Treball de Fi de Màster

Master's degree in Automatic Control and Robotics

Supervision of an Humanoid Robot

MEMÒRIA

Autor: Giulia Angarano
Directors: Dr. Vicenç Puig
Dr. Guillem Alenyà

Convocatòria: July 2017

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Abstract

Robots are physical systems with varying degrees of autonomy that operate in different and dynamic physical environments. Their use in our daily lives is increasing, as it is appealing for tasks that can be referred to as the four Ds —too Dangerous, too Dull, too Dirty, and too Difficult— to be done by humans.

Nevertheless, robotic systems are prone to different types of faults, which have the potential to affect the efficiency and the safety of the robot and/or its surroundings. For these reasons, FDD (Fault Detection and Diagnosis) techniques are nowadays essential in robotics, with the aim of facilitating the system recovery.

Based on such considerations, this thesis addresses the problem of supervision of a humanoid robot, specifically focusing on its head.

With this scope in mind, the robotic system has been modelled and controlled by means of a linear parameter varying (LPV) feedback controller.

Hence, a fault detection and isolation scheme has been implemented using the LPV approach. Such a method has been selected as the one to be followed as it encompasses the performance requirements a humanoid robot implies: it has to detect faults quickly, online and with a low computational burden, according to expectations autonomously generated.

Later, a fault tolerant scheme has been designed to compensate the faulty effect, once the fault is detected and isolated.

Lastly, all the above-mentioned schemes have been tested in simulation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Robots are physical systems with varying degrees of autonomy that operate in different and dynamic physical environments, e.g., satellites, Martian rovers, unmanned aerial, ground, or underwater vehicles.

The use of robots in our daily lives is increasing. Based on data published in World Robotics [1], Guizzo [2] points to an increase in the world robot population from 4.5 million in 2006 to 8.6 million in 2008. Recent reports [3,4] by the International Federation of Robotics (IFR) from 2016 describe yearly increases of 15% and 25% in sales of service and industrial robots, respectively. The use of robots is appealing for tasks that can be referred to as the four Ds —too Dangerous, too Dull, too Dirty, and too Difficult— to be done by humans.

Indeed, in a general sense, a robotic system is engaged in three main processes [9]: sensing, thinking, and acting.

The local environment and the robot's own body are sensed by different sensors. The "thinking" component involves (a) the extraction of information from the sensor-data, and (b) the use of acquired knowledge to decide on a course of action(s) to achieve the goal(s). Then, according to the decisions made by the robot, instructions are issued to different actuators that, in turn, act and affect the robot and its local environment. These effects are sensed by the robot's sensors and this cycle continues until the desired

goals are achieved.

However, like any physical system, these intricate and sometimes expensive machines are prone to different types of faults such as wear and tear, noise, or control failures [5]. In the robotics domain, faults have the potential to affect the robot's efficiency, cause failures, or even jeopardize the safety of the robot or its surroundings [6]. Hence, when a fault is detected, it is imperative to proceed with a diagnosis process to identify which internal components are involved, aiming at facilitating the system recovery [7].

1.2 Thesis objectives

In this master thesis, the problem of supervision of an humanoid robot has been addressed. Specifically, the mobile manipulator robot TIAGo by PAL robotics has been taken as reference and the focus has been put on its head.

In order to achieve the above-mentioned objective, the following tasks have to be carried out:

- by considering the robotic head as a 2-dof manipulator, a dynamic model of the robot head has to be developed by means of the Newton-Euler algorithm;
- a control scheme has to be designed, with the following characteristics:
  - it has to be able to make the robot head reaching a reference orientation given in terms of the two joints angles;
  - it has to account for the real sensors availability on the robot head i.e. for the lack of some of them;
- a fault detection and isolation scheme has to be implemented;
- a fault tolerant control scheme has to be implemented to compensate the isolated faulty effect.

Figure 1.1: *TIAGo robot* [10]

1.3 Thesis outline

The present work is structured as follows.

CHAPTER 2 - State of the art

In this chapter, an overview on the currently mostly used approaches in the field of Fault Detection and Diagnosis (FDD) are pointed out, together with their applications in robotics.

CHAPTER 3 - Robot head model

This chapter deals with the development of the robot head dynamic model, by means of the Newton-Euler approach, whose resulting equations have been verified by imple-

menting the Newton-Euler algorithm in MATLAB, through the use of the Robotics Toolbox.

Moreover, the "energy based" Lagrange approach has been sketched out.

CHAPTER 4 - LPV Model

Starting from the equations of motion obtained in Chapter 3, in this chapter the robot head model is reformulated as an LPV one.

CHAPTER 5 - LPV Control

This chapter covers the development of an LPV controller for the above-mentioned model, whose implementation implies the states full observability. Hence, the construction of an LPV observer is presented to overcome such unrealistic assumption.

CHAPTER 6 - Estimation fault/disturbance

In this chapter, an improvement of the previously explained estimation scheme is pointed out: a LPV UIO is proposed, so that not only the system unobservable states can be estimated, but also eventual unknown faults or disturbances acting on the system.

CHAPTER 7 - Simulation results

This chapter is dedicated to showing the simulations results obtained from the MATLAB and Simulink implementation of the schemes (fault tolerant and not) described in the previous chapters.

CHAPTER 8 - Socioeconomic impact

In this chapter, a brief discussion of possible impacts of this project on economy and society is presented.

CHAPTER 9 - Project budget

In this chapter, some considerations about the development and implementation of the system presented in this work are provided from an economic point of view.

CHAPTER 10 - Conclusions and future work

Lastly, some final remarks are presented, as well as some suggestions about possible future follow-up.

Chapter 2

State of the art

The use of robots in our daily lives is increasing as they can be used to realize those tasks that can be referred to as the four Ds —too Dangerous, too Dull, too Dirty, and too Difficult— to be done by humans.

However, like any physical system, these machines are prone to different types of faults, which have the potential to affect the robot's efficiency, cause failures, or even jeopardize the safety of the robot or its surroundings [5, 6]. Hence, when a fault is detected, it is imperative to proceed with a detection and diagnosis process, aimed at facilitating the system recovery.

This diagnostic information can be used for recovery or for decision making purposes, such as using undamaged redundant systems or re-planning [7]. The ability to recover successfully allows the system to be reliable, robust, and efficient, which is ultimately the reason for applying Fault Detection and Diagnosis (FDD) approaches.

The field of FDD has been studied for many years, leading to three main categories of FDD approaches: data-driven, model-based, and knowledge-based.

Data-driven approaches are model free. Online data is usually used to statistically differentiate a potential fault from historically observed normal behavior, e.g., via Principle Component Analysis [11].

Model-based approaches [12] typically use analytical redundancy to detect and diagnose faults. The correct behavior of each component in the system is modeled analytically and the expected output can be compared to the observed output. Among

them, quantitative models involve mathematical equations, which typically describe the functionality of components. Qualitative models involve logic-functions that describe the behavior of components by representing qualitative relations between the observed variables.

Lastly, knowledge-based [13] approaches associate recognized behaviors with predefined known faults and diagnoses.

Pros and cons of the above-mentioned categories are summarized in Fig.2.1 [8].

Approaches	Advantages	Challenges
Data-driven	Model free Detection of unknown faults	Being quick and online
Model-based	Quick & Online Detection of unknown faults	Modeling complex robotic systems Modeling context (especially environment)
Knowledge-based	Quick & Online Low computational burden	Detection of unknown faults

Figure 2.1: *FDD approaches pros and cons table*

Although the study of FDD for robotics is relatively new, there are many FFD approaches which can be borrowed from other fields of study to be applied to robotic systems, depending on the type of the robotic system itself.

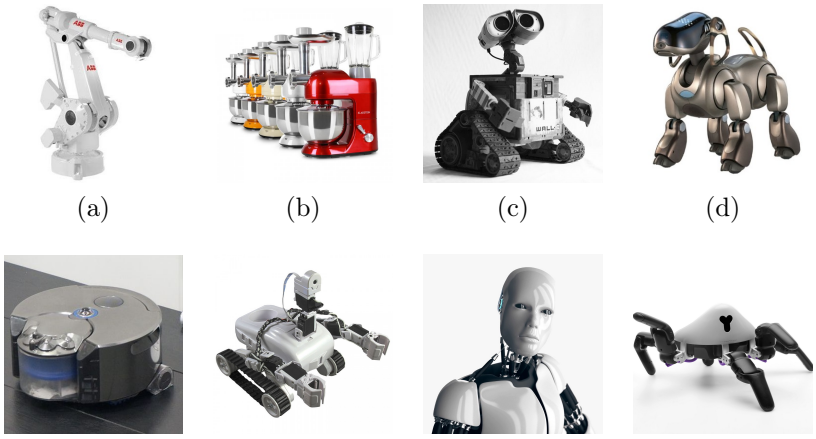
Indeed, nowadays there is a broad range of systems that can be considered of robotic type, characterized by a number of different features. It is actually such variety that makes challenging the application of the FDD techniques to robotic systems.

For the sake of completeness, the following table lists five key characteristics of robotic systems, together with the different FDD challenges they usually imply and the typical FDD approaches used to address them. The analysis of such relations would need a deeper discussion which is out of the scope of this work.

The interested reader is referred to [8] for a deeper understanding.

Nevertheless, as this thesis deals with the implementation of FDD techniques on a humanoid robot, a particular mention is worth giving to autonomy.

Robotic systems can range from being totally controlled (e.g., a robotic exoskeleton or suit) to systems with a high degree of autonomy. Fully autonomous robots do exist; some of these robots carry out a simple task (e.g., vacuum cleaning) or operate within a



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 2.2: *Some examples of robots*

very narrow context under the strict confines of their direct environment (e.g., a robotic arm in an assembly line).

Other more complex systems may have "a human in the loop" with varying degrees of control and monitoring (e.g., a rover on Mars).

A system is considered more autonomous when it is (a) less dependent on human operators and (b) remote supporting systems, and (c) able to operate for extensive periods of time without intervention.

These three attributes challenge FDD approaches in different ways. As the dependency on a human operator and remote support systems decreases, thus, it becomes the role of the robot to monitor itself and detect faults.

The first challenge is to be able to generate expectations using nothing else but the robot's own perceptions; there is no external source to compare with (e.g., human operator, ground radar station).

The next challenges for an FDD mechanism is to detect faults quickly and online. A robot has to continue its operation autonomously even in the presence of faults. This means that faults have to be quickly detected (and handled) as they occur, rather than retrospectively after the operation has ended (and the data of the entire operation is available).

The final challenge is to implement an FDD mechanism that requires low computa-

tional resources (CPU, Memory). Since there is less dependency on remote supporting systems, the FDD process has to be carried out on board the robot. The robot has a limited computational power, most of which is dedicated to mission oriented processes. A computationally demanding FDD process might interfere with other processes and ironically be the cause for faults.

The main difficulty in creating an FDD mechanism for an autonomous robot is to address all of these challenges together, i.e.,generating expectations, quickly, online, with a low computational burden.

