Chapter 2

Fault-tolerant Control of a Service Robot

Alberto San Miguel, Vicenç Puig and Guillem Alenyà [1]

In this chapter, the problem of fault-tolerant control of a service robot is addressed. The proposed approach is based on using a fault estimation scheme based on an Robust Unknown-Input Observer (RUIO) that allows to estimate the fault as well as the robot state. This fault estimation scheme is integrated with the control algorithm that is based on a observer-based state feedback control. After the fault occurrence, from the fault estimation, a feedforward control action is added to the feedback control action to compensate the fault effect. To cope with the robot non-linearity, its non-linear model is transformed into a Takagi-Sugeno model. Then, the state-feedback and RUIO are designed using an LMI-based approach considering a gain-scheduling scheme. To illustrate the proposed fault-tolerant scheme a mobile service robot TIAGo, developed by PAL robotics, is used.

2.1 Introduction

Over the last few years service robots have been increasingly introduced in our daily lives (see as e.g. Figure 2.1). According to the International Federation of Robotics (IFR), since 2016 there has been a yearly increase of 15% on its sales [1].

Although service robots have been designed to successfully perform tasks on highly dynamic and unpredictable anthropic domains some faults can appear. A wide range of faults can be considered regarding interaction, such as misleading interpretations of the human actions or unexpected scenarios beyond nominal operation. Also, their inherent complexity make them prone to failures at all their levels, from the low-level actuators and sensors to the high-level decision layers. All these factors can lead to a degradation on the performance of the robot or imply critical damage to it, that might even jeopardise its safety. Thus, their ability to autonomously overcome most of these situations in a safe and efficient manner must play a fundamental role in their implementation.

[1]Institut de Robòtica i Informàtica Industrial (CSIC-UPC), Llorens i Artigas, 4-6, 08028 Barcelona, Spain.
This work is supported by the Spanish State Research Agency through the María de Maeztu Seal of Excellence to IRI MDM-2016-0656

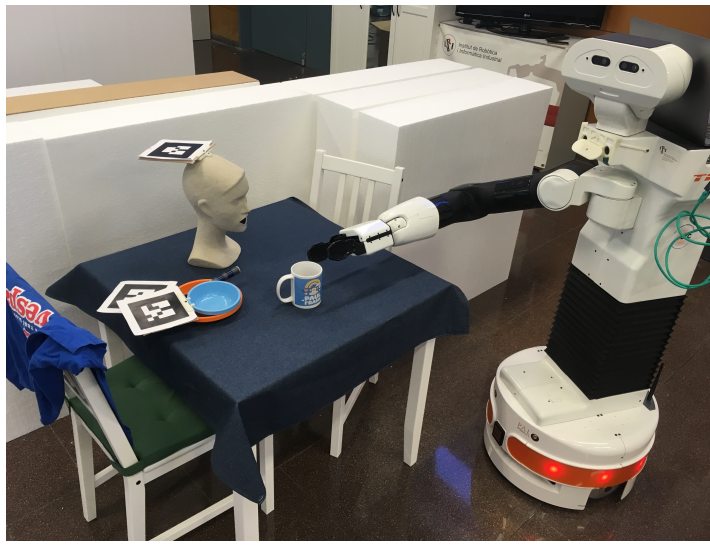

Figure 2.1 TIAGo robot in a domestic environment.

2.1.1 State of the art

Fault Detection and Diagnosis (FDD) field has been widely studied for many years on classic control problems [2]. Only on certain generic robotic platforms some of these approaches have been successfully applied, like for wheeled mobile robots [3], being still considered as a relative new field of study for robotic systems.

Current FDD techniques for robots can be classified into three different categories according to their common key characteristics [4]:

Data-driven: rely on the extraction and processing of data from different parts of the system, in order to detect and determine a faulty behaviour. One example is the work presented in Reference [5], where a Neural Network is trained and integrated within a Sliding Mode Control structure to enhance its robustness.

Model-based: depend on *a priori* analytical model that depicts the behaviour of a certain fault in the system or the nominal behaviour of the system itself. By comparison of the expected performance given by the model against the current one, faults can be detected and isolated. In Reference [6], a residual is computed using a dynamical model of a 7-DOF robotic arm to detect and obtain information on unexpected collisions to determine suitable reaction strategies.

Knowledge-based: mimic the behaviour of a human expert, directly associating certain evidences with their corresponding faults. Some of this methods are seen as hybrid techniques that combine Data-Driven and Model-Based approaches. On this line, in Reference [7], a two-layered structure is used, where a model aims at detection and a decision tree (Data-driven) at isolation of different actuator faults.

On the suggested classification, key advantages and weaknesses of the different approaches are worth to be pointed out. It should be mentioned that this classification does not draw clear boundaries between the different groups but determine certain general characteristics that are usually present, which have been considered on this remark.

