/Volt is assumed to be equal to the torque constant k_t. Finally the motor terminal voltage can be computed by making use of equations (6.1) and (6.2):

$$v = v_m + iR = \omega k_t + \frac{\tau}{k_t}R, \tag{6.3}$$

where ω is known from the encoder, $k_t = 11.5\,\text{mNm/A}$ and $R = 19.8\,\text{ohm}$ are constant and τ comes from the control architecture. In appendix (D) the `C++` code and the data sheet of the motor are attached.

6.2 PD-controller.

The motion control problem for a robot manipulator can be formulated as finding the joint torques which ensure that the end effector attains a desired position and orientation. While task specification (end effector motion and forces) is usually carried out in the operational space, control action (joint actuator generalized forces) are performed in the joint space. This lead to two different kinds of general control schemes: the joint space control and operational space control scheme. Since joint space control schemes are quite adequate in situations where manipulator tasks can be accurately preplanned and little or no online trajectory adjustments are necessary, in this work a cartesian control scheme has been developed. In fact, operational space control scheme works good in more complicated and less certain environments and in particular when controlling the interaction between the manipulator and environment is of concern.[10] Since EEDURO the delta robot developed at NTB assumed to be low-cost for education and research a PD control structures has been chosen for its simple computation. In fact, the PD controllers, despite their simple structures, assure acceptable performances for a wide range of industrial plants. PD control, as explained more in detail in appendix (C) utilize a linear control scheme based on the linearization of the system about an operating point. The PD controller can be seen as a mechanical system consisting on the parallel of a virtual spring and a virtual damper created between the measured and reference positions of a system, see figure (6.2). Such a system can be described informally as an object with

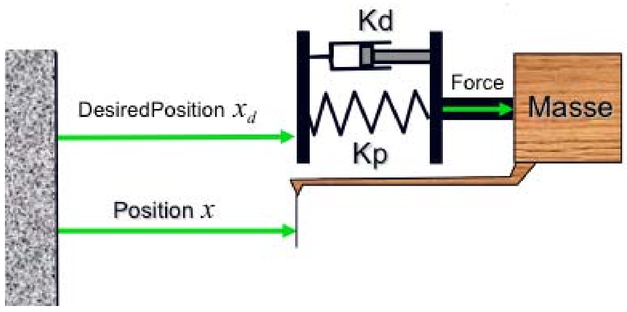


Figure 6.2: 1DOF mechanical system.[15]

mass m that is affected by a spring force and that is located on some surface, i.e. there is friction between the surface and the object. [14] The deflection of the object from its resting position is denoted by x, thus $\dot{x}$ and $\ddot{x}$ are speed and acceleration of the object. The force corresponding to friction is proportional to the speed of the object and the force exerted by the spring is proportional to the deflection of the object from the resting position. In order to control the physical system an extra force f is assumed to exist. Since the friction force is $F_d = d\dot{x}$ and the spring force is $F_p = kx$ it is possible to state the following dynamics equation:

$$m\ddot{x} + d\dot{x} + kx = f \tag{6.4}$$

were the necessary force to accelerate the mass is defined as $F = m\ddot{x}$, k is the spring constant and d is the viscous friction coefficient. This equation is, in the context of control, also called the open-loop-equation, since it does not take into account any feedback from the system (joint position measurements from sensors). In order to have the system carrying out a certain trajectory x_d it is necessary to exert the additional force f on the system (x_d is assumed to be a twice continuously differentiable function). Let $e(t) = x_d(t) - x(t)$ denote the difference between the actual position and the desired position. Movement along the desired trajectory can be achieved by employing the following control rule:

$$f = \alpha\hat{f} + \beta. \tag{6.5}$$

Factors α and β should be chosen such that the system, considering only $\hat{f}$ as input, behaves like a unit mass, governed by equation $\ddot{x} = \hat{f}$. Putting equations (6.4) and (6.5) together the following relation can be obtained:

$$m\ddot{x} + d\dot{x} + kx = \alpha\hat{f} + \beta, \tag{6.6}$$

where $\alpha = m$ and $\beta = d\dot{x} + kx$ guarantee the decoupling of the mass-dependent part in (6.4). Now it is possible to influence the system directly through $\hat{f}$ in order to move from the open-loop form to the closed-loop form. The desired trajectory can be followed by the robot using the control rule:

$$\hat{f} = \ddot{x}_d + K_d\dot{e} + K_p e. \tag{6.7}$$

Substituting equation (6.7) in $\ddot{x} = \hat{f}$, the result is:

$$\ddot{x} = \ddot{x}_d + K_d\dot{e} + K_p e \implies \ddot{e} + K_d\dot{e} + K_p e = 0; \tag{6.8}$$

i.e. the error e behaves like a mass-damping-spring system. This means that the error will tend towards 0, and it will approach zero with a speed depending on K_d and K_p. The (6.8) is a second order differential equation that represents the damped harmonic oscillator and can be rewritten as:

$$\ddot{e} + 2Dw_0\dot{e} + w_0^2 e = 0 \tag{6.9}$$

where $w_0 = \sqrt{\frac{k}{m}}$ is called the "undamped angular frequency of the oscillator"and $D = \frac{d}{2\sqrt{mk}} = \frac{d}{2mw_0}$ is called the "damping ratio". Therefore the PD gains can be seen as $K_p = w_o^2$ and $K_v = 2Dw_0$. The response of a PD controller can be, therefore, characterized by the damping ratio and the natural frequency. If the damping ratio is less than one, then the system will gradually approach the target. If the damping ratio is greater than one, the system will shoot past the target before returning. The natural frequency describes how quickly the system approaches the target. What it is recommended to achieve is the so-called critical damping of the system. There can be three different cases (figure 6.3):

1. under-damping $\Longrightarrow$ oscillation of the system (spring stiffness dominates),
2. over-damping $\Longrightarrow$ slow convergence to the resting point (friction dominates),
3. critical damping $\Longrightarrow$ fastest possible transition of the system to its resting configuration, avoiding both overdamping and underdamping (friction and stiffness are in balance).

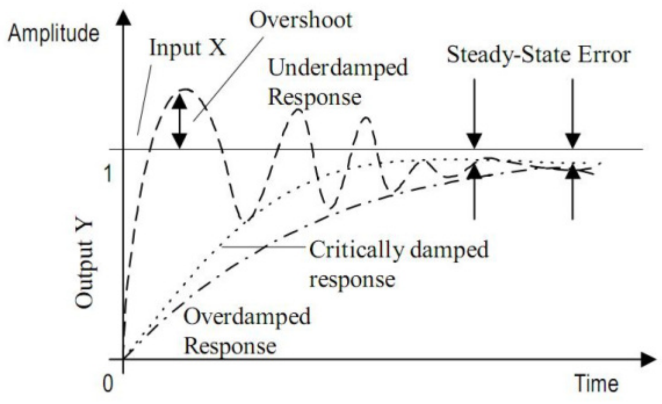


Figure 6.3: Dependence of the system behavior on the value of the damping ratio d.

In this work the goal to achieve was to compensate the speed error in a single control cycle Δt_{task} in order not to have instability. This lead to condition [15]:

$$K_d = \frac{1}{T_d} < \frac{1}{\Delta t_{task}} = f_{task} \tag{6.10}$$

where T_d is the sampling time. In order to achieve equation (6.10) the following choice can be made:

$$K_d = \frac{f_{task}}{2.2} \qquad \frac{2}{3} \leq f_{task}K_d \leq 3. \tag{6.11}$$

In this thesis the following parameters have been used:

$$D = 0.7; \qquad T_d = 0.001s; \qquad f_0 = \frac{f_{task}}{20}; \qquad \omega_0 = 2\pi f_0 = 2\pi 25.$$

Given the values for proportional controller K_p and derivative controller K_d the overall control scheme can be computed.

Figure (6.4) represents a PD controller that minimize the error between the real robot configuration x and the desired one x_d and gives back the acceleration control $\ddot{x}_c$ which gives the control force F_c. Since the measure of the velocity is given (it's measured in the motor) a cascade control have been developed. This particular configuration of the PD control is a convenient way to use extra measurements to improve control performance.(6.5)

The goal of the cascade control is to make the control faster thank to an extra measure, the velocity. In fact, the control action in the common configuration (fig. (6.4)) start acting while the variable error e is not

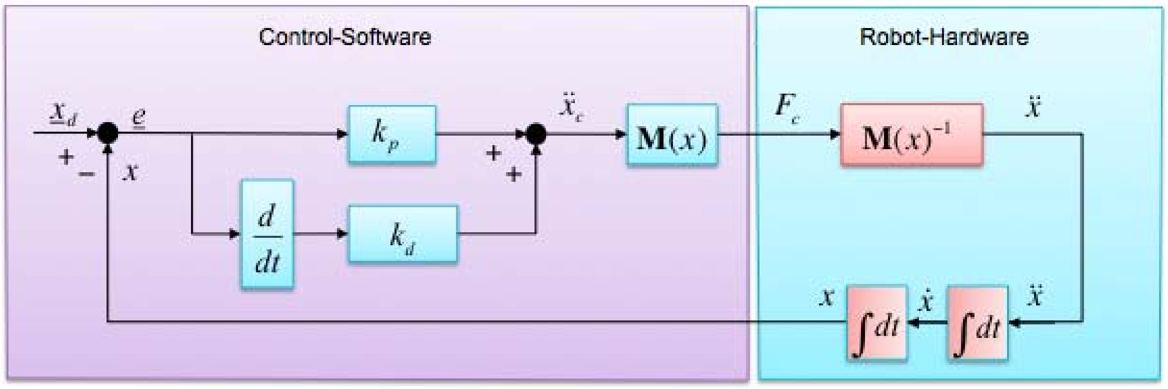


Figure 6.4: PD controller.[15]

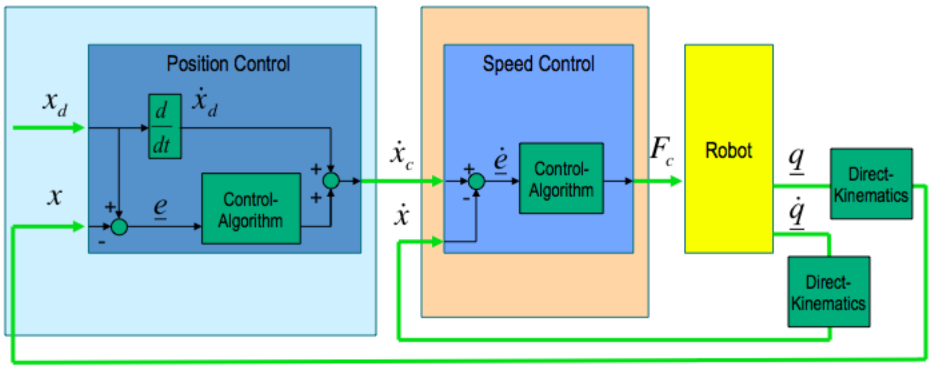


Figure 6.5: PD cascade controller.[15]

negligible and this lead to a slow response. Since in the studied application real time have to be achieved, the velocity while different to zero gives immediately the information of how the robot is moving which turns out in a faster control action. Summarizing the principle of cascaded loops is adding another process loop to more precisely control the intermediate variable. As shown in figure (6.5), the design of a cascade structure implies a dynamical separation of the control loops in two parts: the position control loop and the velocity control loop. The diagram below shows a cascade control system where there is one loop (velocity) nested inside another (position) and with the primary controller's output serving as the secondary controller's set point. For overall stability, the velocity loop must be the fastest responding loop of the two. In fact, if a disturbance affects the process in the velocity loop, its derivative controller can correct the resulting error before the position loop sees the effect and can compensate quickly for the disturbance so that the response in the position, the primary output variable of the process, is small. The improved control with cascade control can be explained by the increased information about the process.

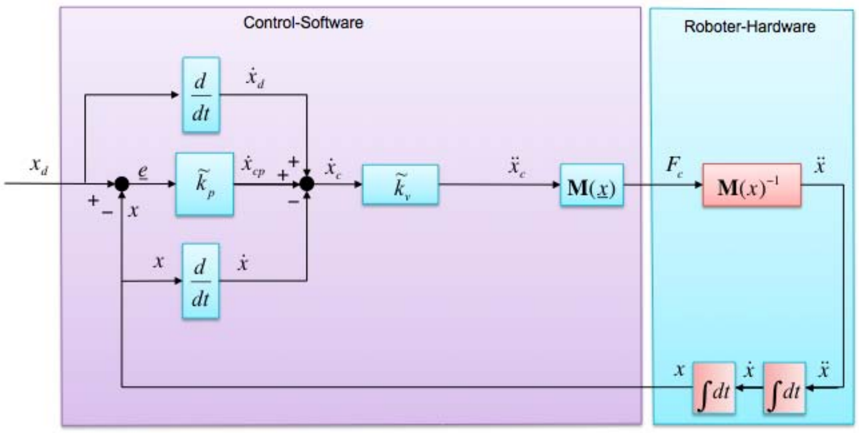


Figure 6.6: PD cascade controller with new gains.[15]

In this thesis the inner velocity loop consists on a derivative controller with gain $\tilde{K}_d$, on the other hand the outer position loop is made of a proportional controller with gain $\tilde{K}_p$. (see figure (6.6)). The new control structure needs different values for the gains $\tilde{K}_p$ and $\tilde{K}_v$ i.e.:

$$\tilde{K}_v = K_d = 2D\omega_0 \qquad \tilde{K}_p = \frac{K_p}{\tilde{K}_v} = \frac{K_p}{K_d} = \frac{\omega_0^2}{2D\omega_0} = \frac{\omega_0}{2D}.$$

This new values can be found watching figure (6.6) and understanding that $\ddot{x}_c = \tilde{K}_v(\tilde{K}_p e + \dot{e})$. This equation compared to $\ddot{x}_c = K_p e + K_d \dot{e}$ of the previous scheme gives back the new values for the gains.

Since in this work a real time controller is needed, an additional action has been added in order not to have a poor control due to relevant disturbances present when high velocity occurs. This "speed advance" feedforward contribution makes the answer to the position set point quicker and it is obtained as the derivative of the desired position which is then added to the velocity loop. (Figure (6.6)).

To sum up, the positive effects related to the use of a feedback controller are:

- noise rejection;
- larger bandwidth for the controlled system if compared to the original system robustness in case of parametric uncertainties;
- the possibility to add a feed-forward control action;
- a more linear relation between the position set point and the overall control output;
- better protection of the system since each intermediate variable can be limited through a saturator.

6.3 Pheriperal interfaces

In this section the interfacing between the computer and the two robots is analyzed. First of all once the Deltas are connected with the Virtual Machine as described in section 3, it is possible to navigate to the folder `/opt/eeduro/bin` through the command `cd /opt/eeduro/bin/` and inside this folder there are all the files that are executable (all the file can be seen using the command `ls` and the executable are underlined with the green color). In the end it is possible to launch the desired program by means of `./delta-server 1` for the server and `./delta-client 146.136.39.234 1` for the client. (These commands were analyzed previously in the description of the communication protocol). Once the program has been launched the robots start the process of initializing position only after that both the green button on the DELTAs have been pressed. In fact this button was programmed to make the robot moving and initializing when pressed. In this particular phase an important error occurred: both the robot where programmed to start sending data after being initialized, but since the operator has to press at first the master green button and then the slave's one; the master ends the initializing phase before the slave. This latter fact was undesirable given that the master started to send data to the slave before the end of slave initializing process. Then an error occurred since the slave starts receiving a position target before being completely initialized. For this reason also the blue button was programmed in order to make the communication start when both the blue button were pressed. Lastly the red button was used to stop the motion of the robot and the LED should lit red in emergency situations already programmed in the EEROS safety system.

Chapter 7

PD Control scheme with gravity compensation and passivity terms

As explained in chapter (1) and (2) bilateral teleoperation is a key technology to allow humans to interact with remote environments by providing the operator with haptic feedback. Haptic feedback improves human perception and therefore the quality of the human robot-interaction on one side, but it can tamper with the stability of the system when the communication between the master and the slave is not instantaneous but affected by delay and packet drops on the other.[17] In this first attempt to compute bilateral teleoperation, a system in which the master and the slave devices are connected through a packet-based communication (see chapter (5.1)) channel which delivers the signals, such as commands from the master and measurements from the slave has been developed. This architecture can lead to instability due to the time delay and an attempt to achieve stability has been made using passivity theory. In order to simplify the problem an assumption has been made: the not negligible time delay in the communication channel has been assumed constant but unknown. This hypothesis, as will be shown in the following chapter, is too strong for this work and even with the use of the passivity theory doesn't lead to satisfactory result in terms of real-time. The used approach exploit the constant time delay hypothesis to design control architectures that guarantee the stability of the system independently of the delay and the interaction with passive environments. The control architecture in this work attempts to guarantee the stability of a position-position bilateral teleoperation by adding a dissipative term to the PD controller at each side of the architecture.[18] Energetically, as illustrated in Fig.(7.1), a closed-loop teleoperator is a two-port system with the master and slave ports being coupled with the human operators and slave environments, respectively. Therefore, the foremost and primary goal of the control (and communication) design for the teleoperation should be to ensure interaction safety and coupled stability when mechanically coupled with a broad class of slave environments and humans. In fact, the control objectives in the teleoperation systems are:

- the stability of the system;
- telepresence, transparency of the teleoperation: position tracking and force reflection.

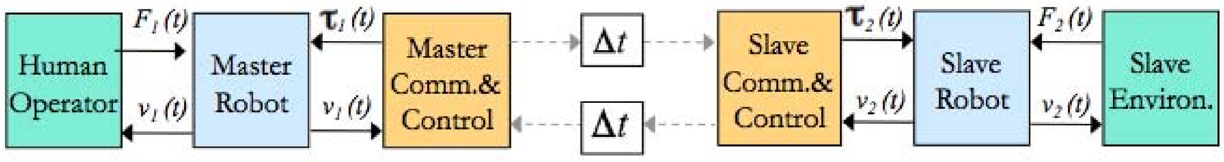


Figure 7.1: Closed loop teleoperator.

Figure (7.1) shows as the teleoperated system can be modeled as a two port system. The equations of the dynamics of the overall system are:

$$\begin{cases} B_1(\theta_1)\ddot{\theta}_1 + C_1(\theta_1, \dot{\theta}_1)\dot{\theta}_1 = \tau_1(t) + F_1(t) \\ B_2(\theta_2)\ddot{\theta}_2 + C_2(\theta_2, \dot{\theta}_2)\dot{\theta}_2 = \tau_2(t) + F_2(t); \end{cases} \tag{7.1}$$

where F_1 and F_2 are the human/environmental forcing, τ_1 and τ_2 are the controls torque and the gravity terms are pre-compensated by the controller. As explained above the forward and backward communications are delayed by finite constant time delays Δt_1 and Δt_2 respectively and this lead to the fact that the control torque

τ_1 and τ_2 can be defined as functions of the current local information and the delayed remote information, i.e.:

$$\begin{cases} \tau_1(t) = \tau_1(\theta_1(t), \dot{\theta}_1(t), \theta_2(t-\Delta t_2), \dot{\theta}_2(t-\Delta t_2)) \\ \tau_2(t) = \tau_2(\theta_2(t), \dot{\theta}_2(t), \theta_1(t-\Delta t_1), \dot{\theta}_1(t-\Delta t_1)). \end{cases} \tag{7.2}$$

Given the network in figure (7.1), it is said to be passive if the sum of energy flows always positive (always more energy flows in than out). Taking into account each component of the system it can be said that:

- the human is assumed a passive controller;
- the environment is assumed a passive spring and a dumper;
- the system highlighted below (7.2), has to being made passive by controlling correspondingly the robot actuators.

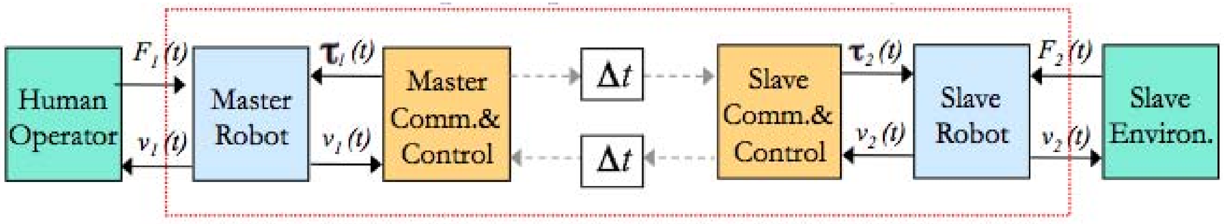


Figure 7.2: Closed loop teleoperator.

The aim of the controller is to achieve the followings:

1. **master-slave position coordination**: i.e. when $(F_1(t), F_2(t)) = 0$ then $e(t) = \theta_1(t) - \theta_2(t) \to 0$;
2. **static force reflection**: i.e. with $(\ddot{\theta}_1(t), \ddot{\theta}_2(t)\dot{\theta}_1(t), \dot{\theta}_2(t)) \to 0$ then $F_1(t) + F_2(t) \to 0$;
3. **energetic passivity**: i.e. there exists a finite constant $d \in \mathbb{R}$ s.t.

$$\int_o^t [F_1(\xi)\dot{\theta}_1(\xi) + F_2(\xi)\dot{\theta}_2(\xi)]\, d\xi \geq -d^2 \qquad \forall t \geq 0; \tag{7.3}$$

i.e., maximum extractable energy from the two-port closed-loop teleoperator is always bounded since equation (7.3) expresses the condition that the amount of energy that can be extracted by the human and work environment is limited by the initial stored energy d^2 (the closed-loop teleoperator does not generate energy by itself).

Then a control τ_i satisfies controller passivity, if there exists a finite constant $c \in \mathbb{R}$ such that:

$$\int_o^t \tau_i(\xi)\dot{\theta}_i(\xi)\, d\xi \leq c^2 \qquad \forall t \geq 0, \tag{7.4}$$

i.e. energy generated by the control action is always limited. Then for the overall system:

$$\int_o^t [\tau_1(\xi)\dot{\theta}_1(\xi) + \tau_2(\xi)\dot{\theta}_2(\xi)]\, d\xi \leq c^2 \qquad \forall t \geq 0. \tag{7.5}$$

For the mechanical teleoperator (7.2), it can be proved that control passivity (7.5) implies energetic passivity (7.3). [18] The latter fact enables to analyze energetic passivity of the closed-loop teleoperator by examining only the controller structure, which is often much simpler than that of the closed-loop dynamics.Furthermore by enforcing controller passivity, energetic passivity will be guaranteed robustly. Then the controller has to be design in such a way that are achieved respectively master slave position coordination, bilateral force reflection, energetic passivity of the closed loop teleoperator. A possible choice of the controller torque is:

$$\begin{cases} \tau_1(t) := \underbrace{-K_d(\dot{\theta}_1(t) - \dot{\theta}_2(t-\Delta t_2))}_{\text{delayedD}-\text{action}} \underbrace{-K_p(\theta_1(t) - \theta_2(t-\Delta t_2))}_{\text{delayedP}-\text{action}} \underbrace{-(K_{diss} + P_\epsilon)\dot{\theta}_1(t)}_{\text{Dissipation}} \\ \tau_2(t) := \underbrace{-K_d(\dot{\theta}_2(t) - \dot{\theta}_1(t-\Delta t_1))}_{\text{delayedD}-\text{action}} \underbrace{-K_p(\theta_2(t) - \theta_1(t-\Delta t_1))}_{\text{delayedP}-\text{action}} \underbrace{-(K_{diss} + P_\epsilon)\dot{\theta}_2(t)}_{\text{Dissipation}}. \end{cases} \tag{7.6}$$

K_{diss} is the dissipation gain and is needed to ensure passivity on the P-control action and P_ϵ is an additional damping ensuring master-slave coordination. Since the inherent device viscous damping can substitute this additional damping in this thesis P_ϵ is assumed equal to 0. The gain K_{diss} must be chosen in such a way to ensure closed-loop passitivity i.e. that must satisfy the condition[18]:

$$K_{diss} \geq \left[\frac{\sin \frac{w(\Delta t_1 + \Delta t_2)}{2}}{w} \right]^2 K_p K_{diss}^{-1} K_p \qquad \forall w \in \mathbb{R}. \tag{7.7}$$

A possible solution for K_{diss} can be chosen as:

$$K_{diss} = \frac{\tau_{rt}}{2} K_p, \tag{7.8}$$

where τ_{rt} is an upper bound of the round-trip delay i.e. $\tau_{rt} \geq (\Delta t_1 + \Delta t_2)$. It is important to note that given the cascade control in this work the K_{diss} had been chosen as

$$K_{diss} = \frac{\tau_{rt}}{2}(K_p K_d).$$

Since the sampling time is $t_s = 0.001s$ the given values for the round-trip delay was set up as $\tau_{rt} = 0.006s$. As will be showed further, this particular control architecture didn't give back the desired result. The stability of the system has been achieved but on the contrary transparency was not satisfied since the overall system was really stiff and hard to move.

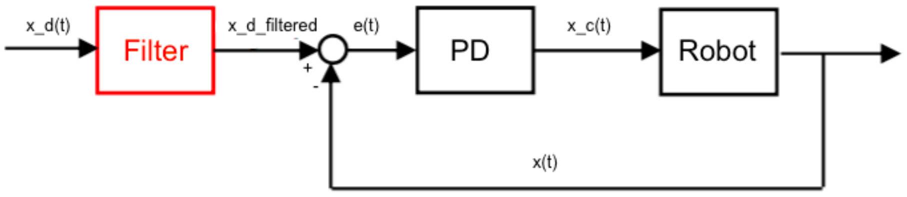


Figure 7.3: Pre-filtering of the target signal.

In order to perform better stability both the target position x_d at the master and slave are pre-filtered in order to avoid over-sudden variations on the actuators and any problems related to the presence of saturation (Fig.7.3). To keep the software computationally light a modified FIR filter of the first order has been implemented of equation in the time domain:

$$x_{dfiltered}(t) = (1-k) \cdot x_{mean} + k \cdot x_d(t), \tag{7.9}$$

where

$$x_{mean} = \frac{x_d(t-1) + x_d(t-2) + x_d(t-3)}{3}$$

is the mean of the previous three desired position configurations, and $k = 0.9$ weights the two contributions.

Then, the overall control scheme is in figure (7.4). The two colored block represent the two robots, the orange stays for the master and the purple for the slave respectively. As explained previously the overall architecture represent a resolved bilateral position-position control. As the name of this architecture suggests, the only information required is the position of the robots. In this way, the path of the master robot x_m is used as reference trajectory for the slave robot. The latter will try to follow it by means of the PD position controller seen in detail in section (6.2). The whole control scheme is totally symmetric, which means that the master robot also receives the position of the slave as a reference trajectory. Force reflection is obtained as a result of the actuation produced by the controller when tracking error grows due to the interaction between the slave robot and the contacted environment. It is important to note that the control is performed in the cartesian space so the blocks `direct kinematic` are required between the robots and the controller. Thank to them the master joint signals are transformed into equivalent cartesian movements of the slave reference point and then again transformed in servo joint command through the block `jacobi`. Similarly, the slave position is transformed and sent back to the master controller. Then again, when the position of the master and slave are different, there is force reflection at the master and force generation at the slave site. Both the positions are inputs of the socket communication blocks `server data` for the master and `client data` for the server.When the reference position is received, then the control acceleration can be computed thank to the PD controller with passivity gain K_{diss}. This acceleration is then converted to a force by means of the block `inertia`. Once this force is transformed in a torque, the gravity contributed is added. (Note that all this three block `inertia`, `jacobi` and `gravity` needs both the direct and the joint configuration as explained in appendix). Finally the control torque is multiplied for the gear gain in order to pass from the gear space to the robot space (the inverse of this operation will be done in order to come back to the gear space). It is important to note that the cascade structure of the PD controller allows the presence of the saturation block and to filter only the velocity leading to better safety and better performance for the manipulators.

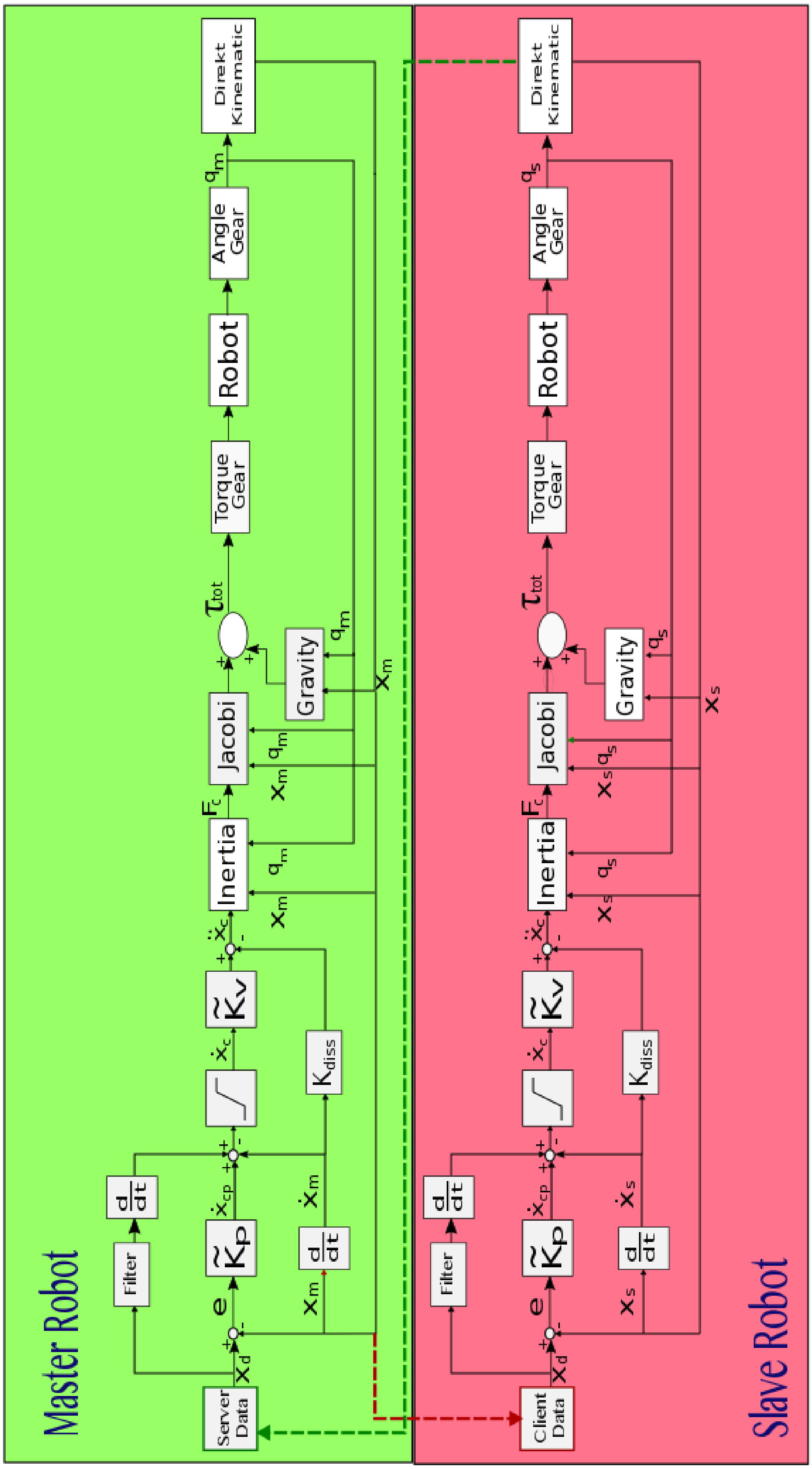


Figure 7.4: Overall control scheme.

Chapter 8

Bilateral Acceleration Control Scheme with Disturbance Observer and Kalman Filters.

In this chapter a novel approach has been developed. The goal to achieve was the same as previous (i.e. enabling a human operator to interact with a remote environment through interface devices) but the controller was a force-position bilateral control instead of a position-position bilateral control as before. As already seen one of the major objectives in designing bilateral control system is achieving transparency which is defined as a correspondence of positions and forces between the master and the slave. It has been pointed out that bilateral controllers cannot preserve transparency and stability simultaneously because of the presence of uncertainties in the system and environment. It has been considered that there is a compromise between these two goals.[19][20] The previous architecture scheme was more focused on stability and less on transparency since, as will showed on the practical result, the human could not move the teleoperator agilely in the free motion since the gains were set in such a way not to perceive a softened force on the master. (Ideally, the perceived force on master should resist the operator's motion in the same way as the force of a direct touch on the operator hand with a manipulated environment would do). Instead, in this force-position bilateral control scheme the goal to reach transparency had been achieved and in order to preserve stability a viscous term has been added for anchoring the robots to the ground. For improve transparency, an acceleration control had been developed in order to decompose the control into the common and differential modes.[21] The force servoing is attained in the common mode and the position error is regulated in the differential mode. Furthermore, the use of an acceleration control is useful in order to consider the conformity of force with position. The bilateral control have been developed sensorless. The use of a force sensor was rejected due to physical feasibility. Furthermore, although force sensors are simple to implement in the force control, they are still not competitive in applications that demands high accuracy, versatility and safety as in the pick and place robots. This is due to some disadvantages such as high noise, narrow bandwidth, vibrations due to the soft structure of the strain gauge and high prices. Instead of measuring the external force, it has been estimated through the use of Kalman filter based state observer (KFSO). The KFSO is made up with a Kalman filter in order to estimate the action/reaction force for each joint and a disturbance observer (DOB)for estimating the disturbances due to errors in the motor's model. Then the chapter can be divided in three different parts: a first section where the Kalman filter is described in order to estimate the position, the velocity, and the torque of each robot; then in the second section the disturbance observer for compensating the disturbances in the motor model is described and finally in the third section the bilateral control is presented.

8.1 Kalman Filtering.

As underlined above, the two Delta robots were provided with encoders that can measure the angle of each arm. Thus no force sensors have been used. As a consequence in order to achieve a bilateral controller the force has to be estimated. Then Kalman filters have been developed for the estimation of action/reaction forces. The Kalman filter is implemented and applied to estimate external torque of each joint of the AC servo motor. This filter is a recursive algorithm that update, at each time step, the optimal estimate of the system's state and the uncertainty associated with that estimate based on the previous estimate and noisy measurement data. The process of the Kalman filter consists of an operation in two phases: prediction and update. In the prediction state, the state estimate from the previous time step is used to produce an estimate of the state at the current time step. In the update state, the actual information at that time is measured and combined with the current prediction to adjust the state estimate. Then the final estimate become a weighted average of the a priori estimate and new values derived from the measurements data. The main goal of the use of Kalman filters is the optimal performances in real-time applications, since only the new measurements data need to be processed on each iteration and old data can be discarded. Figure 8.1 illustrated qualitatively how a Kalman

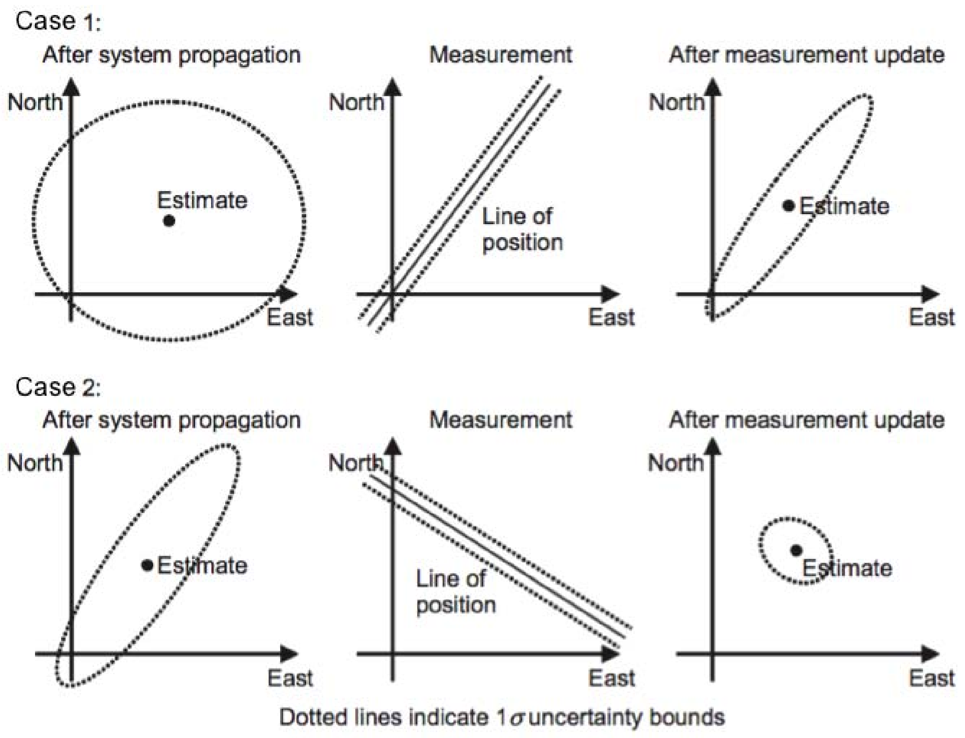


Figure 8.1: Position estimation through a Kalman filter.

filter can determine a position solution from successive incomplete measurements (single line of position). It is clear that thanks to the two steps prediction and update, the final position estimate has a smaller uncertainty. To enable optimal weighting of the data, a Kalman filter maintains a set of uncertainties in its estimates and a measure of the correlations between the errors in the estimates of the different parameters. This is carried forward from iteration to iteration alongside the parameter estimate. It also accounts for the uncertainties in the measurements due to noise. Indeed, in practical applications, the presence of Gaussian noise can deteriorate the performance of the control system and using Kalman filters it is possible to reduce the noise of the measurements systems effectively. The model of the measuring process to be estimated by Kalman filter is formulated as follow:

$$\begin{cases} x(k+1) = Ax(k) + Bu(k) + w(k) \\ y(k) = C(k)x(k) + v(k). \end{cases} \tag{8.1}$$

In (8.1) k represents the sampling instant; $x(k) \in \mathbb{R}^3$ is the vector of states consisting of angular position of motor, angular velocity and the external torque; $u(k) \in \mathbb{R}^2$ is the vector of input variables i.e. the input motor torque τ_{input} and encoder's measurements of motor's angle position; $y(k) \in \mathbb{R}$ represents the output of the system as measured by encoders. To account for errors in the model and also unmodeled disturbances a Gaussian random variable $v \in \mathbb{R}^3$ was introduced: where $v(k) \sim \mathcal{N}(0, Q)$ define the process noise. The sensor measurement model is not perfect either and this is modeled by measurement noise, another Gaussian random variable $w \in \mathbb{R}^2$ and $w(k) \sim \mathcal{N}(0, R)$. It is assumed that the covariance matrix of the process noise Q and the covariance matrix of the measurements noise R are symmetric and positive definite, i.e.:

$$Q = \mathbb{E}\{ww^T\} > 0, \tag{8.2}$$

$$R = \mathbb{E}\{vv^T\} \geq 0. \tag{8.3}$$

The cross-covariance matrix is assumed equal to zero $\mathbb{E}\{wv^T\} = 0$. The matrix $A \in \mathbb{R}^{3\times 3}$ describes the dynamics of the system, that is, how the state evolve with time. The matrix $B \in \mathbb{R}^{2\times 2}$ describes how the inputs are coupled to the system states. The matrix $C \in \mathbb{R}^{1\times 3}$ describes how the system states are mapped to the observed outputs. These matrix can be obtained from the equation of kinematics and dynamics. Indeed from the equation of motion, remembering the Newton's law $\tau_{ext} = J_m\ddot{\theta}(t)$ where J_m is the rotor inertia at the gear side computed in 4.22:

$$\theta(k+1) = \theta(k) + \dot{\theta}(k)T_k + \frac{1}{2}\ddot{\theta}(k)T_k^2 \tag{8.4}$$

$$= \theta(k) + \dot{\theta}(k)T_k + \frac{1}{2}\frac{\tau_{ext}}{J_m}T_k^2. \tag{8.5}$$

Analogously the angular velocity can be obtained as (where $\ddot{\theta}_{input}$ is the contribution to the velocity given by the input):

$$\dot{\theta}(k+1) = \dot{\theta}(k) + \ddot{\theta}(k)T_k + \ddot{\theta}_{input}T_k \tag{8.6}$$

$$= \dot{\theta}(k) + \ddot{\theta}(k)T_k + \frac{T_k}{J_m}\tau_{input}. \tag{8.7}$$

Finally the last relation of the state which gives back the value of the external torque τ_{ext}:

$$\tau_{ext}(k+1) = \tau_{ext}(k). \tag{8.8}$$

The last equation can mislead at a first sight. Indeed, the adaption of the torque is made previously by subtracting the measures that are the second input of the filter. In this way the difference between the measured and estimate force gives back the external torque. Then the model of the robot system can be formulated into the discrete state matrices as:

$$A = \begin{bmatrix} 1 & T_k & \frac{T_k^2}{2J_m} \\ 0 & 1 & \frac{T_k}{J_m} \\ 0 & 0 & 1 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 \\ \frac{K_{tn}T_k}{J_m} \\ 0 \end{bmatrix}, \qquad C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}. \tag{8.9}$$

and the overall model is:

$$\begin{cases} x(k+1) = \begin{bmatrix} \theta(k+1) \\ \dot{\theta}(k+1) \\ \tau_{ext}(k+1) \end{bmatrix} = \begin{bmatrix} 1 & T_k & \frac{T_k^2}{2J_m} \\ 0 & 1 & \frac{T_k}{J_m} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \theta(k) \\ \dot{\theta}(k) \\ \tau_{ext}(k) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{K_{tn}T_k}{J_m} \\ 0 \end{bmatrix} \begin{bmatrix} \tau_{input} \end{bmatrix} + w(k) \\ y(k) = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta(k) \\ \dot{\theta}(k) \\ \tau_{ext}(k) \end{bmatrix} + v(k). \end{cases} \tag{8.10}$$

The output of the filter $y(t)$ is the estimated position and will be subtracted from the encoder's position in order to obtain the innovation as will be explained better later. As explained before the Kalman filter has two different steps: the prediction step and the update step. The former is a prediction of the state based on the previous state and the inputs that were applied. the equation of this phase are:

$$\begin{cases} \hat{x}(k+1|k) = A\hat{x}(k) + Bu(k) \\ P(k+1|k) = AP(k|k)A^T + Q(k-1); \end{cases} \tag{8.11}$$

where $\hat{x}$ is the estimate of the state, $P \in \mathbb{R}^{3\times 3}$ is the estimated covariance, or uncertainty of the difference between the state and the measurements and Q as previously described is the covariance matrix. Then it can be said that equation (8.11) projects the current state estimate ahead in time. This is an open-loop step and its accuracy depends completely on the quality of the model A and B and the ability to measure the inputs u. The notation $k+1|k$ makes explicit that the left-hand side of (8.11) is an estimate at time $k+1$ based on information from time k. The prediction of P involves the addiction of two positive-definite matrices so the uncertainty, given no new information and the uncertainty in the process, has increased. To improve things it is necessary to introduce new information from measurements that comes from the three encoders of the DELTA. This new information added takes the name of innovation and is defined as:

$$M(k+1) = y(k+1) - C\hat{x}(k+1|k). \tag{8.12}$$

Equation (8.12) is the difference between what the sensors measure and what the sensors are predicted to measure: this means that the innovation quantify the real new information the sensors have introduced. Some of the difference will be due to the noise in the sensor, the measurements noise, but the remaining discrepancy indicates that the predicted state was in error and does not properly explain the sensor observations. The second step of the Kalman filter, the update step, uses the Kalman gain L, which is based on the total uncertainty measurement of matrix S:

$$\begin{cases} S(k+1) = CP(k+1|k)C^T + R \\ L(k+1) = P(k+1|k)C^T S^{-1}(k+1). \end{cases} \tag{8.13}$$

The Kalman gain L maps the innovation into a correction for the predicted state, optimally tweaking the estimate based on what the sensors observed. Then the estimated state vector and the error matrix are updated using the Kalman gain at every sampling instant as follows:

$$\begin{cases} \hat{x}(k+1|k+1) = \hat{x}(k+1|k) + L(k+1)M(k+1) \\ P(k+1|k+1) = P(k+1|k) - L(k+1)S(k+1)L^T(k+1). \end{cases} \tag{8.14}$$

Finally the covariance matrix of the process noise Q, the covariance matrix of the measurements noise R and the initial error covariance matrix between the state and the measurements P must be set in order to be able to obtain all the matrix previously analyzed. The matrix R has been set as half of the encoder's resolution. The initial matrix P was set with the values obtained asymptotically in the first experiment made with the Kalman filter. This was made in order to obtain from the very beginning reliable measurements; but since the matrix is updated at each step the matrix will asymptotically tends towards a certain matrix in any case. This lead to:

$$P_0 = \begin{bmatrix} 0.000169289 & 0.0 & 0.00 \\ 0.0 & 0.00167094 & 0.00 \\ 0.0 & 0.00 & 0.0306302 \end{bmatrix}; \tag{8.15}$$

it is important to note how the error covariance between the state and the measurements increases with derived measurements; i.e. the value for the velocity is bigger than the one for the position since the measurements of the velocity is derived from the one of the position as in (8.7). The same holds for the force. Regarding the choice of the process noise Q the question is more complicated. In this thesis a strong assumption has been made i.e. the end-effector moves at constant acceleration. This particular assumption is not satisfied, nevertheless the matrix chosen guarantee stability and great performances of the filter as will be showed is the experimental result section. As in [29] the matrix Q was set as:

$$Q = \mathbb{E}[ww^T] = q \begin{bmatrix} \frac{T_k^5}{20} & \frac{T_k^4}{8} & \frac{T_k^3}{6} \\ \frac{T_k^4}{8} & \frac{T_k^3}{3} & \frac{T_k^2}{2} \\ \frac{T_k^3}{6} & \frac{T_k^2}{2} & T_k \end{bmatrix}, \tag{8.16}$$

where $q = 0.1$ is the value (assumed constant) that comes from the autocorrelation function:

$$Q = \mathbb{E}[w(t)w(\tau)] = q(t)\delta(t-\tau) \tag{8.17}$$

where t and τ are two different instants. Note that also in this matrix the process noise parameters increase as the measurements are derived as explained previously. Since the measurements are derived the noise which affect them increases for the propagation of uncertainty.

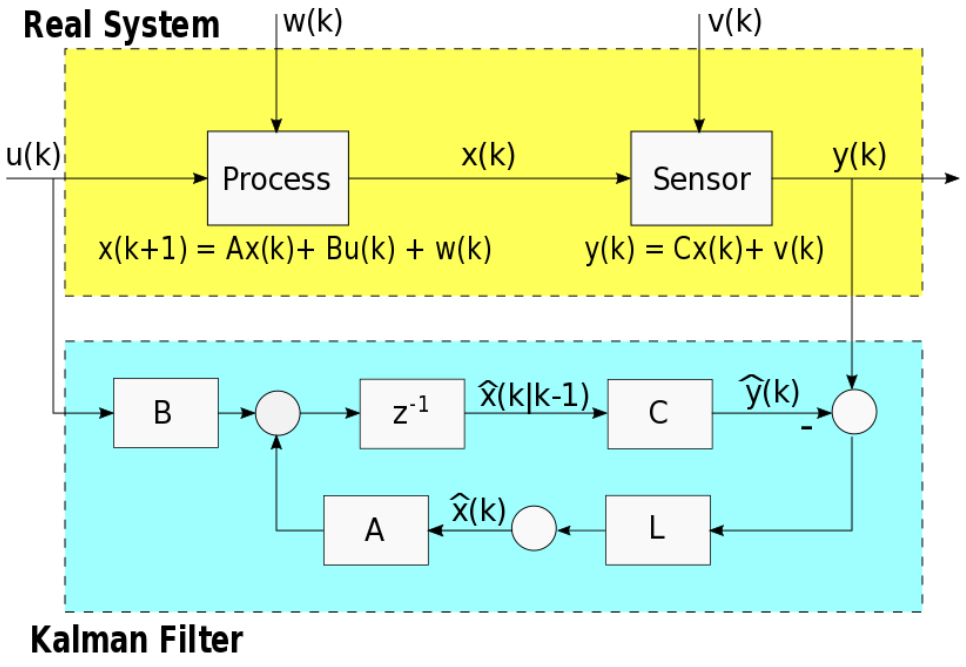


Figure 8.2: Overall Kalman Filter's structure.