Based on such considerations and making reference to Fig.2.1 it is straightforward inferring that one approach to address these challenges is with model-based diagnosis [14], [17].

Indeed, in this case, the diagnostic process relies on an explicit a priori model of the normal system behavior, its structure, and/or its known faults. Given the (online) system input (e.g., instructions), the model, made by a set of analytical equations or logical formulas, can quickly produce an expectation for the robot's behavior (e.g., expected sensor readings). Inconsistencies between the observed behavior and the produced expected behavior are suspected to be caused by faults.

Then, fault isolation is the process in charge of selecting (minimal) sets of components which, if regarded as faulty, may explain these inconsistencies [18]. The computational burden is depended on the type and fidelity of the model.

On the other hand, data-driven and knowledge-based approaches seem to fit more poorly robotics system requirements.

For what concerns data-driven approaches, the statistical ones, such as outlier-detection [19], may not be an attractive choice for an FDD mechanism for a robotic system as large amounts of data are to be processed online, carrying a heavy computational load and might not detect the fault in time [20]. Yet, some techniques do improve traditional approaches and address these challenges.

On the contrary, machine-learning data-driven approaches are able to produce an FDD model, which can be quickly used online as learning offline reduces the online computational load. Nevertheless, it produces a static model, which may not fit new behaviors. Conversely, learning online increases the computational load but produces a

dynamic model. An attractive compromise is investigated in [21] and consists in adding the detected faults to a dynamic model.

This thesis, dealing with a model-based FDD approach, perfectly fits the remarks pointed out in this chapter.

Characteristic	FDD Challenges	Typically Addressed by
Dependency on exteroceptive sensors	Predicting sensor readings for sensor fault detection	Sensor fusion techniques Machine learning techniques
	Exploiting the information of exteroceptive sensors for FDD	Fault injection and supervised learning Particle filtering
Autonomous control	No operator Generating expectations	Model-based approaches Machine-learning approaches (to some extent)
	No remote supporting systems Creating a computationally light FDD mechanism	
	Continuous operation Online fault detection Quick fault detection	
Deliberation	Utilizing the plan for FDD	Model-based reasoning (plan is used as the model)
Dynamic context of operation	Recognizing or distinguish different contexts Detection of contextual faults	Windowing Clustering
Interaction with the environment	Unknown faults FDD for interaction related faults: Human-robot Robot-robot Social diagnosis Team-plan faults Coordination faults Scalability	Research opportunities: Human-robot Ad hoc teams FDD for robotic teams is typically handled with model-based diagnosis

Figure 2.3: *FDD challenges in robotics*

Chapter 3

Robot model

As already mentioned in Chapter 2, this work is aimed at the implementation of a fault detection and isolation scheme by means of a model-based approach. Hence, in order to achieve this goal, the model of the robotic system in analysis has been developed and is reported in the following.

It is worth underlining that the model proposed in this chapter is focused on the robotic head rotation and considers the trunk rotation as a disturbance [22], which can be encompassed in the scheme as any other eventual fault/disturbance (see Chapter 6).

Under such assumption, the system to be taken into consideration is a 2-DOF system. Indeed, TIAGo head is able to perform two rotational movements, namely pan and tilt rotations, as shown in Fig. 3.1.

Regarding what concerns the roll rotation, the robotic head is not equipped with it, so it has not been investigated in this work.

Starting from such knowledge, and for the sake of simplicity, such system has been likened to a 2R manipulator, whose model can be obtained by means of two approaches, i.e. the Lagrange one and the Newton-Euler one.

First, for completeness, the Lagrange approach has been sketched out. Later, the Newton-Euler one has been developed entirely.

Indeed, as it is possible to obtain a software-based prove only of the results got from the Newton-Euler method, the latter has been selected as the one to be taken into account in the following. Further investigations about the Lagrange approach are left

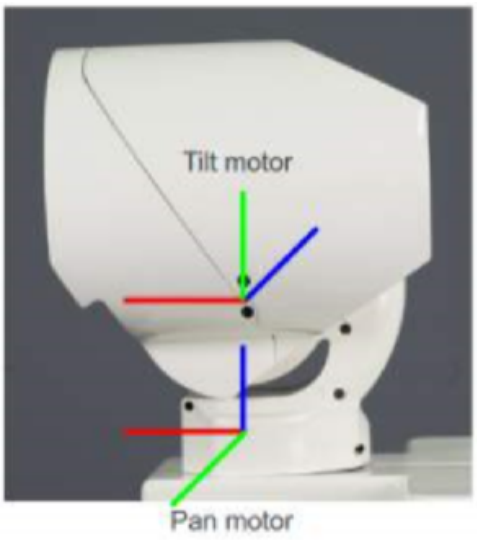


Figure 3.1: *Head degrees of freedom*

to the interested reader.

3.1 Lagrange approach

In this section, the derivation of the robot equations of motion by means of the Lagrangian formulation [23] are sketched out.

Previously to showing its development, it is worth noticing that the inertia tensors of both joints have been considered diagonal, as their off-diagonal terms are negligible.

$$
{}^{C_1}I_1 = \begin{bmatrix} I_{xx_1} & 0 & 0 \\ 0 & I_{yy_1} & 0 \\ 0 & 0 & I_{zz_1} \end{bmatrix}
$$

$$
{}^{C_2}I_2 = \begin{bmatrix} I_{xx_2} & 0 & 0 \\ 0 & I_{yy_2} & 0 \\ 0 & 0 & I_{zz_2} \end{bmatrix}
$$

First of all, as we deal with an "energy-based" approach to model the dynamics, an expression for the joints kinetic energy is obtained. The kinetic energy of the i-th link,

k_i, can be expressed as:

$$k_i = \frac{1}{2} m_i v_{C_i}^T v_{C_i} + \frac{1}{2} {}^i\omega_i^{T\,C_i} I_i \omega_i$$

where the first term is the kinetic energy due to linear velocity of the link center of mass and the second term is the kinetic energy due to angular velocity of the link.

Being the centre of mass of the first joint positioned along its own axis of rotation, the first term is null and the kinetic energy results

$$k_1 = \frac{1}{2} I_{zz_1} \dot{\theta}_1^2.$$

Similarly the kinetic energy corresponding to the second joint can be computed taking into account that in this case each velocity term is composed by both the own velocity (linear and rotational) of link 2 and the velocity propagated from link 1 to link 2.

The total kinetic energy is then the sum of the kinetic energy in the individual links, that is,

$$k = \sum_{i=1}^{2} k_i. \tag{3.1}$$

Moreover, the potential energy of the i-th link, u_i, can be expressed as

$$u_i = -m_i {}^0g^T {}^0P_{C_i} + u_{ref_i}$$

where ${}^0g$ is the 3×1 gravity vector $\begin{bmatrix} 0 & 0 & -g \end{bmatrix}$, ${}^0P_{C_i}$ is the vector locating the center of mass of the i-th link, and u_{ref_i} is a constant chosen so that the minimum value of u_i is zero.

The total potential energy is the sum of the potential energy in the individual links, that is,

$$u = \sum_{i=1}^{2} u_i \tag{3.2}$$

.

Once the expressions in Eq.(3.1) and in Eq.(3.2) have been evaluated, the Lagrangian dynamic formulation provides a means of deriving the equations of motion from a scalar function called the Lagrangian. Such function is defined as the difference between the

kinetic and potential energy of a mechanical system:

$$L(\theta, \dot{\theta}) = k(\theta, \dot{\theta}) - u(\theta). \tag{3.3}$$

The equations of motion are then given by

$$\tau = \frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}} - \frac{\partial L}{\partial \theta} = \frac{d}{dt}\frac{\partial k}{\partial \dot{\theta}} - \frac{\partial k}{\partial \theta} + \frac{\partial u}{\partial \theta} \tag{3.4}$$

where τ is the $n \times 1$ vector of actuator torques.

3.2 Newton-Euler approach

In this section, the robotic head model is developed by means of the Newton-Euler algorithm [23], which is a "force-balance" approach to dynamics, as it describes how forces, inertias and accelerations relate.

It is worth recalling that, as mentioned above, the results reported in this section have been verified by implementing this method in MATLAB, specifically by means of the MATLAB Robotics Toolbox.

Previously to the Newton-Euler algorithm implementation, the computation of the robot forward kinematics is required.

3.2.1 Forward kinematics

As well known in robotics, any robot can be described kinematically by four quantities for each link: the Denavit—Hartenberg parameters. To describe the link itself, and to describe the link connection to a neighboring link.

For a revolute joint, as those present in the robot in analysis, θ_i is called the joint variable and the other three quantities would be fixed link parameters. For prismatic joints, d_i is the joint variable and the other three quantities are fixed link parameters.

In order to obtain their values, a reference frame has been attached to each link, according to the convention in [23] and the DH parameters have been computed according to the following definitions:

d_i = *the distance from* $\hat{X}_{i-1}$ *to* $\hat{Z}_i$ *measured along* $\hat{Z}_i$

θ_i = *the angle from* $\hat{X}_{i-1}$ *to* $\hat{Z}_i$ *measured about* $\hat{Z}_i$

a_{i-1} = *the distance from* $\hat{Z}_{i-1}$ *to* $\hat{Z}_i$ *measured along* $\hat{X}_{i-1}$

α_i = *the angle from* $\hat{Z}_{i-1}$ *to* $\hat{Z}_i$ *measured about* $\hat{X}_{i-1}$

The resulting modified DH parameters for our 2R robot are shown in Table 3.1.

Once the parameters have been obtained, it is possible to construct the transform that defines frame i relative to the frame *i—1* as a function of the i-th joint variable.

i	θ_i	d_i	α_{i-1}	a_{i-1}
1	θ_1	0	0	0
2	$\theta_2 + \frac{\pi}{2}$	l_1	π	0
3	0	l_3	$-\pi$	l_2

Table 3.1: *Modified DH parameters*

State	Min	Max
θ_1	$-75°$	$75°$
θ_2	$-60°$	$45°$
$\dot{\theta}_1$ (m/s)	0	3
$\dot{\theta}_2$ (m/s)	0	3

Table 3.2: *System states physical limits*

For any given robot, the general form of the link transformation ${}^{i-1}_{i}T$ is:

$$ {}^{i-1}_{i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5} $$

Recalling that:

$$ \cos\left(\alpha + \frac{\pi}{2}\right) = -\sin(\alpha) \qquad \sin\left(\alpha + \frac{\pi}{2}\right) = \cos(\alpha) \tag{3.6} $$

and substituting the parameters, the transform for both links are obtained.

$$ {}^{0}_{1}T = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.7} $$

$$ {}^{1}_{2}T = \begin{bmatrix} -s_2 & -c_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ c_2 & -s_2 & 0 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.8} $$

$$ {}^{2}_{3}T = \begin{bmatrix} 1 & 0 & 0 & l_2 \\ 0 & 0 & 1 & l_3 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.9} $$

Lastly, the individual link-transformation matrices can be multiplied together to find the single transformation that relates frame N to frame 0:

$$ {}^{0}_{N}T = {}^{0}_{1}T\,{}^{1}_{2}T \ldots {}^{N-1}_{N}T \tag{3.10} $$

In the TIAGo head case:

$$ {}^{0}_{3}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T \tag{3.11} $$

3.2.2 Newton-Euler algorithm

Aiming at obtaining the robot equations of motion by means of the Newton-Euler algorithm, each link has been considered as a rigid body, whose mass distribution is characterized by its center of mass position and its inertia tensor.

It is worth recalling that the inertia tensors of both joints have been considered diagonal, as in Section 3.1.

Hence, the iterative Newton-Euler dynamics algorithm has been applied. It is composed of two parts: first, link velocities and accelerations are iteratively computed from link 1 out to link n and the Newton-Euler equations are applied to each link. Second, forces and torques of interaction and joint actuator torques are computed recursively from link n back to link 1.

The equations are summarized in Fig.3.2 for the case of all joints rotational (as TIAGo head case).

The effect of gravity loading on the links can be included quite simply by setting

$$
\begin{aligned}
{}^{i+1}\omega_{i+1} &= {}^{i+1}_{i}R\,{}^{i}\omega_i + \dot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1},\\
{}^{i+1}\dot{\omega}_{i+1} &= {}^{i+1}_{i}R\,{}^{i}\dot{\omega}_i + {}^{i+1}_{i}R\,{}^{i}\omega_i \times \dot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} + \ddot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1},\\
{}^{i+1}\dot{v}_{i+1} &= {}^{i+1}_{i}R({}^{i}\dot{\omega}_i \times {}^{i}P_{i+1} + {}^{i}\omega_i \times ({}^{i}\omega_i \times {}^{i}P_{i+1}) + {}^{i}\dot{v}_i),\\
{}^{i+1}\dot{v}_{C_{i+1}} &= {}^{i+1}\dot{\omega}_{i+1} \times {}^{i+1}P_{C_{i+1}}\\
&\quad + {}^{i+1}\omega_{i+1} \times ({}^{i+1}\omega_{i+1} \times {}^{i+1}P_{C_{i+1}}) + {}^{i+1}\dot{v}_{i+1},\\
{}^{i+1}F_{i+1} &= m_{i+1}\,{}^{i+1}\dot{v}_{C_{i+1}},\\
{}^{i+1}N_{i+1} &= {}^{C_{i+1}}I_{i+1}\,{}^{i+1}\dot{\omega}_{i+1} + {}^{i+1}\omega_{i+1} \times {}^{C_{i+1}}I_{i+1}\,{}^{i+1}\omega_{i+1}.
\end{aligned}
$$

(a)

$$
\begin{aligned}
{}^{i}f_i &= {}^{i}_{i+1}R\,{}^{i+1}f_{i+1} + {}^{i}F_i,\\
{}^{i}n_i &= {}^{i}N_i + {}^{i}_{i+1}R\,{}^{i+1}n_{i+1} + {}^{i}P_{C_i} \times {}^{i}F_i\\
&\quad + {}^{i}P_{i+1} \times {}^{i}_{i+1}R\,{}^{i+1}f_{i+1},\\
\tau_i &= {}^{i}n_i^T\,{}^{i}\hat{Z}_i.
\end{aligned}
$$

(b)

Figure 3.2: *Newton-Euler algorithm*

${}^{0}\dot{v}_0 = G$, where G has the magnitude of the gravity vector but points in the opposite direction. This is equivalent to consider that the base of the robot is accelerating upward with 1 g acceleration. This fictitious upward acceleration causes exactly the same effect on the links as gravity would and its expression can be rewritten as:

$$
{}^{0}\dot{v}_0 = g\hat{Z}_0.
$$

Moreover, since there are no external forces acting on the robot head, so we have

$$
f_3 = 0,
$$

$$
n_3 = 0.
$$

In addition, as previously mentioned, the trunk rotation is considered as a distur-

bance, such that the base of the robot head is not rotating. Hence, we have

$$\omega_0 = 0,$$

$$\dot{\omega}_0 = 0.$$

Lastly, before applying the algorithm, it is worth recalling that the following relation holds.

$${}^{i}_{i+1}R = {}^{i+1}_{i}R^T$$

The algorithm has been then applied, leading to the following equations

$$\tau_1 = I_{zz_1}\ddot{\theta}_1 + \frac{1}{2}\ddot{\theta}_1(I_{xx_2} + I_{yy_2}) + \frac{1}{2}\ddot{\theta}_1 c(2\theta_2)(I_{xx_2} - I_{yy_2}) + d_4^2 m_2\ddot{\theta}_1 - d_4^2 m_2\ddot{\theta}_1 c(2\theta_2)^2 - I_{xx_2}\dot{\theta}_1\dot{\theta}_2 s(2\theta_2) + I_{yy_2}\dot{\theta}_1\dot{\theta}_2 s(2\theta_2) + d_4^2\dot{\theta}_1\dot{\theta}_2 m_2 s(4\theta_2) \quad (3.12)$$

$$\tau_2 = I_{zz_2}\ddot{\theta}_2 + \frac{1}{2}s(2\theta_2)\dot{\theta}_1^2(I_{xx_2} - I_{yy_2}) + m_2 d_4^2\ddot{\theta}_2 - \frac{1}{2}m_2 d_4^2\dot{\theta}_1^2 s(4\theta_2) - d_4 m_2 g s(2\theta_2) \quad (3.13)$$

As expected, Eq.3.12 and Eq.3.13 give expressions for the torque at the actuators as a function of joint position, velocity and acceleration.

Also in this case, such equations can be rewritten in state-space form, as follows:

$$\tau = M(\theta)\ddot{\theta} + V(\theta, \dot{\theta}) + G(\theta)$$

where $M(\theta)$ is then $n \times n$ mass matrix of the manipulator, $V(\theta, \dot{\theta})$ is a $n \times 1$ vector of centrifugal and Coriolis terms and $G(\theta)$ is an $n \times 1$ vector of gravity terms.

Each element of $M(\theta)$ and $G(\theta)$ is a complex function that depends on θ, the position of all the joints of the manipulator. Each element of $V(\theta, \dot{\theta})$ is a complex function of both θ and $\dot{\theta}$.

To be even more precise, by writing the velocity-dependent term, $V(\theta, \dot{\theta})$, in a different form, we can rewrite the dynamic equations in the so-called configuration-space formulation:

$$\tau = M(\theta)\ddot{\theta} + B(\theta)[\dot{\theta}\dot{\theta}] + C(\theta)[\dot{\theta}^2] + G(\theta)$$

where $B(\theta)$ is a matrix of dimensions $n \times n\frac{n^-1}{2}$ of Coriolis coefficients, $[\dot{\theta}\dot{\theta}]$ is an $n\frac{n^-1}{2} \times 1$

vector of joint velocity products given by

$$[\dot{\theta}\dot{\theta}] = [\dot{\theta_1}\dot{\theta_2}]^T,$$

$C(\theta)$ is a $n \times n$ matrix of centrifugal coefficients, and $[\dot{\theta}^2]$ is an $n \times 1$ vector given by

$$[\dot{\theta_1}^2\dot{\theta_2}^2]^T.$$

It is worth noticing that, in this formulation, all matrices are a function of only the joints position, fact that is very helpful for updating the model as the robot moves its head.

Indeed, for TIAGo's head the resulting equations are the following ones

$$\begin{bmatrix}\tau_1\\ \tau_2\end{bmatrix} = \begin{bmatrix} I_{zz_1} + m_2 d_4^2 + I_{xx_2} c_2^2 + I_{yy_2} s_2^2 - m_2 d_4^2 c(2\theta_2)^2 & 0 \\ 0 & I_{zz_2} + m_2 d_4^2 \end{bmatrix} \begin{bmatrix}\ddot{\theta}_1\\ \ddot{\theta}_2\end{bmatrix} + \begin{bmatrix} -2c_2 s_2 (I_{xx_2} - I_{yy_2}) + m_2 d_4^2 s(4\theta_2) \\ 0 \end{bmatrix} \dot{\theta}_1 \dot{\theta}_2 + \begin{bmatrix} 0 & 0 \\ c_2 s_2 (I_{xx_2} - I_{yy_2}) - \frac{1}{2} m_2 d_4^2 s(4\theta_2) & 0 \end{bmatrix} \begin{bmatrix}\dot{\theta}_1^2\\ \dot{\theta}_2^2\end{bmatrix} + g \begin{bmatrix} 0 \\ -m_2 d_4 s(2\theta_2) \end{bmatrix}$$

which perfectly correspond to those obtained by implementing the algorithm in Fig.3.2 in MATLAB.

Lastly, such equations can be expressed in a shorter and more intuitive way, as follows:

$$\begin{bmatrix}\tau_1\\ \tau_2\end{bmatrix} = \begin{bmatrix} \alpha + \beta(\theta_2) & 0 \\ 0 & \xi \end{bmatrix} \begin{bmatrix}\ddot{\theta}_1\\ \ddot{\theta}_2\end{bmatrix} + \begin{bmatrix} \delta(\theta_2) \\ 0 \end{bmatrix} \dot{\theta}_1 \dot{\theta}_2 + \begin{bmatrix} 0 & 0 \\ \eta(\theta_2) & 0 \end{bmatrix} \begin{bmatrix}\dot{\theta}_1^2\\ \dot{\theta}_2^2\end{bmatrix} + g \begin{bmatrix} 0 \\ \lambda(\theta_2) \end{bmatrix} \tag{3.14}$$

where the dependency on the joints position is even more emphasized. As a matter of facts, α and ξ are constant terms, while β, δ, η and λ depend on θ_2.

From Eq.(3.14), the model equations of interest are derived straightforwardly:

$$\begin{cases} \ddot{\theta}_1 = -\frac{\delta(\theta_2)}{\alpha+\beta(\theta_2)}\dot{\theta}_1\dot{\theta}_2 + \frac{1}{\alpha+\beta(\theta_2)}\tau_1 \\ \ddot{\theta_1} = -\frac{\eta(\theta_2)}{\xi}\dot{\theta}_1^2 + \frac{1}{\xi}\tau_2 - \frac{\lambda(\theta_2)}{\xi} \\ \dot{\theta}_1 = \frac{d}{dt}\theta_1 \\ \dot{\theta}_2 = \frac{d}{dt}\theta_2. \end{cases} \tag{3.15}$$

Chapter 4

LPV Model

The model obtained in Chapter 3, namely Eq.3.15 can be expressed in the state space formulation

$$\dot{x} = A(x)x + B(x)u + E(x)$$
$$y = C(x)x + D(x)$$

by considering as states, control variables and output respectively the following vectors

$$x = \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \theta_1 \\ \theta_2 \end{bmatrix} \qquad u = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \qquad y = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}.$$

Hence, we obtain the following formulation

$$\begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} = \begin{bmatrix} 0 & -\frac{\delta(\theta_2)}{\alpha+\beta(\theta_2)}\dot{\theta}_1 & 0 & 0 \\ -\frac{\eta(\theta_2)}{\xi}\dot{\theta}_1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \theta_1 \\ \theta_2 \end{bmatrix} + \begin{bmatrix} \frac{1}{\alpha+\beta(\theta_2)} & 0 \\ 0 & \frac{1}{\xi} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{\lambda(\theta_2)}{\xi} \\ 0 \\ 0 \end{bmatrix} \tag{4.1}$$

$$\begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix} = \begin{bmatrix}0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\dot{\theta}_1\\ \dot{\theta}_2\\ \theta_1\\ \theta_2\end{bmatrix} \tag{4.2}$$

where an important consideration results evident: in the case of analysis, contrary to matrices *A(x)* and *B(x)*, matrix C and D are constant.