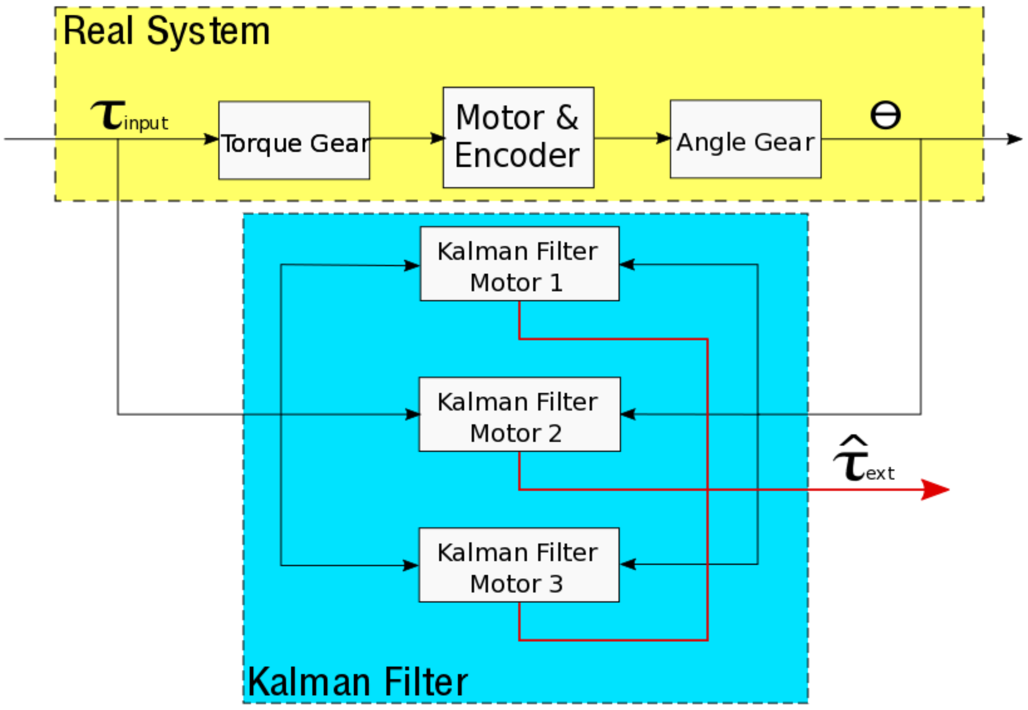


Figure 8.3: Overall structure for each robot.

Finally the overall structure of the kalman filter can be plotted as in figure (8.2). As can be seen all the equation are obtained through this block diagram. Lastly, as showed in figure (8.3) it is important to show that for each robot three kalman filter were required. Each motor indeed needs a filter in order to estimate the position, the velocity and the force of that particular cartesian axes. For this reason in the overall structure there were six different filters: three for the master and three for the slave respectively. In figure (8.3) it is also underlined as the filters takes as input the torque sent to the motors and the encoder's measurements. Then in this particular application the output of the filter is the state that can be taken from the input of the A block (see figure (8.2)). In the experimental result section will be showed how good are the performances of the filter.

8.2 Disturbance Observer.

In the ideal case the parameter variation and the disturbance should not give any significant effect to output in robust control but these cannot be neglected in most of the applications. In this thesis in order to enhance the performance of the controllers and to avoid instability two disturbance observers DOB have been developed (one for the master and one for the slave respectively). The DOB was developed for the first time by Kouhei Ohnishi[28] and from the input and output of the robot is able to estimate the disturbances. In fact, each observer has the model of motor dynamics and the estimated disturbance includes not only external effect such as load torque but also internal influences such as the errors between supposed model and real model. So the system with disturbance observer performs as supposed model with compensation of model variance. In DOB based robust control systems, the estimated disturbances, including system uncertainties, are fedbacked in a feed-back loop, namely inner-loop, so that the robustness of a system is obtained. To achieve performance goals, another feed-back loop, namely outer-loop, is designed independently by considering only nominal plant parameters, since a DOB nominalizes uncertain plant and suppresses external disturbances in the inner-loop. The main advantage of a DOB is that it suppresses external disturbances and system uncertainties without affecting the outer-loop performance controller. In figure 8.4, $G(s)$ and $G_n(s)$ denote uncertain and nominal plant models, respectively; $Q(s)$ denotes the low-pass-filter (LPF) of DOB; $C(s)$ denotes the outer-loop performance controller; r, d, and ξ denote reference, disturbance, and noise external inputs, respectively; d^* denotes system disturbances including external disturbances and plant uncertainties, and $\hat{d}^*$ denotes its estimation; and r_{con} denotes outer loop control signal. Given the main idea which stays beyond the observer it is possible to analyze more in detail the equation in the block in figure and where they come from.

Considering a process consisting of a DC motor affected by a disturbance torque τ_{diss} the equation of the dynamic of the motor is:

$$J_m\ddot{q} + B_m\dot{q} = \tau_m - \tau_{diss}, \tag{8.18}$$

where J_m is the rotor inertia, B_m is the friction coefficient and τ_m is the engine torque. Then from the relations

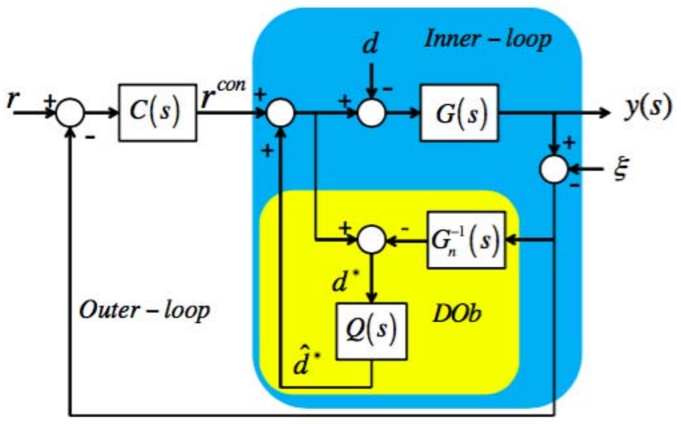


Figure 8.4: Block diagram of a DOB.

$\tau_m = J_m\ddot{q}$ and (6.1) i.e. $i_{ref} = \frac{\tau_m}{k_t}$ the following relation is obtained:

$$i_{ref} = \frac{J_m}{k_t}\ddot{q}, \tag{8.19}$$

where as already seen k_t is the torque constant.

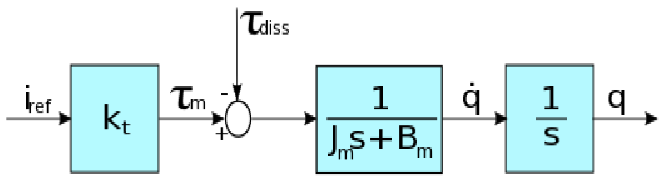


Figure 8.5: Block diagram of the motor.

The idea behind the DOB is to obtain the disturbance torque τ_d by subtracting the real torque that accelerate the load from the torque given as input from the controller.

$$\tau_d = i_{ref}k_t - (J_m s + B_m)\dot{q}. \tag{8.20}$$

Unfortunately the real exact model of the motor is difficult to obtain, since the rotor inertia J_m and the torque constant k_t are parameters that change with the angular position. These variations with respect of the nominal parameters can be seen again as disturbances and finally the disturbance torque can be defined as:

$$\begin{aligned}\tau_{diss} &= \tau_d + (J_m - J_{mn})s\dot{q} + F + (k_{tn} - k_t)i_{ref} && (8.21)\\ &= i_{ref}k_t - (J_m s + B_m)\dot{q} + (J_m - J_{mn})s\dot{q} + F + (k_{tn} - k_t)i_{ref}, && (8.22)\end{aligned}$$

where F denotes the friction torque and the subscript $_n$ denotes the nominal parameters.

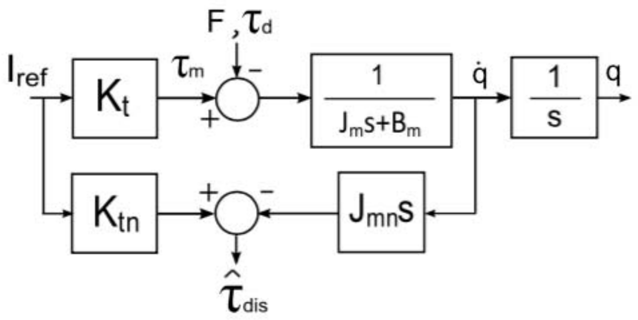


Figure 8.6: Block diagram of DOB.

Then the aim of the disturbance observer is to estimate the torque τ_{diss} in order to compensate it on the controller and to make the system immune to parametric changes and robust against disturbances and friction. It is possible to verify the accuracy of the scheme by computing τ_{diss} from the equation of 8.6, i.e.:

$$\begin{cases} (i_{ref}k_t - \tau_d - F)\left(\frac{1}{J_m s + B_m}\right) = \dot{q} \\ i_{ref}k_{tn} - J_{mn}s\dot{q} = \tau_{dis} \end{cases} \tag{8.23}$$

that becomes:

$$\begin{cases} (i_{ref}k_t - \tau_d - F) - J_m s\dot{q} + B_m\dot{q} = 0 \\ i_{ref}k_{tn} - J_{mn}s\dot{q} = \tau_{dis} \end{cases} \tag{8.24}$$

and subtracting (8.24b) from (8.24a) the (8.22) is obtained. Thus the observer of figure 8.6 has also some high frequency signal component due to the noise (mostly derived from encoders). Then in order to obtain a better estimate for the disturbance torque a low pass filter (LPF) is needed. The cut-off frequency of the filter g_{dis} is a critical parameter that has to be chosen properly: ideally it must be as larger as possible in order to make an accurate and fast estimate, but in reality it must limit as much noise as possible so the chose of this parameters is an important trade-off. In this work that parameter was chosen by trial and errors and the value $g_{dis} = 1000$ was selected. The overall scheme is in figure 8.7.

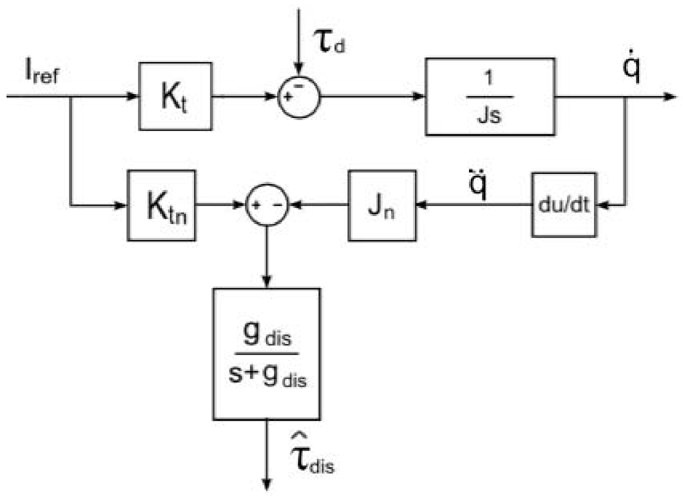


Figure 8.7: Block diagram of DOB.

In order to avoid the double operation of derivative a different scheme can be adopted. In figure 8.8 can be seen that only one derivation is required since the velocity is the only input needed. It is important to note that the input velocity $\dot{q}$ of the DOB was not obtained from sensors, since the Delta robots were provided just with position sensors, but from the Kalman filters state observer studied in the previous section. For this reason the cut-off frequency of the LPF could be set with an high values. Alternatively the measure of the velocity could be obtain as a double derivative of the position measure from encoders; this particular choice was not taken into account since when concerning with quantized signals the derivative operation can lead to deteriorated signal corrupted with noise. Then the overall scheme is shown in figure 8.8. Another important issue that have to marked is that in the control scheme adopted the input given to the robot is a torque. Summarizing the DOB has as input a torque and the estimated velocities and as output the disturbance torque that will be added directly to the engine torque. The final adopted scheme is in figure 8.9.

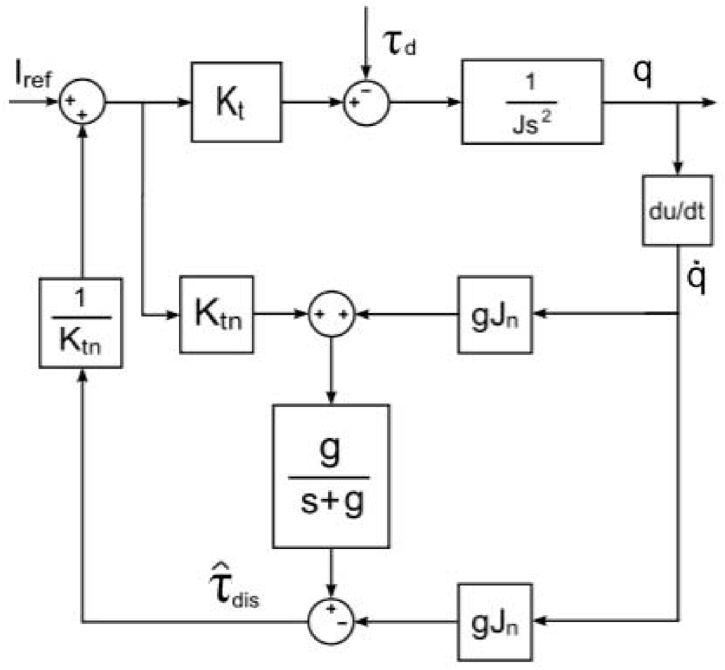


Figure 8.8: Block diagram of improved DOB.

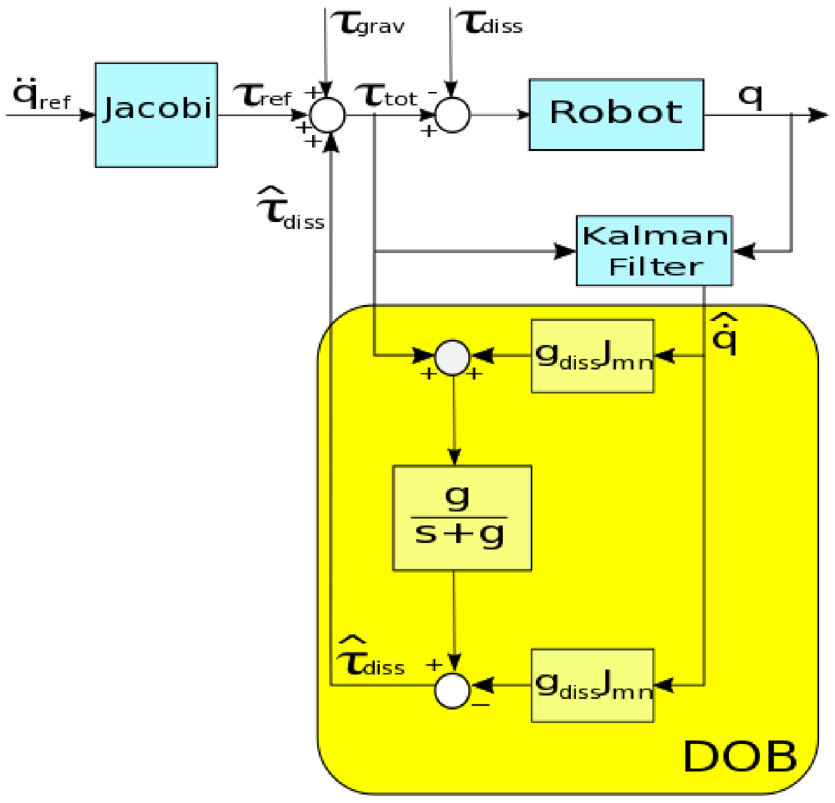


Figure 8.9: Block diagram of adopted DOB.

8.3 Bilateral Acceleration Control.

As already seen previously, the concept of bilateral control is made on the basis of the exchanged informations between motors and has to provide the sense of touch on the operator. The bilateral control supplies force feedback informations to the human operator through a master robot and on the other hand, a slave robot is applied to interact with unknown environments. The system shown in figure 8.10 functionally relates the position and forces of the operator and the environment, having master and slave devices as the interface systems. The general relation between position and forces in bilateral control can be described using the 4ch hybrid matrix H:

$$\begin{bmatrix} f_m \\ x_m \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_s \\ -f_s \end{bmatrix} \tag{8.25}$$

where h_{11} denotes the master side mechanical impedance, h_{22} can be interpreted as the a slave side mechanical admittance, h_{12} corresponds to the accuracy of force reflection and h_{21} refers to the position tracking.[22] The

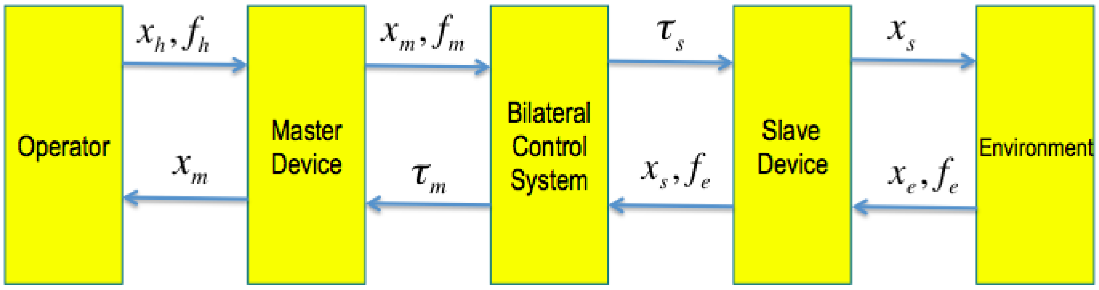


Figure 8.10: Structure and components of a bilateral control.

goal of bilateral control is to approximate the "ideal teleoperator" as closely as possible. The ideal functional relation provides transparency: the operator feels as if he is manipulating the remote environment directly. Position and force matching is important for reproduction of the environmental impedance. Here, positions of the master and slave are defined as x_m and x_s. The operational force and reaction force from the environment are defined as f_m and f_s. In order to satisfy the position-force matching the two equations that have to be satisfied are:

$$f_m + f_s = 0; \tag{8.26}$$

$$x_m - x_s = 0. \tag{8.27}$$

Equation (8.26) denotes that the position error between the master and the slave should be zero, and equation (8.27) denotes that the sum of the master and slave forces should be zero in order to satisfy the law of action-reaction. The bilateral control desired ideal conditions can be represented by:

$$\begin{bmatrix} f_m \\ x_m \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_s \\ -f_s \end{bmatrix}. \tag{8.28}$$

If conditions (8.28) are satisfied, the operator feels the environmental impedance Z_e. Indeed if the force applied to the slave satisfy: $f_s = Z_e x_s$, then when inserted in (8.25) the relationship between the position and the force in the master side becomes:

$$f_m = \frac{h_{11} - h_{12}Z_e}{h_{21} - h_{22}Z_e} x_m = Z_f x_m, \tag{8.29}$$

where Z_f is an impedance which the operators feels. Hence, when Z_f is equal to the environmental impedance Z_e the operator feels the real touch sensation of the remote object. By expansion of equation (8.29), the following equation is obtained:

$$f_m = \left(-\frac{h_{12}}{h_{21} - h_{22}Z_e} Z_e + \frac{h_{11}}{h_{21} - h_{22}} Z_e \right) x_m \tag{8.30}$$

$$= (P_r Z_e + P_o) x_m, \tag{8.31}$$

where parameters P_r and P_o defined respectively as "reproducibility" and "operationality" are parameters that can give back a feedback about the performance of the bilateral controller, and in this way can be used in order to find the values of the gain in the control architecture.[24] Reproducibility shows how precisely environment impedance is reproduced in master side. Operationality means how smoothly the operator manipulates master robot. Because the reproduction of environmental impedance in master side is the most important condition in bilateral teleoperation, $P_r = 1$ should be satisfied. Additionally, when small operational force, ideally $Po = 0$, is realized, the operator can feel real environmental impedance naturally. Later in this section will be showed as this parameters can help in order to find the force gain in the control architecture.

Indeed, as described by equations (8.26) and (8.27) the control has to be synchronized in terms of movements and force. Once it is assumed that disturbance compensation is applied to both slave and master two control modes namely, the common mode and differential mode, are implemented and applied to control both the position and force. This modal decomposition is achieved through the second-order Hadamard matrix $H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$. Then the virtual position of each mode is represented as follows:

$$\begin{bmatrix} x_c \\ x_d \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} x_m \\ x_s \end{bmatrix} = H_2 \begin{bmatrix} x_m \\ x_s \end{bmatrix}; \tag{8.32}$$

where x_c denotes the common mode and x_d the differential mode. The position signals from the optical encoders at the master side x_m and at the slave side x_s are transformed into the virtual positions in the common mode x_c and the differential mode x_d by using H_2. Equally for the force:

$$\begin{bmatrix} f_c \\ f_d \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} \hat{f}_m \\ \hat{f}_s \end{bmatrix} = H_2 \begin{bmatrix} \hat{f}_m \\ \hat{f}_s \end{bmatrix}; \tag{8.33}$$

where the force estimations from the KFSO at the master side $\hat{f}_m$ and at the slave side $\hat{f}_s$ are transformed into the virtual forces in the common mode, f_c, and the differential mode f_d, by using H_2 Hadamard matrix. The force control in the common mode can be separately analyzed as the proportional controller. On the other hand, the position control in the differential mode can be analyzed as the proportional-derivative controller. In order to attain the robustness in the system the modal space design of each mode is controlled in the acceleration dimension, i.e. that equations (8.26) and (8.27) become:

$$\ddot{x}_m - \ddot{x}_s \to 0; \tag{8.34}$$

$$\ddot{x}_m + \ddot{x}_s \to 0. \tag{8.35}$$

Then it can be written:

$$\ddot{x}^c_{ref} = \left(f^c_{cmd} - \hat{f}_c \right) K_f \tag{8.36}$$

$$\ddot{x}^d_{ref} = \left(x^d_{cmd} - \hat{x}_d \right) K_p + \left(\dot{x}^d_{cmd} - \dot{\hat{x}}_d \right) K_d \tag{8.37}$$

where f^c_{cmd} is the force command and x^d_{cmd} is the position command and both must be set to zero in order to solve conditions (8.26) and (8.27). K_f is the proportional gain of force control and K_p and K_d are the gains of the PD-controller. At this point the desired accelerations can be computed using the inverse of the Hadamard matrix H_2^{-1} in order to transform again the acceleration control signals from the modal space to the real space in master and slave robot respectively,i.e.:

$$\begin{bmatrix} \ddot{x}^m_{ref} \\ \ddot{x}^s_{ref} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} \ddot{x}^c_{ref} \\ \ddot{x}^d_{ref} \end{bmatrix} = H_2^{-1} \begin{bmatrix} \ddot{x}^c_{ref} \\ \ddot{x}^d_{ref} \end{bmatrix}; \tag{8.38}$$

where $\ddot{x}^c_{ref}$ and $\ddot{x}^d_{ref}$ are the acceleration control signals in the common and differential modes, respectively; and $\ddot{x}^m_{ref}$ and $\ddot{x}^s_{ref}$ are the acceleration control signals at the master and slave side, respectively. The overall scheme is in figure 8.11.

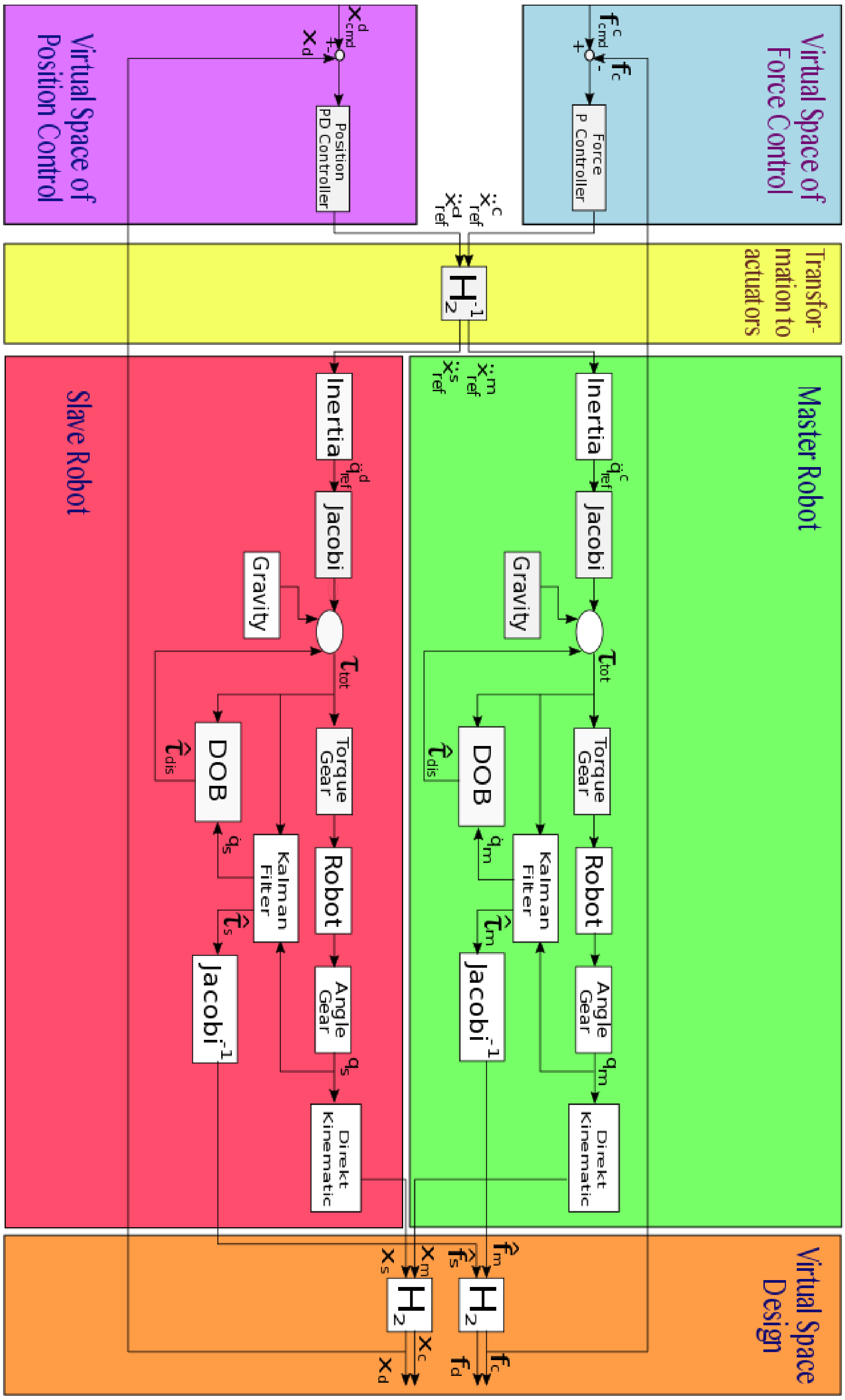


Figure 8.11: Bilateral acceleration control scheme.

Once the overall control architecture had been implemented the P and PD controller gain must be set. As previously anticipated this choice can be made exploiting the parameters of reproducibility P_r and operationality P_o. This values must be chosen taking into account the stability of the overall scheme. Then the three goals of the control can be summarized as:

1. stability,
2. reproducibility,
3. operationality.

Since in this control system, force control and position control work simultaneously in the acceleration dimension the structure is not simple and it is difficult to analyze the stability of the system. To solve this problem the stability of the two modal space has been studied independently.[25] The stability of the PD-position controller has already been studied. In order to improve the transparency the constant value of the angular frequency w_0 is reduced by a factor 2.5. This reduction doesn't influence the stability of the controller, but can slow down the answer of the system; nevertheless the advantage on less stiffness on master exceed the the $\frac{1}{2.5}$ slower factor on the rise time. On the contrary the P-force controller is stable according to the choice of the gain K_f. Indeed, force gain should be as large as possible in order to improve the performance. However, this can deteriorate stability also because force information has some noise. The choice of a big K_f is suggested also from the parameters P_o. As seen before in the ideal case $P_o = 0$ in order to make the operator feel real environmental impedance naturally. However ideal conditions (8.28) cannot be realized actually due to noise and errors in the model, i.e:

$$\begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} = \begin{bmatrix} \frac{1}{K_f}s^2 & -1 \\ 1 & 0 \end{bmatrix} \neq \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}. \tag{8.39}$$

Indeed the parameter K_f should be as large as possible in order to have $P_0 \simeq 0$. In [25] simulations have been made in order find how P_r and P_o are related under time delay and the result is in figure 8.12. So an agreement

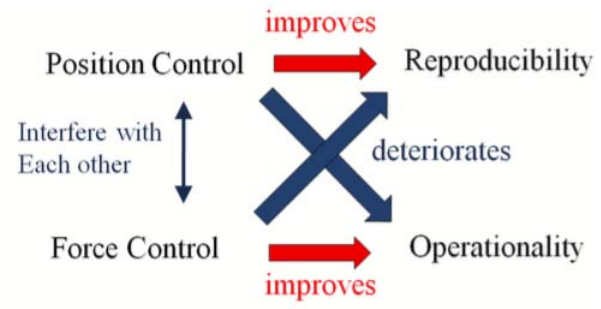


Figure 8.12: Relation of reproducibility and operationality with each control under time delay.

have been made in order to guarantee high performances and the force gain $K_f = \frac{1}{3}$ had been selected by trial and error.

As mentioned above, the 4 channel architecture is optimized for good performance, but can easily become unstable under delay. By injecting to the robots virtual damping, which is obtained by multiplying the velocity of the robots by a damping constant, it is possible to obtain an architecture which is more robust against time delays. Then different values of damping injection have been added to the control as in [26] and equation (8.36) and (8.37) become:

$$\ddot{x}^c_{ref} = \left(f^c_{cmd} - \hat{f}_c\right) K_f - dx_m, \tag{8.40}$$

$$\ddot{x}^d_{ref} = \left(x^d_{cmd} - \hat{x}_d\right) K_p + \left(\dot{x}^d_{cmd} - \dot{\hat{x}}_d\right) K_d - dx_s; \tag{8.41}$$

where b is the ground gain. These damping terms can be used to partially compensate the energy flow caused by time delay, and this makes the damping injection based teleoperation more robust against delay. In [26] it has been demonstrated through an analysis of the $h_{ij} \quad i,j = 1,2$ parameters under time delay, that when there is no delay, the damping injection based system becomes identical to a normal 4 channel teleoperator, except for the h_{11} parameter of the Hadamard matrix, which means that the robots will move in a damped environment, when there is free motion. In a more mechanical view the adding of damping terms in the control means damping to ground the bilateral system. In fact, the overall system can be seen as two masses connected by a spring and a damper (see figure 8.13). Then the two damper can be seen as a way to attach the whole system to the ground. In this way when the operator moves the master, both the system don't suddenly fly out, thank to the dampers which slow it down. By adopting the 4-channel teleoperation and using local damping on the robots

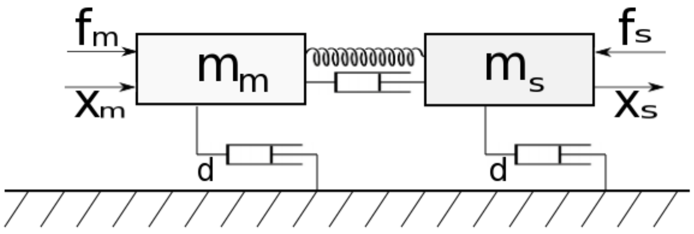


Figure 8.13: Mechanical interpretation of bilateral control.

robust teleoperation with considerably good transparency can be realized. This method does not guarantee global stability, under all possible frequencies, however it greatly improves the robustness to time delay without much performance loss. In [26] a numerical analysis has been made in order to study the connection between stability and transparency in bilateral teleoperated systems. Stability robustness of damping injection at various delay times, damping architectures and damping amounts had been analyzed. The article underlines how the damping improve stability exploiting the Llewellyn's absolute stability criterion which says that the following conditions must hold in order to guarantee the stability of a LTI teleoperation system[27]:

- the hybrid parameters h_{11} and h_{22} should have no poles on the right half plane;
- any poles of h_{11} and h_{22} on the imaginary axis are simple and have real and positive residues,
- the inequalities:

$$\Re(h_{11}) \geq 0; \tag{8.42}$$

$$\eta(\omega) = -\frac{\Re(h_{12}h_{21})}{|h_{12}h_{21}|} + 2\frac{\Re h_{11}\Re h_{22}}{|h_{12}h_{21}|} \geq 1 \tag{8.43}$$

  should hold on the $j\omega$ axis for all $\omega \geq 0$ where $\eta(\omega)$ is called the network stability parameter.

In fact when there is no damping injection, the 4 channel controller has a stability parameter which is upper-bounded by 1, this means that it is certainly unstable. However when there is damping at the master side or both sides the network stability parameter η is greater than or equal to 1, at most times, which means that it is less likely to become unstable. In this thesis different values of the ground gain d have been used in order to find the best match transparency-stability since the minimum transmitted impedance for the damping injection case is close to the constant d. This means that the operator feels the impedance d during free motion. There is a compromise here between stability and transparency.

Chapter 9

Experimental results

In order to evaluate the performances of the different control scheme described previously many experimental test have been studied. For each control scheme (PD control and bilateral acceleration control) the transparency between master and slave has been studied i.e. the correspondence of positions and forces between the two robots. More in details the capability of both master and slave of following the position setpoints given and to generate a correct feedback's action depending on the presence or not of obstacles have been analyzed.

9.1 PD Control scheme with gravity compensation and passivity terms.

In this section the performance of the first control scheme are showed. For this scenario two test have been made; in the first the slave moves in free motion in an open space, in the second the human operates the master to move the slave close to some obstacle and keeps pushing in order to feel the reaction (reflection) force of the object. These free motion and contact behaviors are all stable, as passivity has been enforced. Both experiments begin with master and slave initialized in the same position and end as the operator stop both the device from the terminal. For each experiment the position and the force reflection of master and slave are represented for 5, 10 and 50 seconds respectively.

9.1.1 Case A: free motion.

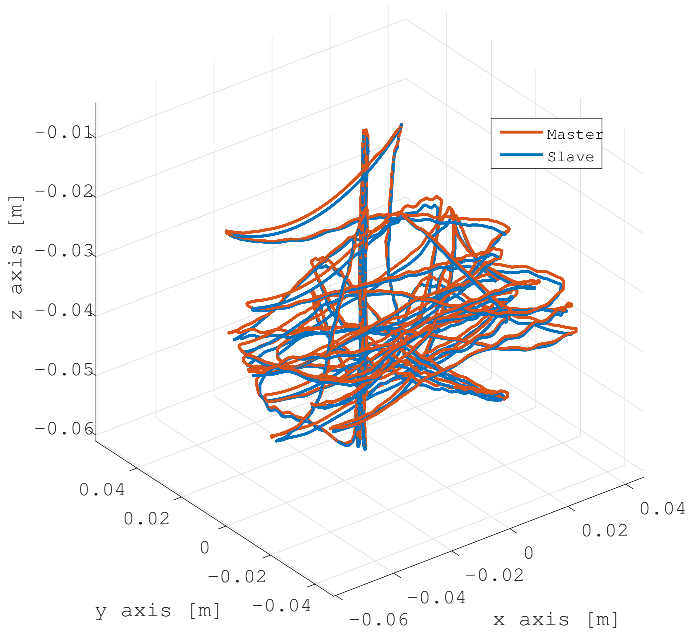
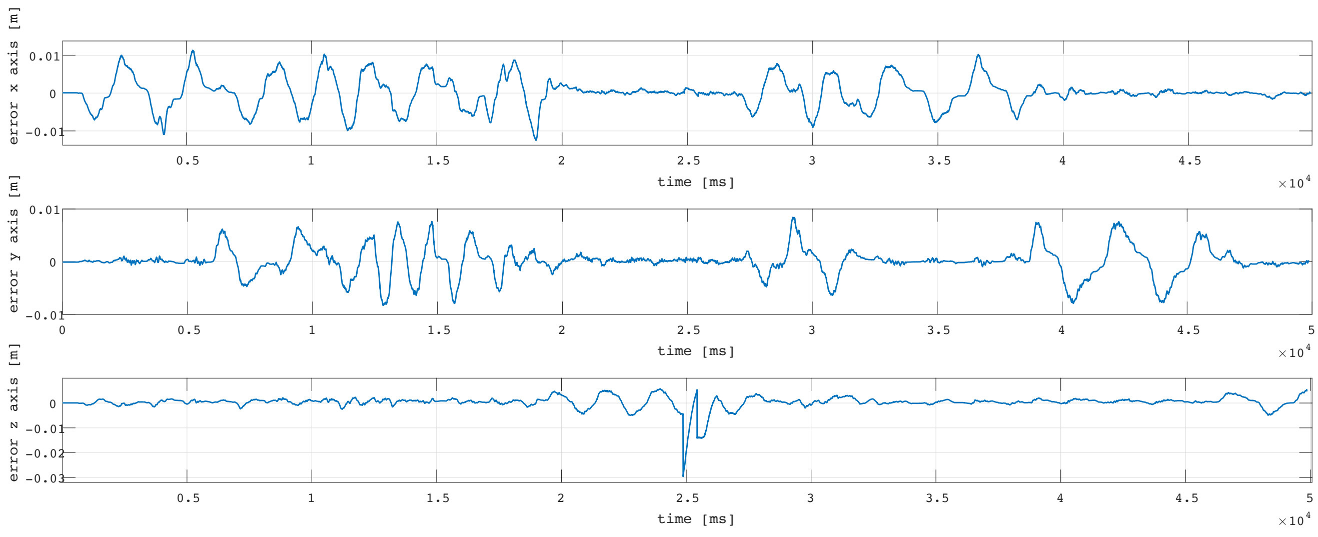


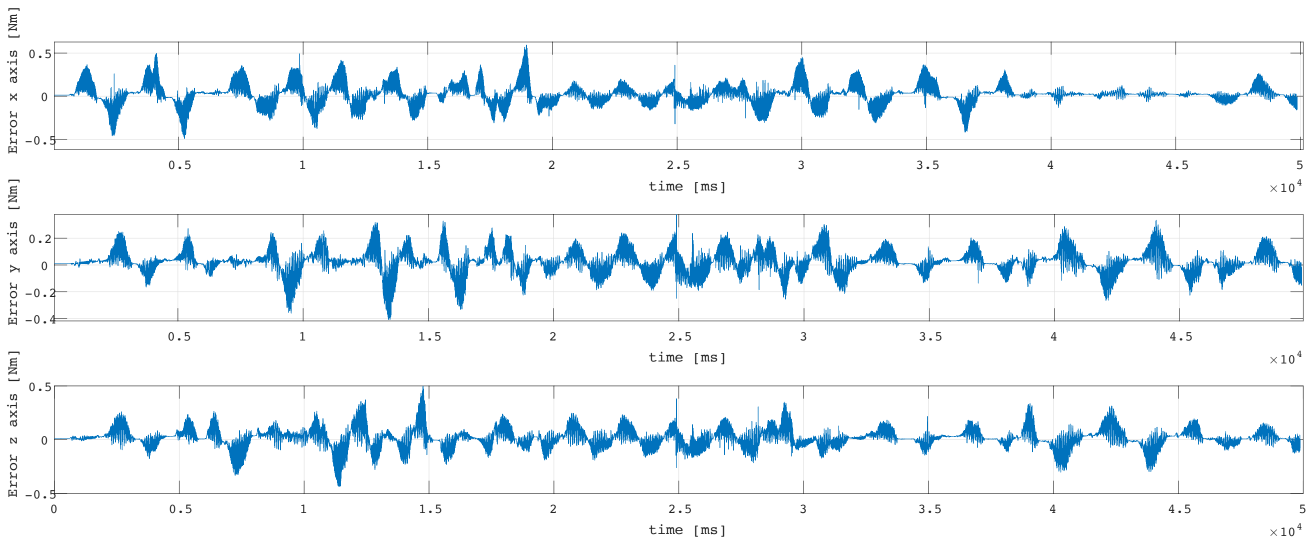
Figure 9.1: 3D workspace position response in absence of obstacle.

In the figure above the 3D workspace position response has been plotted when the human operator interacted with the travelling plate. Free motion means that a human operator manipulates the master robot freely and the slave robot doesn't touch anything. As shown in figure 9.1 the operator can move the master in free motion and the slave follows the path of the master. This can be better illustrated by the following figures. In figure

9.3 the end effector's trajectory respectively in the x,y, and z axis of master and slave have been plotted. The blu is for the master and the orange characterizes the slave robot. As it is possible to see more clearly in the 5 seconds plot there is a delay between the two robot which can be estimate around $\simeq 65[ms]$. From the plots it is clear that despite the delay, in free motion the master and slave positions become coordinated with each other. The delay effect is more clear in the plot of the force. Because of the delay, when the master starts to move, the slave is still stopped and sends back to the master a force feedback consistent with its status. Due to this, the master is hard to move for the operator since it follows the slave trajectory and the resulting movement is short jerking. In figure 9.5 the latter phenomenon is clear. For this reason all the force plot follow this sawtooth trend. In order to made the master less hard to move the derivative gain in the PD control was reduced. This lead to a less transparency in the amplitude of the server's force sent back to the master. Apart from that in the force plot in figure 9.4 it is possible to see how the action-reaction principle is respected. The master and slave force are nearly equal and opposite and the sum of this two force should be zero. On the other hand the difference between the positions should be zero given that the slave follow the master's path. Again because of the delay this two target are far from been achieved as illustrated in figure 9.5. In fact the error in position tracking is quite satisfactory since it is ten times lower in terms of the order of magnitude, but the force reflection order is still very big.

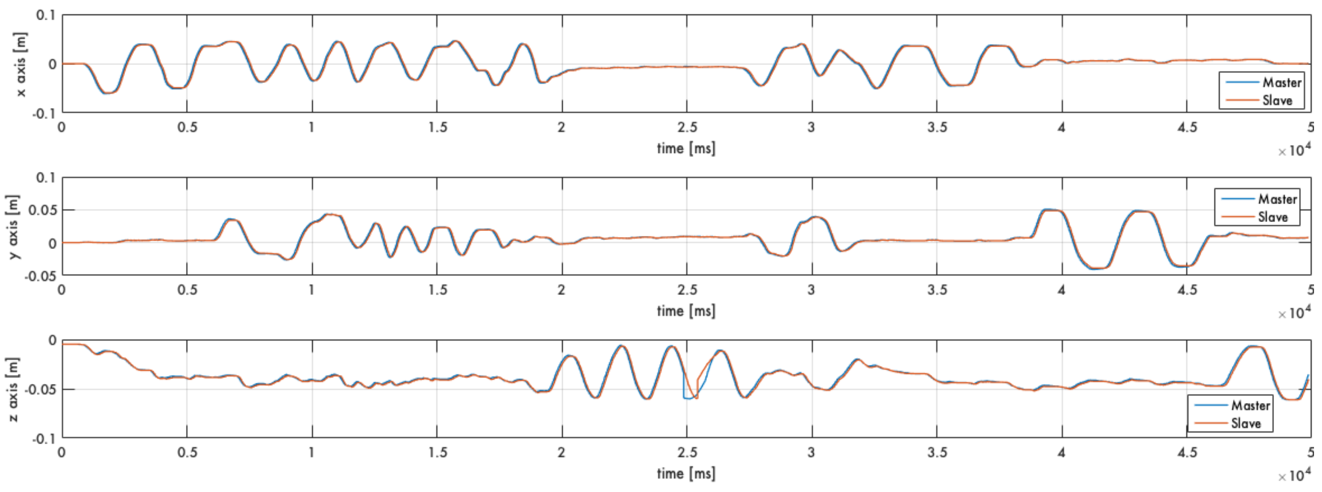


(a) *Range of 50 seconds.*

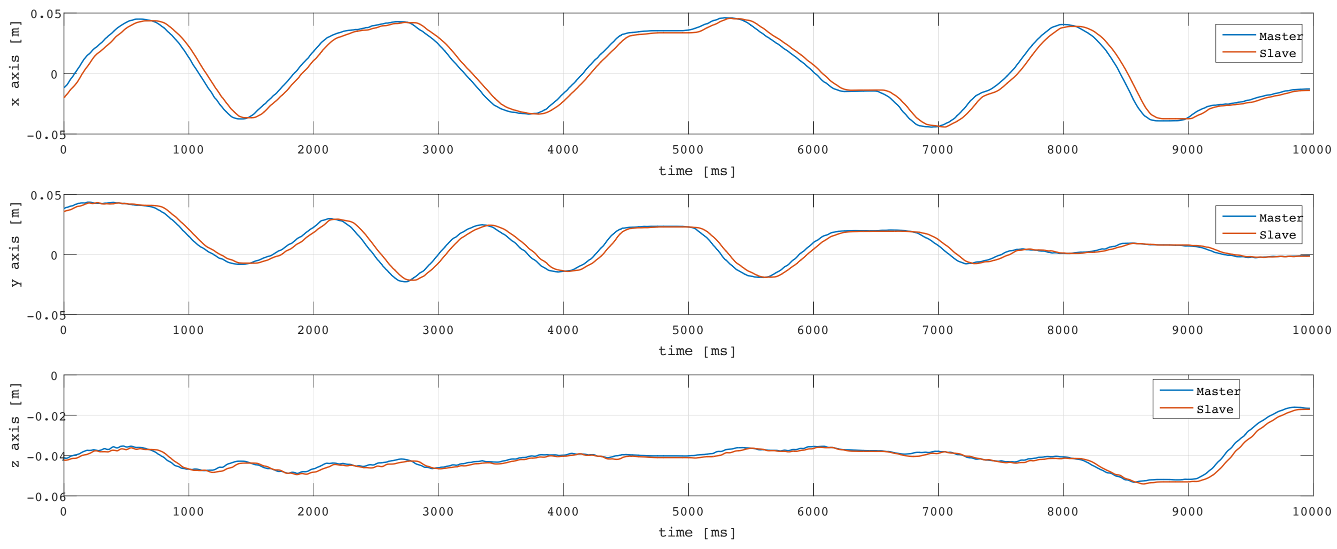


(b) *Range of 50 seconds.*

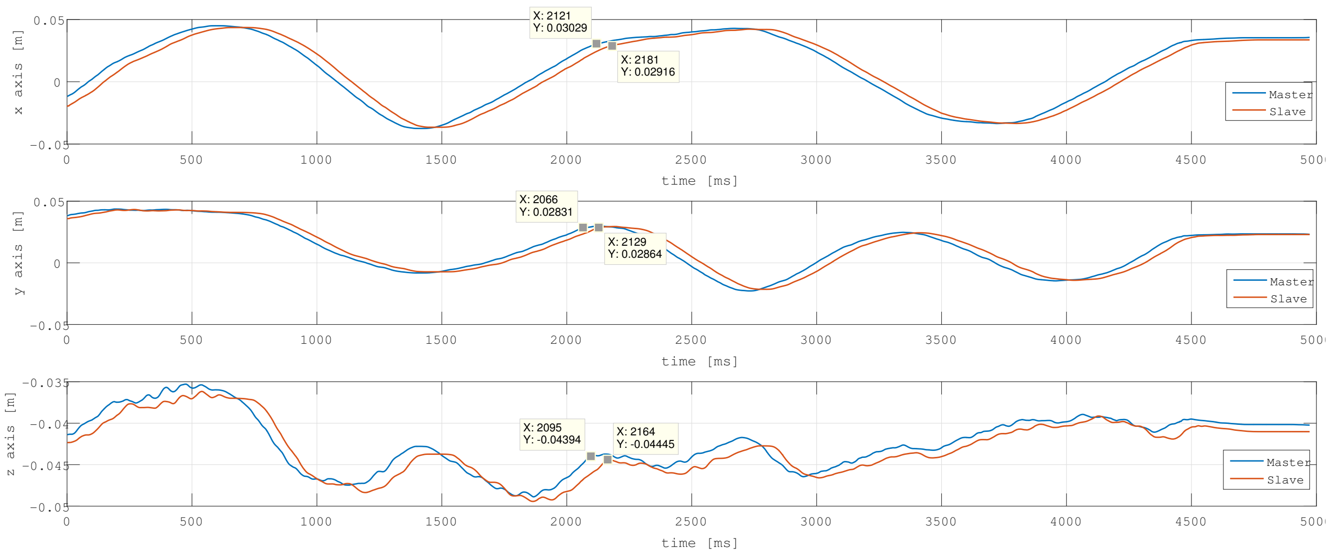
Figure 9.2: Position and force error in absence of obstacle.