Moreover, analyzing the results of the previous chapter, it can be noticed that the following relation holds:

$$\delta(\theta_2) = -2\eta(\theta_2) \tag{4.3}$$

Thus, renaming some terms as follows

$$\alpha + \beta(\theta_2) = \gamma(\theta_2) \tag{4.4}$$

$$\delta(\theta_2)\dot{\theta}_1 = \varphi(\theta_2, \dot{\theta}_1) \tag{4.5}$$

the state space formulation becomes:

$$\begin{bmatrix}\ddot{\theta}_1\\ \ddot{\theta}_2\\ \dot{\theta}_1\\ \dot{\theta}_2\end{bmatrix} = \begin{bmatrix}0 & -\frac{\varphi(\theta_2,\dot{\theta}_1)}{\gamma(\theta_2)} & 0 & 0\\ \frac{\varphi(\theta_2,\dot{\theta}_1)}{2\xi} & 0 & 0 & 0\\ 1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\end{bmatrix}\begin{bmatrix}\dot{\theta}_1\\ \dot{\theta}_2\\ \theta_1\\ \theta_2\end{bmatrix} + \begin{bmatrix}\frac{1}{\gamma(\theta_2)} & 0\\ 0 & \frac{1}{\xi}\\ 0 & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}\tau_1\\ \tau_2\end{bmatrix} + \begin{bmatrix}0\\ -\frac{\lambda(\theta_2)}{\xi}\\ 0\\ 0\end{bmatrix}$$

$$\begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix} = \begin{bmatrix}0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\dot{\theta}_1\\ \dot{\theta}_2\\ \theta_1\\ \theta_2\end{bmatrix} \tag{4.6}$$

From now on the matrix *E(x)* will be neglected and its role will be further investigated later.

At this point, all the matrices elements depend on φ and γ. Such consideration suggests that the system in analysis is an LPV one.

Indeed, following the terminology in [24], LPV systems are linear time-varying plants

whose state-space matrices are fixed functions of some vector of varying parameters $\Psi(t)$ [25].

Generally speaking, LPV systems are described by state-space equations of the form

$$\dot{x} = A(\Psi(t))x + B(\Psi(t))u$$
$$y = C(\Psi(t))x + D(\Psi(t))u$$

where x, u and y denote the state, input and output vectors respectively.

Often the varying parameters $\Psi(t)$ can be measured in real time during system operation. Consequently, the control strategy can exploit their available measurements, adjusting to the variations in the plant dynamics in order to maintain stability and high performance along all trajectories $\Psi(t)$. In other words, the controller is 'self-scheduled', that is, automatically gain-scheduled with respect to $\Psi(t)$, leading to a performance increase.

Such procedure is based on the following LPV systems property: when 'freezing' $\Psi(t)$, to some given value Ψ, the LPV system becomes an LTI system described by the transfer function

$$G(\sigma) = D(\Psi) + C(\Psi)\left[\sigma I - A(\Psi)\right]^{-1}$$

where σ stands for the Laplace variable s in the continuous-time case and for the Z-transform variable z in the discrete-time case.

Hence, in order to overcome the difficulty of managing an infinite number of possible trajectories, we confine ourself to LPV systems [25] where the time-varying parameter $\Psi(t)$ varies in a polytope Θ of vertices $\omega_1, \omega_2, \ldots, \omega_r$; that is,

$$\Psi(t) = \Theta := Co\{\omega_1, \omega_2, \ldots, \omega_r\}.$$

In other words, we deal with a polytopic LPV system, whose state-space matrices range in a polytope of matrices whose vertices are the images of the vertices $\omega_1, \omega_2, \ldots, \omega_r$:

$$\begin{bmatrix} A(\Psi) & B(\Psi) \\ C(\Psi) & D(\Psi) \end{bmatrix} \in Co\left\{ \begin{bmatrix} A_i & B_i \\ C_i & D_i \end{bmatrix} := \begin{bmatrix} A(\omega_i) & B(\omega_i) \\ C(\omega_i) & D(\omega_i) \end{bmatrix} \right\}, \qquad i = 1, \ldots, r.$$

Coming back to TIAGo's head model the time-varying parameters to be considered are φ and γ, which can assume any value respectively in $[\underline{\varphi}, \overline{\varphi}]$ and $[\underline{\gamma}, \overline{\gamma}]$.

Previously to compute the values of such extreme, it is worth recalling their expressions:

$$\gamma(\theta_2) = I_{zz_1} + m_2 d_4^2 + I_{xx_2} c_2^2 + I_{yy_2} s_2^2 - m_2 d_4^2 c(2\theta_2)^2$$

$$\varphi(\theta_2, \dot{\theta}_1) = (-2c_2 s_2 (I_{xx_2} - I_{yy_2}) + m_2 d_4^2 s(4\theta_2))\dot{\theta}_1.$$

At this point it is evident that the parameters extremes depend on the system states values, specifically on θ_2 and $\dot{\theta}_1$, which in turn are bounded by the physical system limits reported in Table 4.1.

State	Min	Max
θ_1	$-75°$	$75°$
θ_2	$-60°$	$45°$
$\dot{\theta}_1$ (m/s)	0	3
$\dot{\theta}_2$ (m/s)	0	3

Table 4.1: *System states physical limits*

Consequently, an optimization process has been carried out, aimed at obtaining the above-mentioned parameters extremes. Such computation has been realized by means of the MATLAB Optimization Toolbox, leading to the results listed in Table 4.2.

Parameter	Min	Max
γ	0.0055	0.0091
φ	0.0110	-0.0110

Table 4.2: *LPV parameters limits*

Those results are also visible in Fig.4.1 and Fig.4.2, where the trend of γ and φ respectively has been plotted with respect to $\theta_2 \in [\underline{\theta_2}, \overline{\theta_2}]$.

For what concerns φ, it is worth underlining that $\dot{\theta}_1$ affects its trend only in terms of amplitude. Hence, it is enough to analyze it fixing $\dot{\theta}_1$ to its maximum value, namely 3.

Once the parameters extremes have been computed, the sought polytope vertexes have been identified.

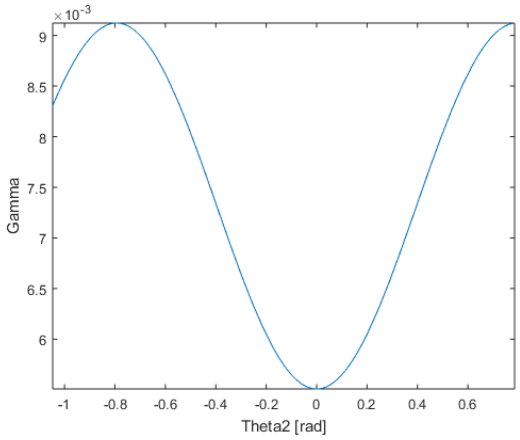


Figure 4.1: *Trend of* $\gamma(\theta_2)$

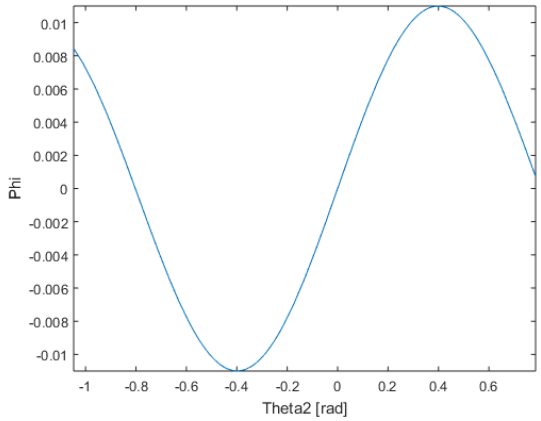


Figure 4.2: *Trend of* $\phi(\theta_2)$ *with* $\dot{\theta}_1 = 3$

Therefore, recalling that only matrices A and B are varying, also the matrices polytopes, having as vertexes A_i and B_i, has been straightforwardly determined, resulting in the bounding boxes shown in Fig.4.3 and Fig.4.4.

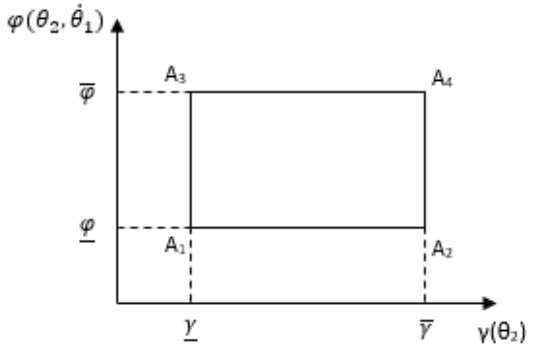


Figure 4.3: *Matrix A bounding box*

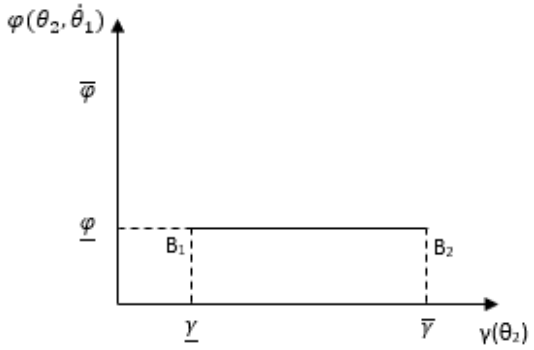


Figure 4.4: *Matrix B bounding box*

Chapter 5

LPV Control

This chapter formulates the control problem for the polytopic LPV systems, modelled as in Chapter 4.

Assuming complete measurement of the varying parameters $\Psi(t)$, the controller is allowed to incorporate these measurements in the same LPV fashion as the plant. The resulting LPV controller exploits all available information on $\Psi(t)$, to adjust to the current plant dynamics. This provides smooth and automatic gain-scheduling with respect to the varying parameters $\Psi(t)$.

As pointed out by [26], in order not to deal with a problem having an infinite number of constraints and, thus, not easily tractable, the following assumption has to be satisfied: matrices B, C and D have to be parameter-independent.