(a) *Range of 50 seconds.*

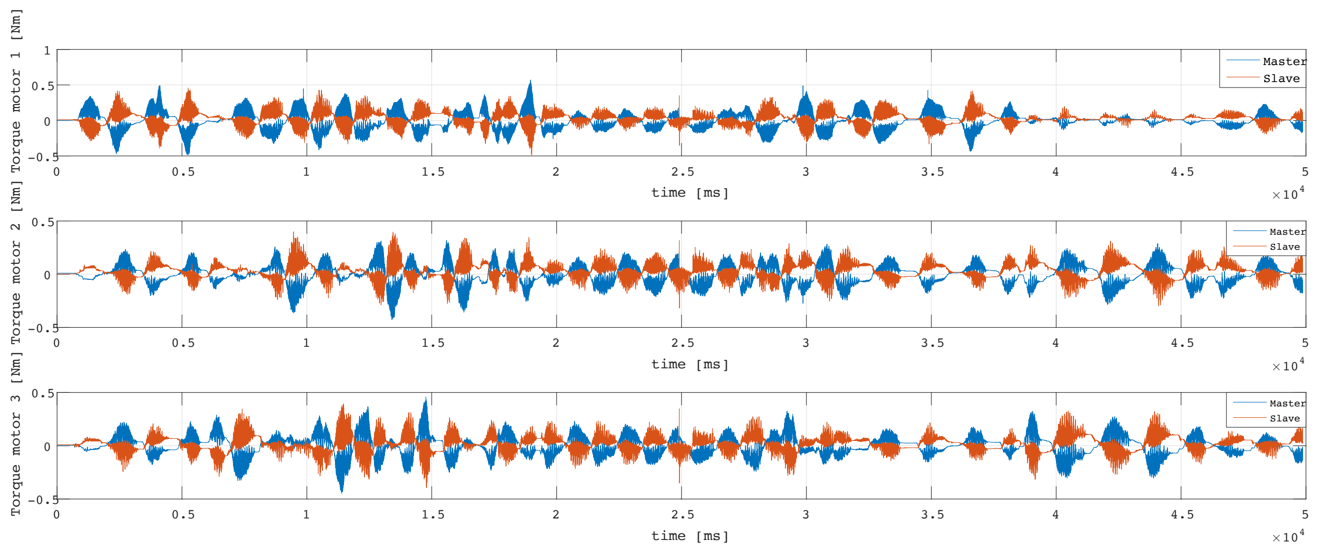


(b) *Range of 10 seconds.*

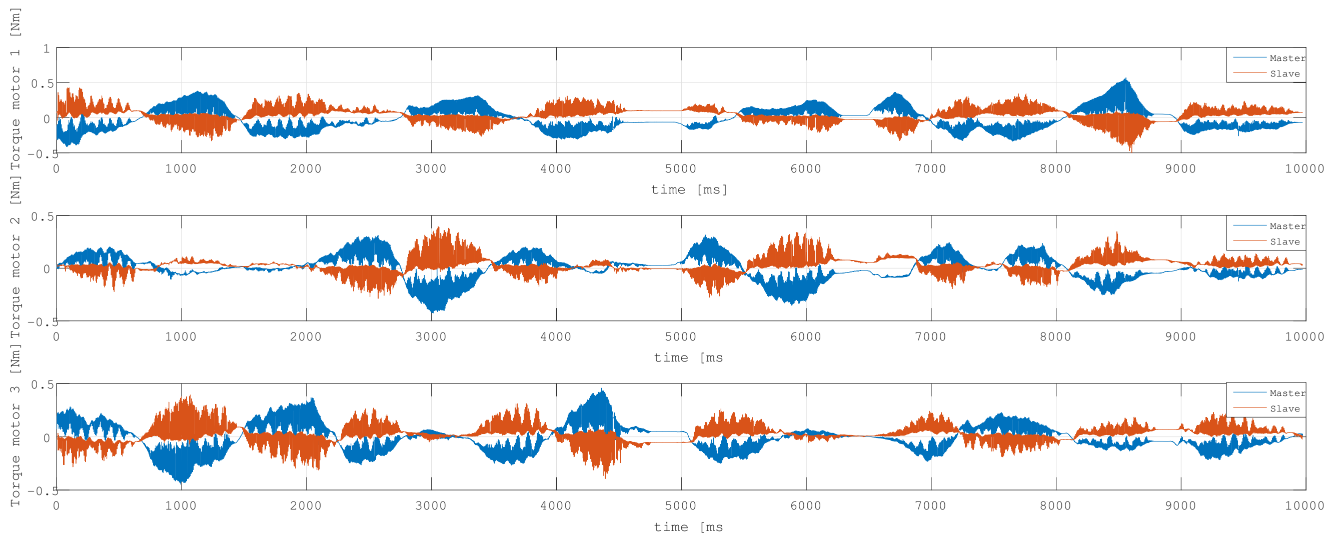


(c) *Range of 5 seconds.*

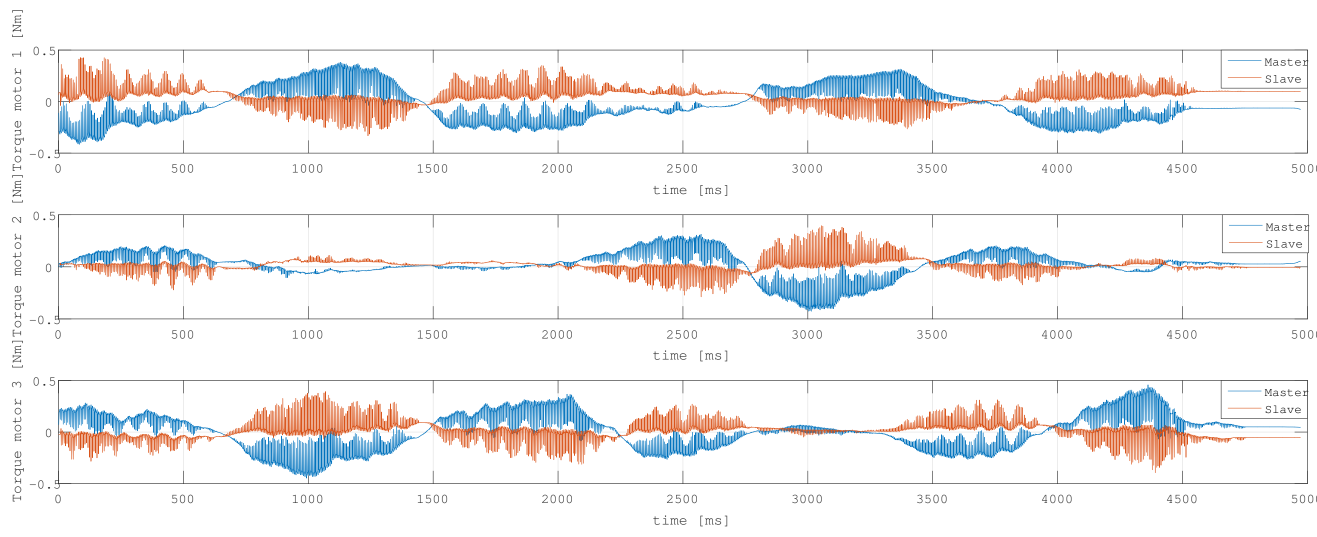
Figure 9.3: Position response in absence of obstacle.



(a) *Range of 50 seconds.*

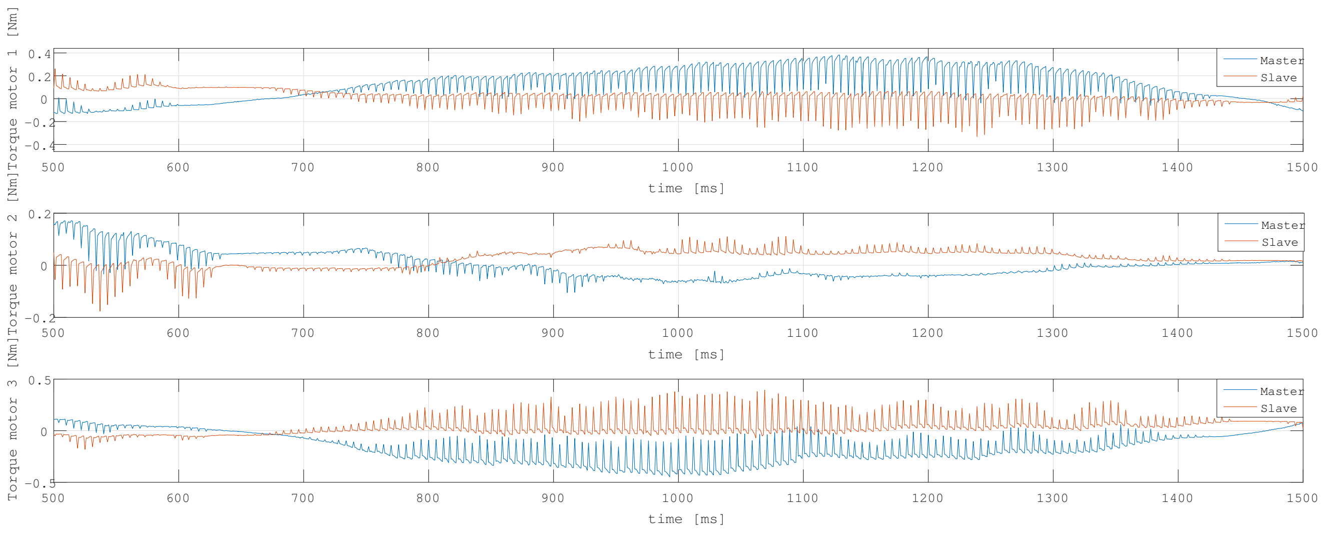


(b) *Range of 10 seconds.*

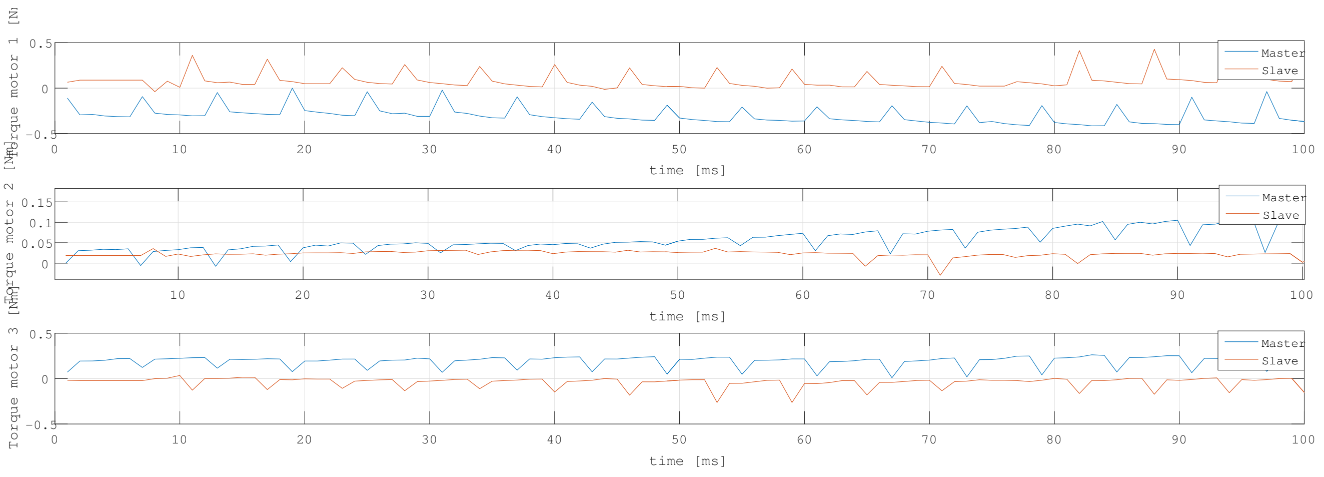


(c) *Range of 5 seconds.*

Figure 9.4: Force reflection in absence of obstacle.



(a) *Range of 1 seconds.*



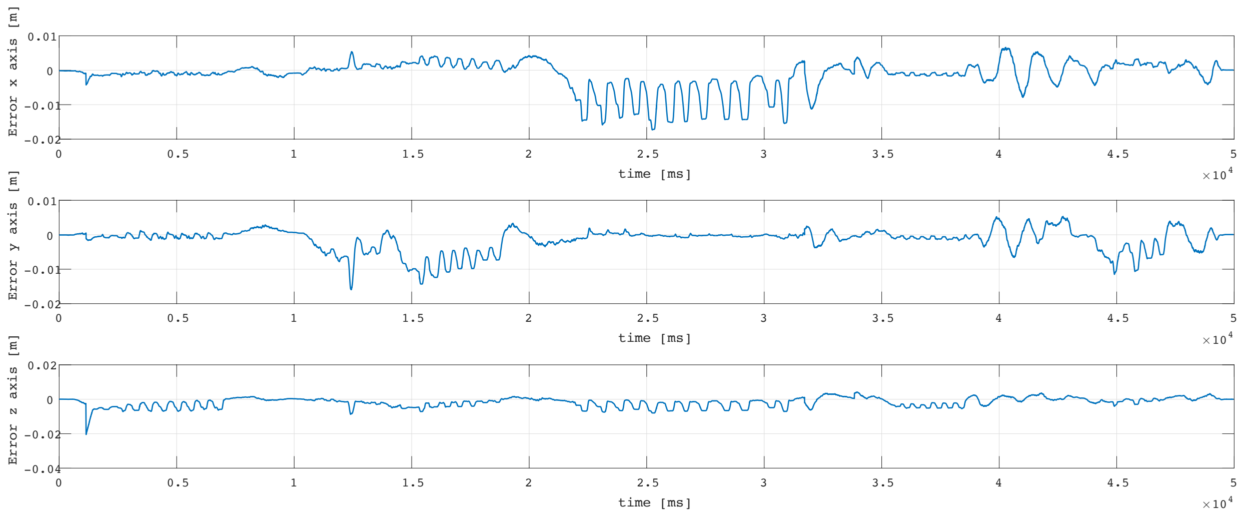
(b) *Range of 100 milliseconds.*

Figure 9.5: Force reflection in absence of obstacle.

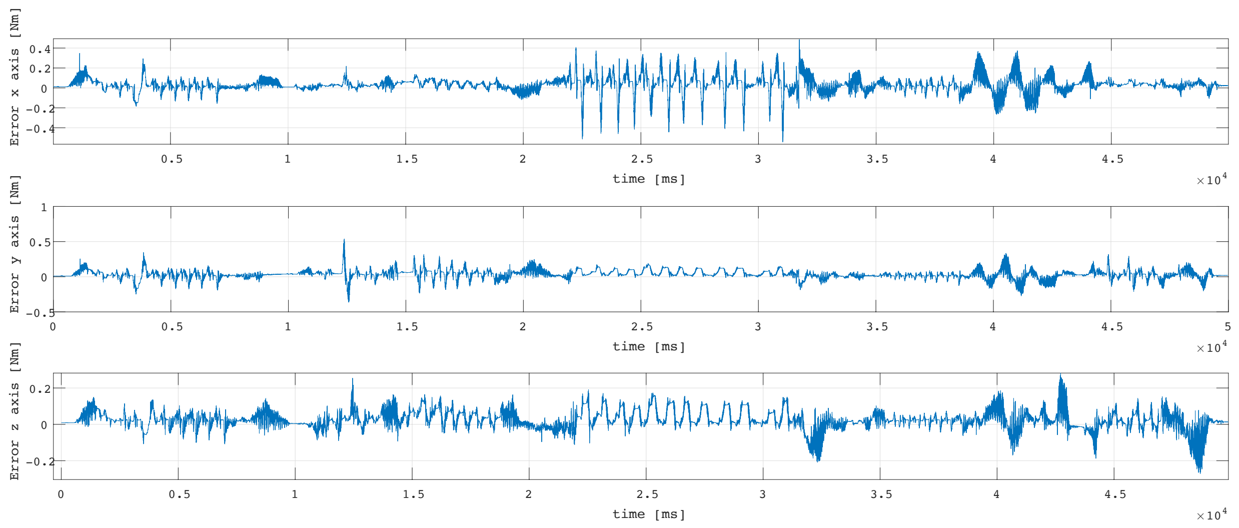
9.1.2 Case B: motion in presence of obstacles.

Secondly, the experiment was conducted on contact motion. Contact motion means that a human operator manipulates the master robot and the slave robot touches environments. In this case all the previous observation made in the free motion remain valid. In fact the time delay is again the undesirable lead, but a new problem can be seen in this new scenario. As explain above the derivative gain has been reduced in order to allow the operator to move the master. In this way the stiff control action was reduced and the proportional action is predominant. For this reason when the slave robot reaches an obstacle and the operator keeps pushing the master in the direction of the obstacle an elastic behavior can be seen. In fact when the master feels the force of the server and the inability to continue the motion in a certain direction if it keeps pushing it can feel the elastic force due to the control architecture, see figure 9.7 especially in the x axis in the 50 seconds range. The latter fact can lead to a wrong interpretation of the operator concerning the consistency of the obstacle reached. This elastic behavior can be seen also in the plot of the force 9.8. Thank to the passivity terms which lead to a dissipation action a stable contact with the environment is achieved which is evident from the positions of both the masters and slaves in figure 9.7. The slave stays at the position where the obstacle is located after the contact happens. In figure 9.6 the position and force error between the two robots have been plotted. Again thanks to the delay between the two, both the error seems not negligible. In the presence of obstacle the performance are even worst and also the position error is not enough satisfactory since in the worst case it reaches nearly half of the position range.

Finally it can be said that the most significant advantage of the position-position controller has proved to be its stability; but at the same time it is unable to offer both large force-reflection ratios and light maneuvering capability. Indeed the lack of high frequency force feedback and transparency limits the capabilities of the teleoperation system.

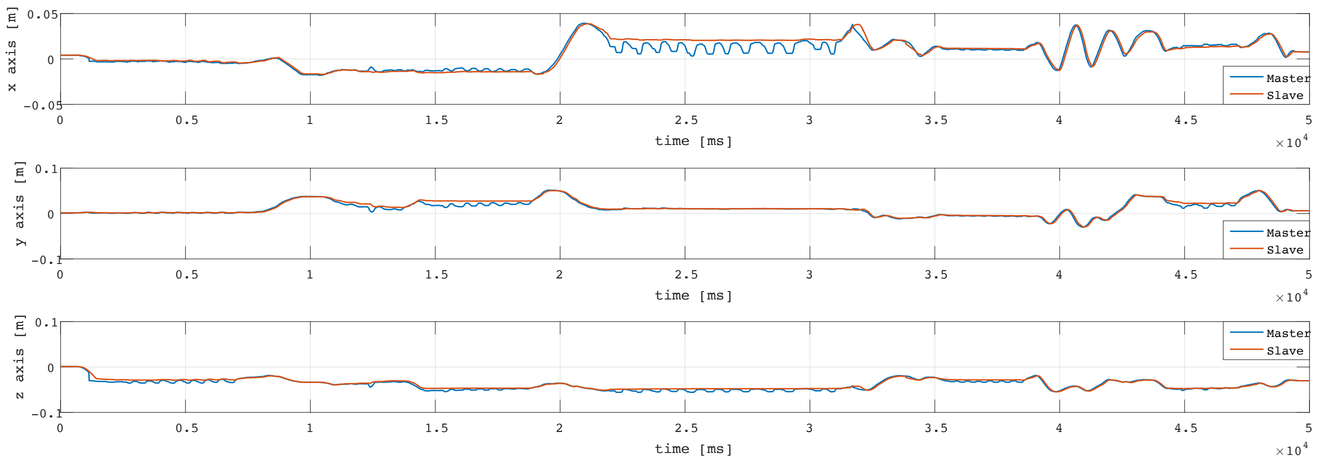


(a) *Range of 50 seconds.*

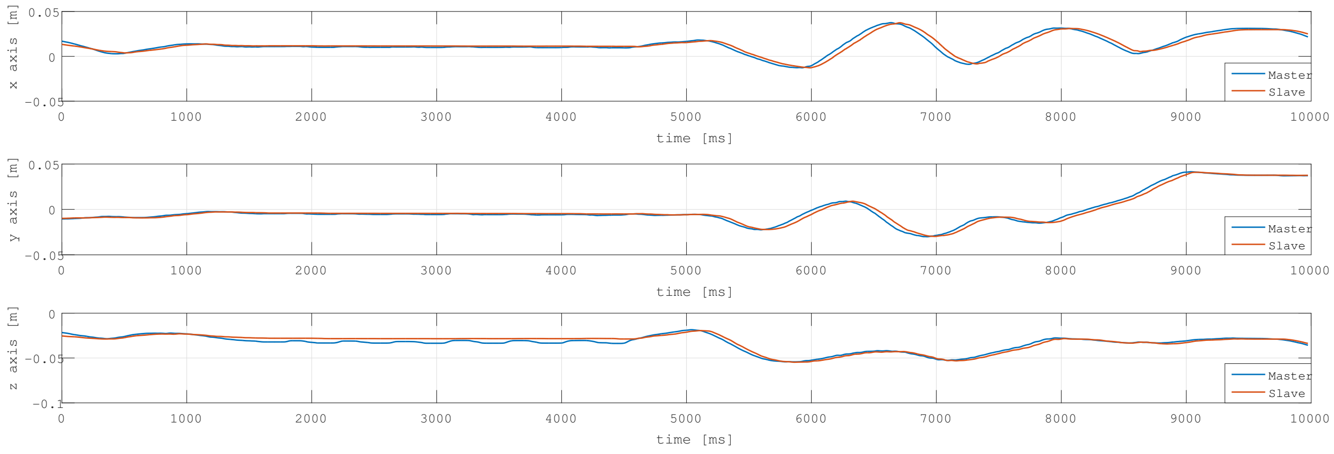


(b) *Range of 10 seconds.*

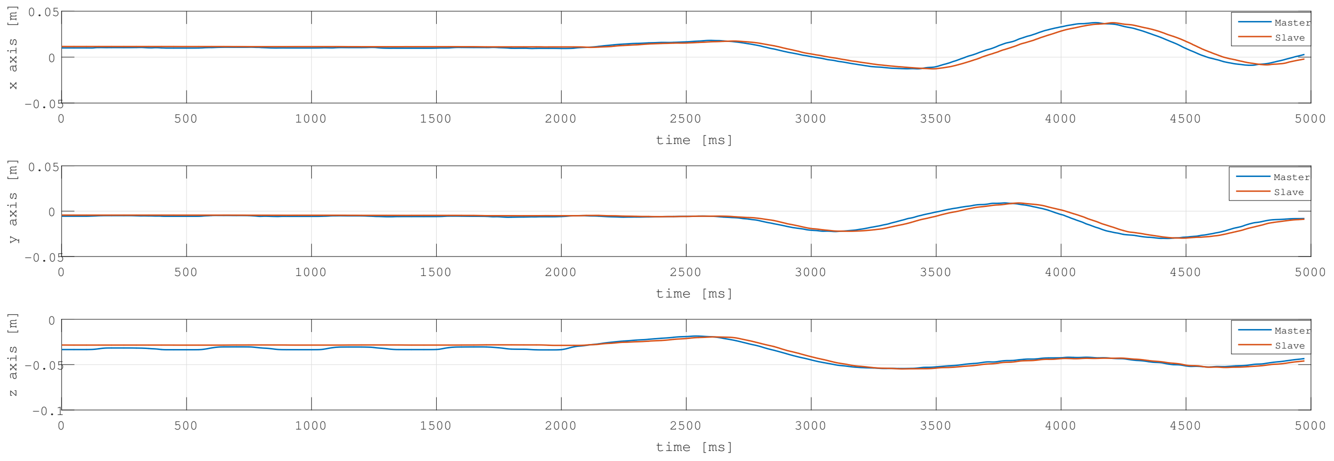
Figure 9.6: Position and Force error in presence of obstacles.



(a) *Range of 50 seconds.*

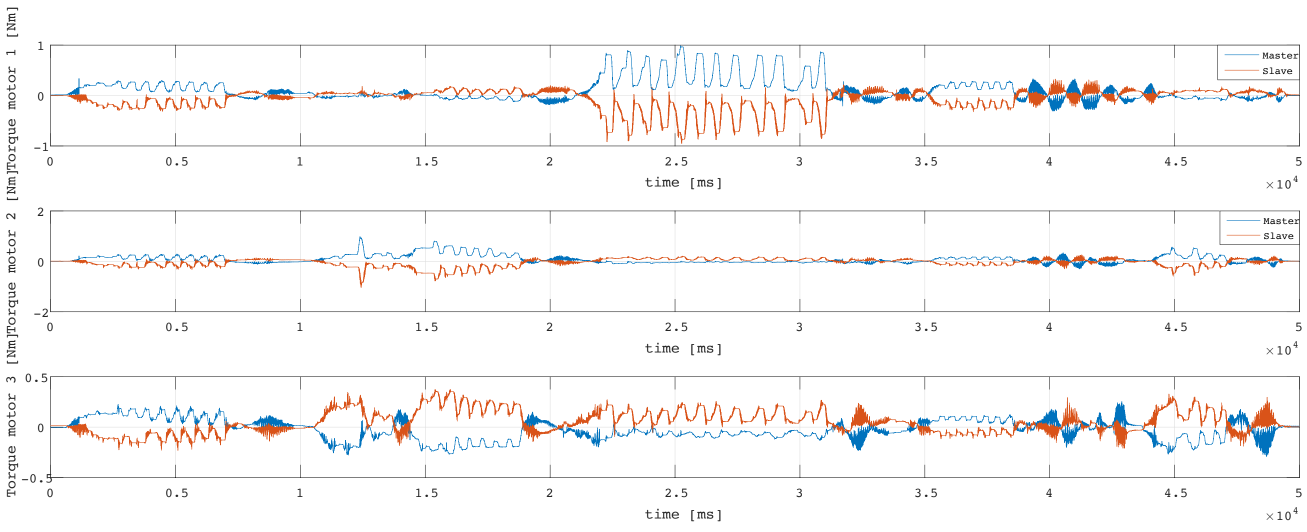


(b) *Range of 10 seconds.*

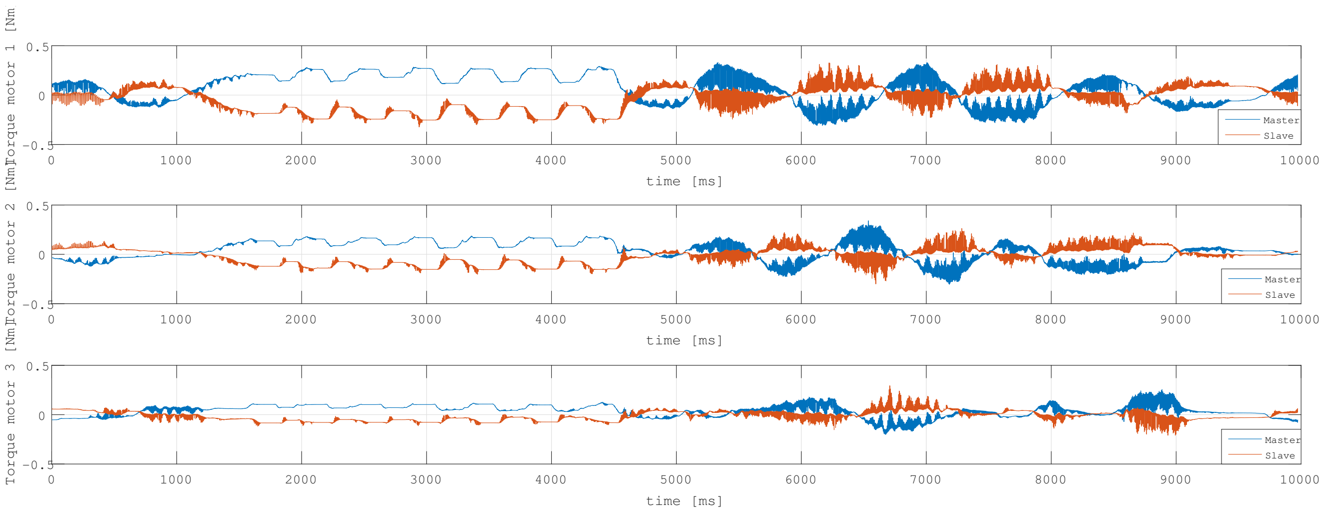


(c) *Range of 5 seconds.*

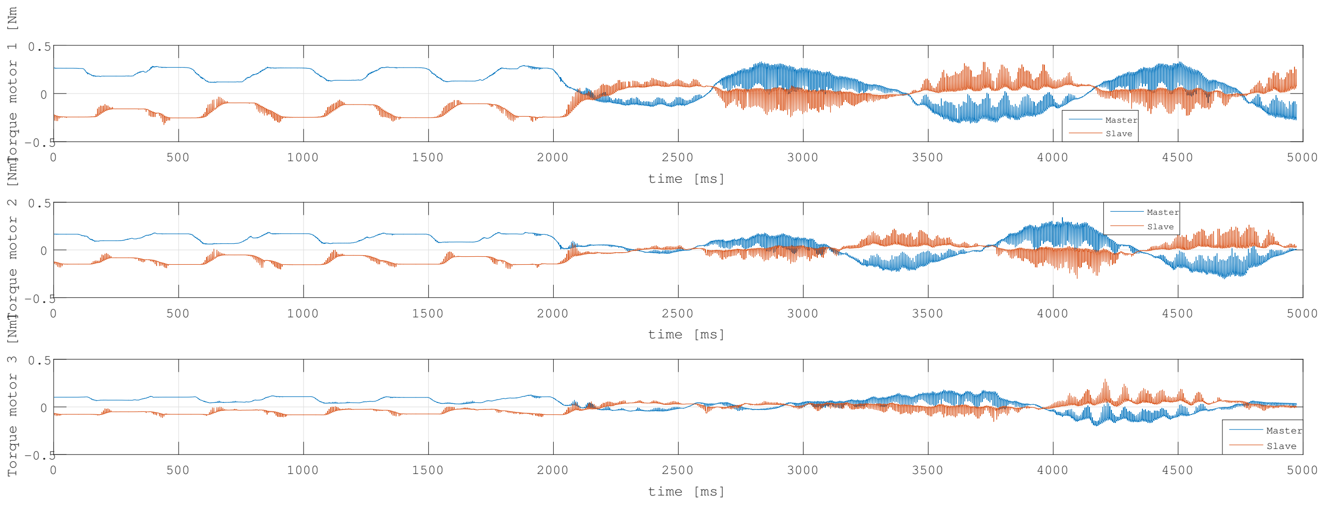
Figure 9.7: Position response in presence of obstacles.



(a) *Range of 50 seconds.*



(b) *Range of 10 seconds.*



(c) *Range of 5 seconds.*

Figure 9.8: Force reflection in presence of obstacles.

9.2 Bilateral Acceleration Control with Disturbance Observer and Kalman Filters.

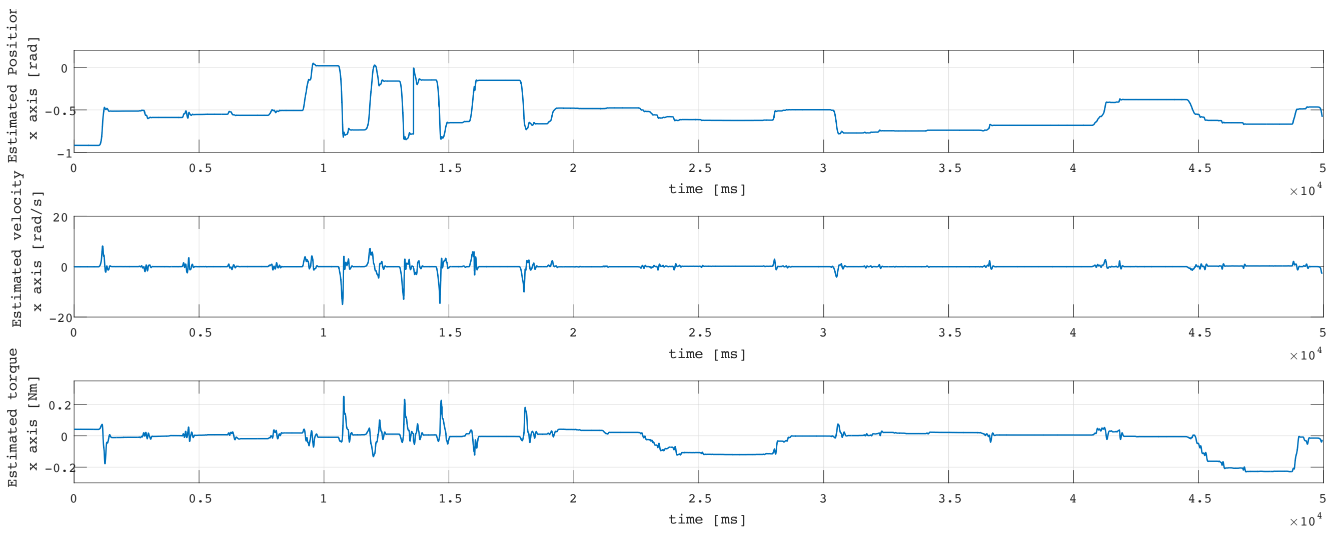
In this section of experimental result the bilateral acceleration control scheme has been analyzed. As explained in section 8.3 this architecture needs the presence of two disturbance observers (one for the master and one for the slave) and 6 Kalman filters (one for each cartesian axis of each robot). The performance of the overall structure is deeply related to the one of these components. For this reason in this thesis these performance have been studied at first, and only once these are validated it is possible to proceed with the analysis of the acceleration bilateral control.

9.2.1 Validation of Kalman Filters and DOB disturbance observers.

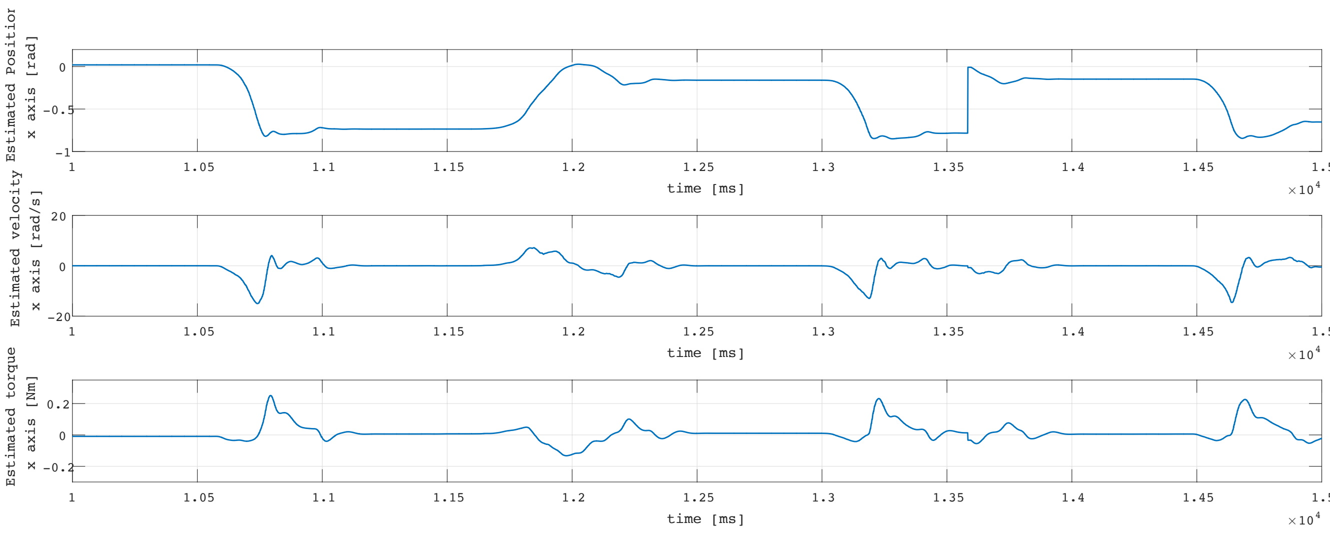
In figure 9.9, 9.10, 9.11, 9.12, 9.13 and 9.14 the estimated output of the kalman filter related respectively to x,y,z axes have been plotted for master and slave in presence and absence of obstacle. In free motion the force peak correspond to velocity peak and changes on the position. On the contrary, on presence of obstacle even if the velocity is negligible it is possible to see some force peak. These behavior can be seen obviously in both robots.
In figure 9.15, 9.16, 9.17 and 9.18 the position's output of the estimator and the encoder's measurements have been compared. It is possible to see that the estimate tracks exactly the encoder for each axes, for both robots and in free motion or in presence of obstacle. The error will be analyzed in the following figures but it will be showed that it is negligible and can be also traced back to the encoder's resolution. For this reason it is possible to suppose that also the velocity and force estimate are correct. The error for these position derived measurements should be bigger but still small.
In figure 9.19 and 9.20 the position error between the estimated position and the encoder measurements for each cartesian axes have been plotted for master and slave and in presence or absence of obstacle. As can be seen the error is one order of magnitude smaller than the position range. This is an indication of the proper functioning of the Kalman filters and it doesn't depend on the presence or not of obstacle and whether the master or the slave is analyzed. This error increase when the operator change the position sharply with high velocity; on the contrary in the case of smooth trajectory the error is such small that may be due to the encoder's resolution.
Finally in figure 9.21, 9.22 and 9.23 the DOB output compared with the estimated velocity for each cartesian axes have been plotted for master and slave. From this plot it can be seen that the estimated velocity and the estimated DOB's output are one the scaled opposite of the other. This comes from equation 9.1 and it is due to the contribution of the Internal back electromotive force which takes position against motion (and it exists only when the robots are moving). It is important to note that when the slave reaches an obstacle, the disturbance estimated torque turns to zero given that the slave and master are not moving anymore.

$$v = v_m + iR = \omega k_t + \frac{\tau}{k_t} R, \tag{9.1}$$

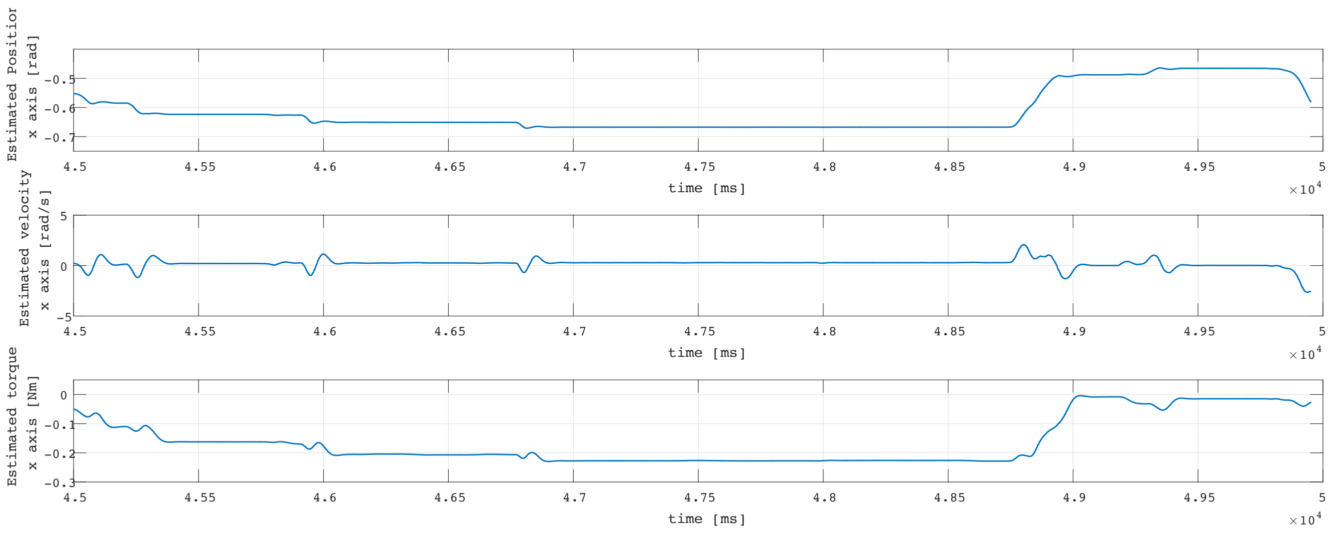
Since in this application the masses which play a role are quite small, the estimated disturbance torque is pretty small and negligible (the order of magnitude is 10^{-4} Nm). But in the case of a bigger manipulator these parameter would play an important role and for this reason the DOB is added to the control scheme of this work.
The last figure 9.24 was added just to show that the reliability of the filter doesn't depend on the ground gain chosen in the acceleration control. Once the filter and DOB are validated in the following section the goodness of the control scheme is analyzed by means of experimental results.



(a) *Range of 50 seconds.*

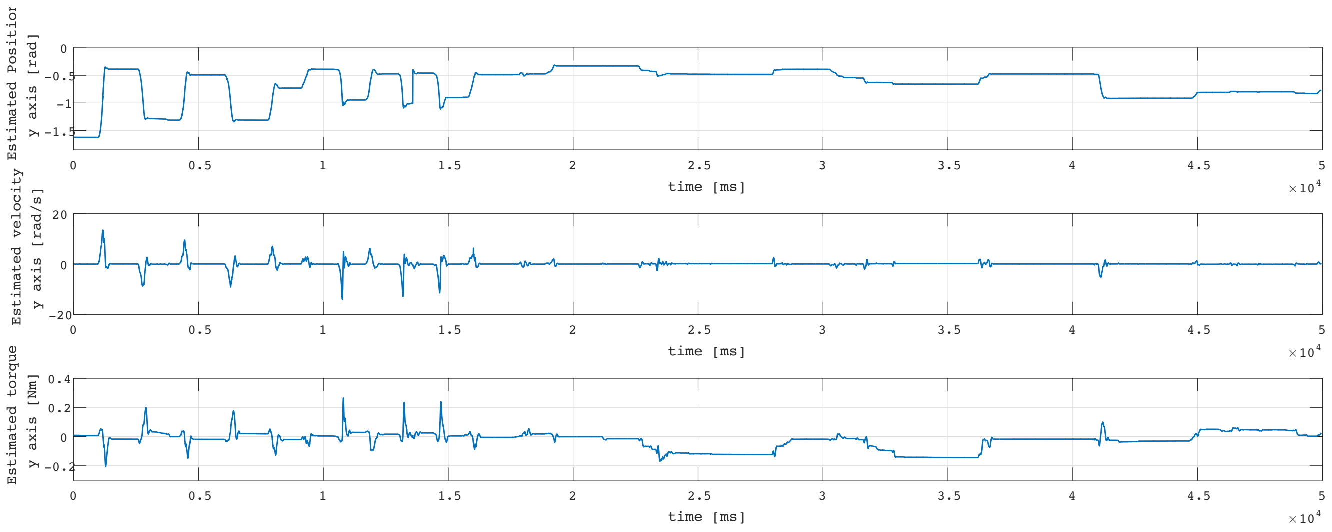


(b) *Range of 5 seconds in absence of obstacle.*

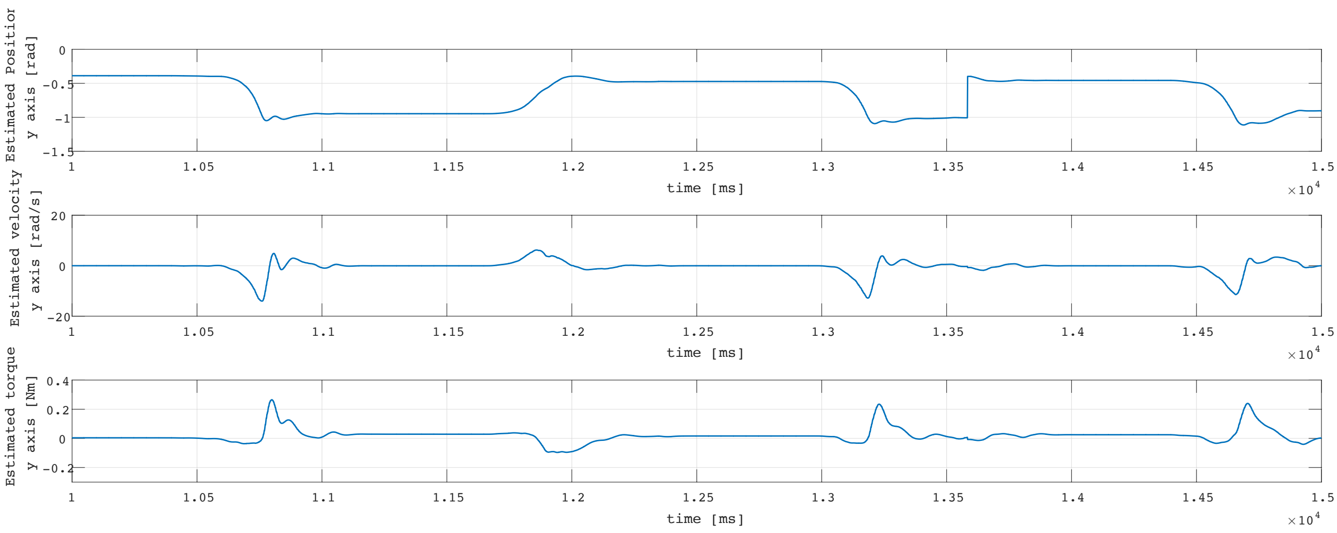


(c) *Range of 5 seconds in presence of obstacle.*

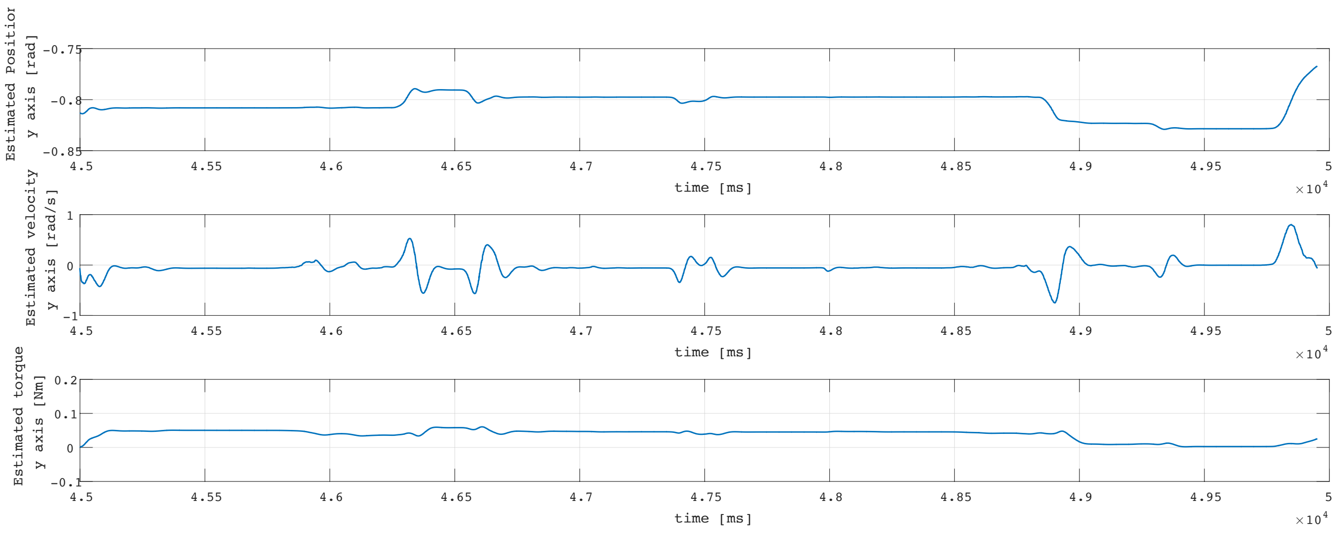
Figure 9.9: Master Estimator's output for x axis in absence and presence of obstacle.



(a) *Range of 50 seconds.*

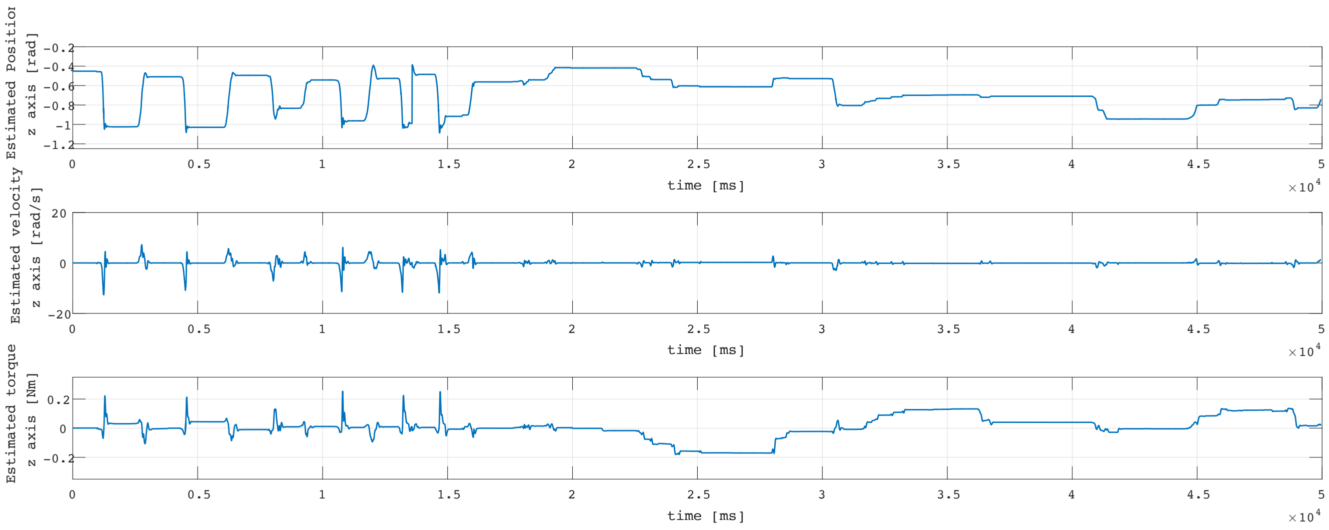


(b) *Range of 5 seconds in absence of obstacle.*

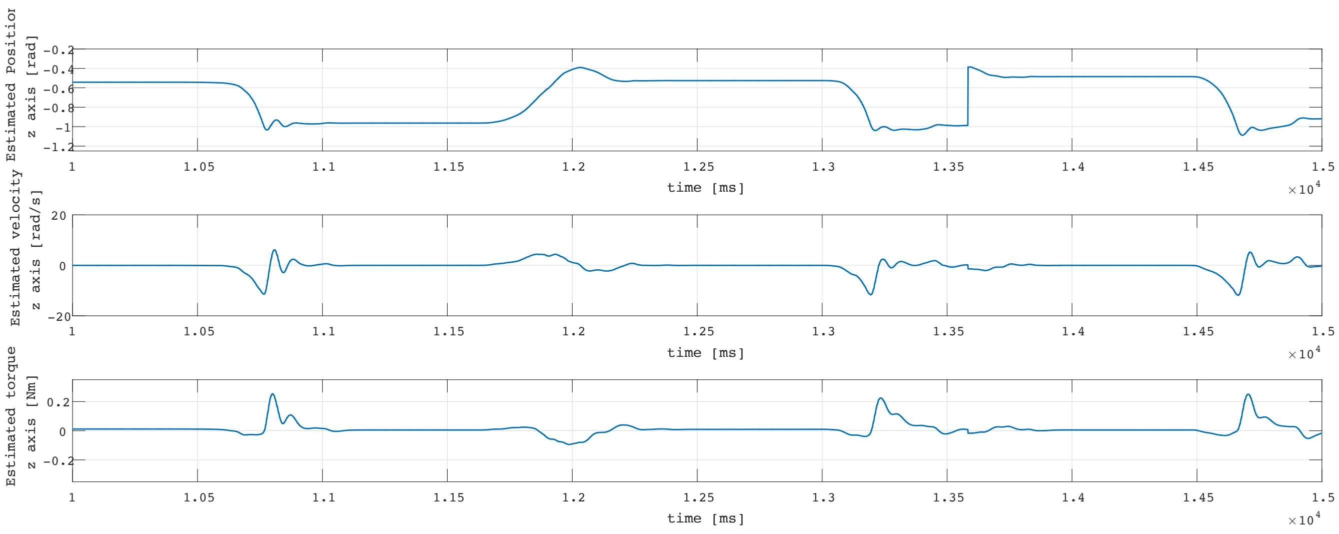


(c) *Range of 5 seconds in presence of obstacle.*

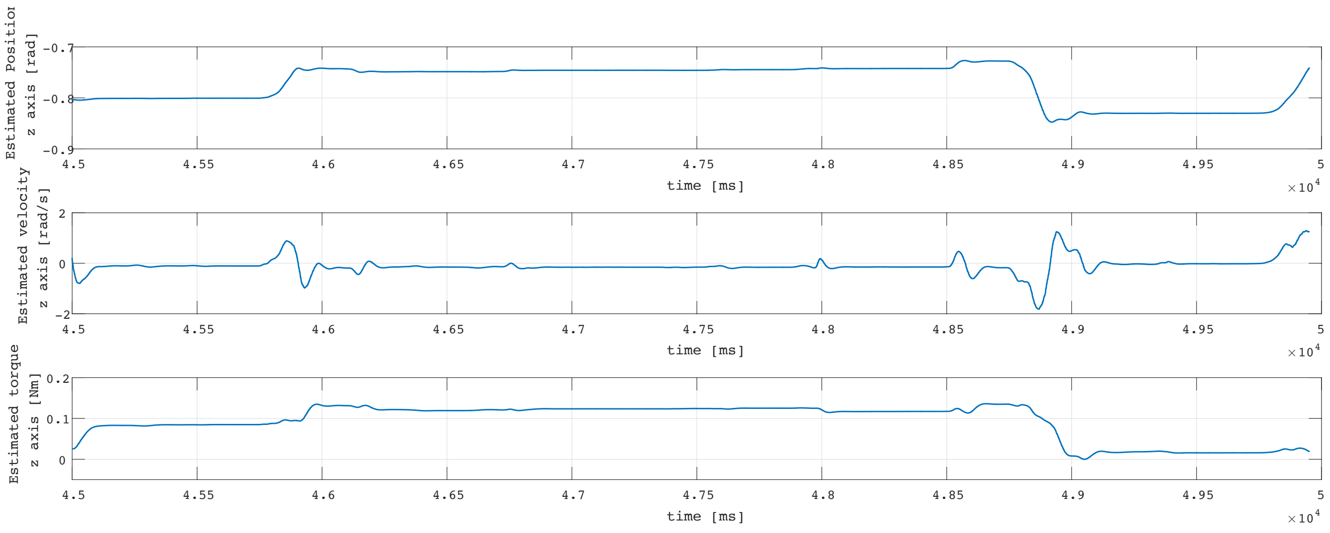
Figure 9.10: Master Estimator's output for y axis in absence and presence of obstacle.



(a) *Range of 50 seconds.*

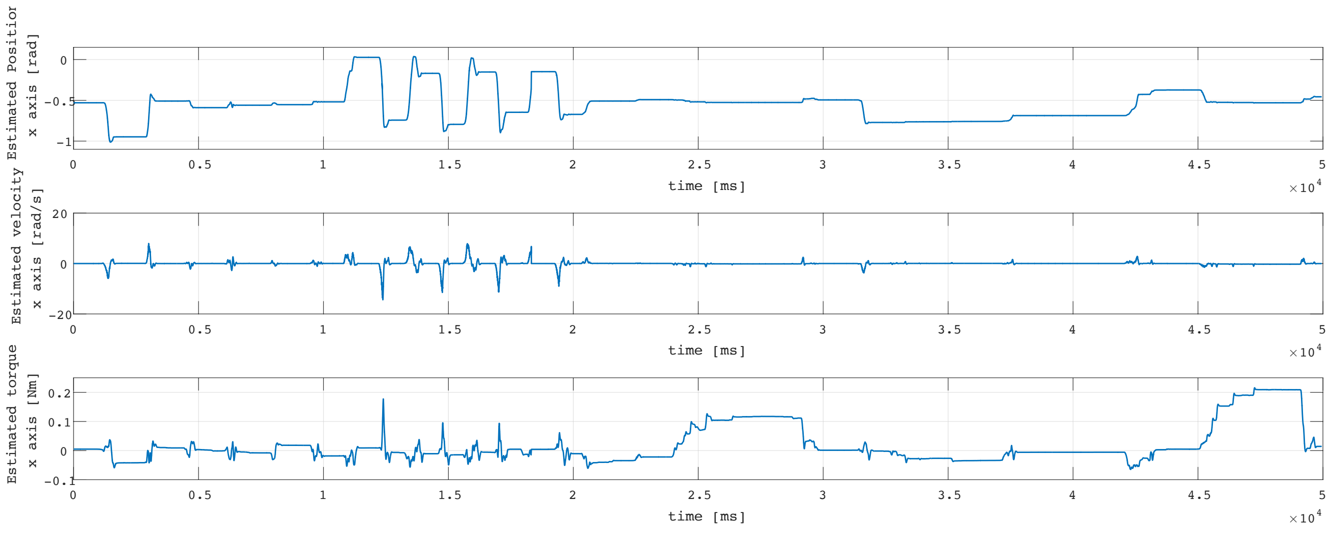


(b) *Range of 5 seconds in absence of obstacle.*

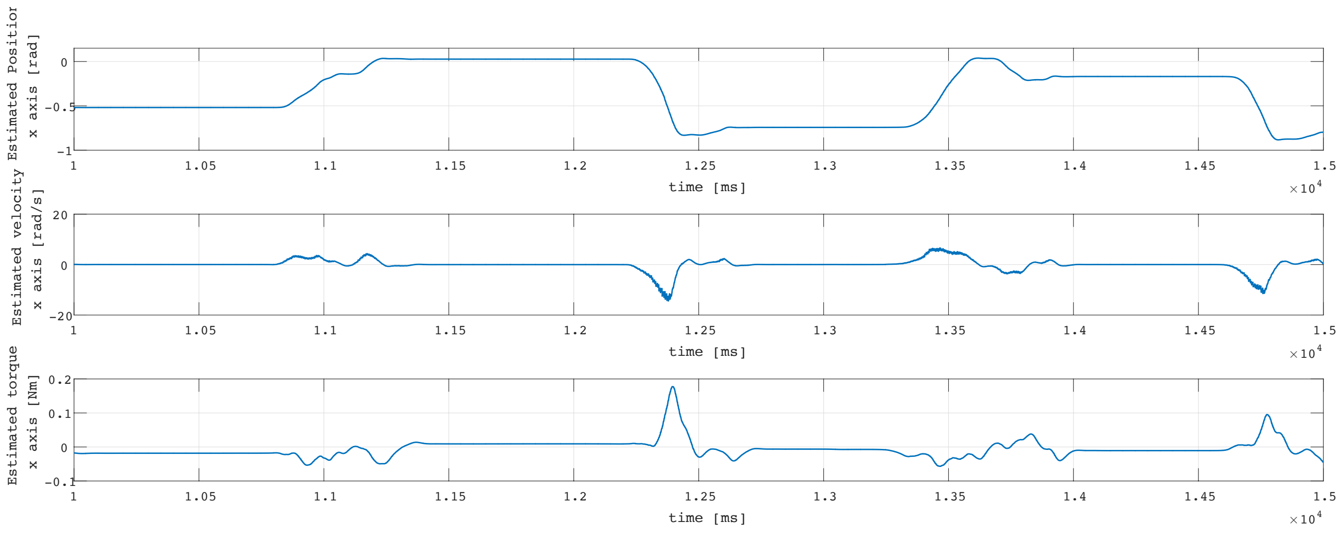


(c) *Range of 5 seconds in presence of obstacle.*

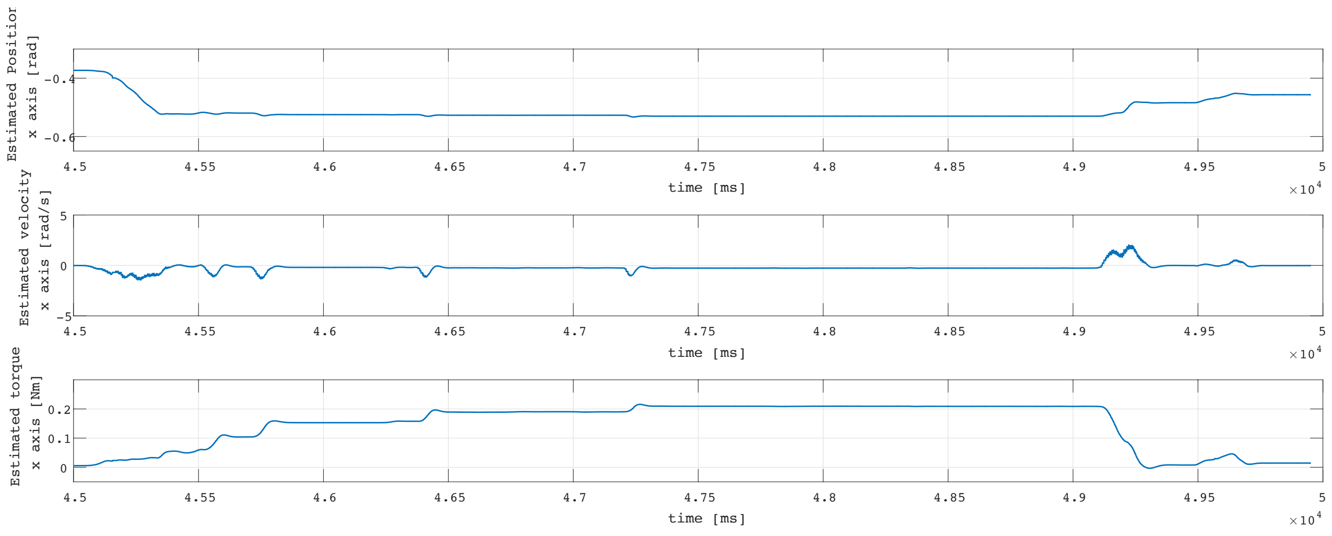
Figure 9.11: Master Estimator's output for z axis in absence and presence of obstacle.



(a) *Range of 50 seconds.*

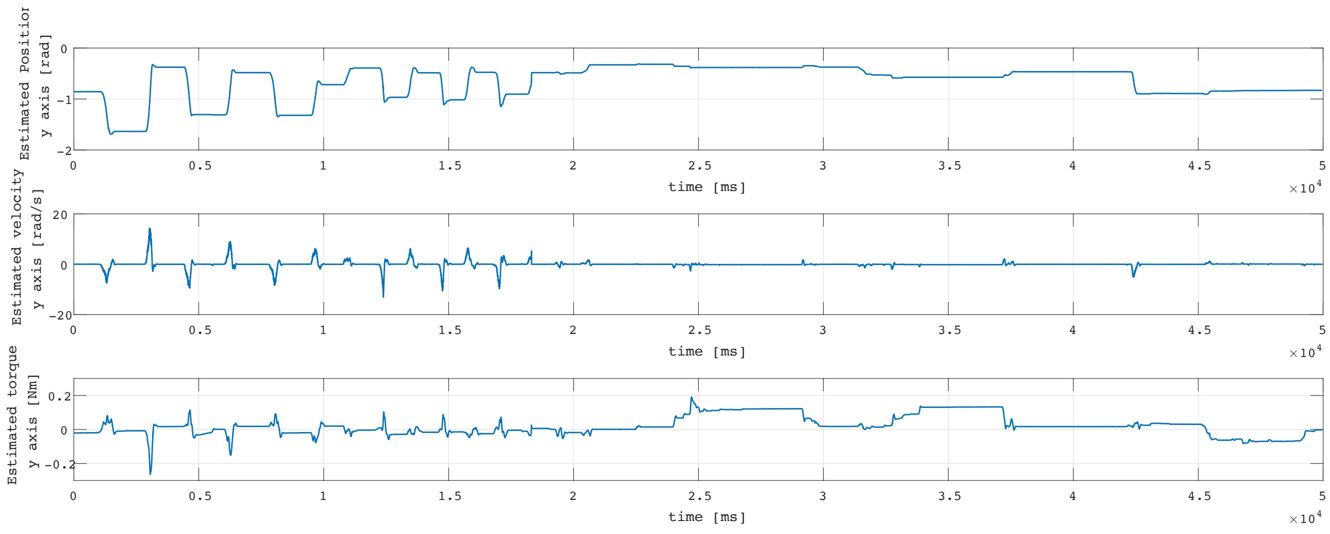


(b) *Range of 5 seconds in absence of obstacle.*

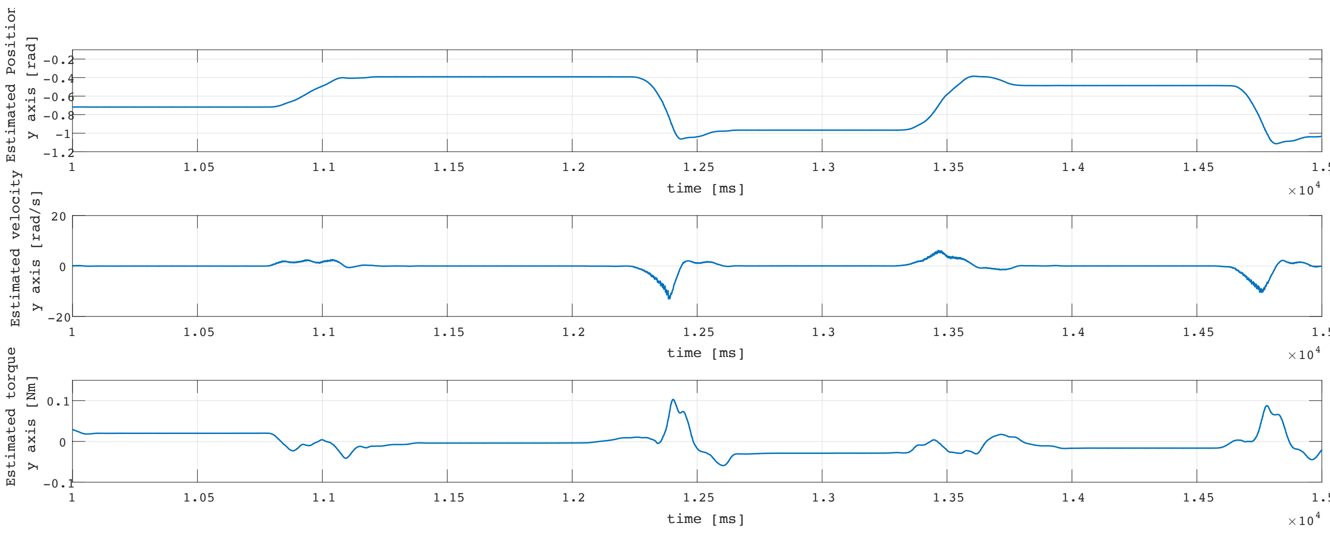


(c) *Range of 5 seconds in presence of obstacle.*

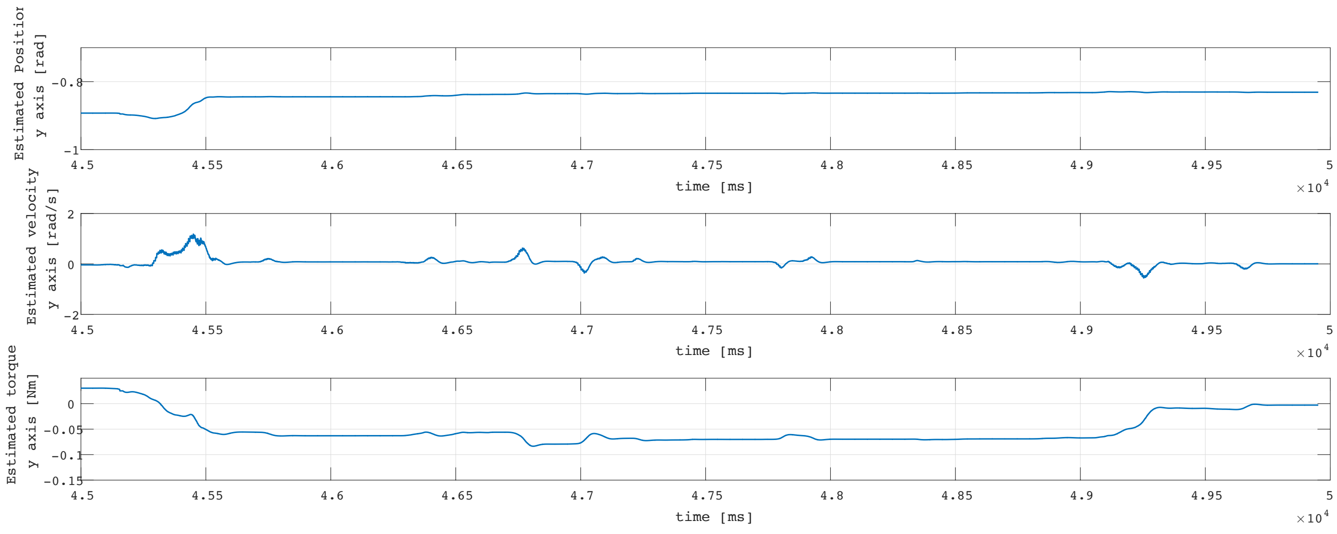
Figure 9.12: Slave Estimator's output for x axis in absence and presence of obstacle.



(a) *Range of 50 seconds.*

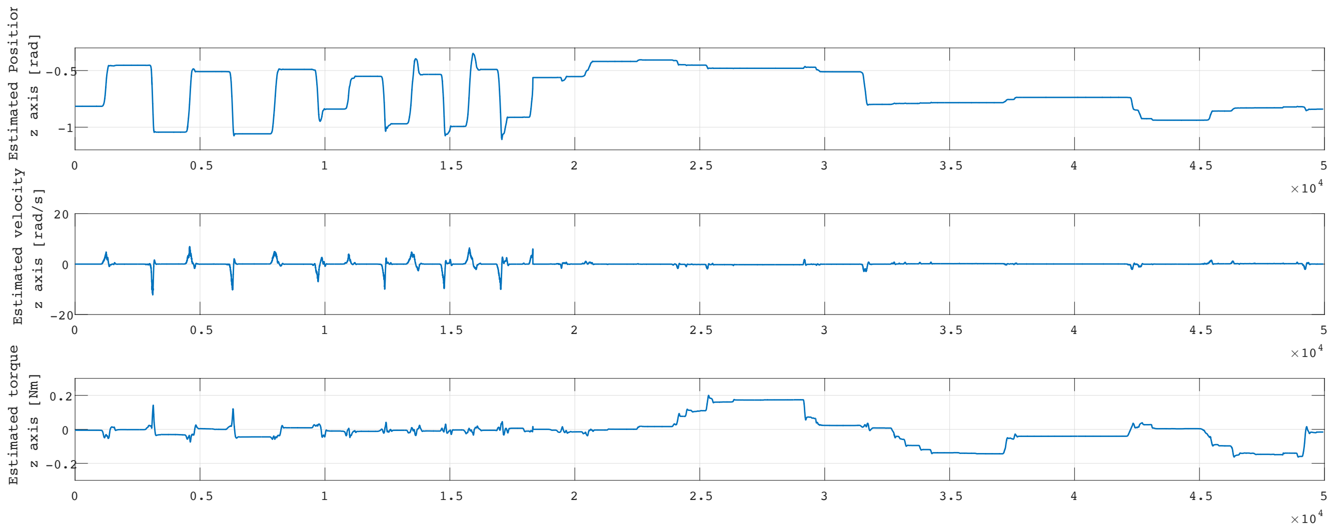


(b) *Range of 5 seconds in absence of obstacle.*

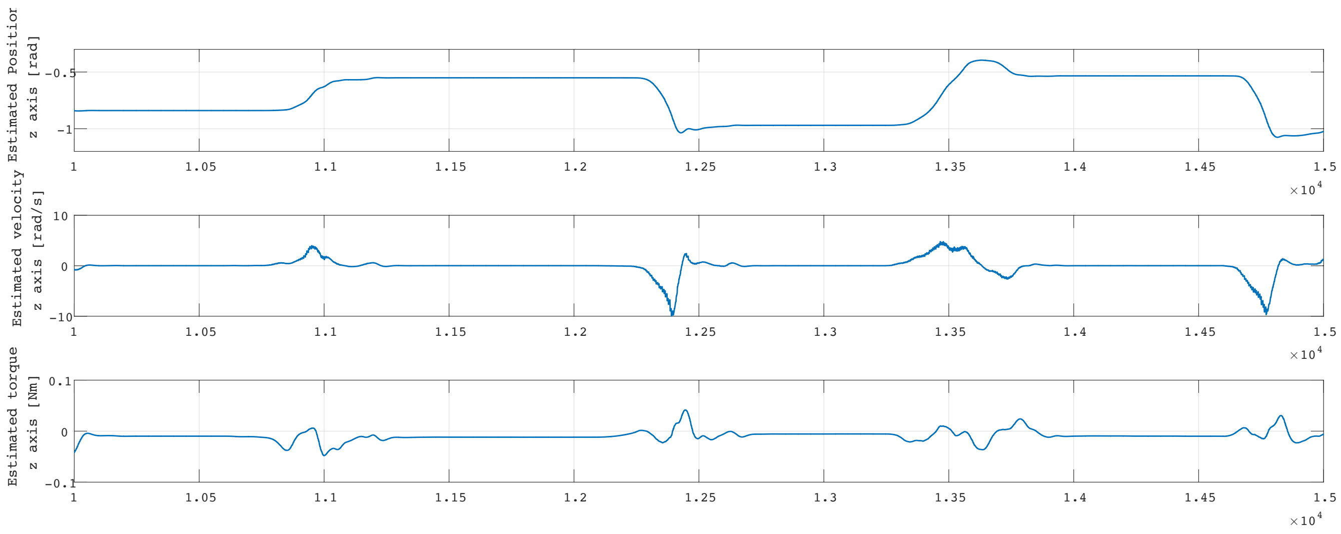


(c) *Range of 5 seconds in presence of obstacle.*

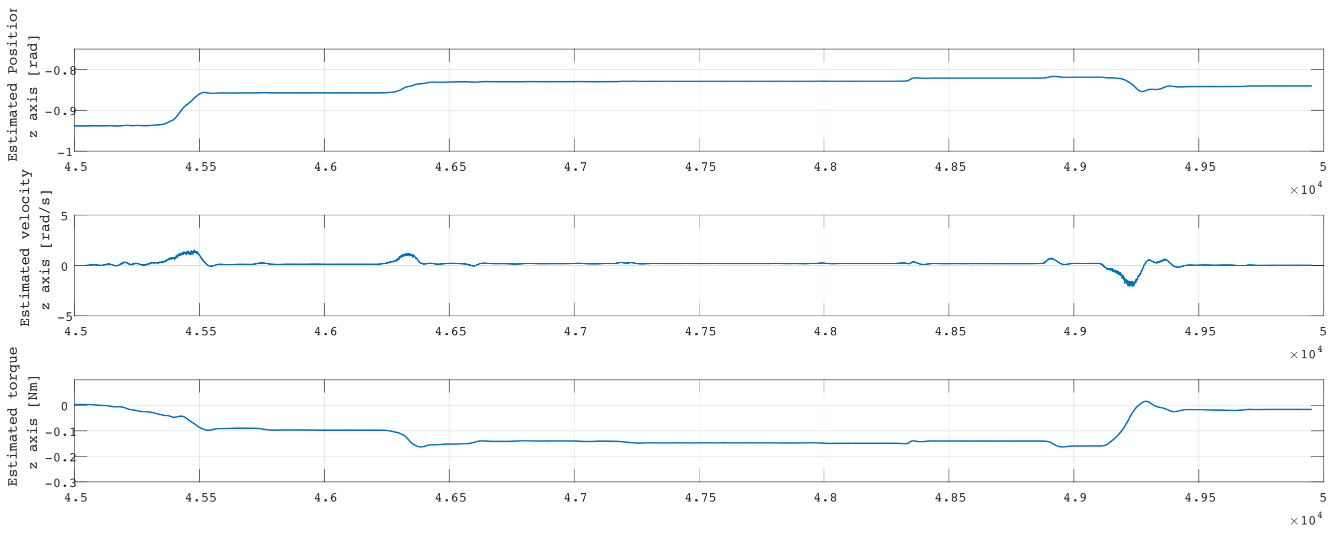
Figure 9.13: Slave Estimator's output for y axis in absence and presence of obstacle.



(a) *Range of 50 seconds.*

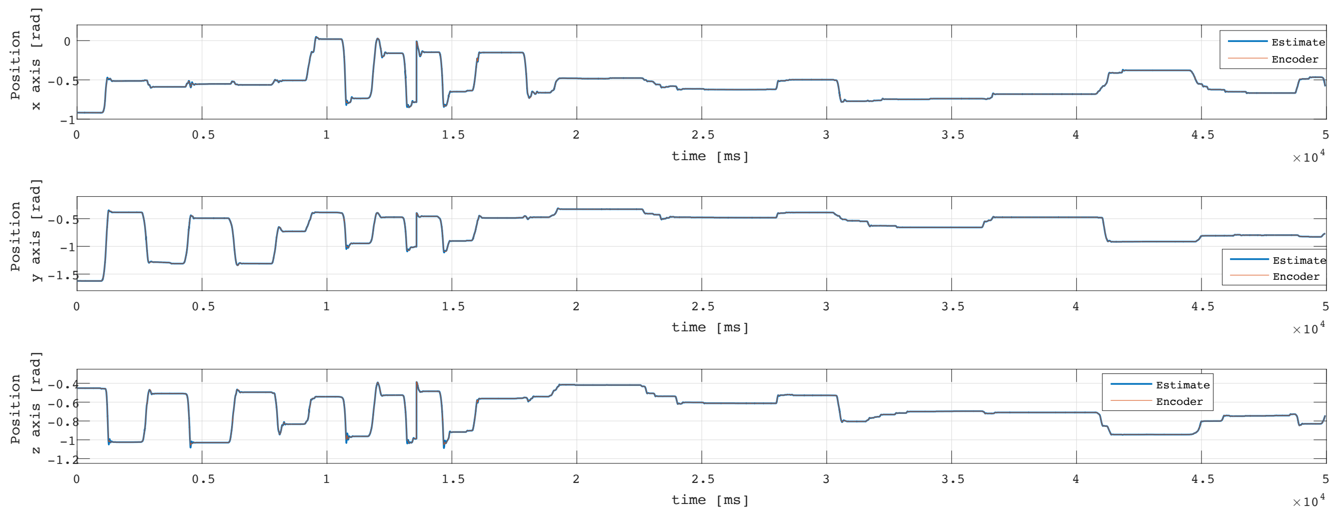


(b) *Range of 5 seconds in absence of obstacle.*

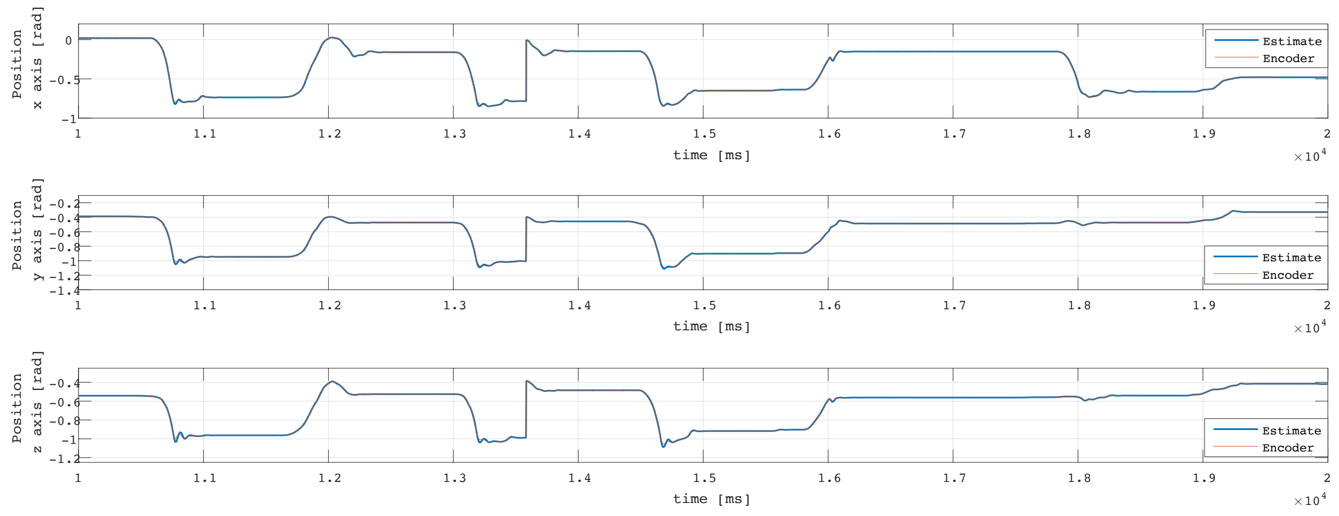


(c) *Range of 5 seconds in presence of obstacle.*

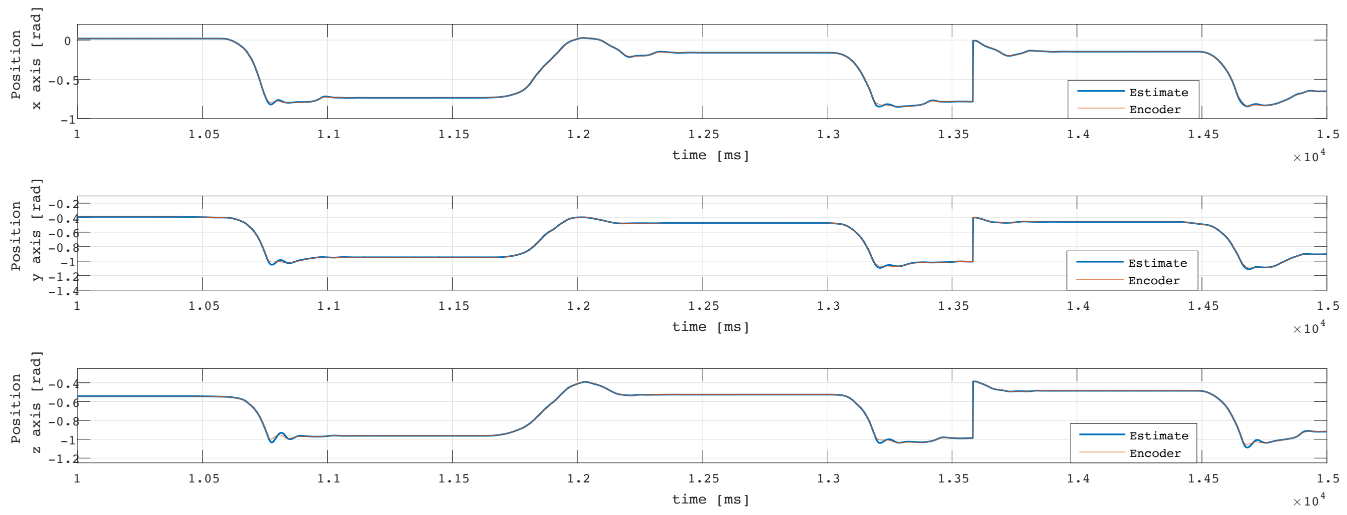
Figure 9.14: Slave Estimator's output for z axis in absence and presence of obstacle.



(a) *Range of 50 seconds.*

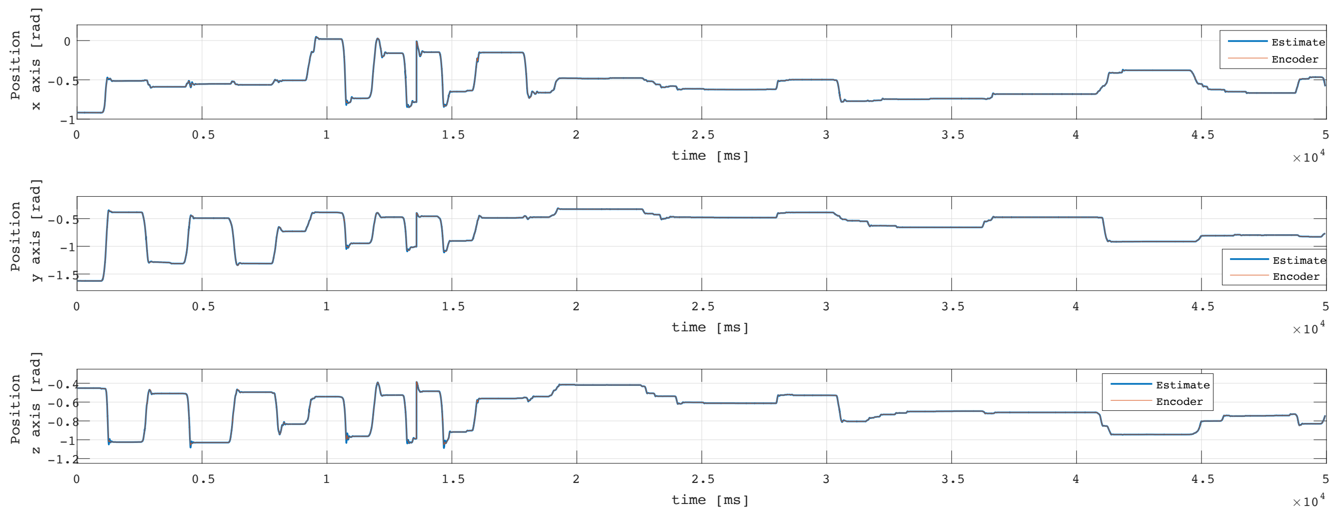


(b) *Range of 10 seconds in absence of obstacle.*

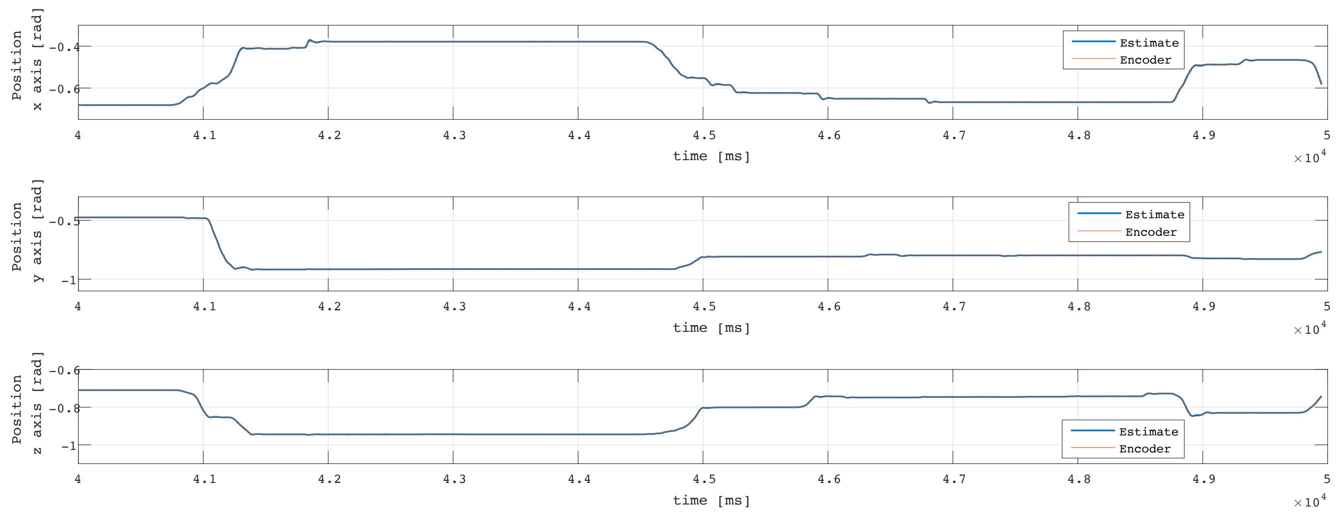


(c) *Range of 5 seconds in absence of obstacle.*

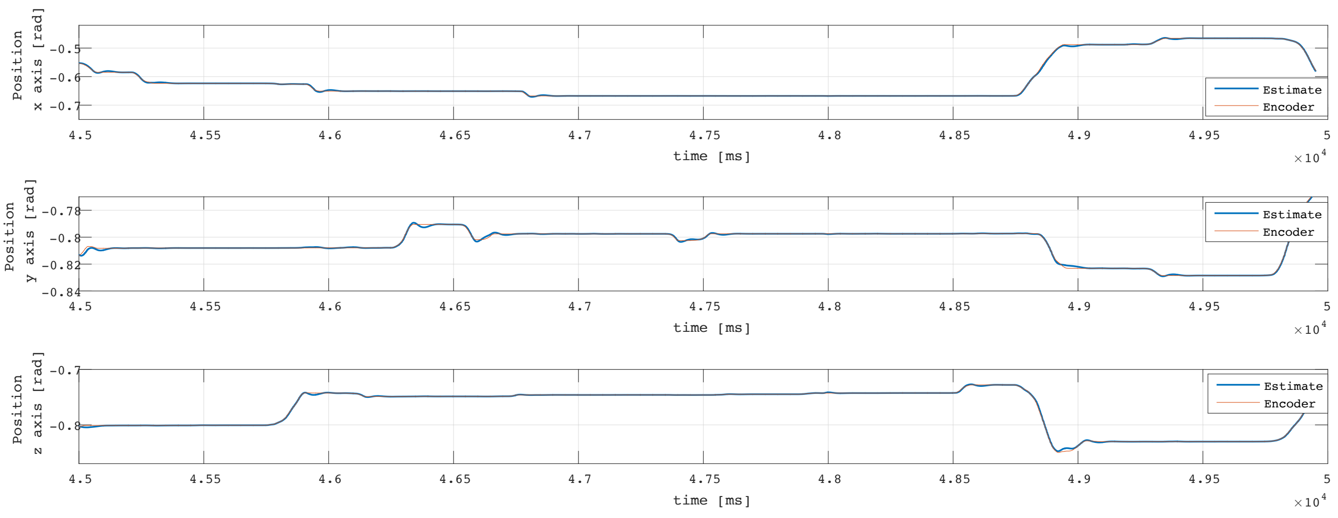
Figure 9.15: Master Estimator's and encoder's outputs in absence of obstacle.



(a) *Range of 50 seconds.*

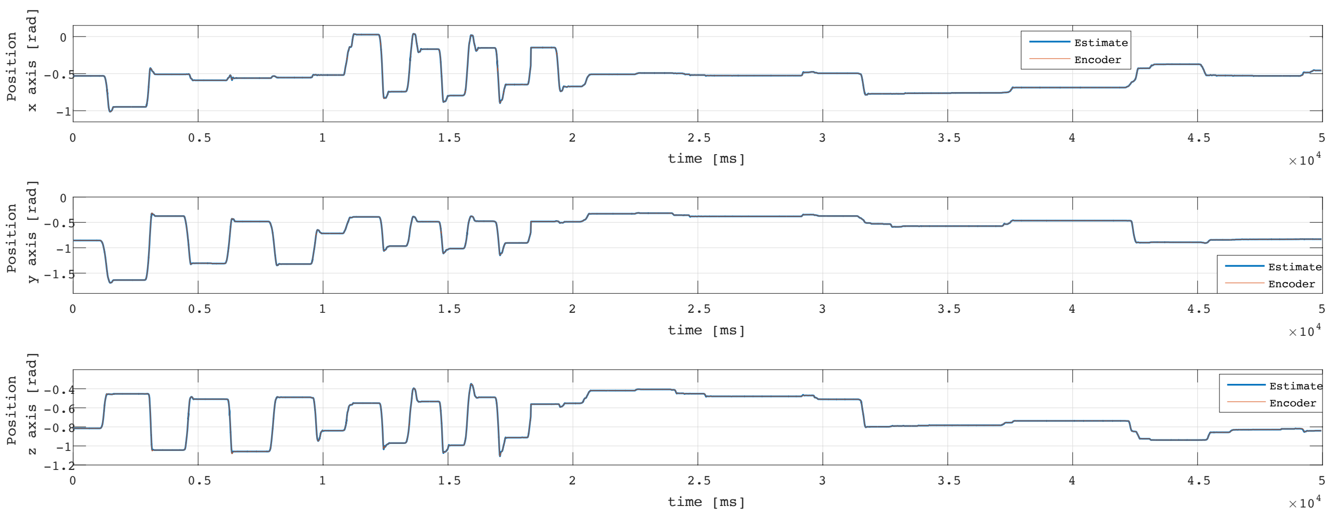


(b) *Range of 10 seconds in presence of obstacle.*

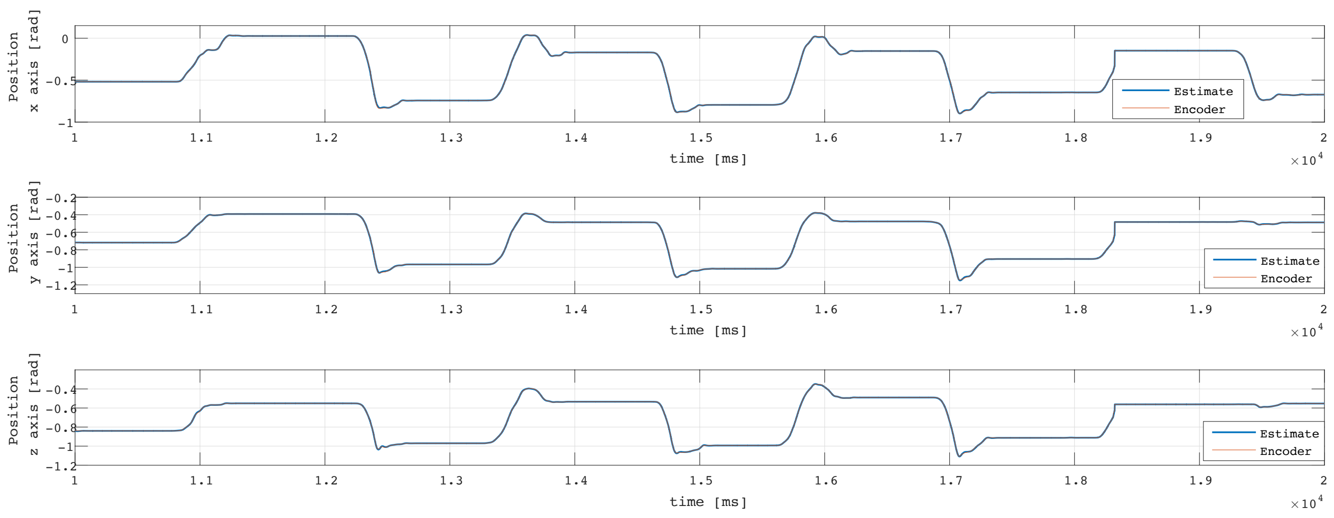


(c) *Range of 5 seconds in presence of obstacle.*

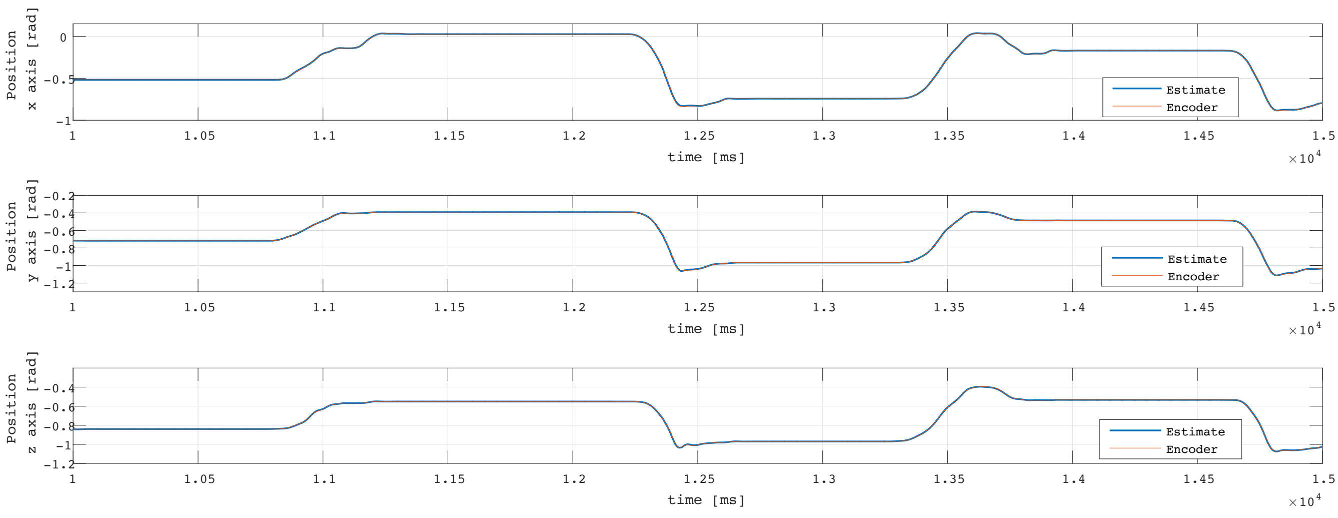
Figure 9.16: Master Estimator's and encoder's outputs in presence of obstacle.