According to the model obtained in Chapter 4, and here reported,

$$\dot{x} = A(\Psi(t))x + B(\Psi(t))u$$
$$y = Cx + Du$$

matrices C and D meet such requirement, while matrix B does not, as it depends on the parameter γ.

$$B = \begin{bmatrix} \frac{1}{\gamma(\theta_2)} & 0 \\ 0 & \frac{1}{\xi} \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

However, this difficulty can be alleviated by pre- and/or post-filtering the control inputs u. Specifically, a fast dynamic filter has been added as suggested by [25] in the form

$$\dot{x_f} = A_f x_f + B_f u_f$$

$$\begin{bmatrix} \dot{\tau_1} \\ \dot{\tau_2} \end{bmatrix} = \begin{bmatrix} -\psi & 0 \\ 0 & -\psi \end{bmatrix} \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} + \begin{bmatrix} \psi & 0 \\ 0 & \psi \end{bmatrix} \begin{bmatrix} u_{\tau_1} \\ u_{\tau_2} \end{bmatrix}$$

where ψ represents the filter gain and u_f is the new control variable vector.

Note that this new added states have fast dynamics, so that do not significantly alter the original problem.

Then, the original fourth order system is transformed into a new sixth order system

$$\dot{\tilde{x}} = \tilde{A}(\Psi)\tilde{x} + \tilde{B}u_f$$

$$y = \tilde{C}\tilde{x} \tag{5.1}$$

with state and input vectors as

$$\tilde{x} = \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \theta_1 \\ \theta_2 \\ \tau_1 \\ \tau_2 \end{bmatrix} \qquad \tilde{u} = \begin{bmatrix} u_{\tau_1} \\ u_{\tau_2} \end{bmatrix}$$

and matrices $\tilde{A}$, $\tilde{B}$ and $\tilde{C}$ as

$$\tilde{A} = \begin{bmatrix} A & B \\ 0_{2x4} & A_f \end{bmatrix} = \begin{bmatrix} 0 & -\frac{\varphi(\theta_2,\dot{\theta}_1)}{\gamma(\theta_2)} & 0 & 0 & \frac{1}{\gamma(\theta_2)} & 0 \\ \frac{\varphi(\theta_2,\dot{\theta}_1)}{2\xi} & 0 & 0 & 0 & 0 & \frac{1}{\xi} \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\psi & 0 \\ 0 & 0 & 0 & 0 & 0 & -\psi \end{bmatrix}$$

$$\tilde{B} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \psi & 0 \\ 0 & \psi \end{bmatrix}$$

$$\tilde{C} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}. \tag{5.2}$$

Note that the control and measurement matrices are now parameter-free as required. Moreover, whenever the plant model includes actuator and sensor dynamics, the control and measurement matrices are still parameter-free. Hence the proposed filtering operations are not restrictive in a practical perspective.

Once these arrangements have been realized in the system model, the problem of generating the appropriate behavior of TIAGo's head has been addressed. In this work a feedback LPV controller is proposed for performing such task, given a desired reference in terms of joints position.

Such controller K is in charge of computing smooth control actions (the joint torques) such that the robotic head is capable of achieving the required orientation, while respecting the system physical limits.

To design K a polytopic approach has been followed. Hence the controller gain is

obtained in the form

$$K(\Psi) = \sum_{i=1}^{2^{n_\Psi}} \mu_i(\Psi) K_i \tag{5.3}$$

where n_Ψ is the number of scheduling variables contained in $\Psi := [\psi_1(t), \ldots, \psi_{n_\Psi}]$ -here $n_\Psi = 2$ and $\mu_i(\Psi)$ is given by

$$\mu_i(\Psi) = \prod_{j=1}^{n_\Psi} \zeta_{ij}(\eta_0^j, \eta_1^j), \qquad i = 1, \ldots, 2^{n_\Psi}$$

$$\eta_0^j = \frac{\bar{\psi_j} - \psi_j(t)}{\bar{\psi_j} - \underline{\psi_j}}$$

$$\eta_0^j + \eta_1^j = 1, \qquad j = 1, \ldots, n_\Psi$$

where, as already explained, each variable ψ_j is known and varies in a defined interval $\psi_j(t) \in [\underline{\psi_j}, \bar{\psi_j}]$.

Moreover, K_i are determined by studying the LMI based LPV stability problem presented in the following.

Given a continuous state-space system in the form $\dot{x} = Ax + Bu$ a state feedback controller $u = Kx$ is sought, such that the closed-loop system is stable. Based on the Lyapunov theory, the problem can be reformulated in terms of the following LMI:

$$AP + PA' + BW + W'B' < 0 \tag{5.4}$$

$$P > 0 \tag{5.5}$$

which, if satisfied, has as solution the above-mentioned controller K:

$$K = WP^{-1}$$

When applied to an LPV system, the stability problem in Eq.5.4 becomes:

$$A(\Psi)P + PA(\Psi)' + BW + W'B' < 0$$

$$P > 0$$

being characterized by an infinite number of constraints. As already done in the modeling phase, a polytopic approach can be applied by confining the analysis to the bounding box vertexes:

$$A_i P + P A_i' + BW + W'B' < 0 \tag{5.6}$$

$$P > 0 \tag{5.7}$$

where $i = 1, \ldots, n_\Psi$ and $A_i(\Psi)$ are the vertexes of matrix A bounding box.

Solving such problem for TIAGo's head by means of the software MATLAB, the vertex controllers $K_i(\Psi)$ have been computed. The correspondent closed-loop systems poles are shown in Fig. 5.1. It is evident that stability is ensured.

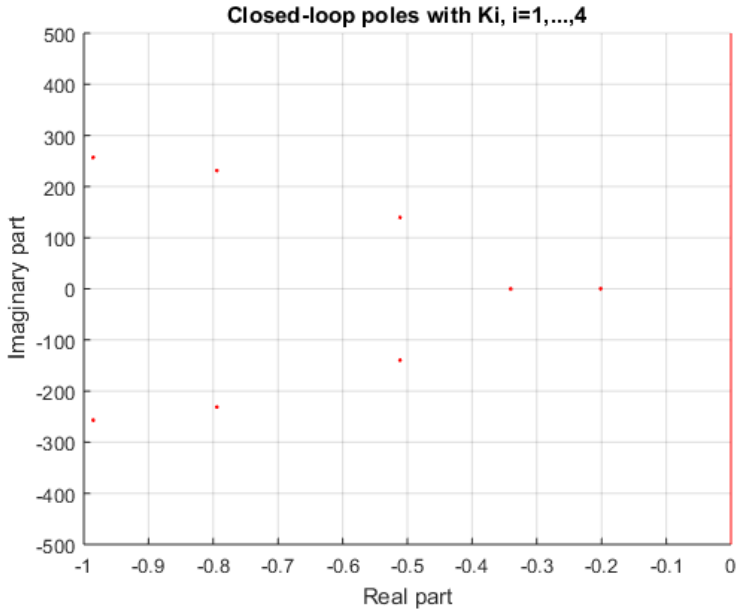


Figure 5.1: *Closed-loop poles for K=K_i, i=1,...,4*

The proposed state feedback control scheme has been then integrated with a feed-forward control, in order to make the system gain unitary. Such feedforward matrix is computed following the next expression:

$$M = -\left[\tilde{C}(-\tilde{B}K + \tilde{A}(\Psi))^{-1}\tilde{B}\right]^{-1}. \tag{5.8}$$

It is worth pointing out that while the vertex controllers K_i and matrices $\tilde{A}_i$ can be computed off-line, the LPV matrices $\tilde{A}(\Psi)$ and $K(\Psi)$ must be updated in real time based on the parameter measurement Ψ.

Indeed, the real time value of matrix $\tilde{A}(\Psi)$ can be computed in the same fashion as $K(\Psi)$:

$$\tilde{A}(\Psi) = \sum_{i=1}^{2^{n_\Psi}} \mu_i(\Psi)\tilde{A}_i$$

The control strategy presented till now perfectly fits the ideal case of sensors availability for measuring all the system states. As a matter of fact, only the measurements of the joints positions, i.e. the angles θ_1 and θ_2, are accessible, as shown by matrix $\tilde{C}$ expression in Eq.5.2.

Due to the lack of the other sensors, i.e. there is no one that measures joints velocities, the design of a state estimator has been developed through a polytopic approach, analogously to the controller:

$$L(\Psi) = \sum_{i=1}^{2^{n_\Psi}} \mu_i(\Psi)L_i$$

where the vertex observers L_i can be obtained solving the LMI based problem dual to the K_i controllers one in Eq.5.6:

$$A_i'P + PA_i + C'W + W'C < 0 \tag{5.9}$$

$$P > 0 \tag{5.10}$$

Moreover, for performance reasons, the observer has been sped up by slightly modifying the LMI in Eq.5.9 as follows:

$$A_i'P + PA_i + C'W + W'C + 2\alpha_{obs}P < 0$$

$$P > 0$$

where the parameter α_{obs} can be chosen to arbitrarily improve the observer performances. Here a dynamic 10 times faster than the controller one has been selected, as demonstrated

by the plot of the new closed-loop poles in blue in Fig.5.2.

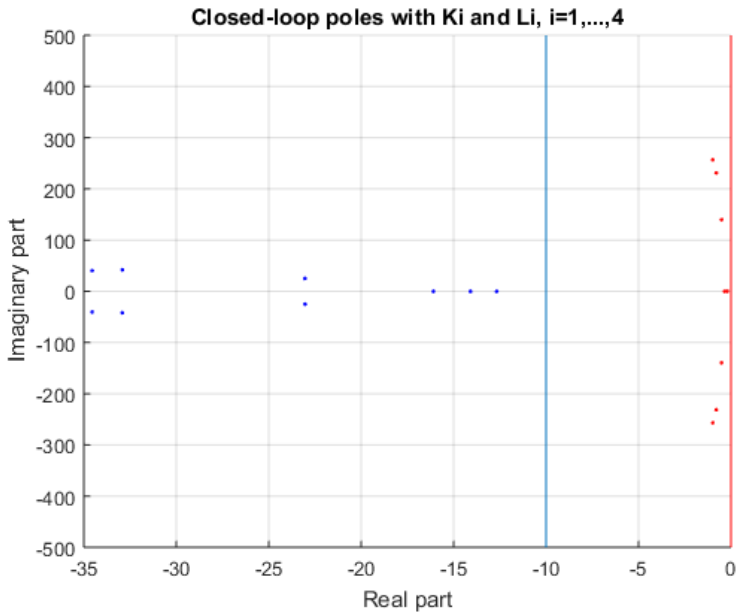


Figure 5.2: *Closed-loop poles for K=K_i (red) and L=L_i (blue) , i=1,...,4*

Note that the same approach could have been followed to speed up also the controller.

Chapter 6

Fault/disturbance estimation

The present chapter deals with the design of a Linear Parameter Varying Unknown Input Observer (LPV UIO), proposed as an improvement to the estimation scheme developed in Chapter 5. Such observer, indeed, tackles the problem of estimating both the dynamic states and the eventual disturbance actuating over the system.