(a) *Range of 50 seconds.*

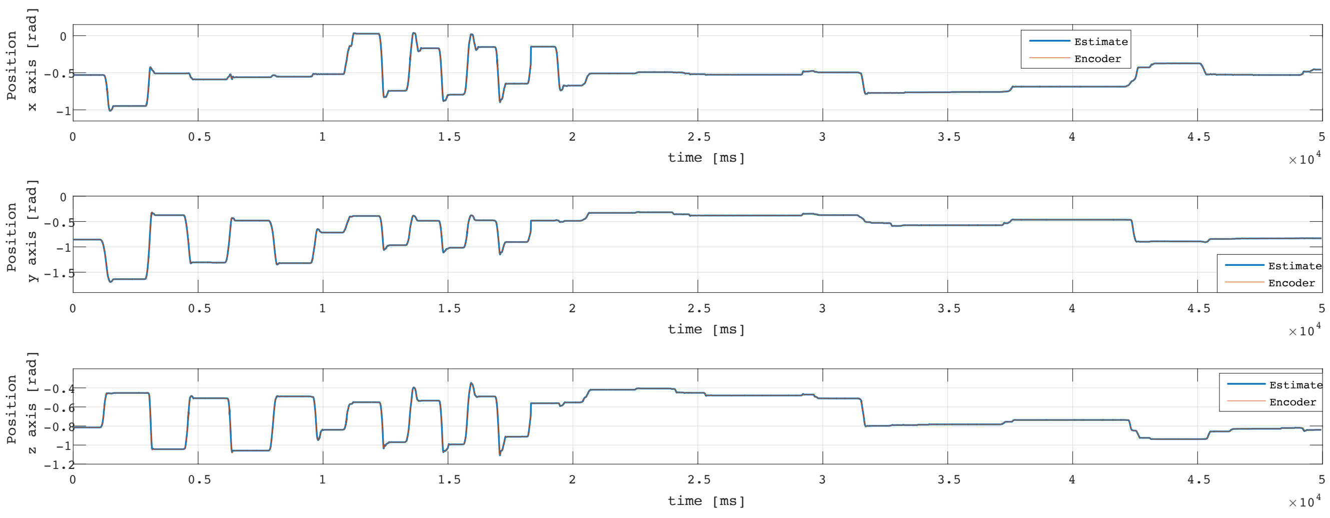


(b) *Range of 10 seconds in absence of obstacle.*

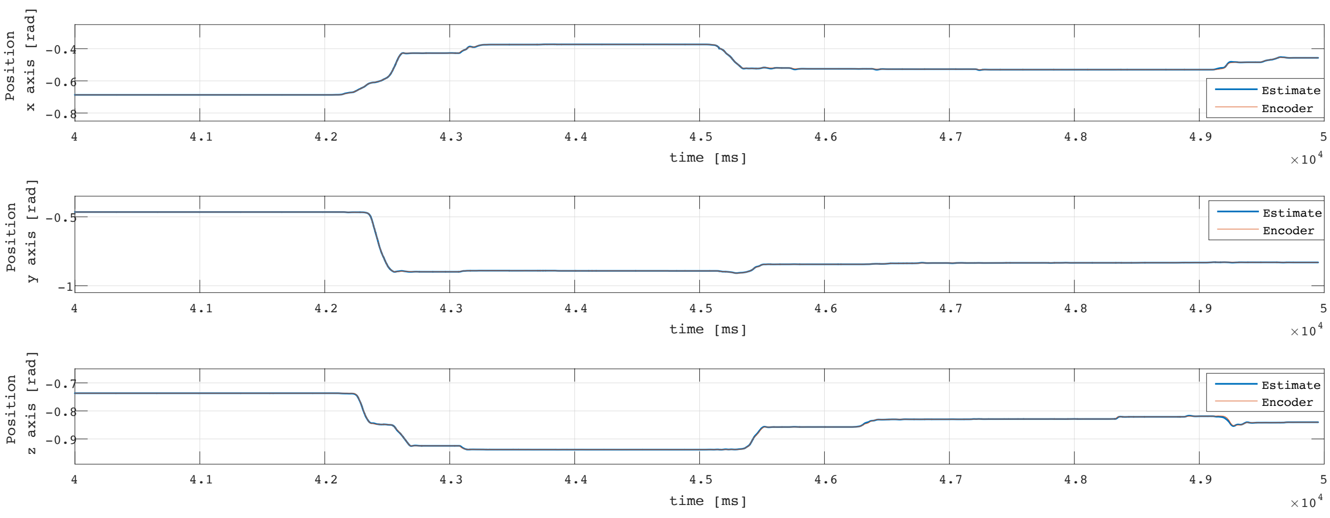


(c) *Range of 5 seconds in absence of obstacle.*

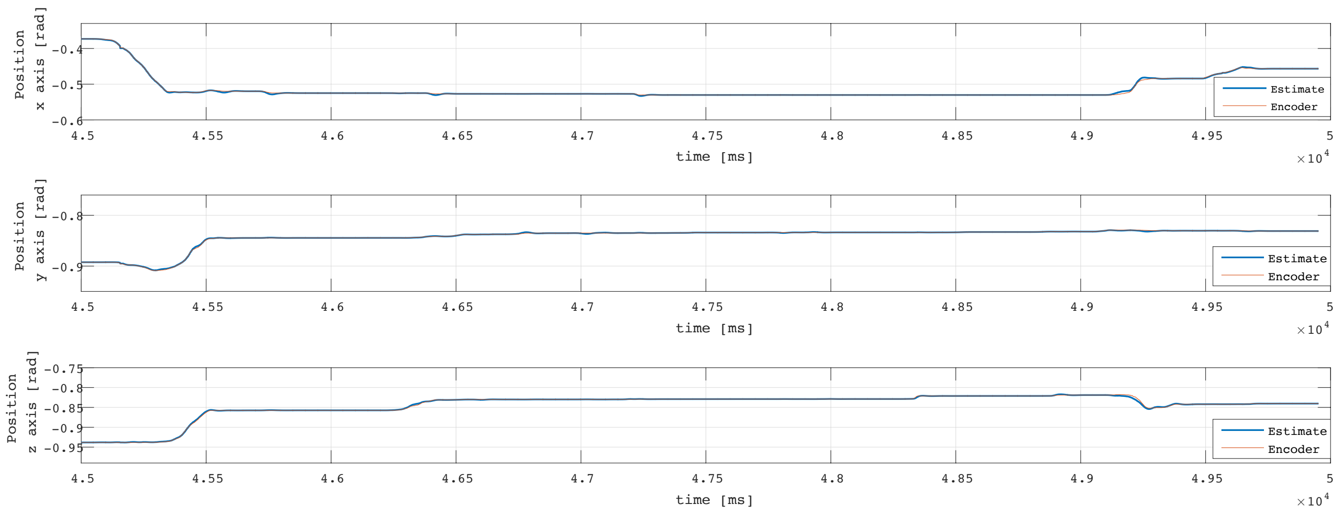
Figure 9.17: Slave Estimator's and encoder's outputs in absence of obstacle.



(a) *Range of 50 seconds.*

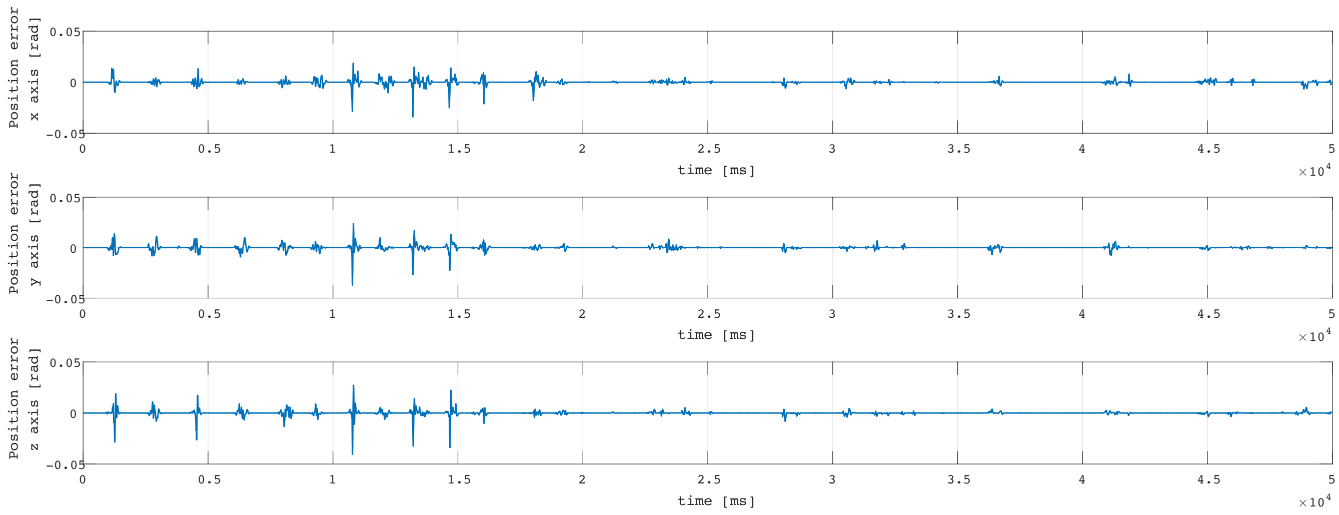


(b) *Range of 5 seconds in presence of obstacle.*

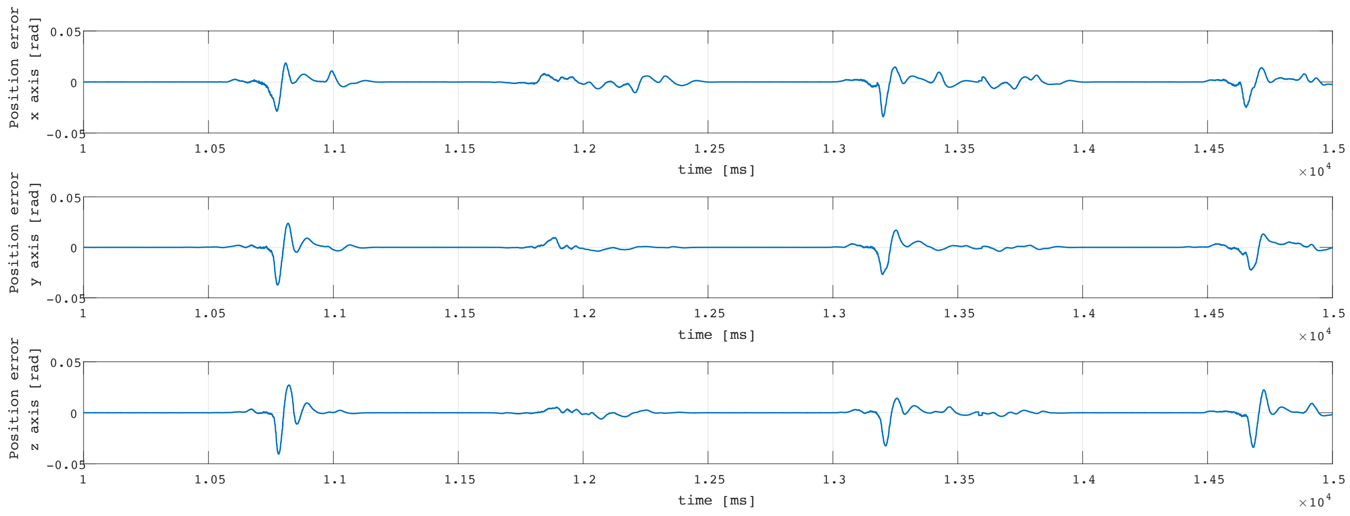


(c) *Range of 5 seconds in presence of obstacle.*

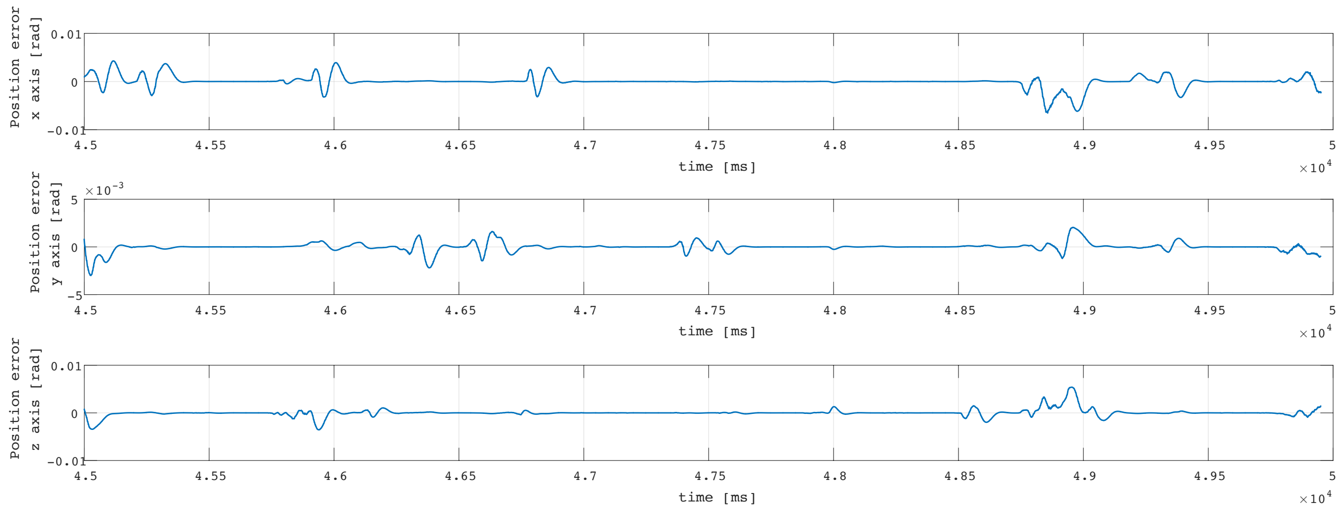
Figure 9.18: Slave Estimator's and encoder's outputs in presence of obstacle.



(a) *Range of 50 seconds.*

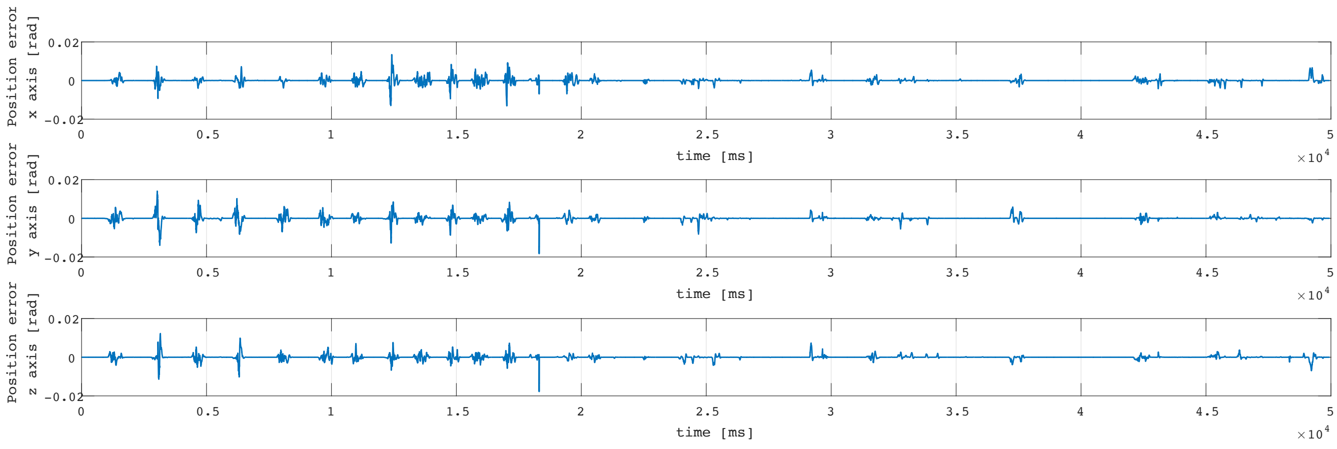


(b) *Range of 5 seconds in absence of obstacle.*

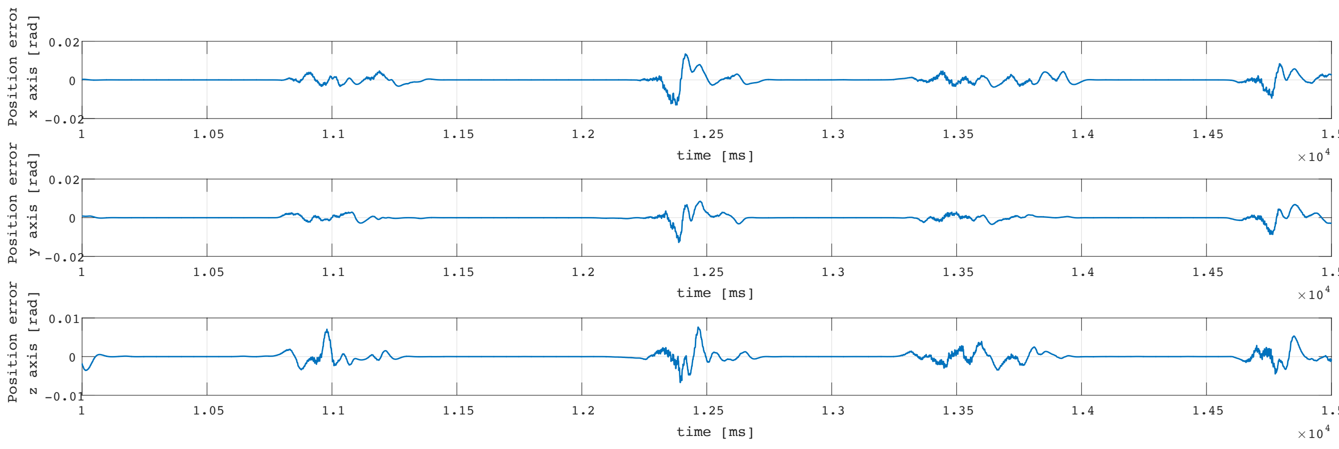


(c) *Range of 5 seconds in presence of obstacle.*

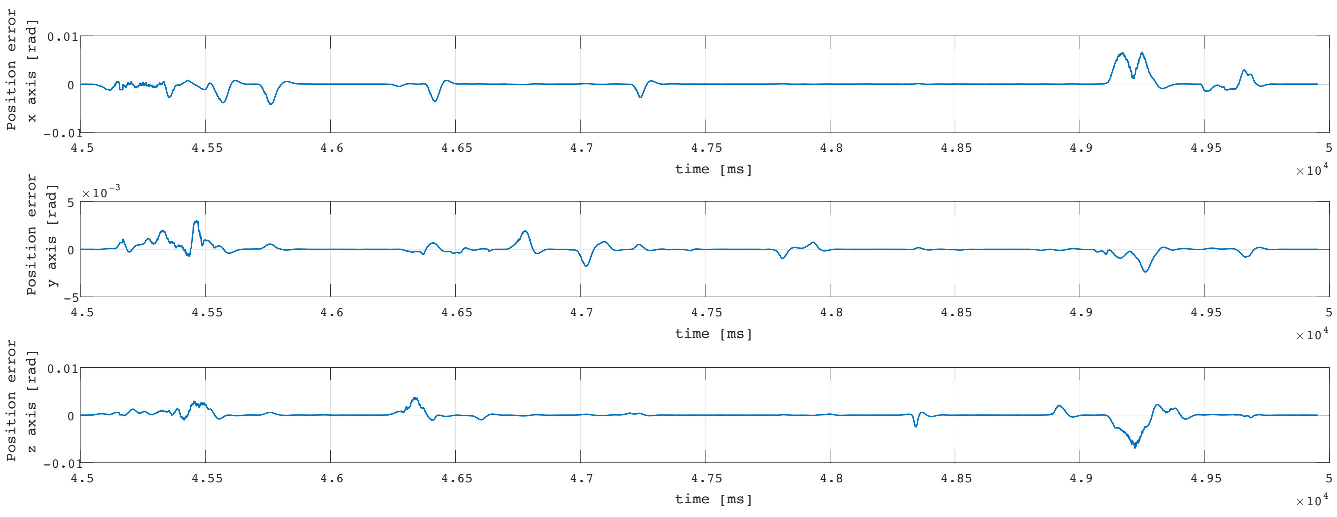
Figure 9.19: Master Error between Estimator's and encoder's outputs in absence and presence of obstacle.



(a) *Range of 50 seconds.*

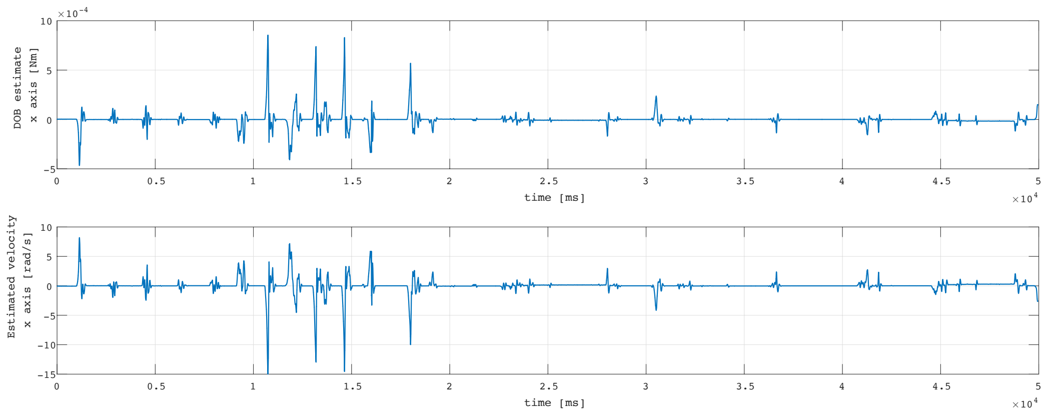


(b) *Range of 5 seconds in absence of obstacle.*

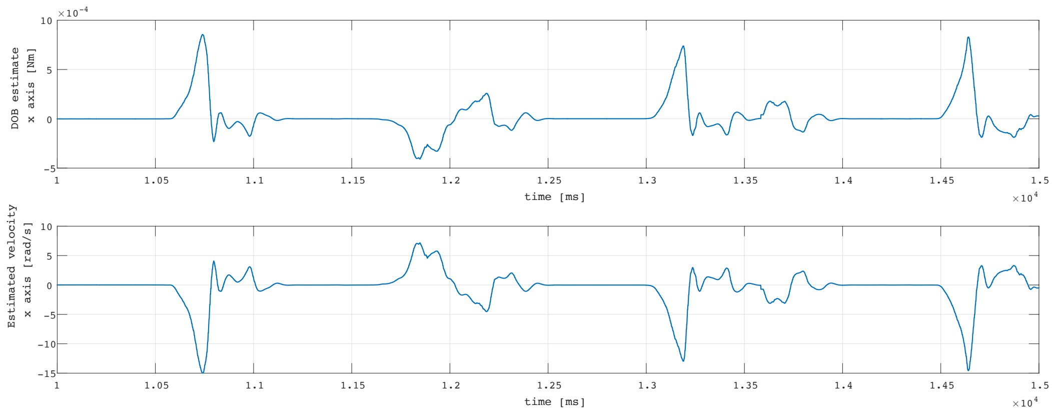


(c) *Range of 5 seconds in presence of obstacle.*

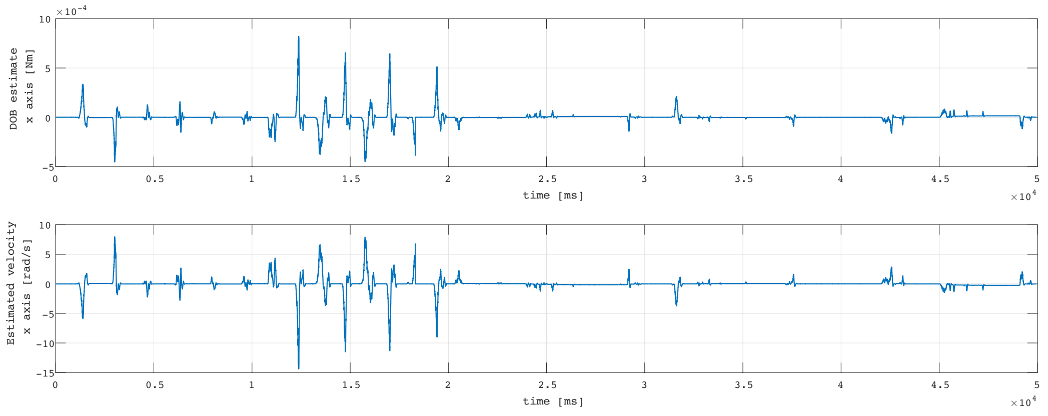
Figure 9.20: Slave Error between Estimator's and encoder's outputs in absence and presence of obstacle.



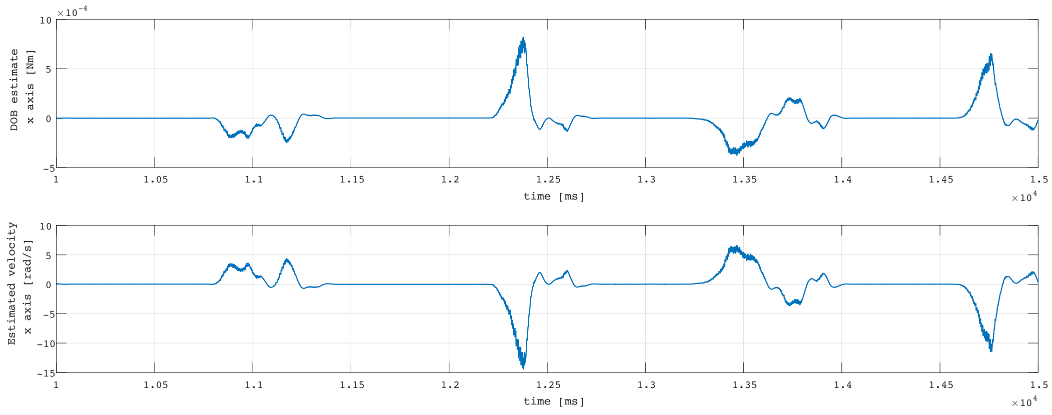
(a) *Master: Range of 50 seconds.*



(b) *Master: Range of 5 seconds.*

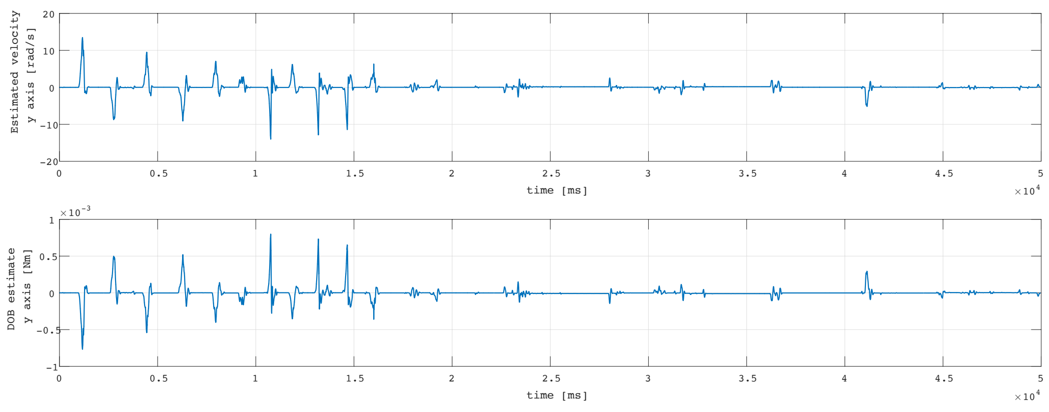


(c) *Slave: Range of 50 seconds.*

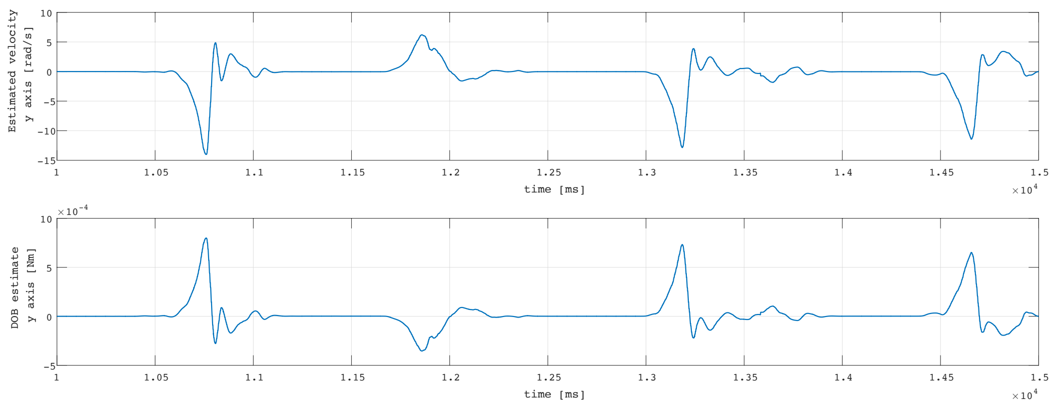


(d) *Slave: Range of 5 seconds.*

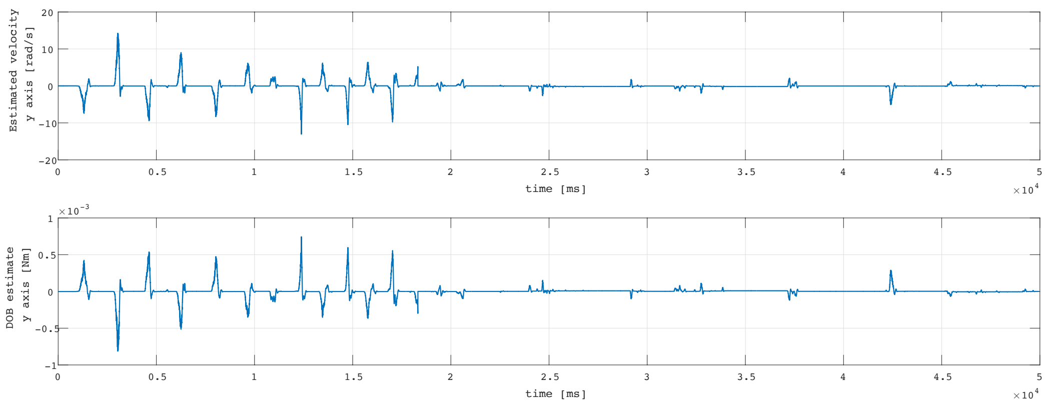
Figure 9.21: DOB output compared with the velocity estimated for x axis.



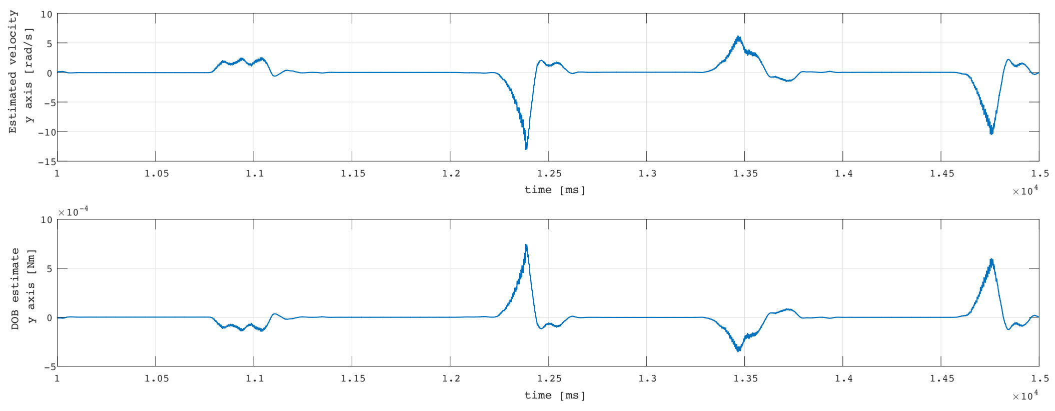
(a) *Master: Range of 50 seconds.*



(b) *Master: Range of 5 seconds.*

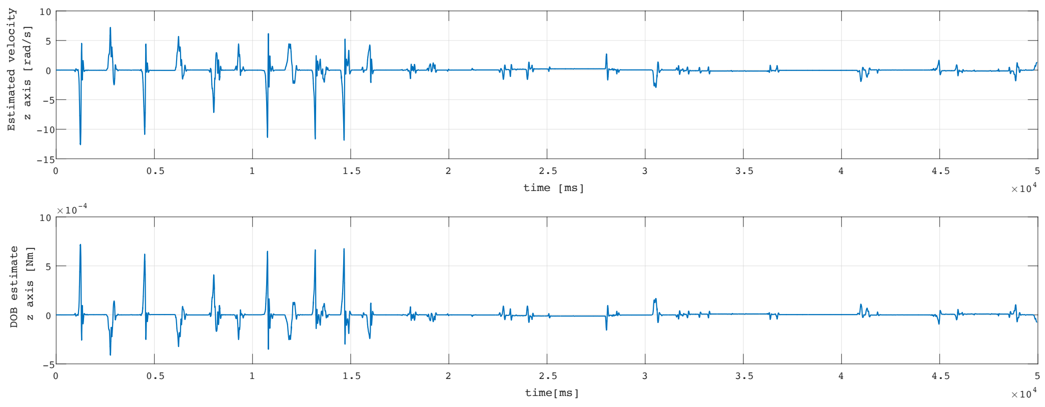


(c) *Slave: Range of 50 seconds.*

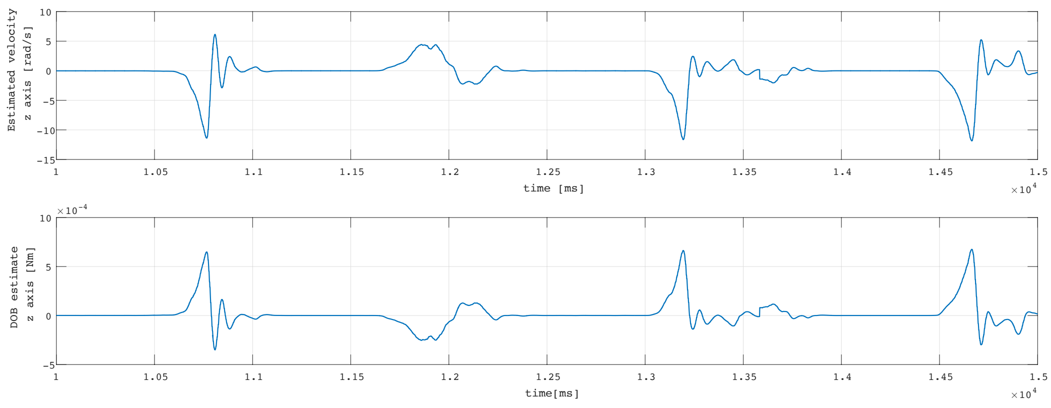


(d) *Slave: Range of 5 seconds.*

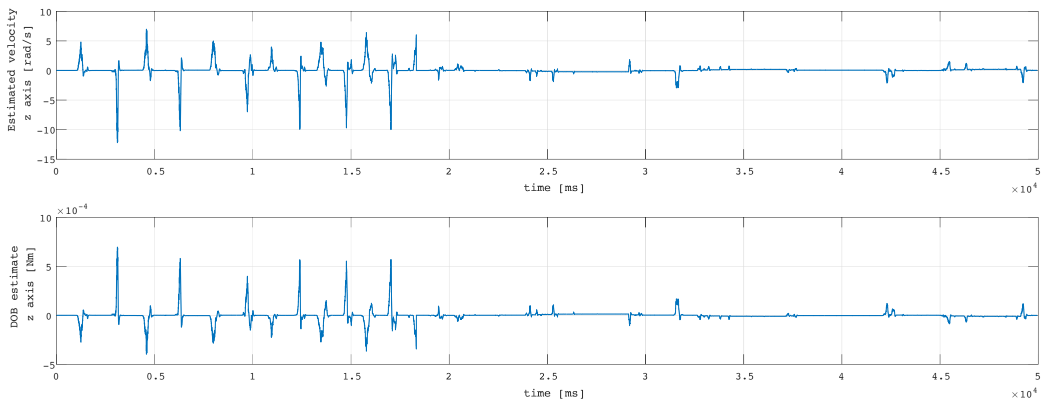
Figure 9.22: DOB output compared with the velocity estimated for y axis.



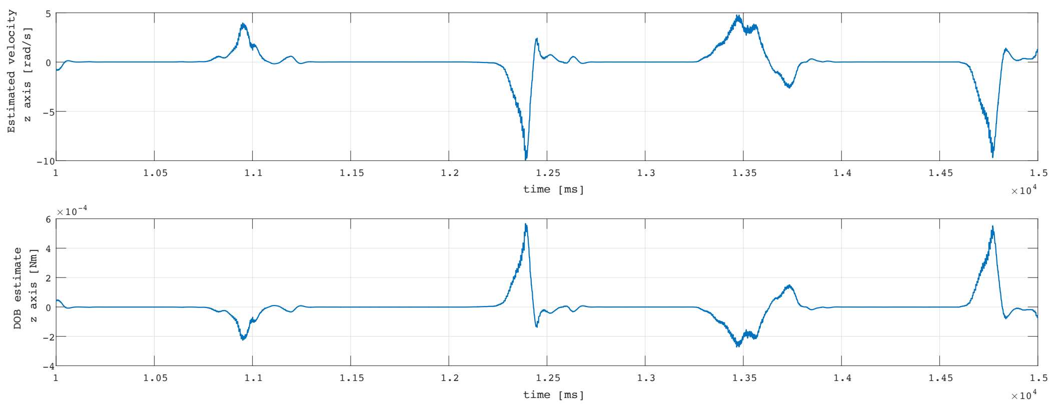
(a) *Master: Range of 50 seconds.*



(b) *Master: Range of 5 seconds.*

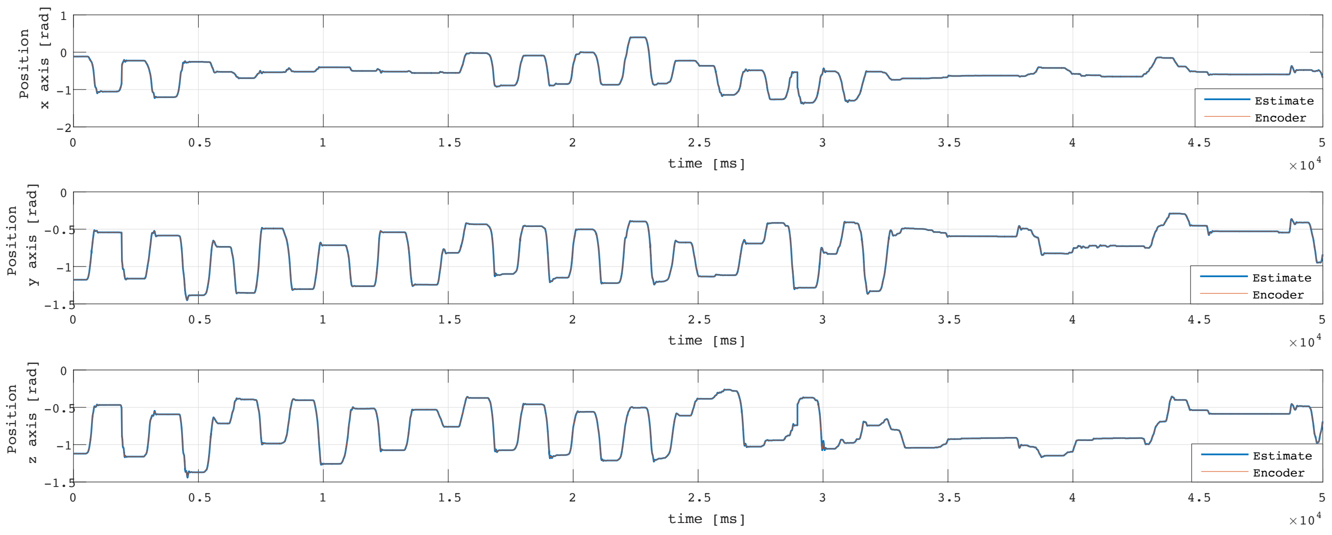


(c) *Slave: Range of 50 seconds.*

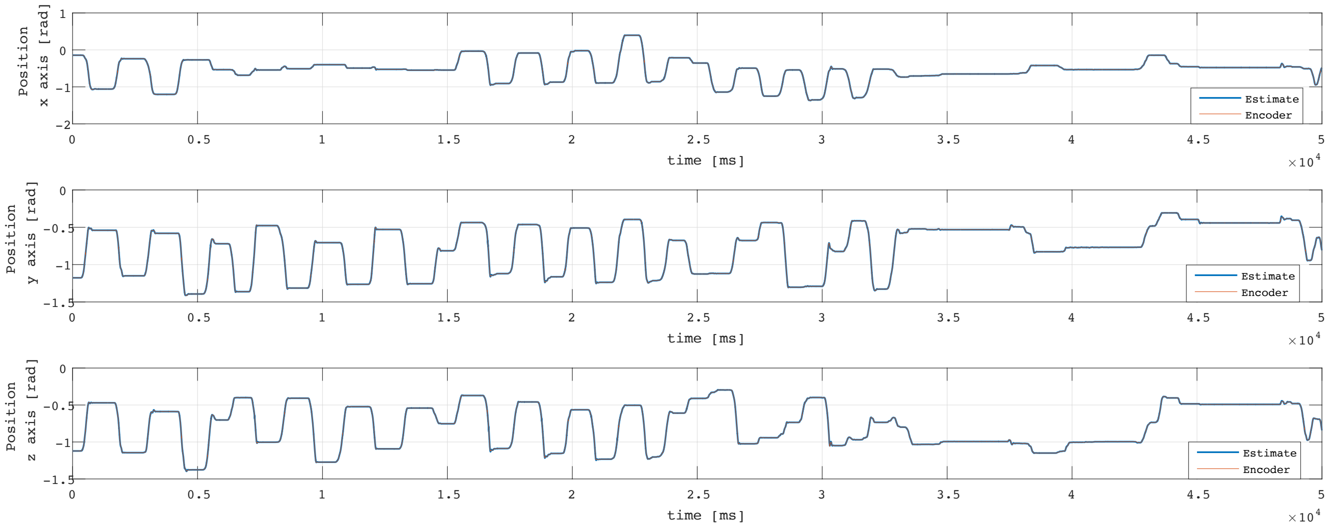


(d) *Slave: Range of 5 seconds.*

Figure 9.23: DOB output compared with the velocity estimated for z axis.



(a) *Master: Range of 50 seconds.*



(b) *Slave: Range of 50 seconds.*

Figure 9.24: Master and Slave estimator's and encoder's outputs in presence of ground gain 20.

9.2.2 Validation of bilateral acceleration control.

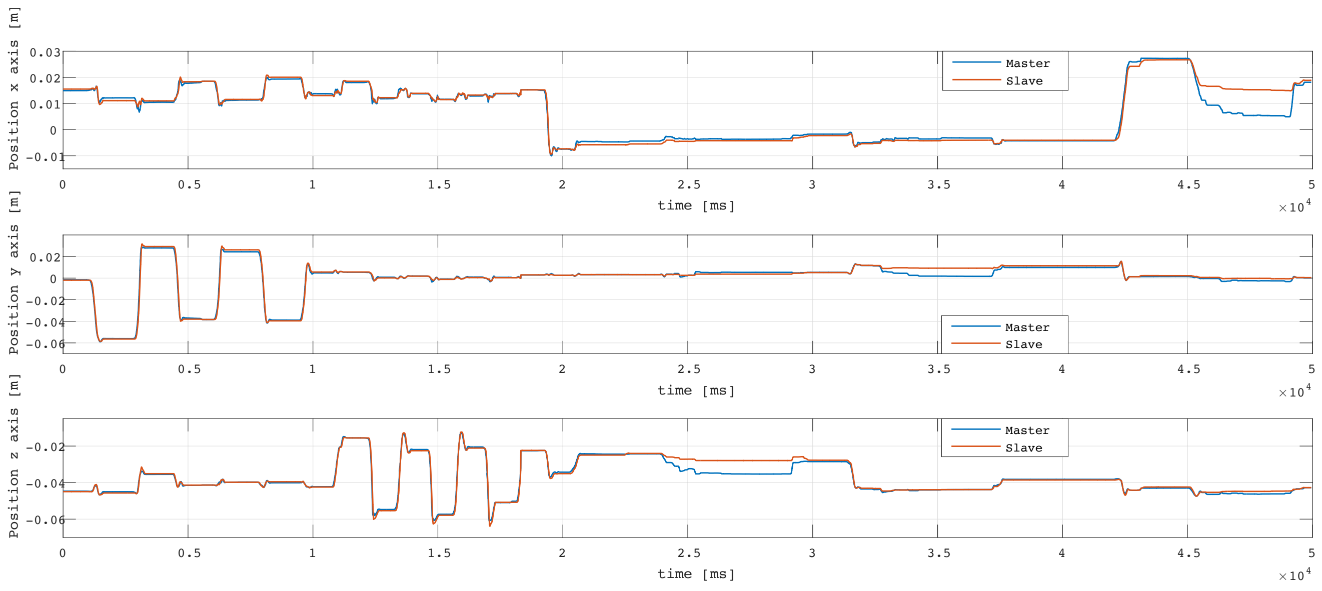
As explained in the previous chapter the aim of bilateral control is to allow the human to perceive remote environment in high transparency. Thanks to the Kalman filters, which have been validated in the previous section, there isn't too much noise on the position and force response. As explained in section 8.3 the bilateral acceleration control architecture can easily become unstable under delay. By injecting to the robots virtual damping it is possible to obtain an architecture which is more robust against time delays and lead to a compromise between stability and transparency. In this section different values of damping injection have been added to the control i.e. $b = 0$, 20, 40 and 50 where b is the ground gain. The addiction of a ground gain means that the robots will move in a damped environment and for this reason the choice of the best gain is really important in order to preserve transparency. This choice have been made by trials and error and had the goal to achieve a stable system where the slave follows the master's path and gives back a real feel of the environment. In the following figures for the range of 50, 10 and 5 seconds the encoder's position and estimated force of master and slave are plotted in comparison, in absence and presence of obstacles. In all cases examined the slave follows the master in free motion with a negligible time delay which depends on the damping injection (figure 9.25, 9.29, 9.33). In presence of obstacle it can be seen a different scenario: when an obstacle is reached the slave stop its moving and holds the current position. However, the master is not able to hold the same position and there is a error between the two coordinates. This could result from the feedback force of the slave in hard contact motion which stops the master. It must be said that this error is still small and doesn't deteriorate the performance of the bilateral control (figure 9.27, 9.31, 9.35). Finally it is important to note that in spite of impact contact to hard environment, stable contact motion is realized. Regarding the force it must be said that the force response estimated at the slave side is in opposite direction with respect to the one sensed at the master side. This guarantee that the action/reaction principle is satisfied. Of course this is not satisfied at any given time but the trend is the one described. In absence of obstacle (figure 9.26, 9.30, 9.34) it is important to note how the force becomes relevant when in the position plot there are rising and arising front i.e. in presence of sharp position changes. Instead, in the force graph in presence of obstacle there are some force peaks even in absence of position changes (figure 9.28, 9.32, 9.36). Finally the last two figure represent the case of ground gain 50 for a 6 seconds experiments. From figure 9.37 it is possible to see that the two robots became unstable after a short time. This may due to time delay that has become too big because of an high damping injection. Then even if the Kalman filters still have optimal performance (9.38) the slave is not fast enough in following the master's path leading to an oscillating behavior. A last comment can be done regarding the piecewise constant trend of the two signals: position of the master and estimated force of the slave. This trend is a consequence of the loss of information due to the communication process.

ground 0 Figure 9.25, 9.26, 9.27 and 9.28 represents respectively the position and force tracking in absence of ground gain. From these it is possible to see how the delay in free motion between the two can be estimated around ≤ 10 milliseconds. This parameter is really low and confirm the good performance of the control architecture. The position and the force almost perfectly track the ones of the other part because of good reproducibility. What cannot being showed through graph is the tactile feeling: when the operator starts to manipulate the master, the slave receives as input a step signal which can lead to instability when this input signal is too high. On the contrary the human operator can feel vivid touch sensation of the environment.