As a matter of fact, in this chapter the role of the matrix E in Eq.4, neglected till now, has been investigated. In order to carry out an analysis as extensive as possible, a totally generic additive disturbance acting on both angles has been taken into account.

Hence, the matrix E_d actually implemented has the following expression:

$$E_d = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

Consequently, the system model in Eq.5 becomes:

$$\begin{aligned} \dot{\tilde{x}} &= \tilde{A}(\Psi)\tilde{x} + \tilde{B}\tilde{u} + E_d d \\ y &= \tilde{C}\tilde{x} \end{aligned}$$

where $d = [d_{\theta_1}; d_{\theta_2}]$ is the disturbance vector.

The UIO approach is based on computing the difference between the real system and the model used for observation:

$$\tilde{C}E_d d = \dot{y} - \tilde{C}(A(\tilde{\Psi})\hat{\tilde{x}} + \tilde{B}\tilde{u}).$$

Thus, considering $\Theta = (\tilde{C}E_d)^+$, the disturbance vector can be obtained as:

$$d = \Theta\left(\hat{y} - \tilde{C}(A(\tilde{\Psi})\hat{\tilde{x}} + \tilde{B}\tilde{u}\right)$$

And consequently, decoupling the considered disturbance, the system model can be rewritten as follows:

$$\begin{aligned}\dot{\tilde{x}} &= A_o(\Psi)\tilde{x} + B_o\tilde{u} + E_d\Theta\dot{y}\\ y &= \tilde{C}\tilde{x}\end{aligned}$$

where:

$$A_o = (I - E_d\Theta\tilde{C})\tilde{A}$$

$$B_o = (I - E_d\Theta\tilde{C})\tilde{B}$$

Then, the state estimation depends on the observer gain $L(\Psi)$ and presents the form:

$$\dot{\hat{\tilde{x}}} = \left(A_o - L\tilde{C}\right)\hat{\tilde{x}} + B_o\tilde{u} + E_d\Theta\dot{y} + Ly$$

Chapter 7

Simulation results

This chapter is devoted to the description of the simulation schemes implemented in order to test the LPV control and fault estimation scheme proposed in this thesis.

All simulations have been realized by means of the softwares MATLAB and Simulink and are presented in the following as listed below:

1. the model has been simulated considering full observability, i.e only the feedback controller K and the feedforward matrix M have been tested;

2. the observer L has been added to the model in 1), aimed at the estimation of not measurable states;

3. the UIO observer is added to the model in 1), aimed at the estimation of not measurable states and of eventually present fault/disturbance.

It is worth underlining that in all simulations the physical limits in Table4.1 have been taken into account both for the real and estimated states. Indeed, in this first case, their are embedded in the robot head subsystem, while in the latter case, they are represented by the subsystem "Limits".

Lastly, for comparison ease, all models have been simulated with the orientation references set to the same value, namely $\theta_{1_{ref}} = 0.3$ and $\theta_{2_{ref}} = 0.3$.

7.1 Case I. Full observability

As first test, the robotic head system has been considered fully observable, i.e. full sensors availability has been supposed. Obviously this case does not represent a realistic situation, but it has been useful as fist approach to the LPV controller implementation.

The Simulink scheme used for such simulation is shown in Fig.7.1.

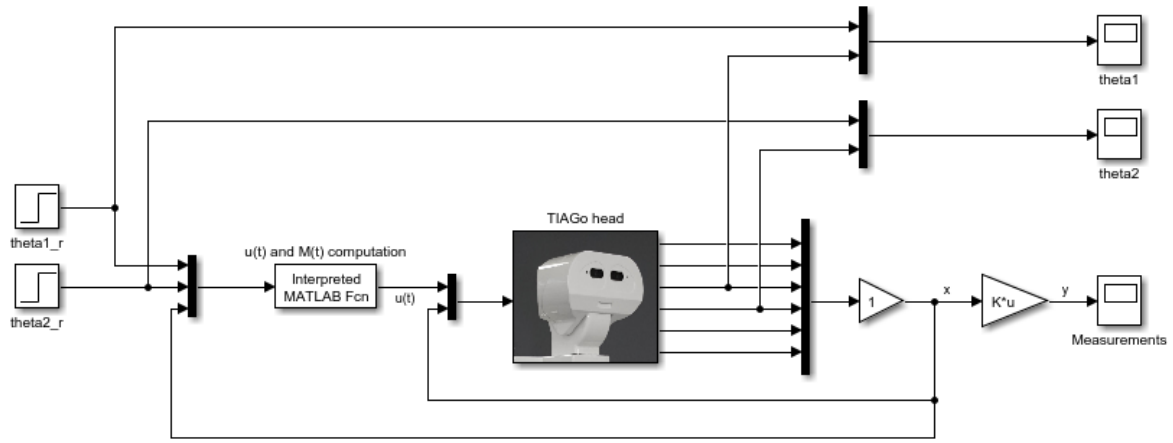


Figure 7.1: *Case I. Simulink scheme*

The proposed control scheme leads to the following results:

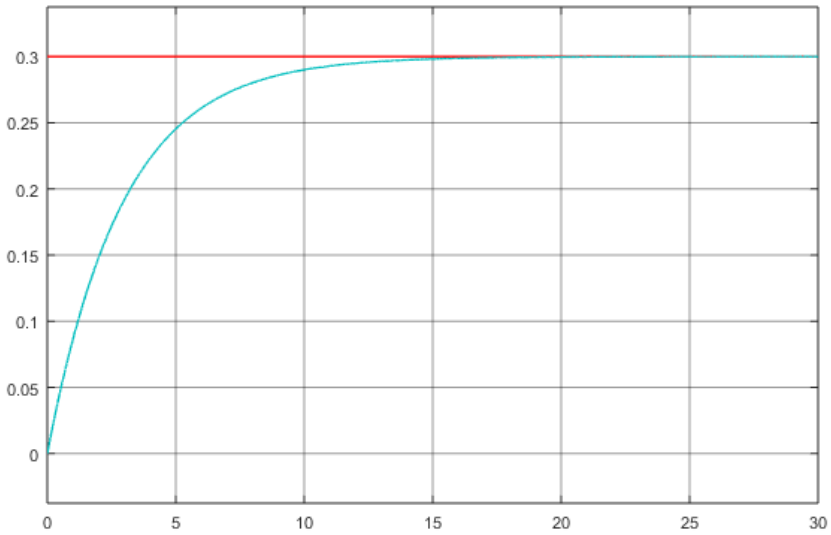


Figure 7.2: *Case I.* θ_1 *trend*

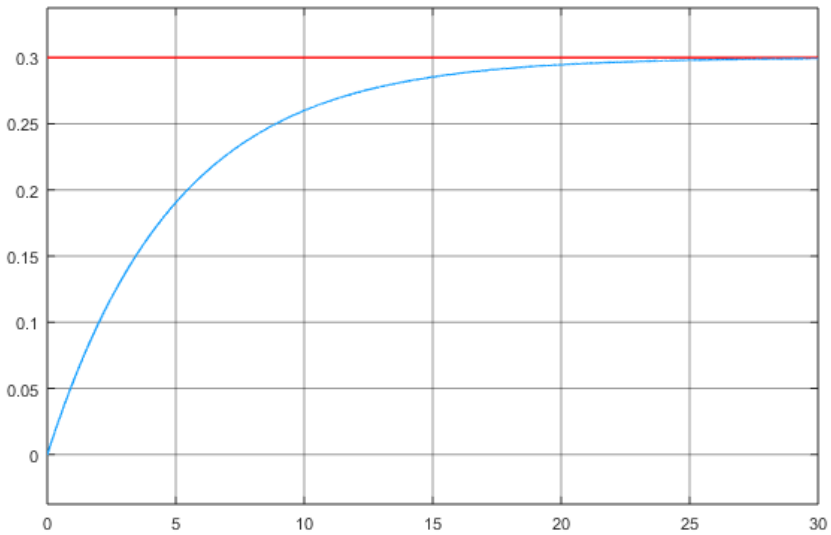


Figure 7.3: *Case I.* θ_2 *trend*

From Fig.7.2 and Fig.7.3 it is evident that both angles, namely θ_1 and θ_2, perfectly reach the reference with satisfying performances -i.e. no oscillation and/or overshoot is present.

7.2 Case II. Partial observability

Once the feedback controller has been checked together with the feedforward compensation, a more realistic case has been analyzed.

As the robot head is equipped with two encoders, one per each rotation movement, the only states which have been considered measurable are the two angles θ_1 and θ_2.

Consequently, the joints velocities have to be estimated, as explained in Chapter 5. Then, the feedback controller receives as input no more the real states, but the estimated ones.

The Simulink scheme used for such simulation is shown in Fig.7.4.

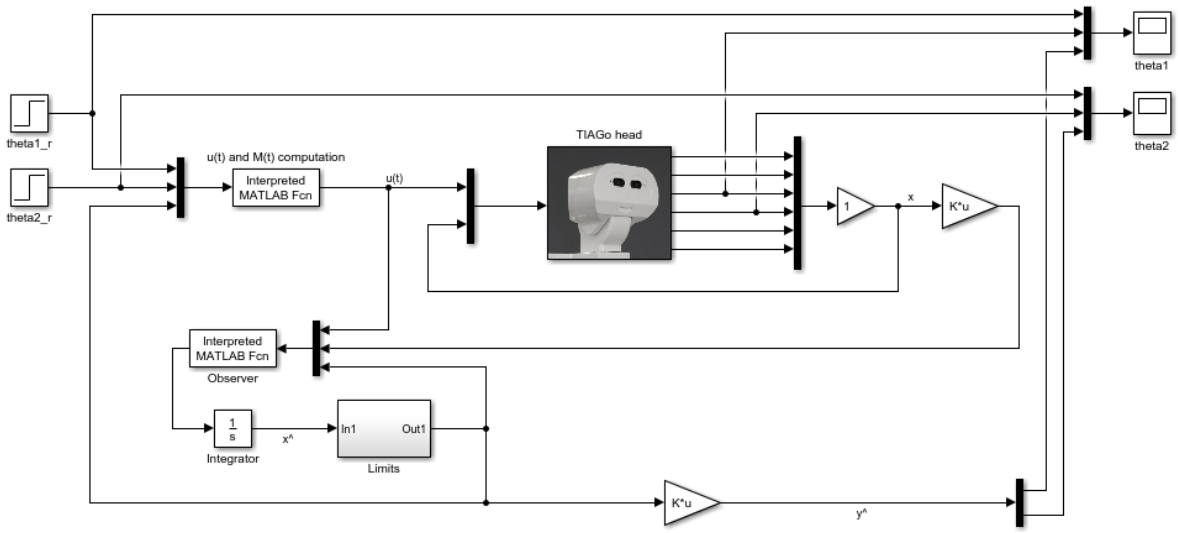


Figure 7.4: *Case II. Simulink scheme*

First of all, the real and estimated trends of the two angles θ_1 and θ_2 have been compared. As shown in Fig.7.5 and Fig.7.6, the observer perfectly accomplishes its estimation task.

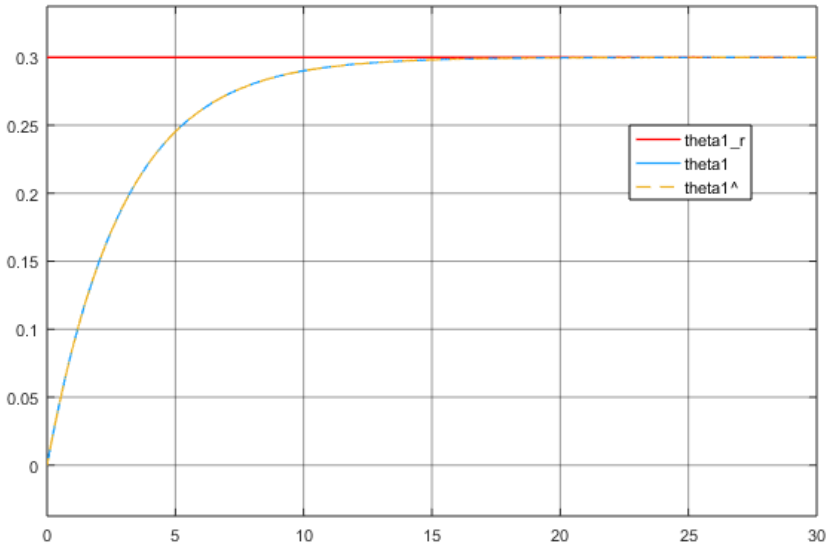


Figure 7.5: *Case II. Real and estimate θ_1 trend*

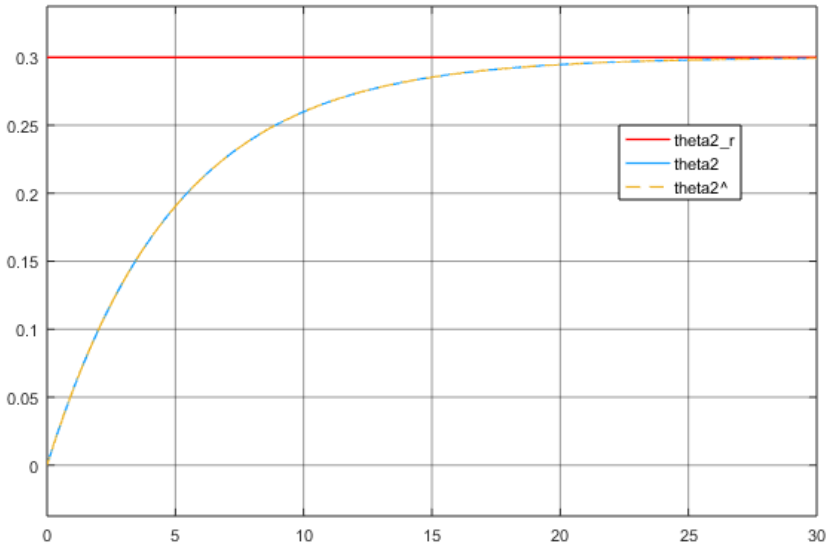


Figure 7.6: *Case II. Real and estimate θ_2 trend*

Moreover, it is visible that also in this case both angles perfectly reach the reference. The same considerations on performances as in Case I are still valid here.

7.3 Case III. Partial observability - UIO Observer

Based on the same discussion about observability presented in Case II, the state and disturbance estimation is here realized by means of the UIO Observer, as represented in the following Simulink scheme:

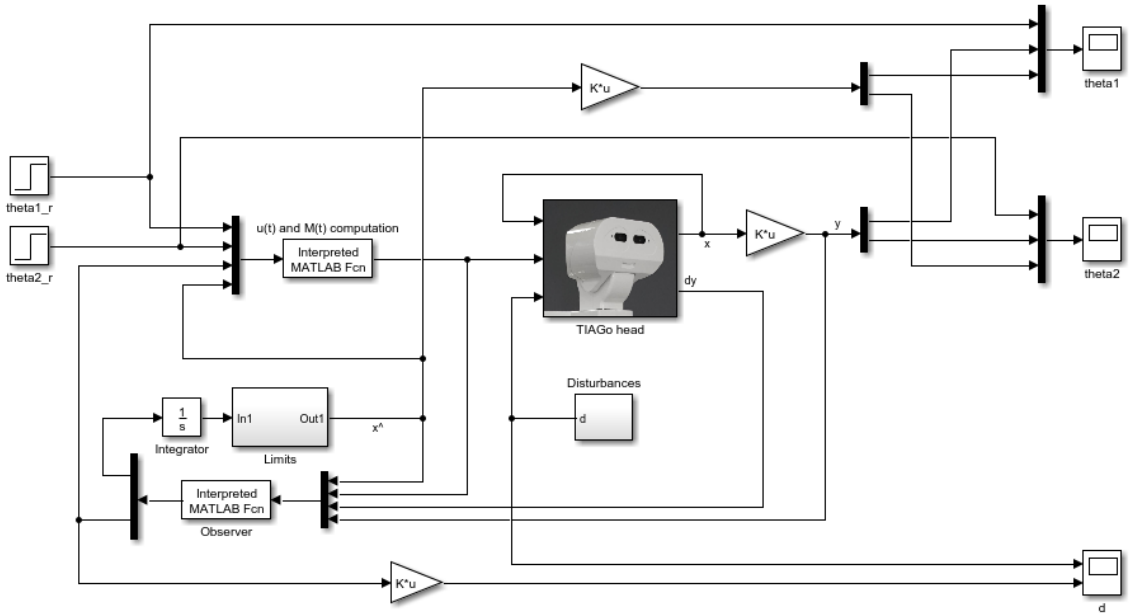


Figure 7.7: *Case III. Simulink scheme*

The simulation conditions for such model are the same as for the previous ones apart from the presence of constant additive disturbances acting on both joints angles. As an example, their values have been set respectively to 0.3 and -0.3.

As shown in Fig.7.8, the new observer perfectly accomplish the disturbance estimation task.

Moreover, the comparison between the real and estimated trends of the two angles θ_1 and θ_2 is reported in Fig.7.9 and Fig.7.10, demonstrating once again a perfect estimation and control of the head orientation.

Note that the reference value is reached by both joints angles, in spite of the disturbance, proving the validity of the proposed fault tolerant scheme.

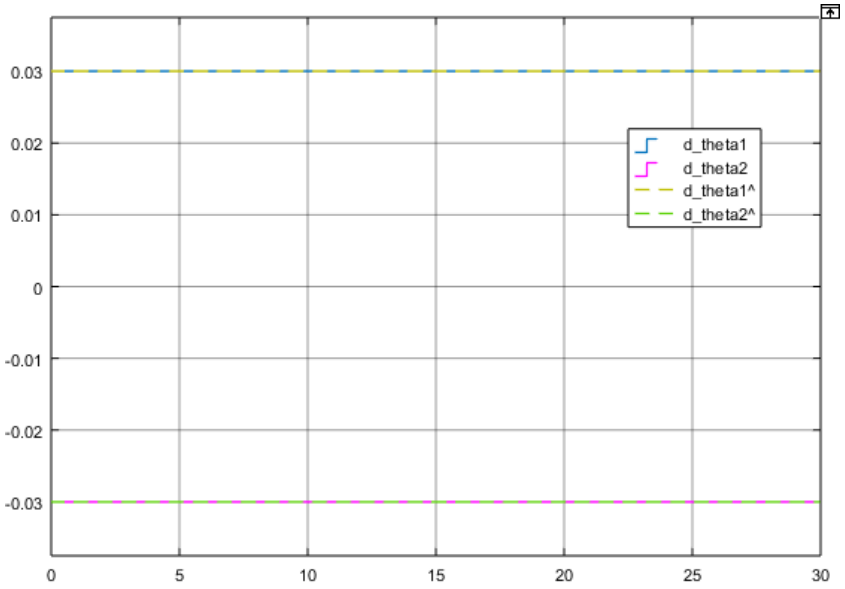


Figure 7.8: *Case III. Real and estimate θ_2 trend*

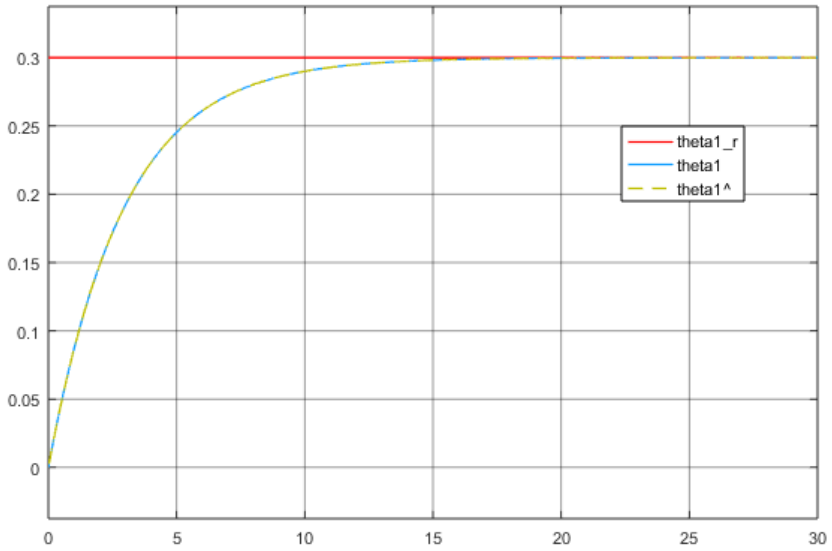


Figure 7.9: *Case III. Real and estimate θ_1 trend*

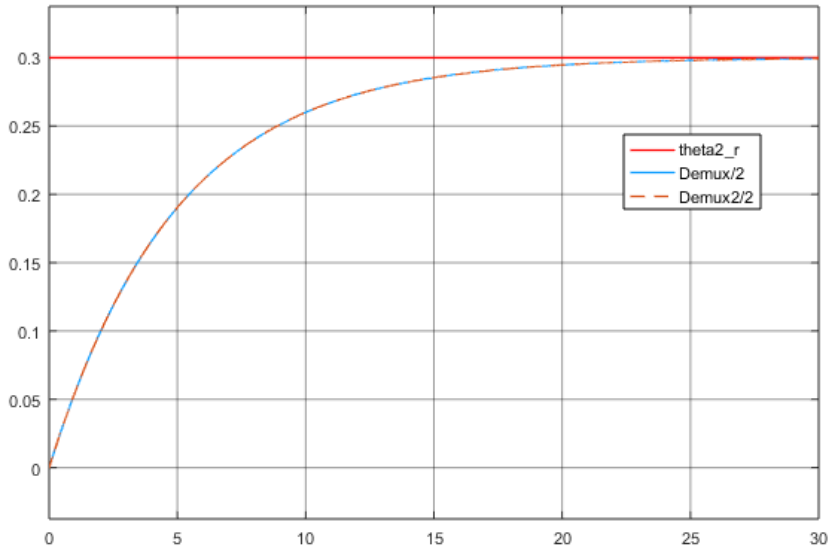


Figure 7.10: *Case III. Real and estimate θ_2 trend*

Chapter 8

Socioeconomic impact

The present chapter deals with an overview of possible social and economics impacts of introducing systems like the one proposed in this work into the market.