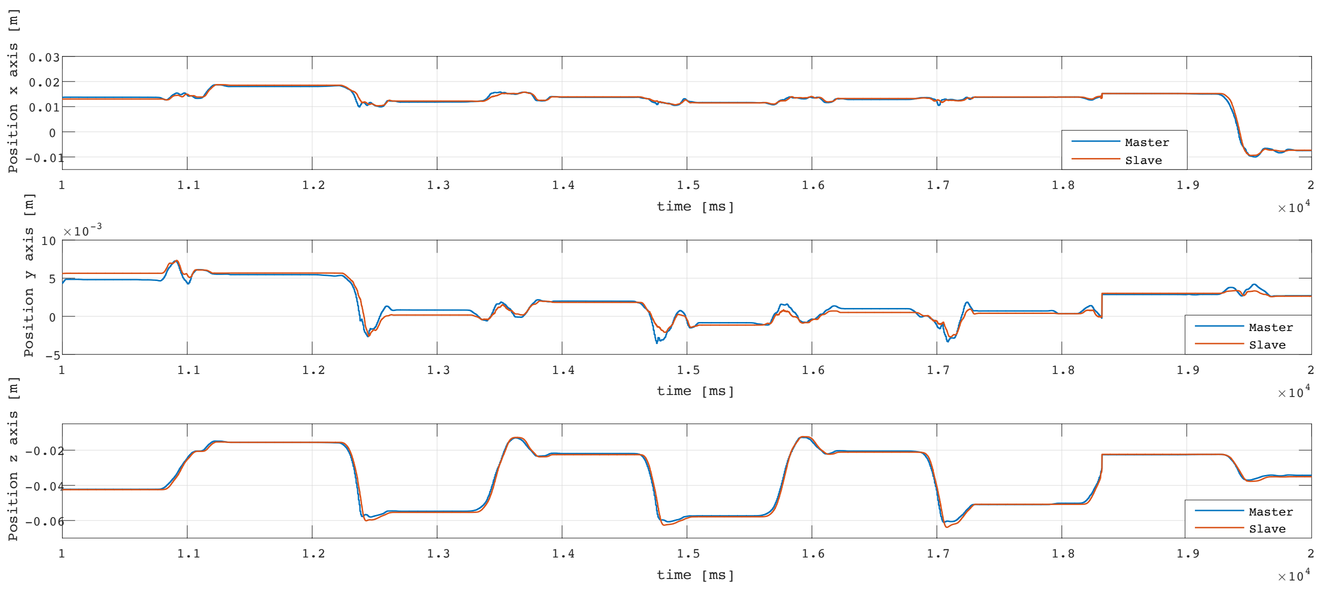
ground 20 Figure 9.29, 9.30, 9.31 and 9.32 represents respectively the position and force tracking with ground gain set equal to 20. From these it is possible to see how the delay between the two in free motion can be estimated around ≤ 30 milliseconds. This parameter even if negligible, is three times the former parameter. This still guarantee good performance. Force reflection is achieved and thank to the ground gain added, when the master starts to move the control of the slave robot doesn't receive a peak. This lead to better stability performance since the gain is small enough to preserve the sense of touch on the operator.

ground 40 Figure 9.33, 9.34, 9.35 and 9.36 represents respectively the position and force tracking with ground gain set equal to 40. From these it is possible to see how the delay between the two in free motion can be estimated around ≤ 40 milliseconds. Even though the delay starts to be much bigger than the first case, stability is achieved. The operator can still feel the touch, but he feels a damped environment.

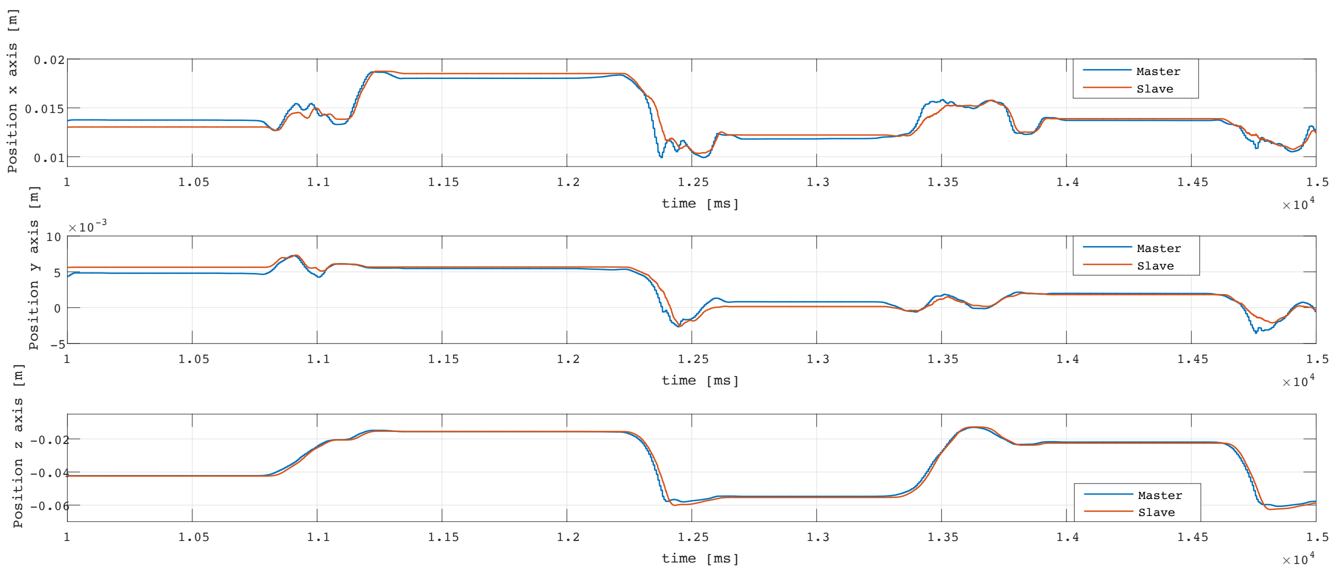
For the reason explained before, the value for ground gain chosen was 20. With the introduction of the latter gain the control scheme was considered completed.



(a) *Range of 50 seconds.*

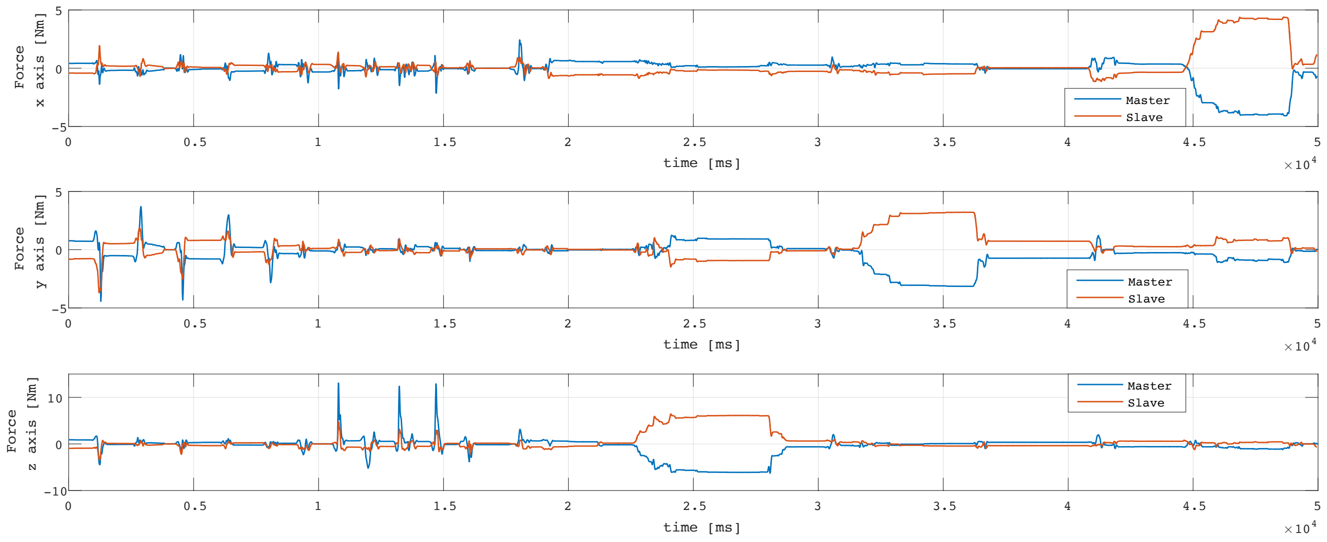


(b) *Range of 10 seconds in absence of obstacle.*

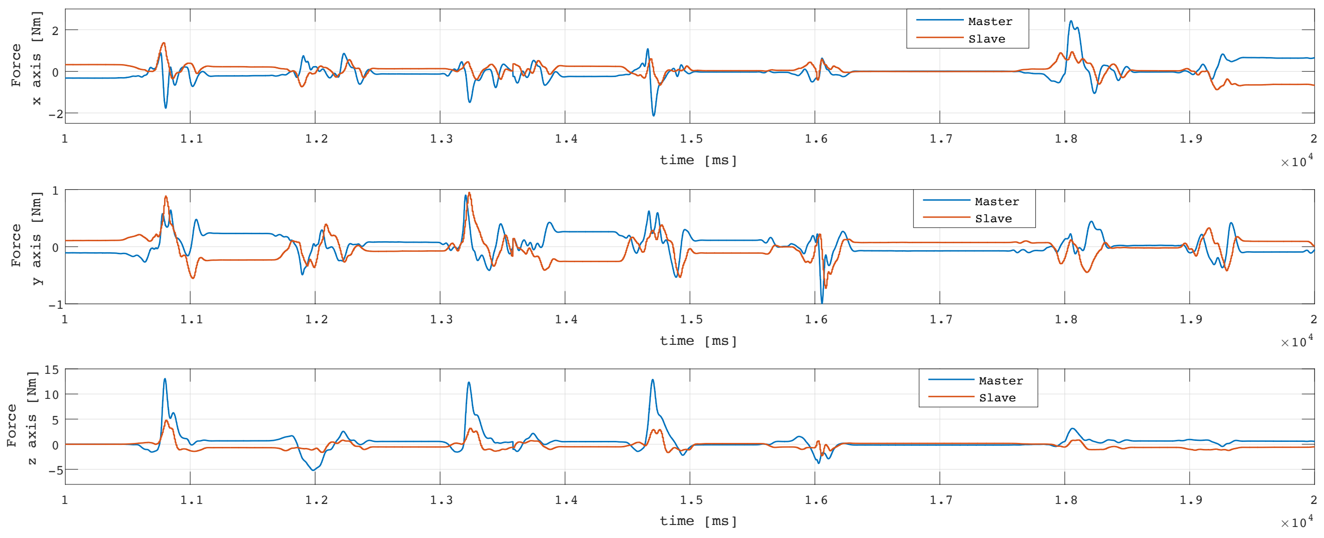


(c) *Range of 5 seconds in absence of obstacle.*

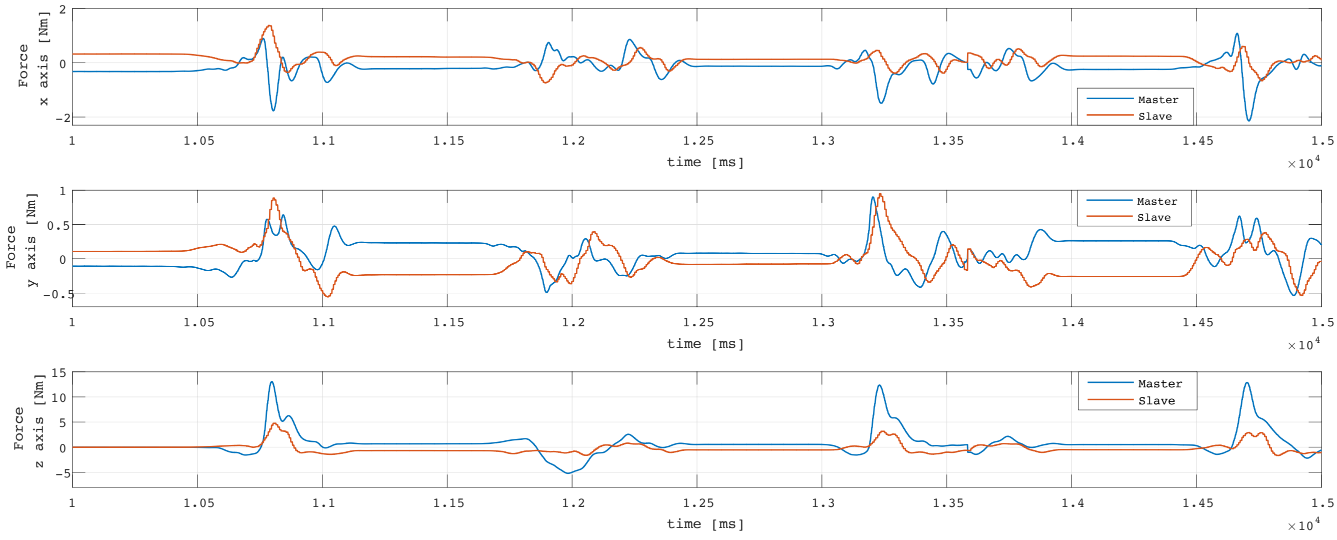
Figure 9.25: Master and Slave encoder's outputs in absence of obstacle with ground gain 0.



(a) *Range of 50 seconds.*

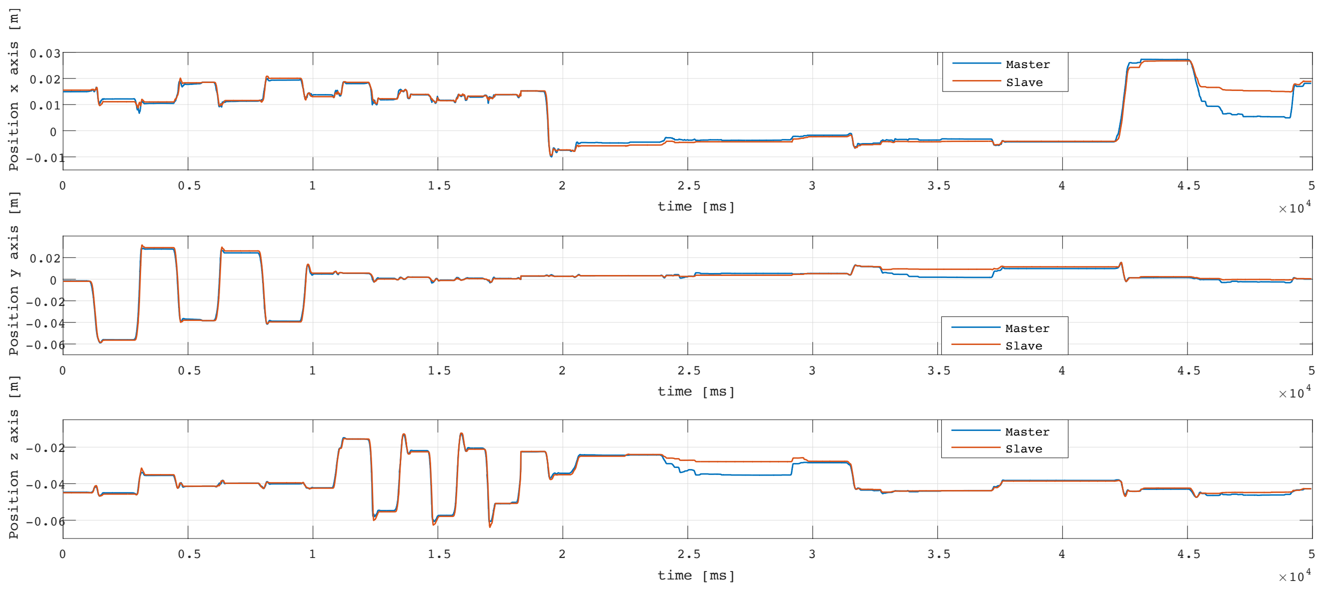


(b) *Range of 10 seconds in absence of obstacle.*

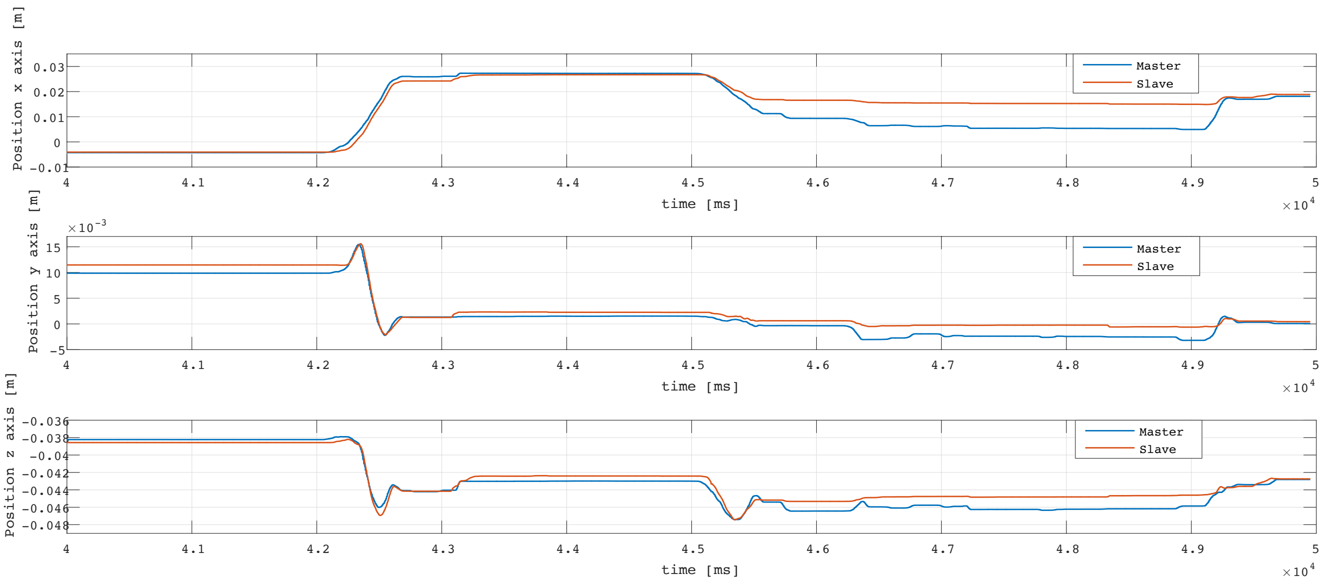


(c) *Range of 5 seconds in absence of obstacle.*

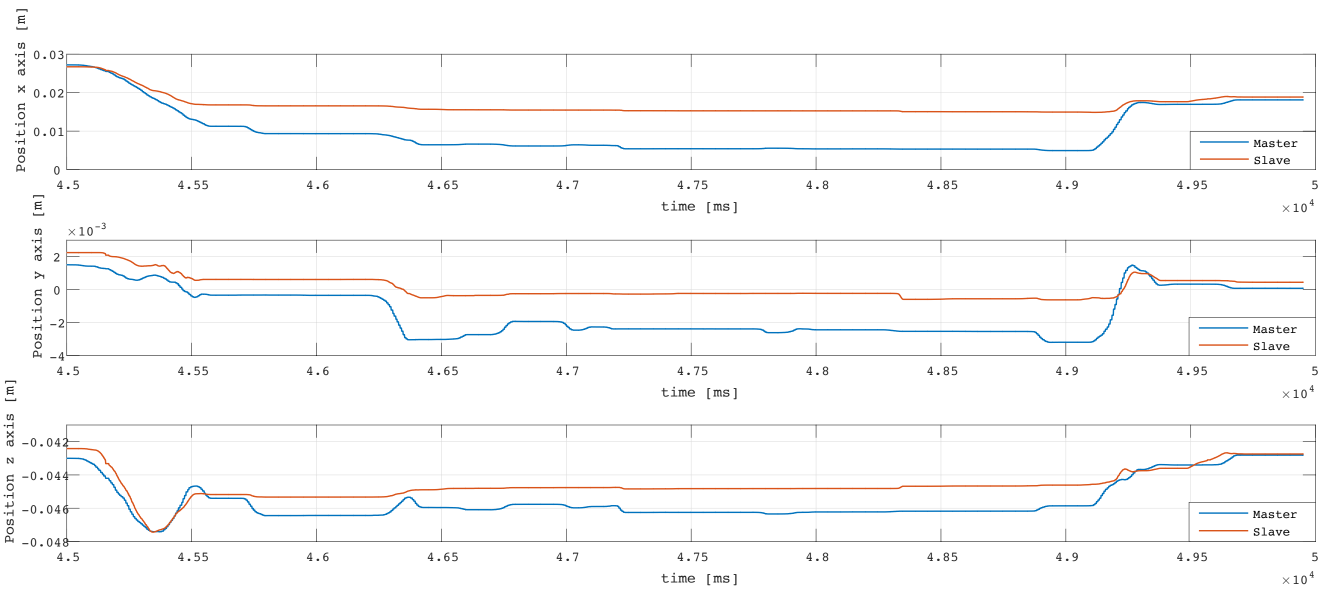
Figure 9.26: Master and Slave force estimator's outputs in absence of obstacle with ground gain 0.



(a) *Range of 50 seconds.*

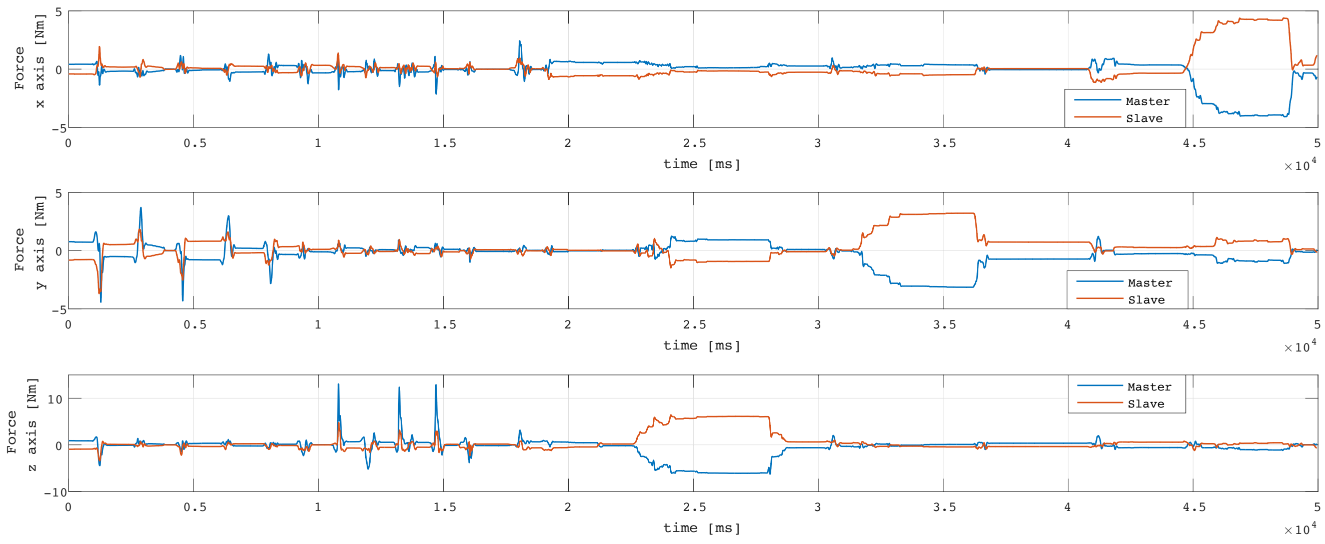


(b) *Range of 10 seconds in presence of obstacle.*

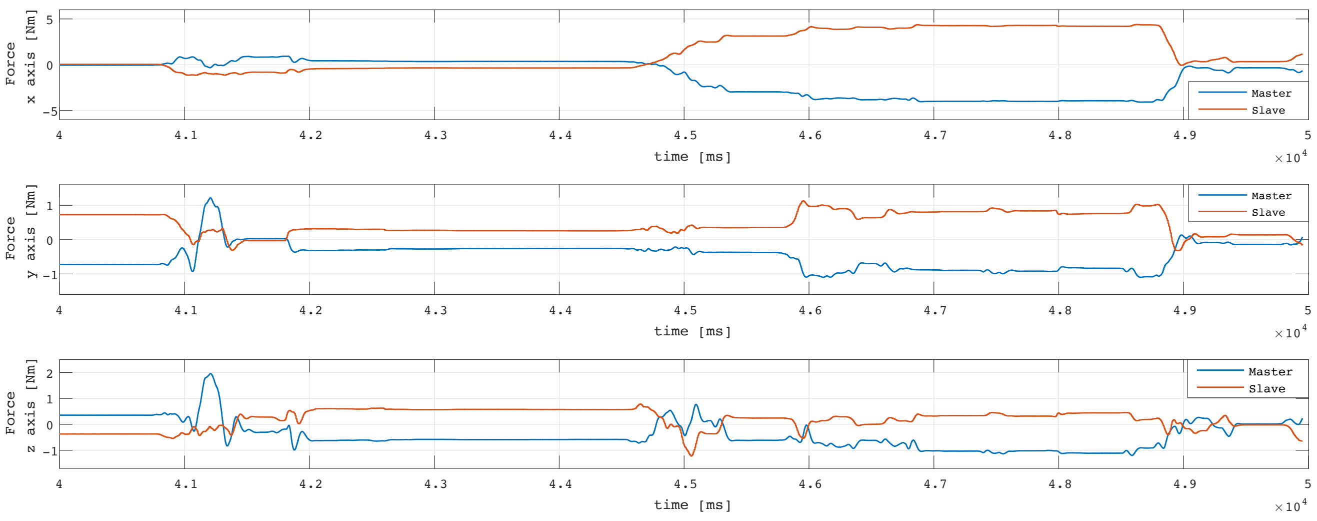


(c) *Range of 5 seconds in presence of obstacle.*

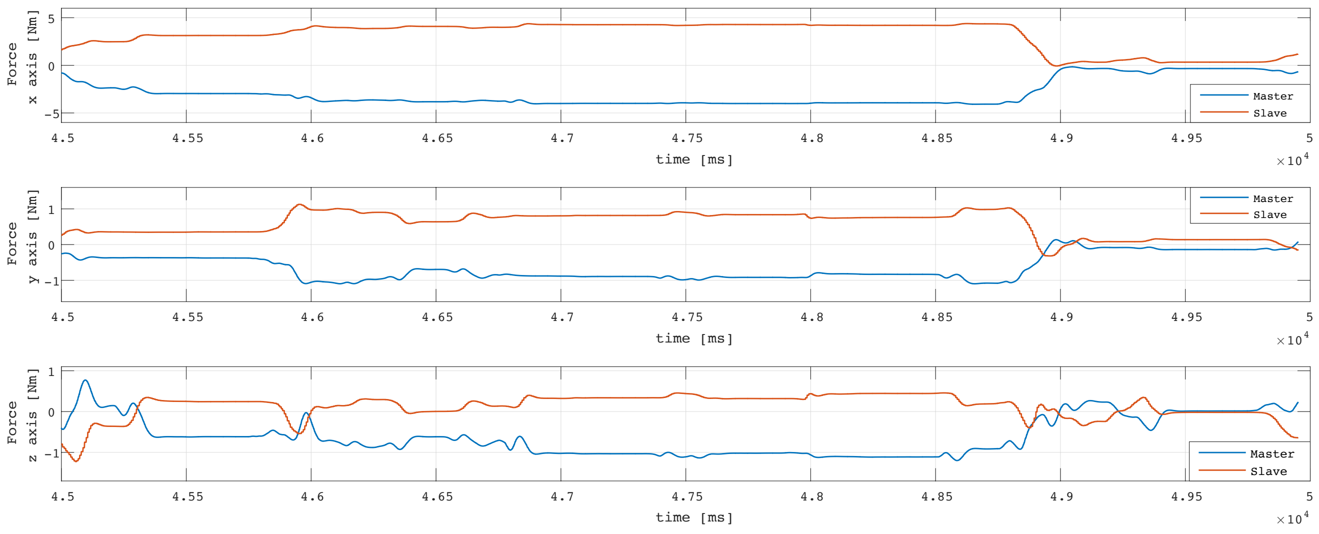
Figure 9.27: Master and Slave encoder's outputs in presence of obstacle with ground gain 0.



(a) *Range of 50 seconds.*

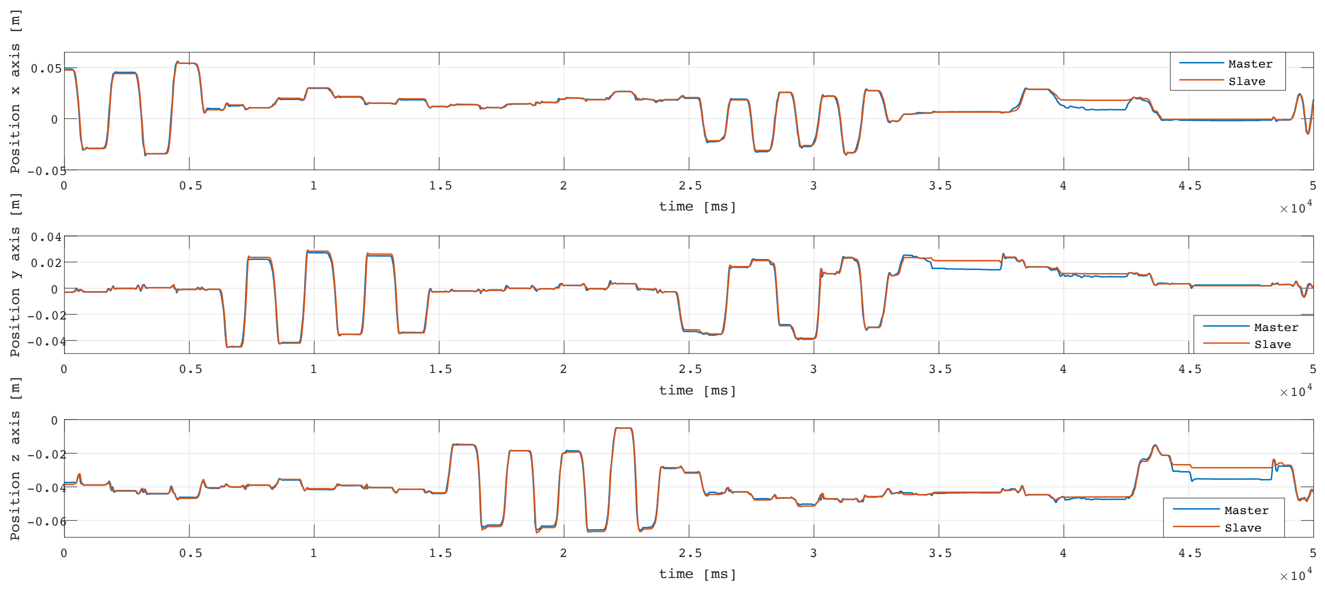


(b) *Range of 10 seconds in presence of obstacle.*

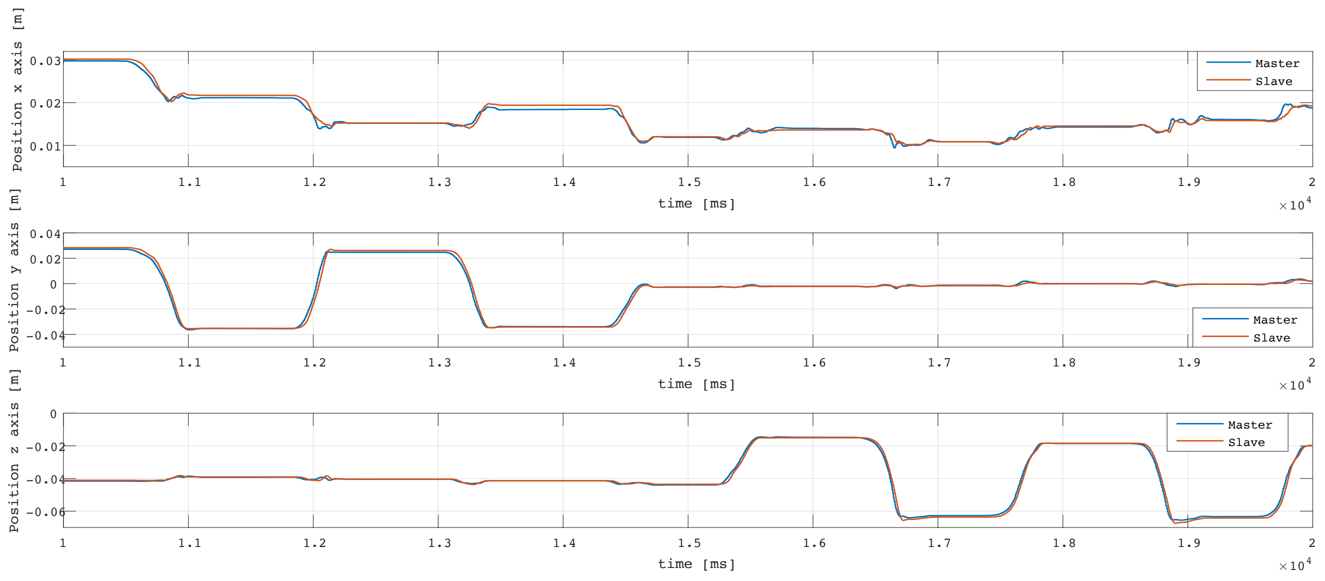


(c) *Range of 5 seconds in presence of obstacle.*

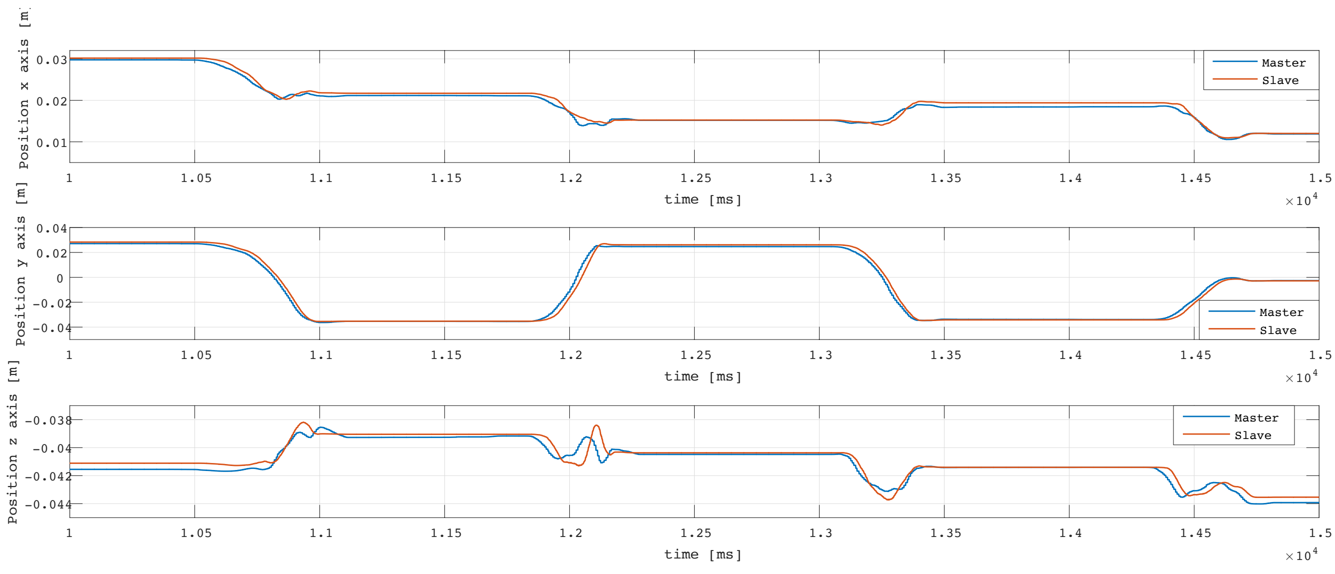
Figure 9.28: Master and Slave force estimator's outputs in presence of obstacle with ground gain 0.



(a) *Range of 50 seconds.*

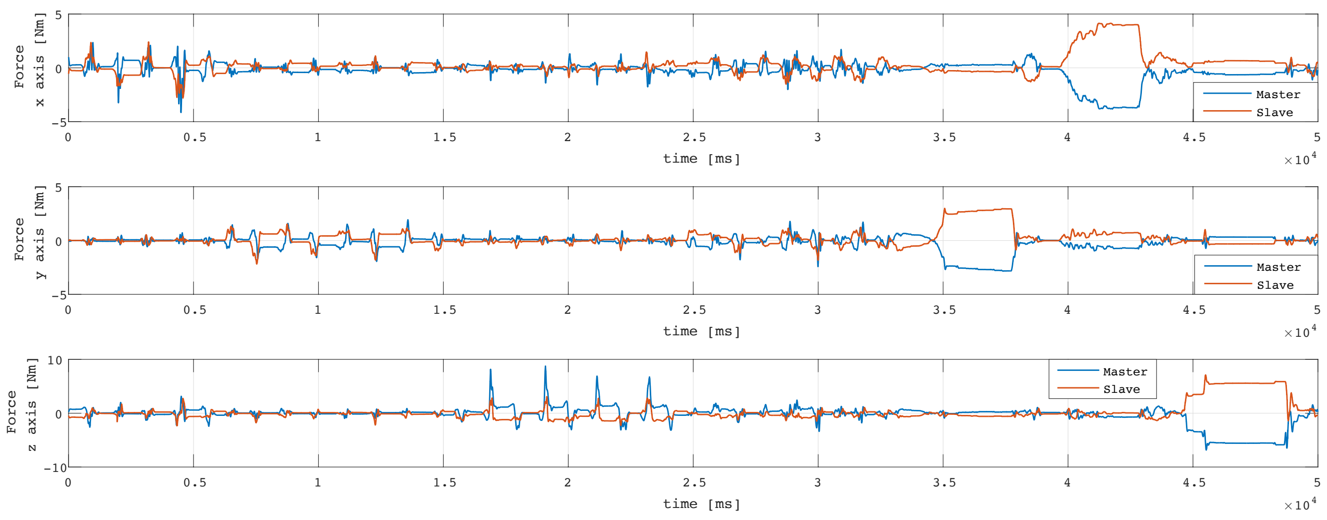


(b) *Range of 10 seconds in absence of obstacle.*

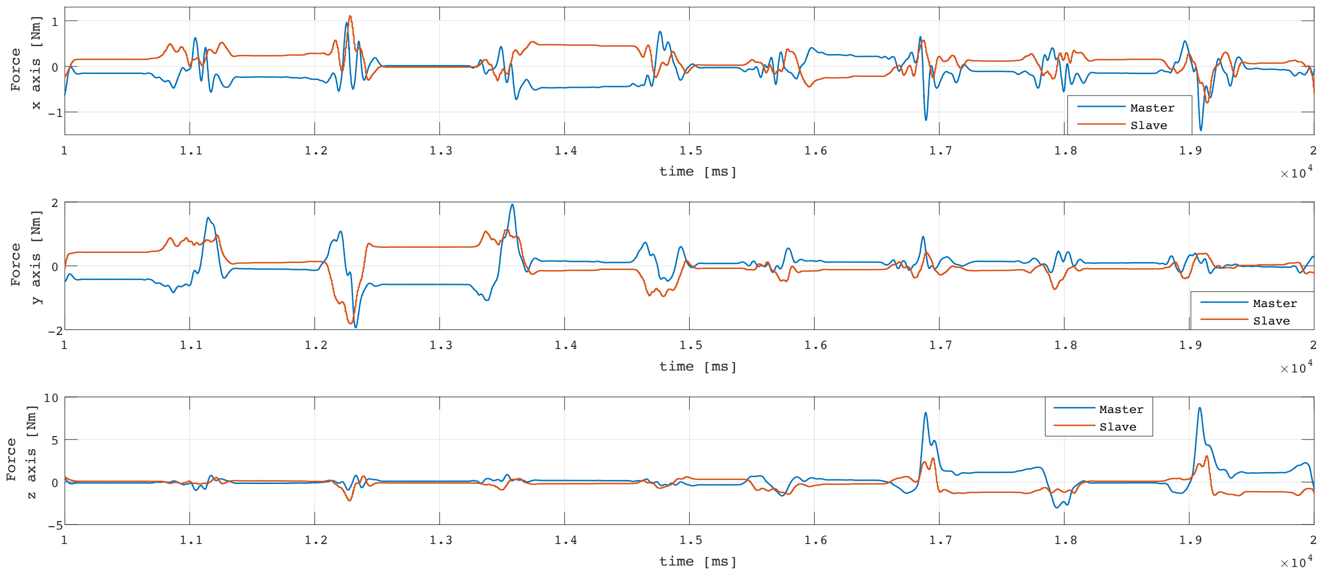


(c) *Range of 5 seconds in absence of obstacle.*

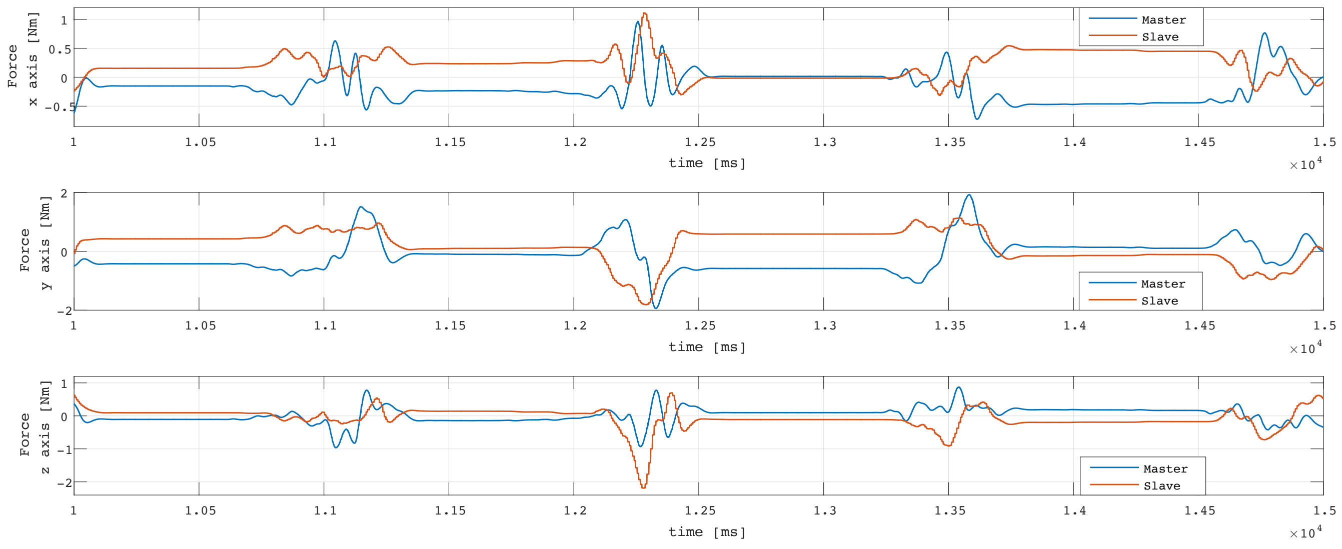
Figure 9.29: Master and Slave encoder's outputs in absence of obstacle with ground gain 20.



(a) *Range of 50 seconds.*

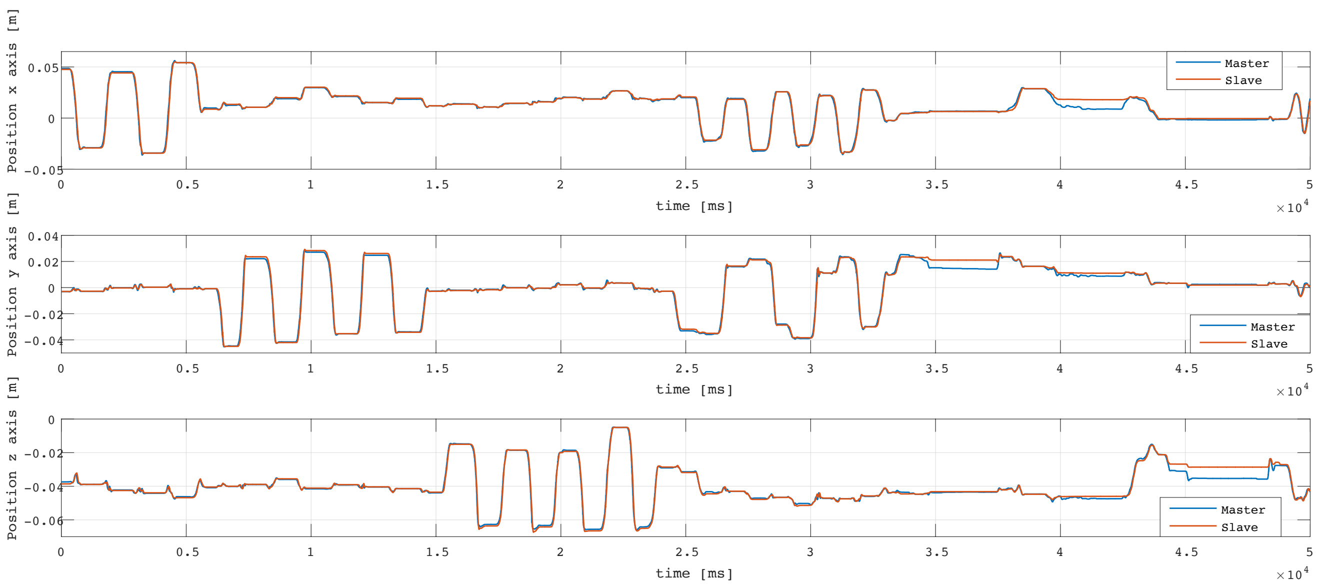


(b) *Range of 10 seconds in absence of obstacle.*

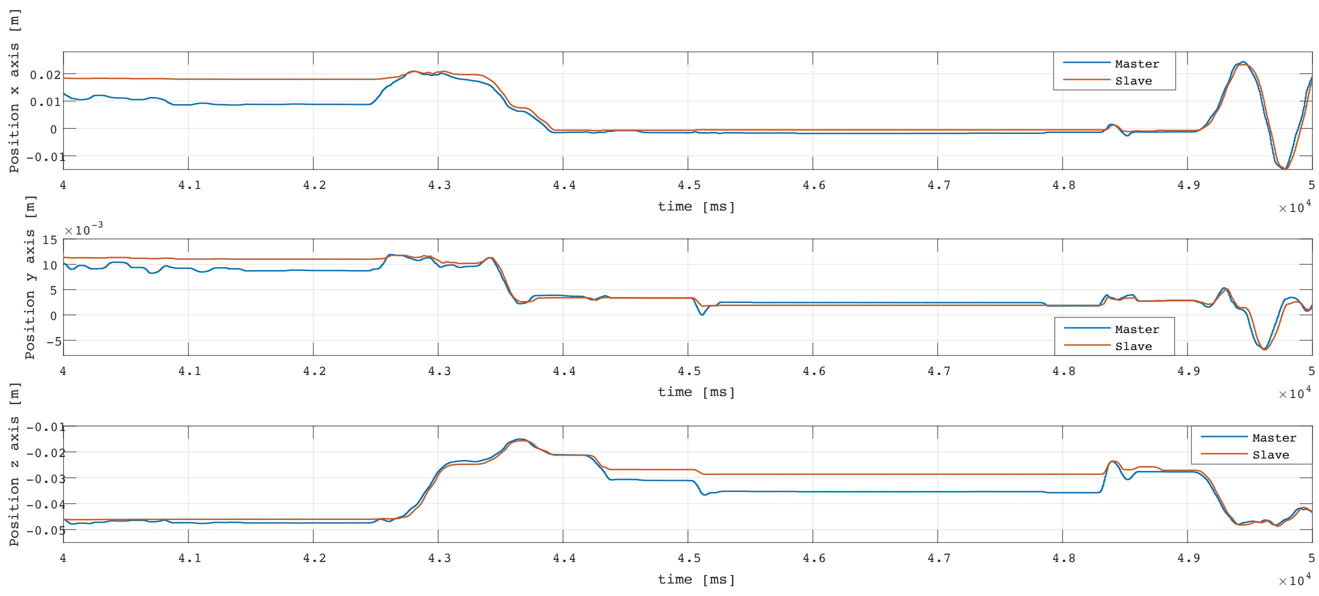


(c) *Range of 5 seconds in absence of obstacle.*

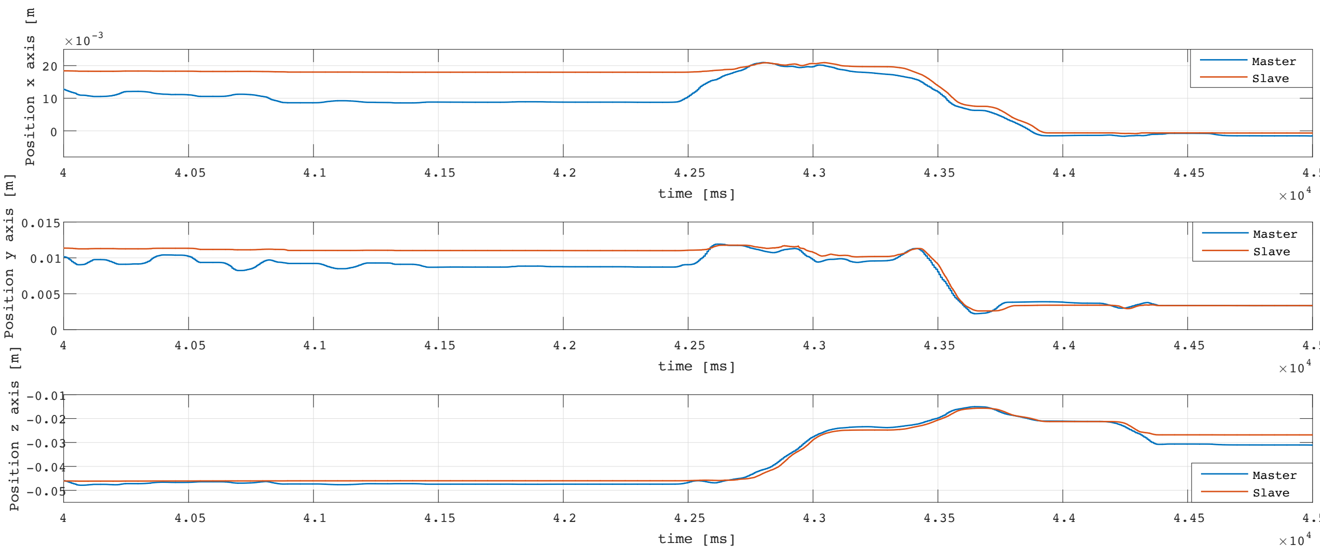
Figure 9.30: Master and Slave force estimator's outputs in absence of obstacle with ground gain 20.



(a) *Range of 50 seconds.*

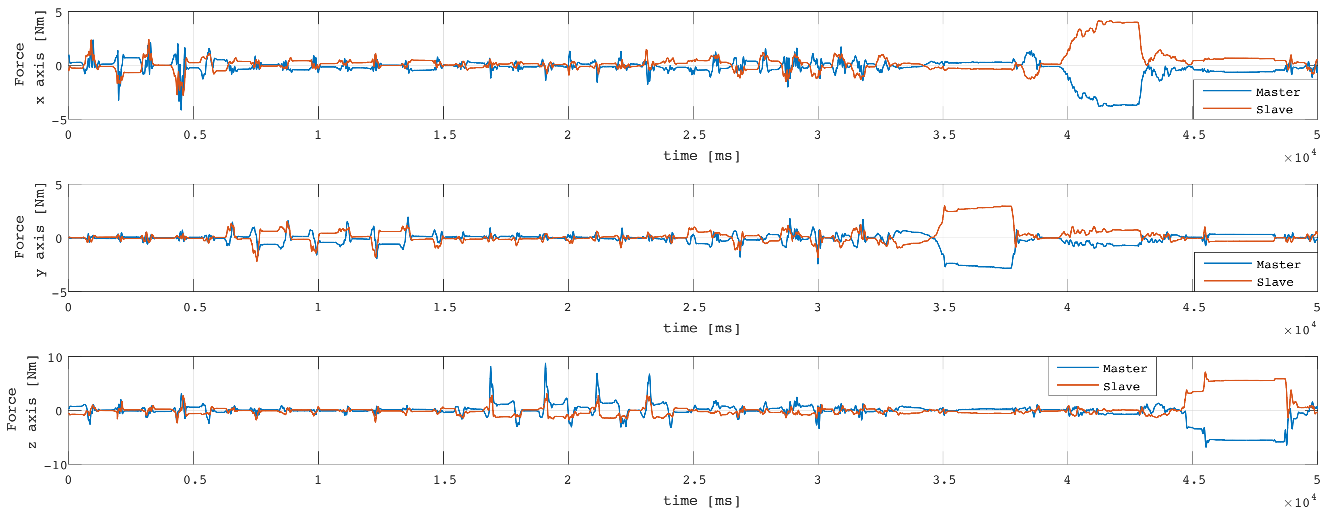


(b) *Range of 10 seconds in presence of obstacle.*

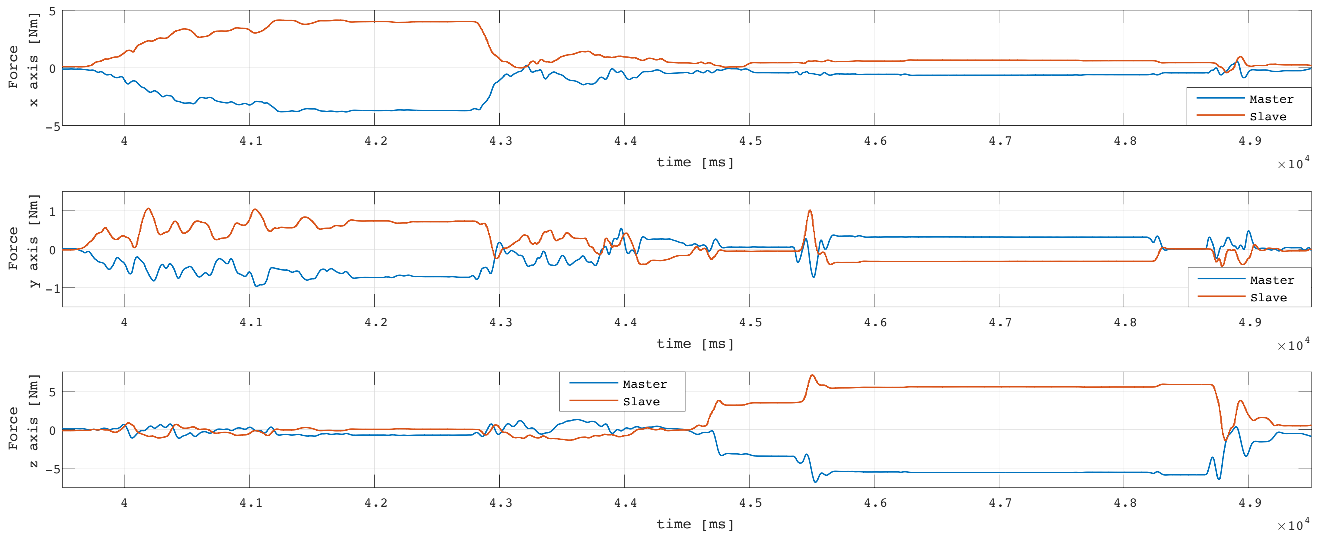


(c) *Range of 5 seconds in presence of obstacle.*

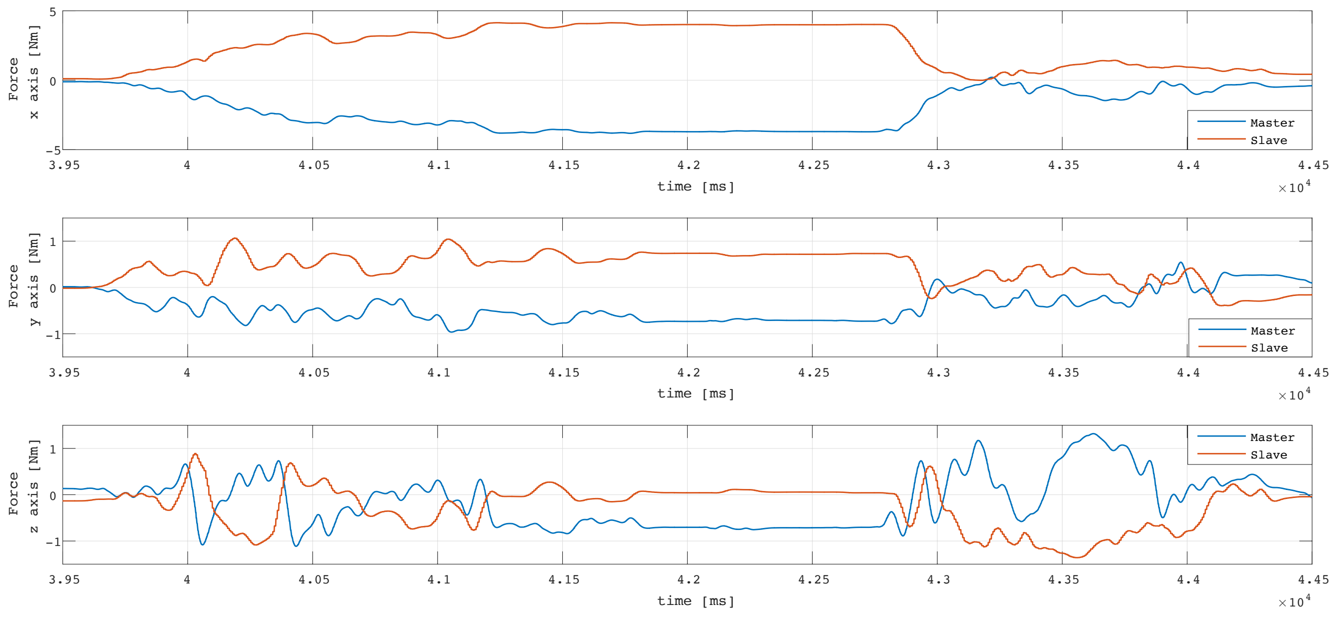
Figure 9.31: Master and Slave encoder's outputs in presence of obstacle with ground gain 20.



(a) *Range of 50 seconds.*

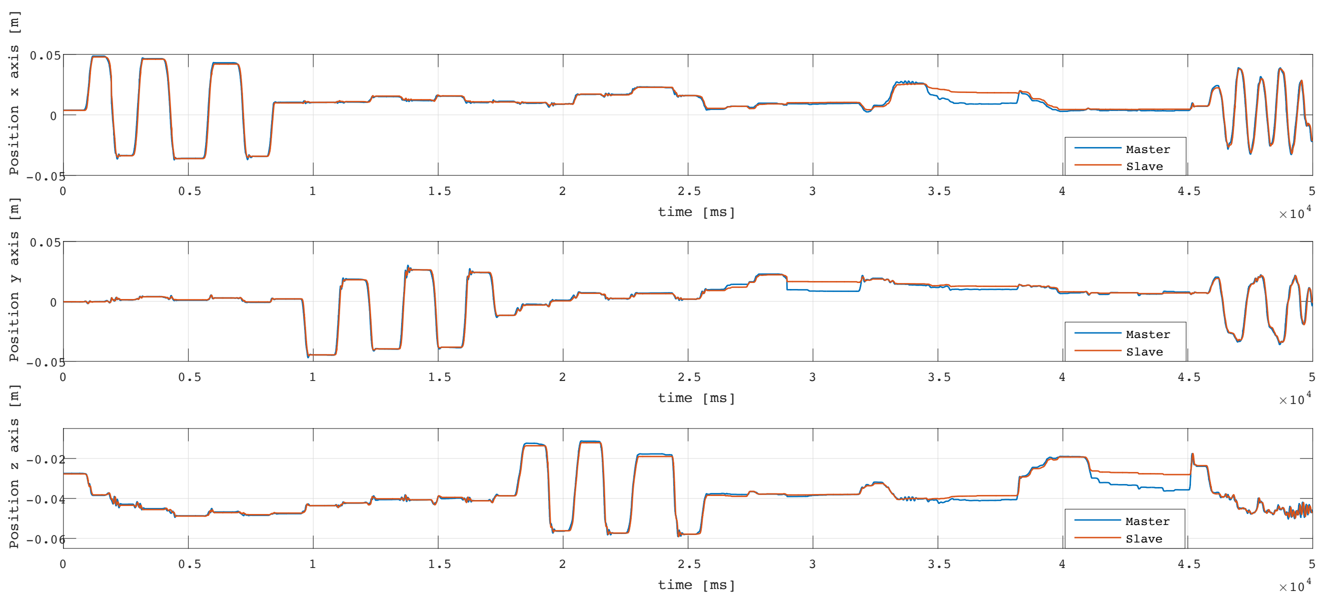


(b) *Range of 10 seconds in presence of obstacle.*

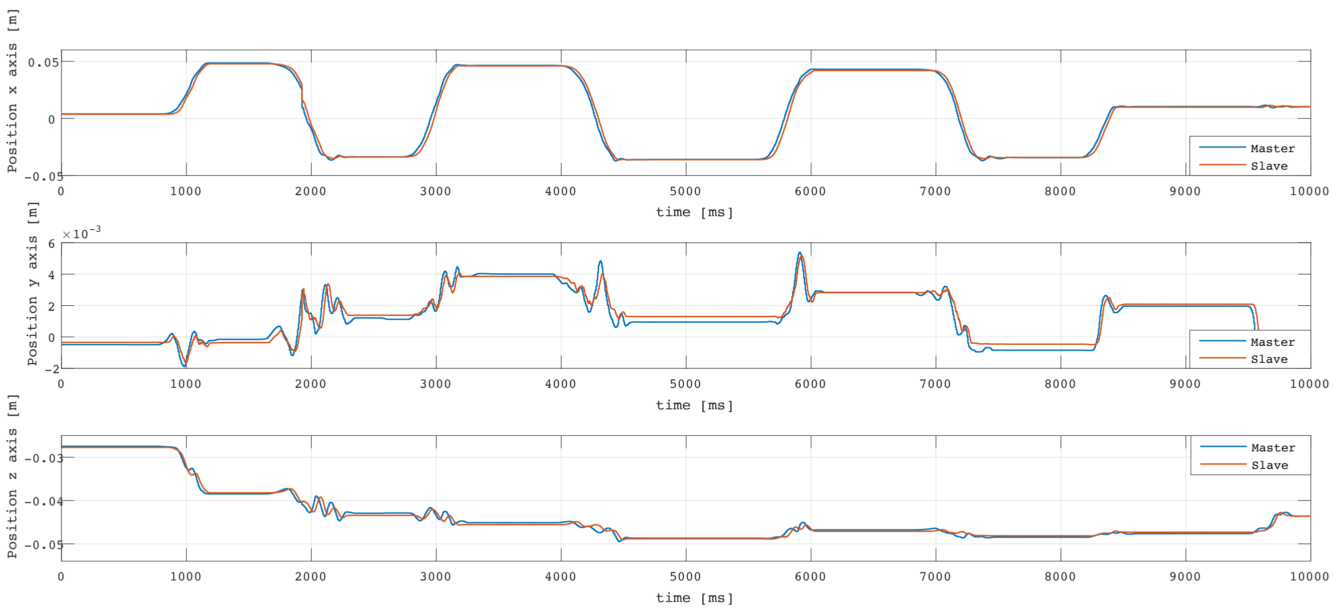


(c) *Range of 5 seconds in presence of obstacle.*

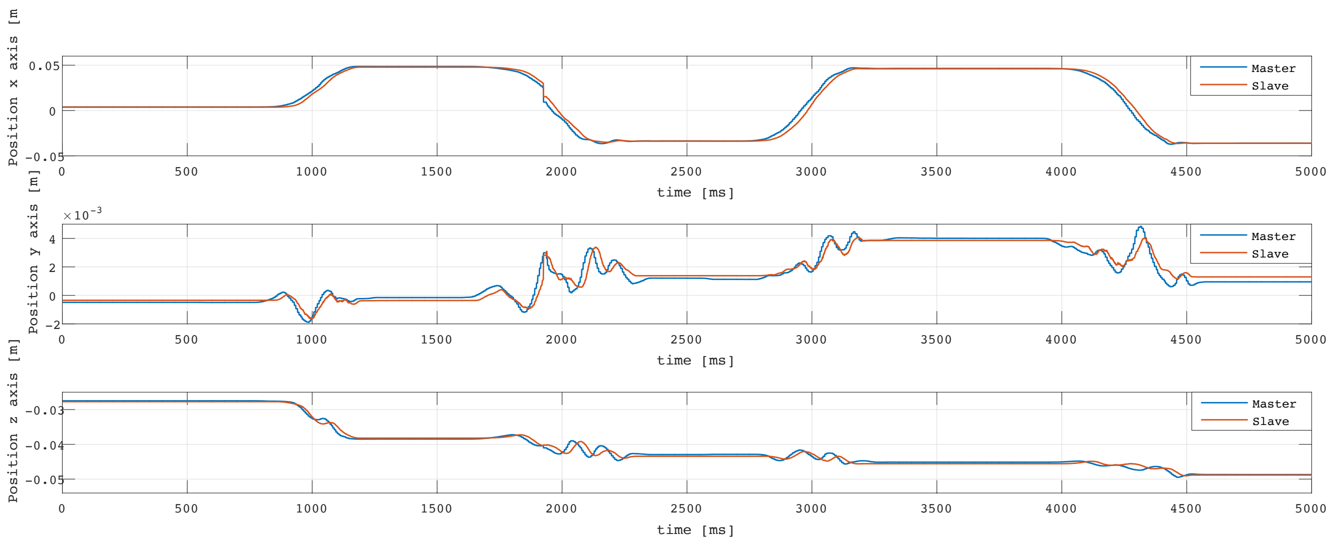
Figure 9.32: Master and Slave force estimator's outputs in presence of obstacle with ground gain 20.



(a) *Range of 50 seconds.*

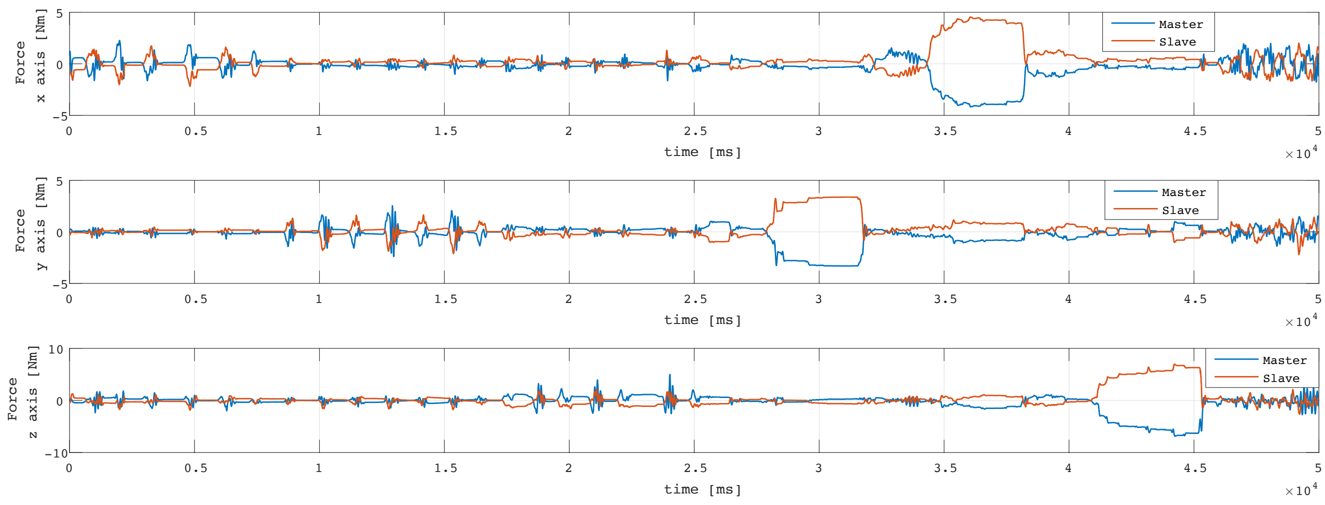


(b) *Range of 10 seconds in absence of obstacle.*

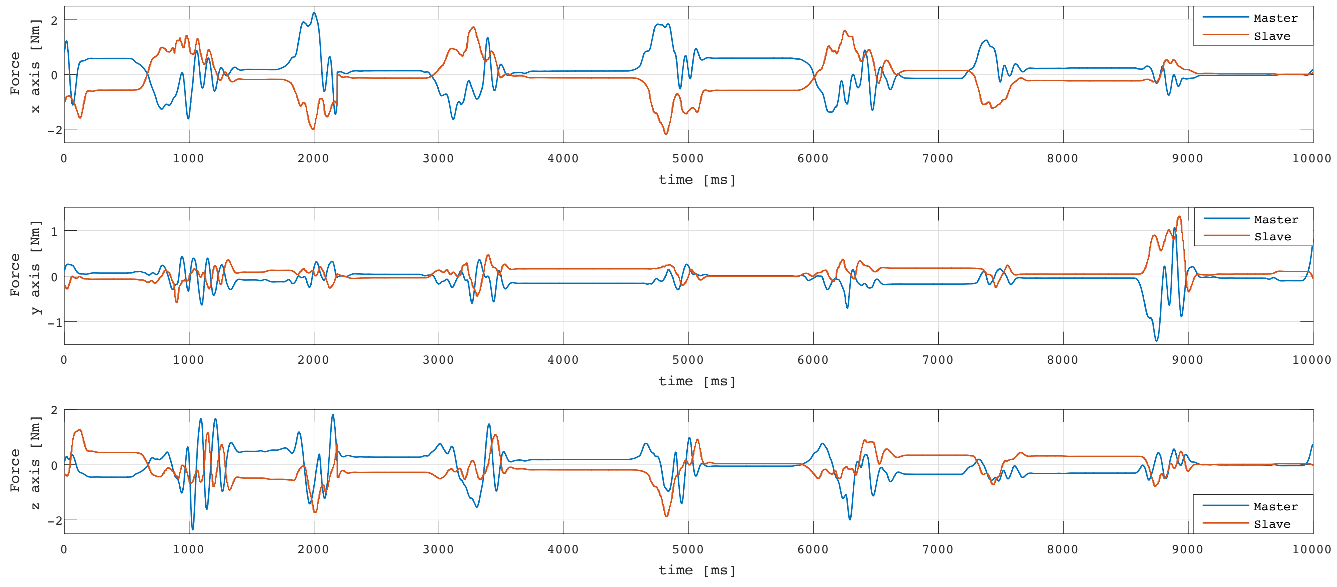


(c) *Range of 5 seconds in absence of obstacle.*

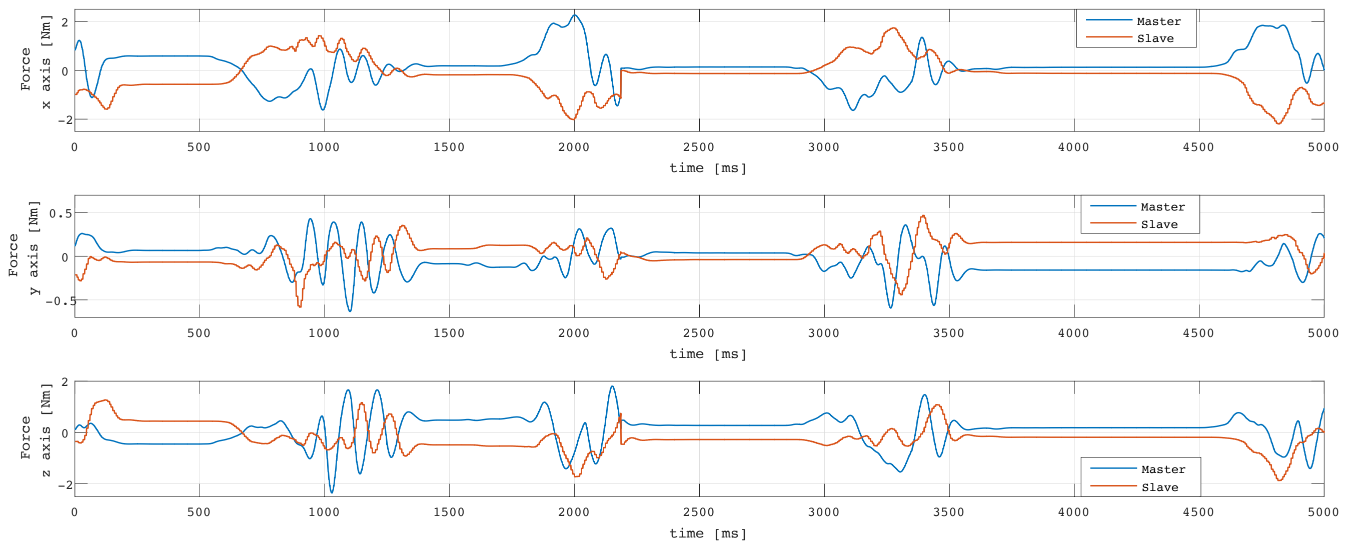
Figure 9.33: Master and Slave encoder's outputs in absence of obstacle with ground gain 40.



(a) *Range of 50 seconds.*

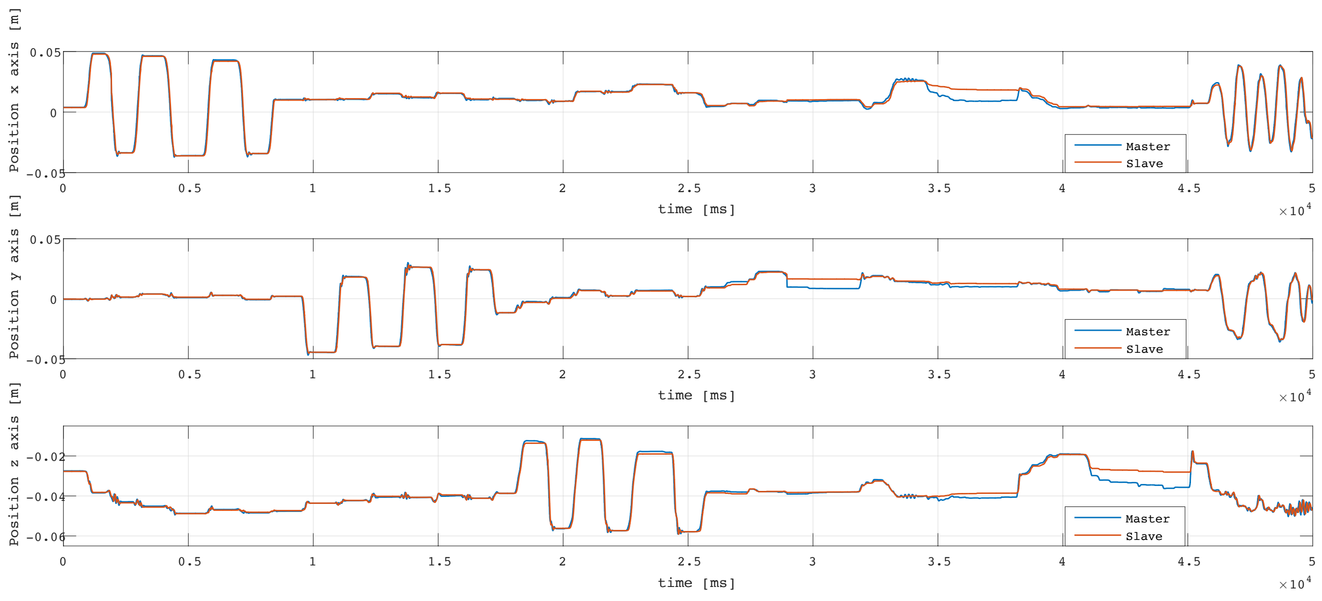


(b) *Range of 10 seconds in absence of obstacle.*

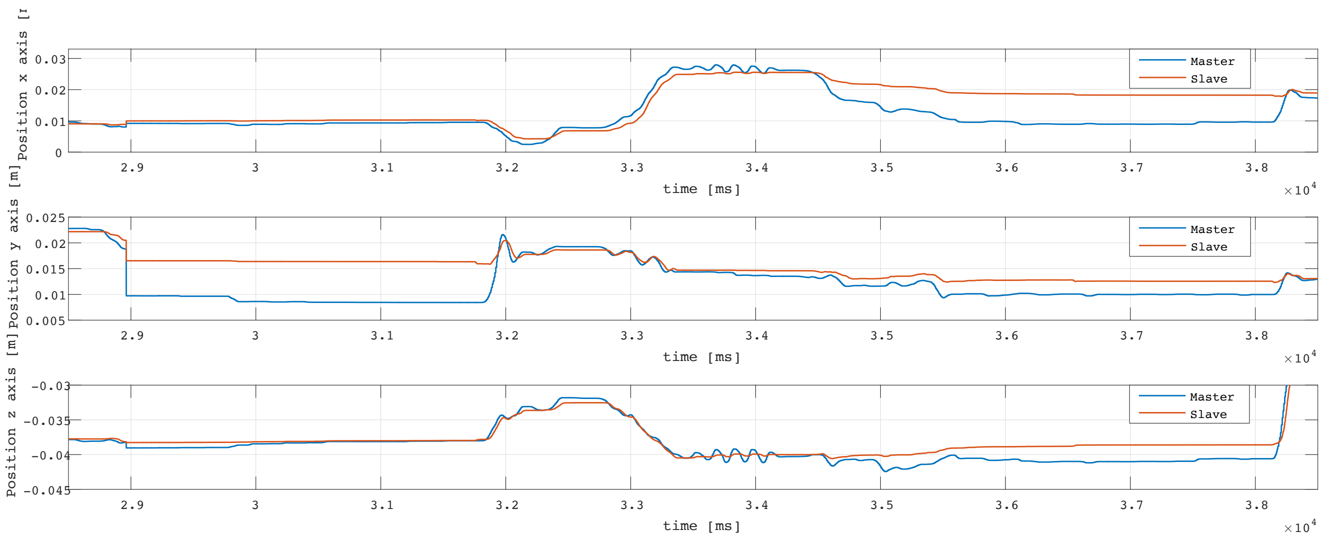


(c) *Range of 5 seconds in absence of obstacle.*

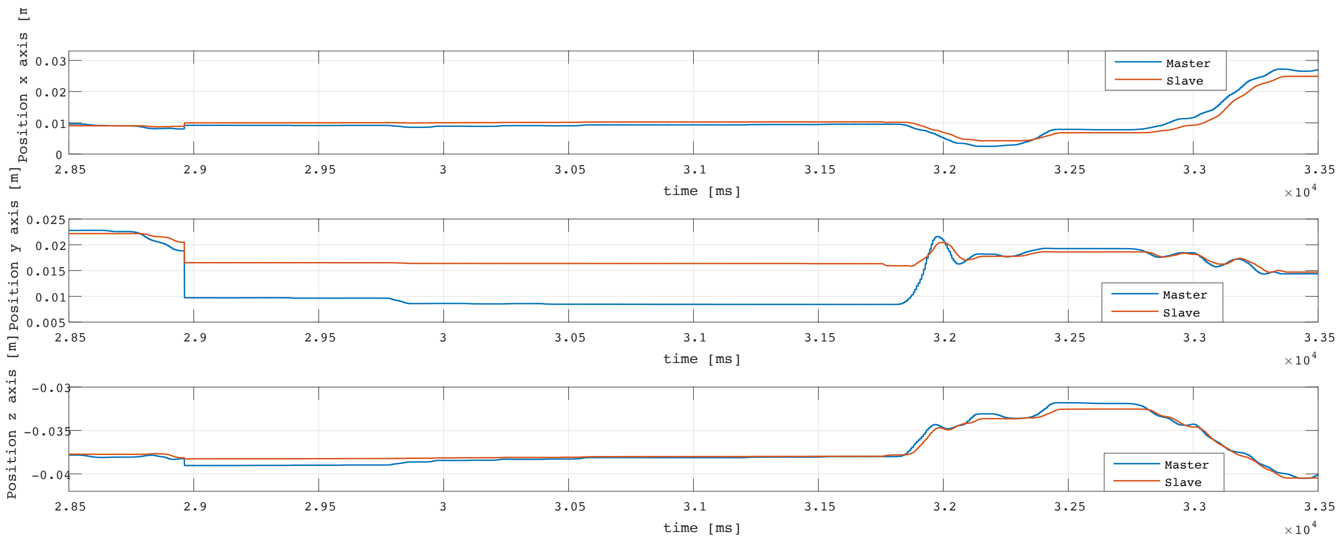
Figure 9.34: Master and Slave force estimator's outputs in absence of obstacle with ground gain 40.



(a) *Range of 50 seconds.*

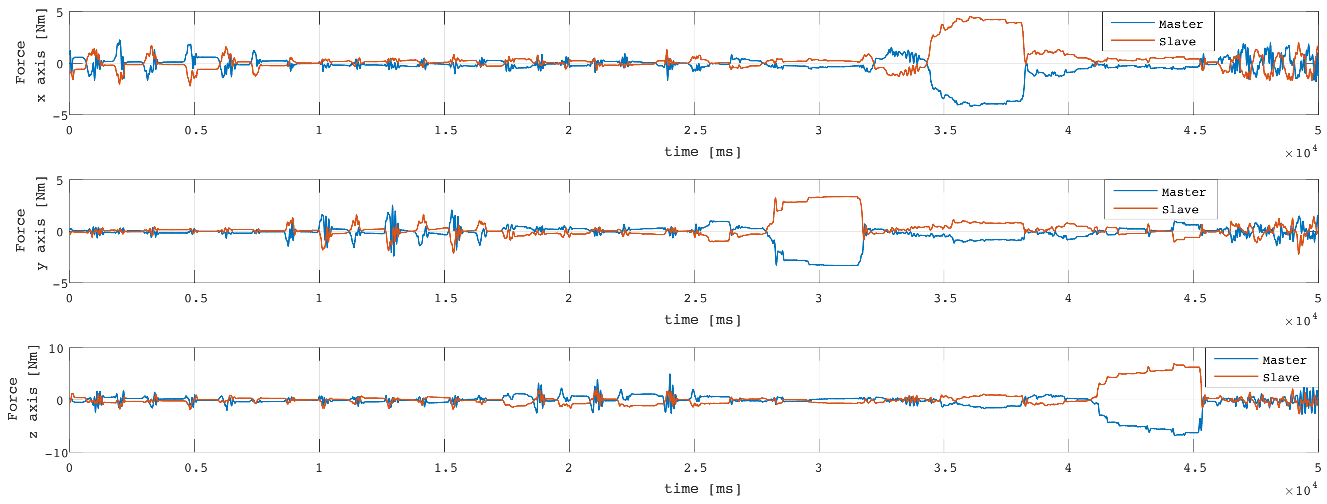


(b) *Range of 10 seconds in presence of obstacle.*

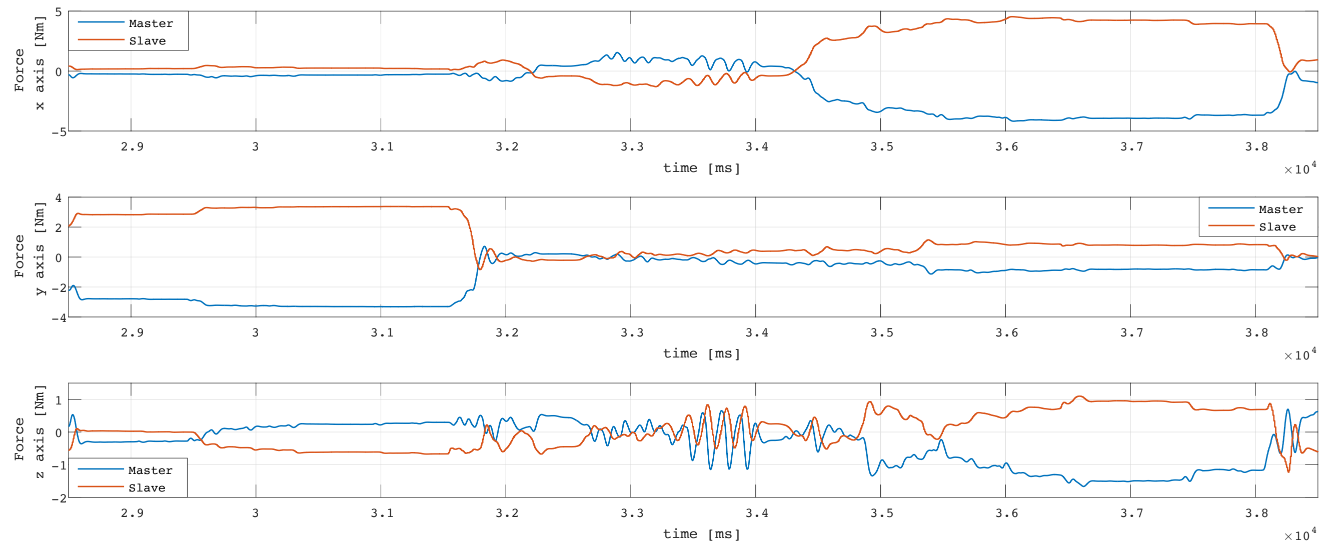


(c) *Range of 5 seconds in presence of obstacle.*

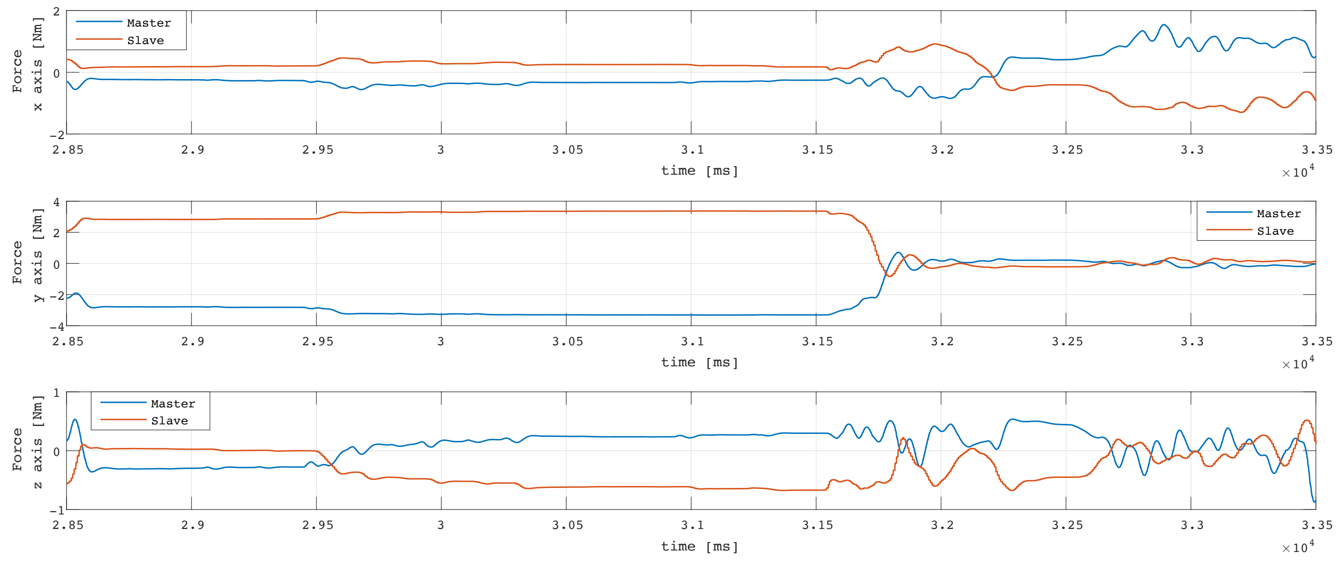
Figure 9.35: Master and Slave encoder's outputs in presence of obstacle with ground gain 40.



(a) *Range of 50 seconds.*

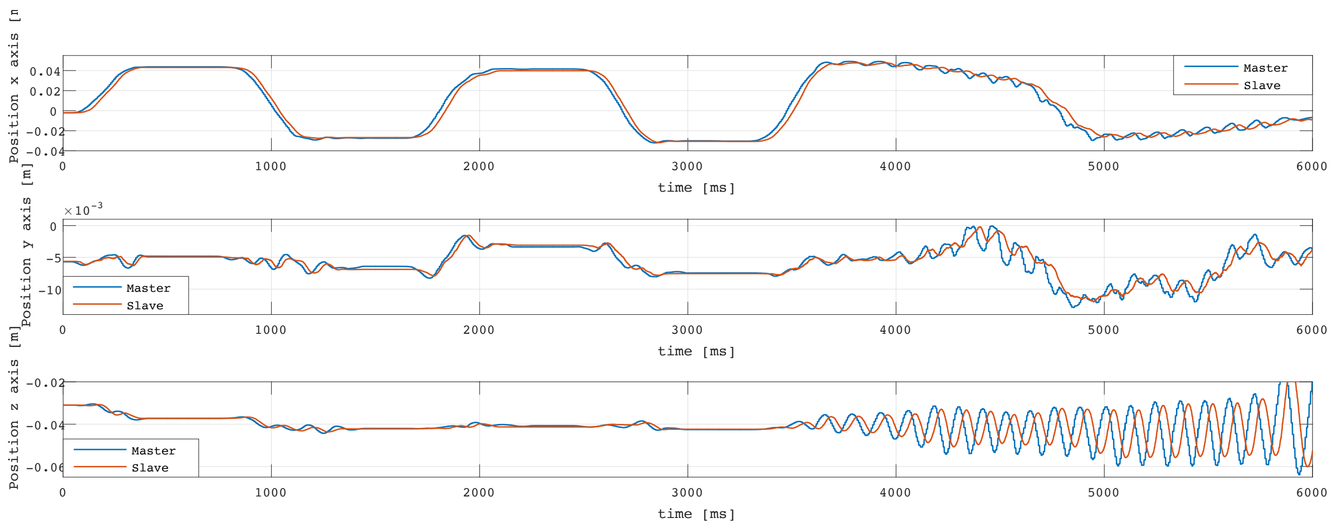


(b) *Range of 10 seconds in presence of obstacle.*

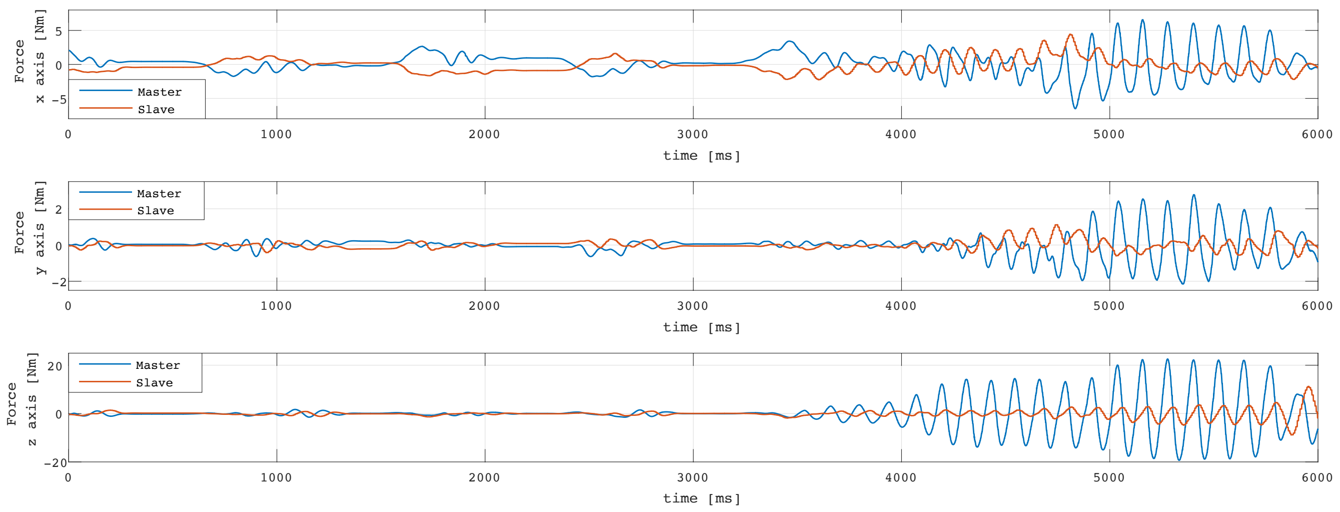


(c) *Range of 5 seconds in presence of obstacle.*

Figure 9.36: Master and Slave force estimator's outputs in presence of obstacle with ground gain 40.