Rapid advances in technology have led to a surge of public interest in automation and robotics. Indeed, the number of robots being used by businesses to boost productivity has increased rapidly in recent years and there is no reason to believe that this pace of robotization will begin to slow any time soon.

On the contrary, as the cost of robots continues to fall while their capabilities go up, and with the robot density in most industries still relatively low, the International Federation of Robotics (IFR) anticipates that yearly robot installations will continue to grow at double-digit rates for the time being.

Considerations like those have lead part of public opinion and various scholars to paint a dark picture of what could happen if machines are able to entirely substitute for jobs, resulting in downward pressure on the wages of low-skilled workers and increasing returns to owners of capital (Sachs and Kotlikoff 2012), (Berg, Buffie and Zanna 2016). But even these scholars agree that the link between automation and wage inequality – and the probability of a downward spiral – are not a given.

On the contrary, there is ample evidence that automation does not lead to job substitution, but rather to a re-allocation of both jobs and tasks in which robots complement and augment human labour by performing routine or dangerous tasks.

As a matter of fact, industrial robots have until recently been separated from humans

-often by physical cages. Due to recent advances in technology, a newer trend, which is also spilling out of the factory into non-manufacturing sectors and into the home, is for collaborative robots that respond to and work alongside humans safely. These collaborative robots are not replacing human work, but are increasing the productivity of human workers, whilst simultaneously reducing the risk of workplace injury – for example due to repetitive heavy lifting. Humans are still needed to carry out tasks such as refinishing and quality check, together with those requiring high levels of creativity, empathy, persuasion or an understanding of which knowledge to apply in which situation to reach a productive decision.

As well as industrial robots, the category of service robots is predicted to grow rapidly in both professional and domestic usage. The IFR projects an increase of 42 million service robots for personal and domestic use between 2016 and 2019 in categories such as floor cleaning, lawn-mowing, entertainment and elderly assistance.

Among others, health care is a particularly promising sector for service robots, with applications ranging from exoskeletons that enable workers to deal ergonomically with heavy loads as well as recover from injury or substitute for limbs that are no longer mobile - to robot-assisted surgery.

Systems like the one proposed in this thesis perfectly fits in this scenario as they are part of those technology advances above-mentioned, that are radically changing the way people and machines interact.

Indeed, the collaborative environment described till now requires high level safety technology that allows humans and robots to share the same workspace with less risk of human injury.

From this perspective it is evident the essential role of sophisticated sensors, safety controllers and communication networks embedded in the robot, that provide real-time data allowing robots to automatically respond to potential incidents (such as coming into contact with a person) or faults and, even, recover the system integrity.

Chapter 9

Project budget

In this chapter, we provide some considerations about the development and implementation of the control and fault estimation scheme proposed in this work from an economic point of view. To this aim, a basic cost analysis is presented, considering a possible introduction of such system in the market.

Note that, as a controller is already installed in TIAGO's head, the robot is already provided with the necessary sensors and actuators -i.e. encoders and motors for both joints- by default. Hence, no hardware implementation cost has to be paid.

The unitary cost of the system would be composed by:

1. the cost of the master computer in which the algorithms runs (C_PC): 800€
2. the cost of development, which is in turn composed by:
    - Software development (C_SW): 20000€
    - Technical support (C_Ts): 3000€
    - Tests (C_Tests): 1000€
    - General costs, such as water, electricity and more (C_Gen): 2000€

Thus, the total unitary cost of equipping a the head of a specimen of the robot TIAGo with the proposed scheme can be calculated as:

$$C_{tot} = C_{PC} + C_{SW} + C_{Ts} + C_{Tests} + C_{Gen}$$

resulting in 27000€ approximately.

Taking into account that the robot prize varies between 30000€ and 60000€, depending on the chosen version, the proposed system can be considered too expensive to be implemented on a single specimen.

However, the system prize would be substantially reduced by equipping more robots with it. Indeed, making a largely underestimate forecast of 100 specimens, the unitary cost in Eq.9.1 would reduce to approximately 1000€, not affecting so significantly the overall robot price.

$$C_{tot} = C_{PC} + \frac{C_{SW} + C_{Ts} + C_{Tests} + C_{Gen}}{100} \tag{9.1}$$

Lastly, it is important underlying that in order to obtain a precise cost estimation, a deeper cost analysis should be carried out.

Conclusions

In recent years the use of robots in our daily lives has increased rapidly and there is no reason to believe that this pace of robotization will begin to slow any time soon.

As a matter of fact, they can easily accomplish many of those tasks that can be referred to as the four Ds —too Dangerous, too Dull, too Dirty, and too Difficult— to be done by humans.

For such reason, and thanks to recent advances in technology, collaborative robots, that respond to and work alongside humans safely, are spilling out of the factory into non-manufacturing sectors and into the home.

Nevertheless, robotic systems are prone to different types of faults, which have the potential to affect the efficiency and the safety of the robot and/or its surroundings. That is why, FDD (Fault Detection and Diagnosis) techniques are getting a more and more prominent role in robotics, allowing humans and robots to share the same workspace with less risk of human injury.

Based on such considerations, in this master thesis, the problem of supervision of an humanoid robot has been addressed, taking as reference the mobile manipulator robot TIAGo by PAL robotics. Particularly, the focus has been put on controlling and supervising its head motion.

With this aim in mind, a dynamic model of the robot head has to be developed by means of the Newton-Euler algorithm. It has been possible to apply such approach as the robotic head in analysis has been likened to a 2-dof manipulator. The resulting equations have been verified by implementing the above mentioned algorithm by means of the MATLAB Robotic Toolbox.

Hence, an LPV feedback control scheme has been designed, together with the cor-

respondent feedforward compensation, in order to make the robotic head reaching a reference orientation given in terms of the two joints angles.

Then, this scheme has been improved to account for a more realistic situation: the lack of some of necessary sensors has been embedded into the control system by designing an observer able to estimate the value of the not measurable states.

A further upgrade has been implemented later through the design of an LPV UIO (Unkown Input Observer), in charge for the estimation of not only the not measurable states, but also the eventually present fault/disturbance acting on the robot head.

At this point, a fault tolerant control scheme has been implemented to compensate the faulty effect, detected and isolated by the fault detection and isolation system explained till now.

9.1 Future work

During the development of this thesis many interesting follow-up have emerged.

Some of them are reported in the following as suggestions for possible enlargements of this work:

- to go through the computations related to the system modeling by means of the Lagrangian approach (see Section3.1), keeping in mind that a result analogous to the Newton-Euler approach one is expected;
- to estimate and account for some frictional terms into the robot head dynamic model;
- to expand the LPV model obtained in Chapter4, to encompass a third time-varying parameter, for example the robot head mass. Such idea follows from the real need of embedding a voice command apparatus on TIAGo head, which obviously would modify the system mass;
- to investigate the impact of applying the filter proposed in Chapter5 in presence of disturbances.

Bibliography

[1] World Robotics. 2010. [En ligne]. Retrieved from http://www.worldrobotics.org/.

[2] E. Guizzo. 2010. World robot population reaches 8.6 million. *IEEE Spectrum.*

[3] IFR. 2016. *Executive Summary World Robotics. 2016 Service Robot.* The International Federation of Robotics (IFR).

[4] IFR. 2016. *Executive Summary World Robotics. 2016 Industrial Robots.* The International Federation of Robotics (IFR).

[5] G. Steinbauer. 2013. A survey about faults of robots used in robocup. In *RoboCup 2012: Robot Soccer World Cup XVI.* Springer, Berlin, 344–355.

[6] B. S. Dhillon. 1991. *Robot Reliability and Safety.* Springer.

[7] J.-H. Shin and J.-J. Lee. 1999. Fault detection and robust fault recovery control for robot manipulators with actuator failures. In *Proceedings of the IEEE International Conference on Robotics and Automation.*

[8] Eliahu Khalastchi and Meir Kalech. 2018.*On Fault Detection and Diagnosis in Robotic Systems.* ACM Comput. Surv. 51, 1, Article 9 (January 2018), 24 pages.

[9] R. Brooks. 1986. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation 2*, 14–23.

[10] http://tiago.pal-robotics.com

[11] I. Jolliffe. 2005. *Principal Component Analysis.* Wiley Online Library.

[12] R. Isermann. 2005. Model-based fault-detection and diagnosis—status and applications. *Annual Reviews in Control* 29, 71–85.

[13] R. Akerkar and P. Sajja. 2010. *Knowledge-Based Systems.* Jones and Bartlett Publishers.

[14] R. Isermann. 2005. Model-based fault-detection and diagnosis—status and applications. *Annual Reviews in Control* 29, 71–85.

[15] S. Zaman, G. Steinbauer, J. Maurer, P. Lepej, and S. Uran. 2013. An integrated model-based diagnosis and repair architecture for ROS-based robot systems. In *Proceedings of the International Conference on Robotics and Automation (ICRA'13).*

[16] D. Stavrou, D. G. Eliades, C. G. Panayiotou, and M. M. Polycarpou. 2016. Fault detection for service mobile robots using model-based method. *Autonomous Robots* 40, 383–394.

[17] M. Hashimoto, H. Kawashima, and F. Oba. 2003. A multi-model based fault detection and diagnosis of internal sensors for mobile robot. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'03).*

[18] R. Reiter. 1987. A theory of diagnosis from first principles. *Artificial Intelligence* 32, 57–95.

[19] V. J. Hodge and J. Austin. 2004. A survey of outlier detection methodologies. *Artificial Intelligence Review 22*, 85–126.

[20] V. Chandola, A. Banerjee, and V. Kumar. 2009. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)* 41, 15.

[21] G. Fagogenis, V. De Carolis, D. M. Lane. 2016. Online fault detection and model adaptation for underwater vehicles in the case of thruster failures. In *Proceedings of the International Conference on Robotics and Automation (ICRA'16).*

[22] E.Falotico et al. 2011. *Using trunk compensation to model head stabilization during locomotion*

[23] John J. Craig *Introduction to Robotics*

[24] Shamma, J. F. and J. R. Cloutier (1992). A linear parameter-varying approach to gain scheduled missile autopilot design. In *Proc. American Control Conf*, Chicago, IL, pp. 1317-1321.

[25] Apkarian, P. and P. Gahinet (1995). *Self-scheduled H_∞ Control of Linear Parameter-varying Systems: a Design Example.*

[26] Becker, G., A. Packard, D. Philbrick and G. Balas (1993). Control of parametrically-dependent linear systems: a single quadratic Lyapunov approach. In *Proc. American Control Conf*, San Francisco, CA, pp. 2795-2799.