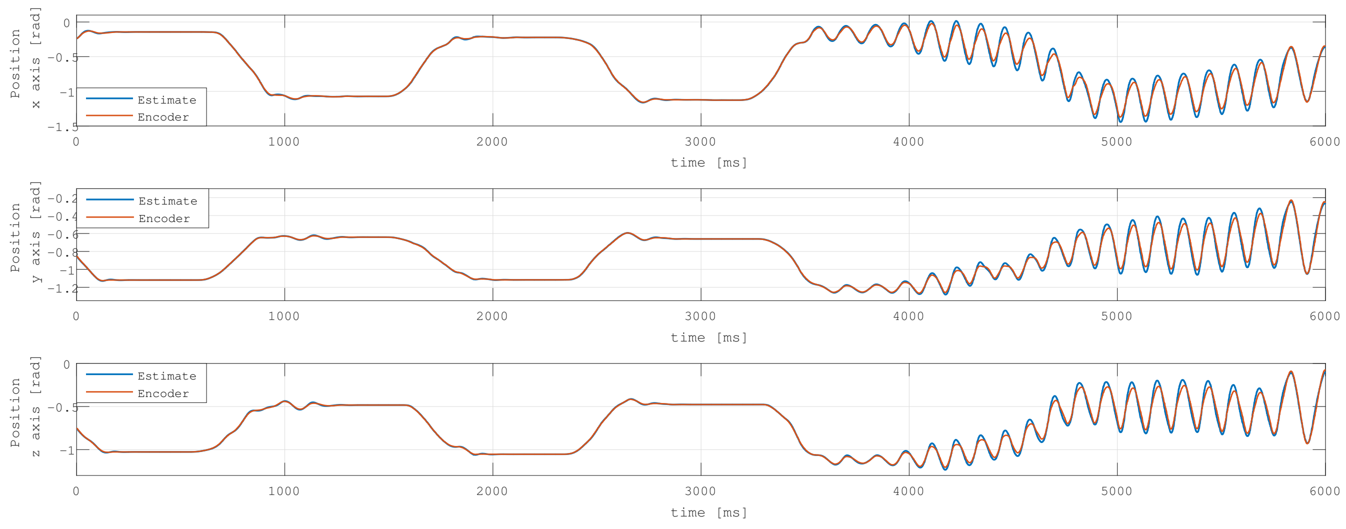


(a) *Range of 6 seconds.*

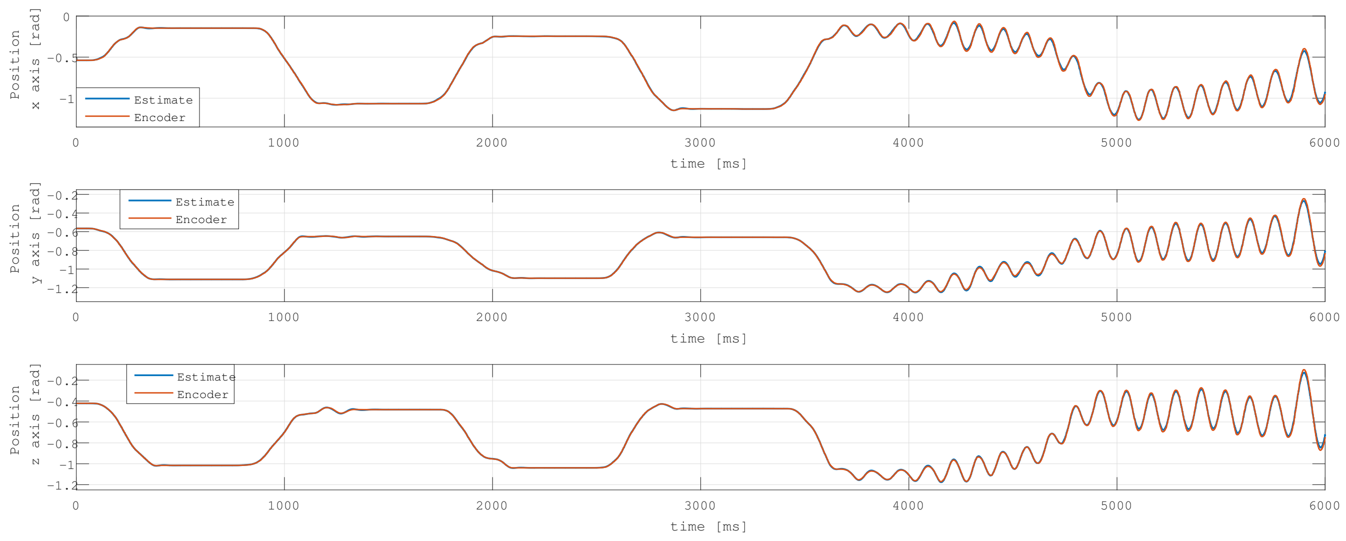


(b) *Range of 6 seconds.*

Figure 9.37: Master and Slave encoder's outputs with ground gain 40.



(a) *Range of 6 seconds.*



(b) *Range of 6 seconds.*

Figure 9.38: Master and Slave force estimator's outputs with ground gain 40.

Chapter 10

Conclusion and future works

In this thesis two different control scheme have been designed and compared in order to implement an haptic force feedback application to two identical delta robots. More in details the first control scheme consists on a position-position bilateral teleoperation control with passivity term architecture. As explained in the experimental section this solution was not satisfactory since the robots were slow, stiff to move and the scheme cannot provide a vivid feel of touch on the operator because of the high time delay. Then a second control architecture was implemented. This consists of an acceleration bilateral control with Disturbance Observer and Kalman Filters in order to estimate the position, the velocity and especially the acting force. Bilateral teleoperation system was decomposed into the common and differential mode by acceleration based controller. The force servoing was attained in the common node and the position error was regulated in the differential mode. In order to consider the conformity of force with position, the force servoing and the position regulator were integrated in the acceleration. Finally a ground anchorage was added to the scheme in order to made the system more stable. The positive performance of this control scheme and the Kalman filters were deeply analyzed. Finally it can be said that the second proposed controller implemented with two identical DELTA robots can be used for human-robot interaction application. In fact the transparency was more than succeeded and the operator could feel a vivid sense of touch.

Nevertheless some improvements can be made. First of all the communication protocol can be improved in order to reduce the time delay that affect so much the performance of the control architecture. Then another issue affects the work: the fact that the Kalman filters are only working when the robots are moving. If the force which affect the robot is very small and cannot exceed the viscous and friction terms, the kalman filters and the DOB disturbance observer don't feel a torque acting. For this reason the control scheme cannot compensate the disturbances and as soon as the force acting overcome the friction term an input step signal is sent as reference in the slave and this lead to an initial jerk which could lead to instability in application with bigger and heavier robots. This problem could be solved by making the motors always moving a little even if the robot is still, in order not to have lack of information about the dynamics. This latter problem wouldn't exists in the presence of force sensor, given that the sensor is always measuring; but as showed in section 8.3 the force sensors have many disadvantages. Last but not the least this control scheme could be tested even in the case of not identical master-slave. In this latter case in must be taken into account a scaling factor between the two robots.

Appendix A

Kinematic, code and details.

A.1 Forward Kinematic.

As described in chapter 5 the main idea in order to solve forward kinematic is to find the intersection of three sphere that have origin point at the elbow of each robot arm chain and forearms length l_2 as radius. This led to system (4.1):

$$\begin{cases} (x-x_1)^2+(y-y_1)^2+(z-z_1)^2=l_2^2 \\ (x-x_2)^2+(y-y_2)^2+(z-z_2)^2=l_2^2 \\ (x-x_3)^2+(y-y_3)^2+(z-z_3)^2=l_2^2. \end{cases} \tag{A.1}$$

Since this system is difficult to solve numerical methods are employed to find the solutions to the non-linear equations with multi-variables. For the purpose of reducing variables the first equation in (A.1) has been subtracted from the latter two.[11] This led to the following two equations:

$$\begin{cases} z_2z+(x_2-x_1)x+(y_2-y_1)y=\frac{w_2-w_1}{2} \\ z_3z+(x_3-x_1)x+(y_3-y_1)y=\frac{w_3-w_1}{2}, \end{cases} \tag{A.2}$$

where $w_i=x_i^2+y_i^2+z_i^2$. According to equation (A.2), x and y can be expressed by z leading to:

$$\begin{cases} x=a_1z+b_1 \\ y=a_2z+b_2, \end{cases} \tag{A.3}$$

where:

$$a_1=\frac{(y_2-y_1)z_3-(y_3-y_1)z_2}{(x_2-x_1)(y_3-y_1)-(x_3-x_1)(y_2-y_1)}, \tag{A.4}$$

$$b_1=\frac{1}{2}\frac{(w_2-w_1)(y_3-y_1)-(w_3-w_1)(y_2-y_1)}{(x_2-x_1)(y_3-y_1)-(x_3-x_1)(y_2-y_1)}, \tag{A.5}$$

$$a_2=\frac{(x_2-x_1)z_3-(x_3-x_1)z_2}{(x_3-x_1)(y_2-y_1)-(x_2-x_1)(y_3-y_1)}, \tag{A.6}$$

$$b_2=\frac{1}{2}\frac{(w_2-w_1)(x_3-x_1)-(w_3-w_1)(x_2-x_1)}{(x_3-x_1)(y_2-y_1)-(x_2-x_1)(y_3-y_1)}. \tag{A.7}$$

Substituting Eqn. (A.3) into the first equation in Eqn. (A.1), that is,

$$(1+a_1^2+a_2^2)z^2+2(a_1b_1-a_1x_1+a_2b_2-a_2y_1)z+(b_1-x_1)^2+(b_2-y_1)^2-l_2^2=0. \tag{A.8}$$

a second order equation in the variable z is obtained. Once z is acquired also y and x are known and the resulting coordinate (x, y, z) for the end effector are:

$$\begin{cases} x=a_1z+b_1 \\ y=a_2z+b_2 \\ z=\frac{-b\pm\sqrt{b^2-4ac}}{2a}, \end{cases} \tag{A.9}$$

where

$$a=1+a_1^2+a_2^2, \tag{A.10}$$

$$b=2(a_1b_1-a_1x_i+a_2b_2-a_2y_1), \tag{A.11}$$

$$c=(b_1-x_1)^2+(b_2-y_1)^2-l_2^2. \tag{A.12}$$

Equation (A.9) is a second order equation i.e. that has two different solution and as explained in chapter 5 only the one with the lower value for z has been kept. Below the C++ code of the forward kinematic has been reported.

```
#include "Kinematic.hpp"
#include <cmath>

using namespace eeduro::delta;
using namespace eeros::math;

const double Kinematic::q_min = -1.75;
const double Kinematic::q_max = 0.85;

const double Kinematic::tcp_min = -0.02;
const double Kinematic::tcp_max = 0.02;
const double Kinematic::tcpz_min = -0.14;
const double Kinematic::tcpz_max = -0.05;

const double Kinematic::length_A(0.05);
const double Kinematic::length_B(0.1);
const double Kinematic::r_A(0.0525);
const double Kinematic::r_B(0.026);
const double Kinematic::r(r_A-r_B);
const double Kinematic::alpha1(0);
const double Kinematic::alpha2(2*M_PI/3);
const double Kinematic::alpha3(-2*M_PI/3);

const Matrix<3, 3> Kinematic::rotz1 = Matrix<3, 3>::createRotZ(Kinematic::alpha1);
const Matrix<3, 3> Kinematic::rotz2 = Matrix<3, 3>::createRotZ(Kinematic::alpha2);
const Matrix<3, 3> Kinematic::rotz3 = Matrix<3, 3>::createRotZ(Kinematic::alpha3);

Kinematic::Kinematic(const Vector3 offsetv) {
        offset = offsetv;
}

Kinematic::Kinematic() {
        offset.zero();

        Vector3 tcp;
        Vector3 q;
        q.zero();

        if (forward(q, tcp)) {
                offset = tcp;
        }
}

Kinematic::~Kinematic() { }

bool Kinematic::forward(const Vector3 q, Vector3& tcp) {
        Vector3 temp1;
        temp1(0) = r + length_A * cos(q(0));
        temp1(1) = 0;
        temp1(2) = length_A * sin(q(0));

        Vector3 temp2;
        temp2(0) = r + length_A * cos(q(1));
        temp2(1) = 0;
        temp2(2) = length_A * sin(q(1));

        Vector3 temp3;
        temp3(0) = r + length_A * cos(q(2));
        temp3(1) = 0;
        temp3(2) = length_A * sin(q(2));

        Vector3 pc1 = rotz1 * temp1;
        Vector3 pc2 = rotz2 * temp2;
        Vector3 pc3 = rotz3 * temp3;

        double rspher = length_B;

        double w1 = pc1.transpose() * pc1;
        double w2 = pc2.transpose() * pc2;
        double w3 = pc3.transpose() * pc3;

        double x1 = pc1(0);
        double y1 = pc1(1);
        double z1 = pc1(2);

        Vector3 vektordiff = pc2-pc1;
        double A = vektordiff(0);
        double B = vektordiff(1);
        double C = -vektordiff(2);
        double G = (w2-w1)/2;

        vektordiff = pc3-pc1;
        double D = vektordiff(0);
        double E = vektordiff(1);
        double F = -vektordiff(2);
        double H = (w3-w1)/2;

        double dnm = A*E-B*D;

        // x = (a1*z + b1)
        double a1 = (C*E-F*B)/dnm;
        double b1 = (G*E-H*B)/dnm;

        // y = (a2*z + b2)
        double a2 = (A*F - C*D)/dnm;
        double b2 = (A*H - G*D)/dnm;

        // a*z^2 + b*z + c = 0
        double a = a1*a1 + a2*a2 + 1;
        double b = 2*(a1*(b1-x1) + a2*(b2-y1) - z1);
        double c = -rspher*rspher + w1  +b1*(b1-2*x1) +b2*(b2-2*y1);

        double d = b*b - 4.0*a*c;

        // no solution exist
        if (d < 0) {
                return false;
        }
        else {
                // take only the solution with the lower value for Z
                double poseZ = (-b - sqrt(d))/(2*a);
                tcp(0) = a1*poseZ + b1 - offset(0);
                tcp(1) = a2*poseZ + b2 - offset(1);
                tcp(2) = poseZ - offset(2);
                return true;
        }
}
```

A.2 Inverse Kinematic.

As described in chapter 5 the main idea in order to solve inverse kinematic is to find the intersection of two circle of equation(4.2):

$$\begin{cases}(x_{ji}-(R-r))^2+z_{ji}^2=l_1^2,\\ (x_{ji}-x_i)^2+y_{i^2}+(z_{ji}-z_i)^2=l_2^2.\end{cases} \tag{A.13}$$

In order to solve system(A.13) and to find the $\boldsymbol{\theta}=[\theta_1 \quad \theta_2 \quad \theta_3]^T$ vector that all satisfies the specific travel plate position it is necessary at first to pre-multiply the TCP position by the inverse of the rotation matrix for each angle α_i. In this way the problem can be solved for each motor separately leading to three different system in the form(A.13). For the purpose of reducing variables the first equation has been subtracted from the latter leading to a second order equation in the variable x_{ji} and z_{ji} since $y_{ji} = 0$. Solving these equations, it can be obtained:

$$\begin{cases} x_{ji} = \frac{-B_i \pm \sqrt{D_i}}{2A_i} = \frac{((R-r)-b_i a_i) \pm \sqrt{l_1^2(1+b_i^2)-(b_i(R-r)+a_i)^2}}{1+b_i^2} \\ z_{ji} = b_i x_{ji} + a_i \end{cases} \tag{A.14}$$

where:

$$\begin{aligned} a_i &= \frac{x_i^2 + y_i^2 + z_i^2 + l_1^2 - l_2^2 - (R-r)^2}{2z_i} \\ b_i &= \frac{(R-r) - x_i}{z_i}, \\ A_i &= b_i^2 + 1, \\ B_i &= 2(a_i b_i + (R-r)), \\ C_i &= a_i^2 - l_1^2 + (R-r)^2, \\ D_i &= B_i^2 - 4A_i C_i; \end{aligned}$$

and remembering that (x_i, y_i, z_i) is the position of end-effector and (x_{ji}, y_{ji}, z_{ji}) is the position of point j_i in the i-th coordinate frame. Once the coordinate x_{ji} and z_{ji} are known it is easy to compute the angle $q(i)$ as:

$$q(i) = \tan^{-1} \frac{z_{ji}}{x_{ji} - (R-r)}. \tag{A.15}$$

Below the C++ code of the forward kinematic has been reported.

```
bool Kinematic::inverse(const Vector3 tcp, Vector3& q) {
        Vector3 tempTCP;

        tempTCP(0) = tcp(0) + offset(0);
        tempTCP(1) = tcp(1) + offset(1);
        tempTCP(2) = tcp(2) + offset(2);

        Vector3 tcp_1 = rotz1*tempTCP;
        Vector3 tcp_2 = rotz3*tempTCP;                              // invers rotz_2
        Vector3 tcp_3 = rotz2*tempTCP;                      // invers rotz_3
        double x1 = r;
        double LA_sqr = length_A*length_A;

        // 1. arm:

        double x0_1 = tcp_1(0);
        double y0_1 = tcp_1(1);
        double z0_1 = tcp_1(2);

        double a_1 = (x0_1*x0_1 + y0_1*y0_1 + z0_1*z0_1 +
                        LA_sqr - length_B*length_B - x1*x1)/(2*z0_1 );
        double b_1 = (x1-x0_1)/z0_1;

        double A_1 = (b_1*b_1 +1);
        double B_1 = 2*(a_1*b_1 - x1);
        double C_1 = x1*x1 +a_1*a_1 - LA_sqr;
        // discriminant
        double d_1 = B_1*B_1 - 4*A_1*C_1;

        // no solution exist
        if(d_1 < 0) {
                return false;
        }
        else {
                double xj_1 = (-B_1 + sqrt(d_1))/(2*A_1);
                // choosing outer point
                double zj_1 = a_1 + b_1*xj_1;
                // 1. joint coordinate
                q(0) = atan2(zj_1,(xj_1 - x1));

                // 2. arm:
                double x0_2 = tcp_2(0);
                double y0_2 = tcp_2(1);
                double z0_2 = tcp_2(2);

                double a_2 = (x0_2*x0_2 + y0_2*y0_2 + z0_2*z0_2 +
                                 LA_sqr - length_B*length_B - x1*x1)/(2*z0_2 );
                double b_2 = (x1-x0_2)/z0_2;

                double A_2 = (b_2*b_2 +1);
                double B_2 = 2*(a_2*b_2 - x1);
                double C_2 = x1*x1 +a_2*a_2 - LA_sqr;
                // discriminant
                double d_2 = B_2*B_2 - 4*A_2*C_2;

                // no solution exist
                if(d_2 < 0) {
                        return false;
                }

                else {
                        double xj_2 = (-B_2 + sqrt(d_2))/(2*A_2);
                        // choosing outer point
                        double zj_2 = a_2 + b_2*xj_2;

                        // 2. joint coordinate
                        q(1) = atan2(zj_2,(xj_2 - x1));

                        // 3. arm:
                        double x0_3 = tcp_3(0);
                        double y0_3 = tcp_3(1);
                        double z0_3 = tcp_3(2);

                        double a_3 = (x0_3*x0_3 + y0_3*y0_3 + z0_3*z0_3 +
                                         LA_sqr - length_B*length_B - x1*x1)/(2*z0_3 );
                        double b_3 = (x1-x0_3)/z0_3;

                        double A_3 = (b_3*b_3 +1);
                        double B_3 = 2*(a_3*b_3 - x1);
                        double C_3 = x1*x1 +a_3*a_3 - LA_sqr;
                        // discriminant
                        double d_3 = B_3*B_3 - 4*A_3*C_3;

                        // no solution exist
                        if(d_3 < 0) {
                                return false;
                        }
                        else {
                                double xj_3 = (-B_3 + sqrt(d_3))/(2*A_3);
                                // choosing outer point
                                double zj_3 = a_3 + b_3*xj_3;

                                // 3. joint coordinate
                                q(2) = atan2(zj_3,(xj_3 - x1));

                                return true;
                        }
                }
        }
}

const eeros::math::Vector3& Kinematic::get_offset() {
        return offset;
}
```

Appendix B

Dynamic, code and details.

B.1 Jacobian.

Below the C++ code for the evaluation of the jacobian matrix has been reported.

```
#include "Jacobian.hpp"
#include "Kinematic.hpp"
#include <cmath>

using namespace eeduro::delta;
using namespace eeros::math;

Jacobian::Jacobian(eeros::math::Vector3 offset) : offset(offset) { }

Jacobian::~Jacobian() { }

Vector3 Jacobian::getCartesianVelo(const Vector3& q, const Vector3& tcp,
                                const Vector3& jointvelo) {
        calculate(q,tcp);
        return jacobi * jointvelo;
}
Vector3 Jacobian::getJointVelo(const Vector3& q, const Vector3& tcp,
                                const Vector3& cartesianvelo) {
        calculate(q,tcp);
        return !jacobi * cartesianvelo;
}
Vector3 Jacobian::getDrivetorque(const Vector3& q, const Vector3& tcp,
                                const Vector3& F_tcp) {
        calculate(q, tcp);
        return getDrivetorque(F_tcp);
}
Vector3 Jacobian::getDrivetorque(const Vector3& F_tcp) {
        return jacobi.transpose() * F_tcp;
}
Vector3 Jacobian::getDriveforce(const Vector3& torque_tcp){
         return !jacobi.transpose() * torque_tcp;
}

bool Jacobian::calculate(const Vector3& q, const Vector3& tcp_offset) {
        Vector3 tcp = (tcp_offset + offset);
        Vector3 temp1;
        temp1(0) = Kinematic::r+Kinematic::length_A*cos(q(0));
        temp1(1) = 0;
        temp1(2) = Kinematic::length_A*sin(q(0));

        Vector3 temp2;
        temp2(0) = Kinematic::r+Kinematic::length_A*cos(q(1));
        temp2(1) = 0;
        temp2(2) = Kinematic::length_A*sin(q(1));

        Vector3 temp3;
        temp3(0) = Kinematic::r+Kinematic::length_A*cos(q(2));
        temp3(1) = 0;
        temp3(2) = Kinematic::length_A*sin(q(2));

        Vector3 s1 = tcp - Kinematic::rotz1*temp1;
        Vector3 s2 = tcp - Kinematic::rotz2*temp2;
        Vector3 s3 = tcp - Kinematic::rotz3*temp3;

        Vector3 temp2_1;
        temp2_1(0) = -Kinematic::length_A*sin(q(0));
        temp2_1(1) = 0;
        temp2_1(2) = Kinematic::length_A*cos(q(0));

        Vector3 temp2_2;
        temp2_2(0) = -Kinematic::length_A*sin(q(1));
        temp2_2(1) = 0;
        temp2_2(2) = Kinematic::length_A*cos(q(1));

        Vector3 temp2_3;
        temp2_3(0) = -Kinematic::length_A*sin(q(2));
        temp2_3(1) = 0;
        temp2_3(2) = Kinematic::length_A*cos(q(2));

        Vector3 b1 = Kinematic::rotz1*temp2_1;
        Vector3 b2 = Kinematic::rotz2*temp2_2;
        Vector3 b3 = Kinematic::rotz3*temp2_3;

        Matrix<3,3> tempA;
        tempA(0,0) = s1.transpose()*b1;
        tempA(1,0) = 0;
        tempA(2,0) = 0;

        tempA(0,1) = 0;
        tempA(1,1) = s2.transpose()*b2;
        tempA(2,1) = 0;

        tempA(0,2) = 0;
        tempA(1,2) = 0;
        tempA(2,2) = s3.transpose()*b3;

        Matrix<3,3> tempS;
        tempS(0,0) = s1(0);
        tempS(1,0) = s2(0);
        tempS(2,0) = s3(0);

        tempS(0,1) = s1(1);
        tempS(1,1) = s2(1);
        tempS(2,1) = s3(1);

        tempS(0,2) = s1(2);
        tempS(1,2) = s2(2);
        tempS(2,2) = s3(2);

        if(!tempS.isInvertible()) {
                return false;
        }
        Matrix<3,3> tempSi;   // inverse tempS
        tempSi = !tempS;

        jacobi = tempSi*tempA;
        return true;
}
Matrix<3,3> Jacobian::getJacobian() {
        return jacobi;
}
Matrix<3,3> Jacobian::getJacobianInv() {
```

```
          if (jacobi.isInvertible()){
  return !jacobi;}
}
```

B.2 Inertia.

In this work the dynamic model of the system has been approximated only through the inertia matrix and the gravity compensation. The inertia matrix have been computed in the cartesian space and the gravity in joint space and thats why there are two different functions for the computations. The inertia as explained in section (4.3) is described by equation $B(\theta) = (I_b + m_{nt}J^T J)$ thus for the computation the output of the jacobian are needed. Since the calculation happened in the cartesian space also the values of accelerations are needed in order to have the values of the forces as the first equation of dynamics explain $F = ma$.

```
#include "Inertia.hpp"
#include "constants.hpp"
#include <iostream>
using namespace eeduro::delta;
using namespace eeros::math;
using namespace eeros::control;

Inertia::Inertia(Jacobian &jacobi) : jacobi(jacobi) {
        tcpMass = 0;
        for (int i = 0; i < 3; i++)
                tcpMass(i,i) = mtcp;

        motorInertia = 0;
        for (int i = 0; i < 3; i++)
                motorInertia(i,i) = jred;
}
void Inertia::run() {
        Vector3 q012, xyz;

        q012 = jointPosIn.getSignal().getValue().getSubMatrix<3,1>(0, 0);
        xyz = tcpPosIn.getSignal().getValue().getSubMatrix<3,1>(0, 0);
        Vector3 a = accelerationIn.getSignal().getValue().getSubMatrix<3,1>(0, 0);
        double qdd = accelerationIn.getSignal().getValue()[3];

        if (jacobi.calculate(q012, xyz)) {
                Matrix<3,3> jacobian = jacobi.getJacobian();
                if (jacobian.isInvertible()) {
                        Matrix<3,3> inverseJacobian = !jacobian;
                        Matrix<3,3> inverseJacobianTransposed = inverseJacobian.transpose();

                        Matrix<3,3> M = tcpMass + inverseJacobianTransposed *
                                                motorInertia * inverseJacobian;
                        Vector3 F = M * a;
                        double tau = jred * qdd; // magnet axis

                        vector_inertia = M(0,0), M(1,1), M(2,2);

                        forceOut.getSignal().setValue(Vector4(F[0], F[1], F[2], tau));
                        forceOut.getSignal().setTimestamp(accelerationIn.getSignal().
                                                getTimestamp());
                        return;
                }
        }
        forceOut.getSignal().setValue(0);
        forceOut.getSignal().setTimestamp(accelerationIn.getSignal().getTimestamp());
}
Matrix<3,1> Inertia::getInertia() {
        return vector_inertia;
}
Input<AxisVector>& Inertia::getAccelerationInput() {
        return accelerationIn;
}
```

```
Input<AxisVector>& Inertia::getTcpPosInput() {
        return tcpPosIn;
}
Input<AxisVector>& Inertia::getJointPosInput() {
        return jointPosIn;
}
Output<AxisVector>& Inertia::getOut() {
        return forceOut;
}
```

B.3 Gravity.

The computation of the gravity compensation's torque has been described in section (??) and has been computed in joint space. For this reason the torque is just added to the values of torque computed by the PID controller and that it is necessary to fulfill the task.

```
#include "Gravity.hpp"
#include "constants.hpp"
#include <iostream>
using namespace eeros;
using namespace eeros::control;
using namespace eeros::math;
using namespace eeduro::delta;

const double Gravity::length_A(0.05);
const double Gravity::m_br(0.013);
const double Gravity::m_c(0.003);
const double Gravity::m_ab(0.005);
const double Gravity::m_n(0.0453);
const double Gravity::r(2/3);

Gravity::Gravity(Jacobian& j): jacobi(j) { }

Gravity::~Gravity() { }

void Gravity::run() {
        AxisVector variable_theta = jointPosIn.getSignal().getValue();
        Vector3 q012, xyz;

        q012 = jointPosIn.getSignal().getValue().getSubMatrix<3,1>(0, 0);
        xyz = tcpPosIn.getSignal().getValue().getSubMatrix<3,1>(0, 0);
        double theta_4 =  jointPosIn.getSignal().getValue()[3];

        Vector3 grav;
        grav(0) = 0;
        grav(1) = 0;
        grav(2) = -g;

        if (enabled){
                double m_b = m_br + m_c + r*m_ab;
                double r_Gb = length_A * ((0.5*m_br + m_c + r*m_ab)/m_b);
                double m_nt = m_n +3*(1-r)*m_ab;
                Vector3 tau_Gb;

                tau_Gb(0) = m_b* r_Gb * g *cos(q012(0));
                tau_Gb(1) = m_b* r_Gb * g *cos(q012(1));
                tau_Gb(2) = m_b* r_Gb * g *cos(q012(2));

                                if (jacobi.calculate(q012, xyz)) {
                        Matrix<3,3> jacobian = jacobi.getJacobian();
                                Matrix<3,3> JacobianTransposed = jacobian.transpose();
                                Vector3 temp = -m_nt*JacobianTransposed*grav;
                                torque_tot(0) = temp(0)- tau_Gb(0);
                                torque_tot(1) = temp(1)- tau_Gb(1);
                                torque_tot(2) = temp(2)- tau_Gb(2);
                                torque_tot(3) = 0;

                                out_torque.getSignal().setValue(torque_tot);
                                out_torque.getSignal().setTimestamp(jointPosIn.getSignal().
                                                                        getTimestamp());
                                return;
                }
                else{
                  out_torque.getSignal().setValue(0);
                  out_torque.getSignal().setTimestamp(jointPosIn.getSignal().getTimestamp());
                }
        }
        else{
                  out_torque.getSignal().setValue(0);
                  out_torque.getSignal().setTimestamp(jointPosIn.getSignal().getTimestamp());
        }
}
Input<AxisVector>& Gravity::getTcpPosInput() {
        return tcpPosIn;
}
Input<AxisVector>& Gravity::getJointPosInput() {
        return jointPosIn;
}
Output<AxisVector>& Gravity::getTorqueOut() {
        return out_torque;
}
void Gravity::enable() {
        this->enabled = true;
}
void Gravity::disable() {
        this->enabled = false;
}
```

Appendix C

Stability of the PD controller.

In section (6.2) the PD controller scheme has been described in a mechanical approach. In this section a more theoretical analysis will be made. In fact the control law has to ensure global asymptotic stability around a constant equilibrium configuration θ_d for the manipulator of equation:

$$B(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} + g(\theta) = \tau. \tag{C.1}$$

This will be made exploiting the Lyapunov method.[16] The idea is to compensate the gravity and then to find a control law that ensures a minimum in the potential energy of the pose θ_d, due to the fact that manipulators tend to move into the configuration of minimum potential energy. Considering the new variable $\tilde{\theta} = \theta_d - \theta$ it is possible to define a new state:

$$\begin{bmatrix} \tilde{\theta} \\ \dot{\theta} \end{bmatrix}$$

and considering this new state the asymptotical stability of configuration θ_d can be studied analyzing the stability of the origin in new coordinates. Taking as candidate Lyapunov function the following:

$$V(\dot{\theta}, \tilde{\theta}) = \frac{1}{2}\dot{\theta}^T B(\theta)\dot{\theta} + \frac{1}{2}\tilde{\theta} K_p \tilde{\theta} > 0 \qquad \forall \dot{\theta}, \tilde{\theta} \neq 0 \tag{C.2}$$

where the condition that K_p is definite positive leads to a positive definite function $V(\dot{q}, \tilde{\theta}) > 0$ ($B(\theta)$ is also definite positive). This equation can be seen as the sum of two different contribution:

- $\frac{1}{2}\dot{\theta}^T B(\theta)\dot{\theta} \quad \Longrightarrow$ the kinetic energy of the system;
- $\frac{1}{2}\tilde{\theta} K_p \tilde{\theta} \quad \Longrightarrow$ potential elastic energy set up to the system through the control action.

Differentiating equation (C.2) with respect to time, and recalling that θ_d is constant ($\dot{\tilde{\theta}} = -\dot{\theta}$), yields:

$$\dot{V}(\dot{\theta}, \tilde{\theta}, \ddot{\theta}) = \dot{\theta}^T B(\theta)\ddot{\theta} + \frac{1}{2}\dot{\theta}^T \dot{B}(\theta)\dot{\theta} - \dot{\theta}^T K_p \tilde{\theta}. \tag{C.3}$$

The dynamic model of a general system given by a manipulator and drives is described by [16]:

$$B(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} + F\dot{\theta} + g(\theta) = u \tag{C.4}$$

where given F_v the matrix of the viscous friction's coefficients:

$$F = F_v + k_r k_t R_a^{-1} k_v k_r \tag{C.5}$$

$$u = k_r k_t R_a^{-1} G_v v_c; \tag{C.6}$$

and the variables in (C.6) described in (C.1) are combined in the following formula:

$$k_r^{-1}\tau = k_t i_a, \tag{C.7}$$

$$v_a = R_a i_a + k_v \dot{\theta_m}, \tag{C.8}$$

$$v_a = G_v v_c, \tag{C.9}$$

$$k_r \theta = \theta_m; \tag{C.10}$$

and represent the equations of the driving system and the gearing transmission systems. Given the dynamic

Variable	**Meaning**
k_t	Diagonal matrix of torque constants.
k_r	Matrix of the reduction gear ratio.
R_a	Diagonal matrix of armature resistances.
k_v	Diagonal matrix of voltage constans of the motors.
G_v	Diagonal matrix of gains of the amplifers.
v_c	Vector of control voltages of the servomotors.
v_a	Vector of armature voltages.

Table C.1: Variables in actuators and transmission.

model of a general system by (C.4) it is possible to obtain the values of $B\ddot{\theta}$ and substituting in (C.3) the following holds:

$$\dot{V} = \frac{1}{2}\dot{\theta}^T(\dot{B}(\theta) - 2C(\theta, \dot{\theta}))\dot{\theta} - \dot{\theta}^T F\dot{\theta} + \dot{\theta}^T(u - g(\theta) - K_p\tilde{\theta}). \tag{C.11}$$

The first term on the right-hand side is null since the matrix $(\dot{B}(\theta) - 2C(\theta, \dot{\theta}))$ is a skew-symmetric matrix. The second term is negative definite. Then, the choice:

$$u = g(\theta) + K_p\tilde{\theta} - K_d\dot{\theta} \tag{C.12}$$

describing controller with compensation of gravitational terms, a proportional action and a derivative action, leads to a negative semi-definite Lyapunov function $\dot{V}$ (K_d is positive definite). In fact:

$$\dot{V} = -\dot{\theta}^T K_d\dot{\theta} \leq 0. \tag{C.13}$$

Since $\dot{V} \leq 0$ in order to establish the asymptotic stability of the equilibrium configuration it is necessary to verify that the only trajectory of the system that gives back $\dot{V} = 0$ is the equilibrium pose $\tilde{\theta} = 0$. Given equation (C.13) it can be seen that when $\dot{V} = 0$ it is $\dot{\theta} = 0$ and $\ddot{\theta} = 0$. Then the only trajectory which is compatible is:

$$K_p\tilde{\theta} = 0$$

i.e. $\tilde{\theta} = 0$ remembering that K_p is definite positive. Then it has been shown that the equilibrium state is characterized by $[\tilde{\theta} \quad \dot{\theta}]^T = [0 \quad 0]^T$ i.e. the manipulator is stationary with null position error and the system is asymptotically globally stable.

Appendix D

Motor's model code and datasheet.

```
#include "MotorModel.hpp"

using namespace eeduro::delta;
using namespace eeros::control;

MotorModel::MotorModel(const AxisVector kM, const AxisVector RA) : kM(kM), RA(RA) { }

Input<AxisVector>& MotorModel::getSpeedIn() {
        return speed;
}

Input<AxisVector>& MotorModel::getTorqueIn() {
        return torque;
}

Output<AxisVector>& MotorModel::getOut() {
        return voltage;
}

void MotorModel::run() {
        AxisVector u, M, w;
        M = torque.getSignal().getValue();
        w = speed.getSignal().getValue();
        for(unsigned int i = 0; i < u.size(); i++) {
                u[i] = RA[i] * M[i] / kM[i] + w[i] * kM[i];
        }
        voltage.getSignal().setValue(u);
        voltage.getSignal().setTimestamp(torque.getSignal().getTimestamp());
}
```

DC-Micromotors

Precious Metal Commutation

2,9 mNm

5,3 W

Series 1524 ... SR

Values at 22°C and nominal voltage	1524 T	003 SR	006 SR	009 SR	012 SR	018 SR	024 SR	
1 Nominal voltage	U_N	3	6	9	12	18	24	V
2 Terminal resistance	R	1,1	5,1	10,6	19,8	43,9	79,3	Ω
3 Efficiency, max.	$\eta_{max.}$	80	80	80	80	80	80	%
4 No-load speed	n_0	10 600	9 500	10 000	9 800	9 800	9 800	min^{-1}
5 No-load current, typ. (with shaft ø 1,5 mm)	I_0	0,03	0,013	0,009	0,007	0,005	0,004	A
6 Stall torque	M_H	6,95	6,98	7,18	6,92	7,07	6,91	mNm
7 Friction torque	M_R	0,08	0,08	0,08	0,08	0,08	0,08	mNm
8 Speed constant	k_n	3 577	1 592	1 117	827	548	414	min^{-1}/V
9 Back-EMF constant	k_E	0,28	0,628	0,895	1,21	1,83	2,42	mV/min^{-1}
10 Torque constant	k_M	2,67	6	8,55	11,5	17,4	23,1	mNm/A
11 Current constant	k_I	0,374	0,167	0,117	0,087	0,057	0,043	A/mNm
12 Slope of n-M curve	$\Delta n/\Delta M$	1 530	1 350	1 380	1 420	1 380	1 420	min^{-1}/mNm
13 Rotor inductance	L	22	110	230	420	950	1 670	µH
14 Mechanical time constant	τ_m	8,5	8,2	8,3	8,3	8,2	8,3	ms
15 Rotor inertia	J	0,53	0,58	0,57	0,56	0,57	0,56	gcm^2
16 Angular acceleration	$\alpha_{max.}$	131	120	126	124	124	123	$\cdot 10^3 rad/s^2$
17 Thermal resistance	R_{th1} / R_{th2}	10 / 29						K/W
18 Thermal time constant	τ_{w1} / τ_{w2}	5,6 / 220						s
19 Operating temperature range:								
– motor		-30 ... +85 (optional version -55 ... +125)						°C
– winding, max. permissible		+125						°C
20 Shaft bearings		sintered bearings			ball bearings, preloaded			
21 Shaft load max.:		(standard)			(optional version)			
– with shaft diameter		1,5			1,5			mm
– radial at 3 000 min^{-1} (3 mm from bearing)		1,2			5			N
– axial at 3 000 min^{-1}		0,2			0,5			N
– axial at standstill		20			10			N
22 Shaft play:								
– radial	≤	0,03			0,015			mm
– axial	≤	0,2			0			mm
23 Housing material		steel, black coated						
24 Mass		18						g
25 Direction of rotation		clockwise, viewed from the front face						
26 Speed up to	$n_{max.}$	13 000						min^{-1}
27 Number of pole pairs		1						
28 Magnet material		NdFeB						

Rated values for continuous operation								
29 Rated torque	M_N	1,7	2,9	2,9	2,9	2,9	2,9	mNm
30 Rated current (thermal limit)	I_N	0,7	0,56	0,38	0,28	0,19	0,14	A
31 Rated speed	n_N	7 800	3 860	4 500	4 130	4 330	4 110	min^{-1}

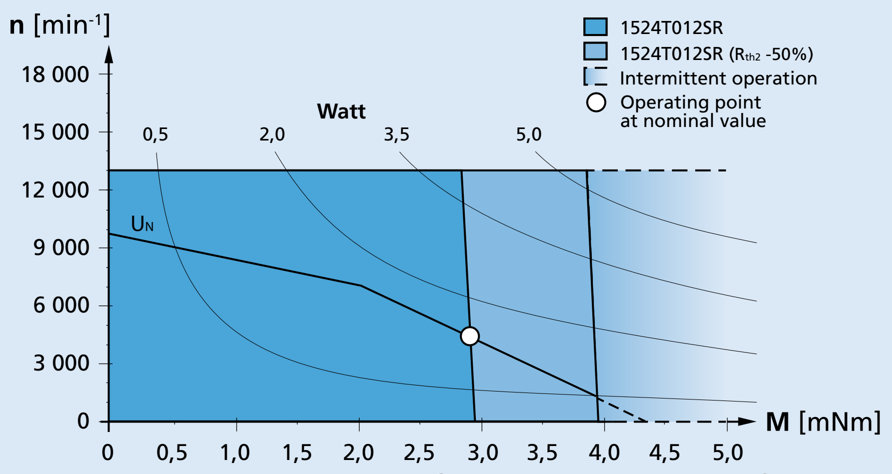
Note: Rated values are calculated with nominal voltage and at a 22°C ambient temperature. The R_{th2} value has been reduced by 0%.

Note:

The diagram indicates the recommended speed in relation to the available torque at the output shaft for a given ambient temperature of 22°C.

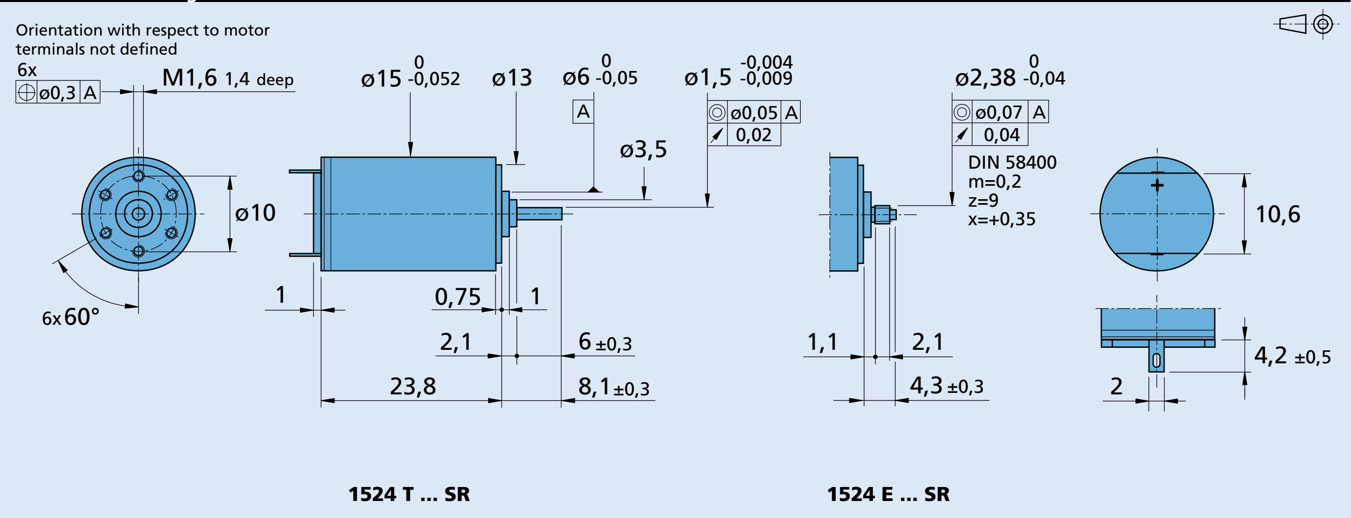
The diagram shows the motor in a completely insulated as well as thermally coupled condition (R_{th2} 50% reduced).

The nominal voltage (U_N) curve shows the operating point at nominal voltage in the insulated and thermally coupled condition. Any points of operation above the curve at nominal voltage will require a higher operating voltage. Any points below the nominal voltage curve will require less voltage.



Recommended operation areas (example: nominal voltage 12V)

Dimensional drawing



Options

Example product designation: **1524T012SR-277**

Option	Type	Description
L	Twin Leads	For motors with twin leads (PVC), length 150 mm, red (+) / black (-)
4924	Twin Leads	For motors with twin leads (PVC), length 300 mm, red (+) / black (-)
X4924	Twin Leads	For motors with twin leads (PVC), length 600 mm, red (+) / black (-)
4925	Twin Leads	For motors with twin leads (PVC), length 150 mm, red (+) / black (-), with connector AMP 179228-2
X4925	Twin Leads	For motors with twin leads (PVC), length 300 mm, red (+) / black (-), with connector AMP 179228-2
Y4925	Twin Leads	For motors with twin leads (PVC), length 600 mm, red (+) / black (-), with connector AMP 179228-2
F	Single Leads	For motors with single leads (PTFE), length 150 mm, red (+) / black (-)
277	Bearings	2 preloaded ball bearings

Product combination

Precision Gearheads / Lead Screws	Encoders	Drive Electronics	Cables / Accessories
15A	IE2-16	SC 1801	To view our large range of accessory parts, please refer to the "Accessories" chapter.
15/5	IE2-1024	MC 5004	
15/5 S	IEH2-4096	MCDC 3002	
15/8	IEH3-4096		
15/10			
16A			
16/7			

© DR. FRITZ FAULHABER GMBH & CO. KG
Specifications subject to change without notice.

Bibliography

[1] Marcia K. O'malley , Abhishek Gupta *HCI: Beyond the GUI* chapter: Haptic Interfaces, Morgan- Kaufman, 2007.

[2] W. Provancher. *Haptics and Embedded Mechatronics Laboratory*, University of Utah, Salt Lake City, Utah, 2014.

[3] Melchiorri Claudio *Robotic telemanipulation systems: an overview on control aspects*, University of Bologna, Italy, 2003.

[4] Nevio Benvenuti and Michele Zorzi *Communications Networks and Systems*, John Wiley & Sons Ltd, 2011.

[5] Andrea Claudi, Aldo Franco Dragoni, *Testing Linux-based real-time systems: Lachesis* Service-Oriented Computing and Applications (SOCA), 2011 IEEE International Conference.

[6] https://www.netlab.tkk.fi/opetus/s38130/s98/tcp/TCP.htm

[7] Elkady, Ayssam. *Robotics Middleware: A Comprehensive Literature Survey and Attribute-Based Bibliography.* Journal of Robotics. 2012.

[8] André Olsson, *Modeling and control of a Delta-3 robot* Department of Automatic Control Lund University February 2009

[9] John J. Craig *Introduction to Robotics. Mechanics and control.* Pearson Education International, 2005.

[10] Siciliano, Kathib *Springer Handbook of Robotics* Springer, 2008.

[11] Liang Yan, Delong Liu, Zongxia Jiao *Novel Design and Kinematics Modeling for Delta Robot with Improved End Effector* Industrial Electronics Society , IECON 2016 - 42nd Annual Conference of the IEEE

[12] Alain Codourey *Dynamic Modelling and Mass Matrix Evaluation of the DELTA Parallel Robot for Axes Decoupling Control* Intelligent Robots and Systems '96 IROS 96.

[13] Bruno Siciliano, Luigi Villani *Robot Force Control* Kluwer Academic Publishers, Boston/Dordrecht/London,1999.

[14] http://www6.in.tum.de/burschka/courses/robotics/aufgaben/solution05.pdf.

[15] Einar Nielsen Skript "Roboterkinematik-4", NTB Interstaatliche Hochschule für Technik Buchs, 2014.

[16] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, Giuseppe Oriolo *Robotica, modellistica, pianificazione, controllo* McGraw-Hill, 2008.

[17] Riccardo Muradore, Paolo Fiorini *A review of bilateral teleoperation algorithms* Acta Polytechnica Hungarica, Vol. 13, No.1, 2016.

[18] D. Lee, M.W.Spong *Passive bilateral teleoperation with constant time delay*, Robotics, IEEE Transactions on, vol. 22 no.2, pp.269-281, 2006.

[19] Hashtrudi-Zaad K., Salcudean S.E. *Bilateral parallel force/position teleoperation control.*, Journal of Robotic Systems, pp. 155-167, 2002.

[20] Hashtrudi-Zaad K., Salcudean S.E. *Transparency in time-delayed systems and effect of local force feedback for transparent teleoperation.* IEEE Transactions on Robotics and Automation, pp. 108-114, 2002.

[21] Seiichiro Katsura, Wataru Iida, Kouhei Ohnishi *Medical mechatronics- An application to haptic forceps.* Annual Reviews in Control, Vol. 29, pp. 237-245, 2005.

[22] D. A. Lawrence *Designing Teleoperator Architectures for Transparency* Proc. IEEE Conf. Rob. Auto., pp. 1406-1411, 1992.

[23] Asif Sabanovic, Kouhei Ohnishi *Motion Control Systems* John Wiley & Sons (Asia) Pte Ltd., 2011.

[24] Wataru Iida, Kouhei Ohnishi *Reproducibility and Operationality in Bilateral Teleoperation* roc. of the 8th IEEE Int. Workshop 0.004 on Advanced Motion Control, pp. 217-222, 2004.

[25] Suzuki Atsushi *Acceleration-Based Bilateral Teleoperation System under Time Delay Based on Modal Space Analysis*, Keio University, 2013.

[26] Ugur Tumerdem, Kouhei Ohnishi *Robust Four Channel Teleoperation under Time Delay by Damping Injection* Proceedings of the 2009 IEEE International Conference on Mechatronics. Malaga, Spain, April 2009.

[27] K. Hashtrudi-Zaad and S. E. Salcudean, *Analysis and evaluation of stability and performance robustness for teleoperation control architectures* IEEE International Conference on Robotics and Automation, San Francisco, pp. 3107-3113, 2000.

[28] Kouhei Ohnishi, Masaaki Shibata, Toshiyuki Murakami *Motion Control for Advanced Mechatronics* IEEE/ASME Transactions on Mechatronics, Volume 1, March 1996.

[29] Ian Reid *Estimation 2* Hilary Term, 2001.

[30] www.eeros.org.

[31] Chowarit Mitsantisuk, Sorawit Stapornchaisit, Nakhon Niramitvasu, Kiyoshi Ohishi *Force Sensorless Control with 3D workspace analysis for haptic devices based on delta robot* Industrial Electronics Society, IECON, Yokohama, 2015.