The main drawback of Model-based techniques is the use of a model itself. For some robotic platforms is extremely complex to determine analytic expressions that describe their behaviour or establish relationship between their components. This issue is even more significant when trying to describe interactions with the environment or the effect of a fault in the unaffected parts, for example. Here Data-Driven approaches present their main advantage, as no knowledge about the robot is assumed and relies on data of the particular robot where the method is applied to. But most Data-Driven methods require a high computational expense which might make them unfeasible for an on-board and on-line implementation that some robots could require (e.g. an spatial exploration robot). When a learning phase (supervised or unsupervised) is involved, part (or all) of it is carried out offline, as in Reference [8] where high dimensional data of a robotic arm is recorded to obtain dimensional reduction transformations and train a binary Support Vector Machine (SVM) model to be used online. It should be considered that online learning allows to obtain a dynamic method able to capture unexpected behaviours. Computational cost issue can also appear on some Model-based methods but is usually overcome by simplification or reformulation of the model, as in Reference [9], where a modification of the classical Newton-Euler is introduced in order to reduce its computation time for on-line execution purposes. Some Knowledge-based approaches that consist on combining Data and Model-Driven methods are able to establish a trade-off between the discussed issues, and suggest feasible solutions for FDD on robotic systems.

Regarding the characteristics that are required for service robots, autonomy is one of the most relevant ones. Service robots have to usually perform without the intervention of a human that supervises its actions or include a human in its operation loop that is not a robotic expert [10]. Thereby, being able to detect faults, identify them and overcome their effect on the execution of a task is crucial to achieve autonomy. As FDD methods should operate on a supervision level concurrently with all the techniques used for the desired performance, the computational demand on the method plays an important role on its implementation. High demanding processes might interfere with others and be the source of faults themselves. Thus, according to the discussion above, Model-Based approaches are preferred, although Data-Driven methods can be also applied if they have a low-computational burden or some of its implementation is carried out offline.

2.1.2 Objectives

In this chapter, the problem of fault-tolerant control of the TIAGo humanoid robot by PAL Robotics [11] is addressed. Specifically and as a proof of concept, the focus has been put into its 2-DOF head subsystem, to tackle the scenario where external forces (e.g. a mass, a contact with a human) generate torques on its joints such that desired configurations aren't reached.

Figure 2.2 TIAGo Robotic Platform by Pal Robotics.

The proposed Model-based approach relies on a Takagi-Sugeno (TS) model of the robot, in order to cope with its non-linearity. From it, a state-feedback plus feed-forward control strategy will be designed using a fault estimation scheme based on a Robust Unknown-Input Observer (RUIO). To obtain the desired performance of the implemented method, a combination of Linear Matrix Inequalities and Equalities (LMI and LME, respectively) is used for the design. Additionally, the affecting fault is estimated using a reference model and its compensation will be included within the control loop, in order to overcome possible errors derived from the design process. All the procedures will be developed in the discrete-time domain, in order to bring the implementation on the real platform together with the proposed approach.

2.2 Takagi-Sugeno Model

2.2.1 Robot model

The method presented in this chapter considers the implementation of a fault detection and isolation (FDI) scheme by means of a Model-based approach. Thereby, an analytic model which describes the behaviour of our system has to be determined. As aforementioned, the target system on this chapter is the 2-DOF Head subsystem of the TIAGo robotic platform, presented in Figure 2.3. This type of systems are usually named Pan and Tilt structures, where the Pan movement corresponds to the rotation around an axis and the Tilt to the rotation around its perpendicular one.

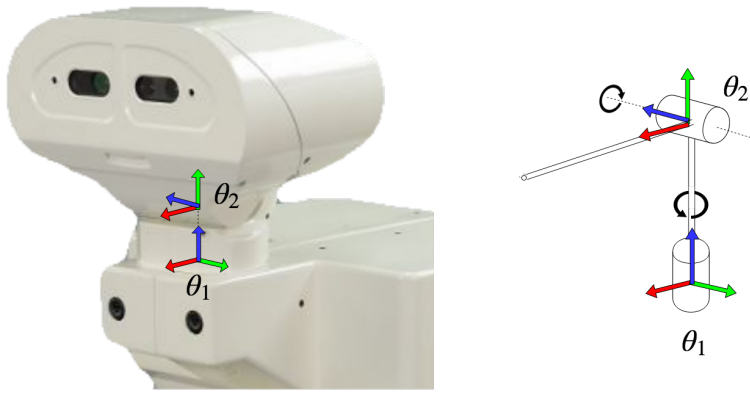

Figure 2.3 TIAGo's head subsystem representation as a two-manipulator link.

To obtain this analytic model, the Newton-Euler method [12] has been applied considering the system as a two-link manipulator with two rotational joints θ_i ($i = 1,2$), represented on the left part of Figure 2.3. For the sake of brevity this process is omitted, presenting only on this chapter the final expression for the model's dynamics.

The model can be stated in the so called configuration-space form, which gives the joints torque vector τ as a function of $\ddot{\theta}$, $\dot{\theta}$ and θ, which are the joint acceleration, velocity and position vectors

$$\tau = M(\theta)\ddot{\theta} + B(\theta)[\dot{\theta}\dot{\theta}] + C(\theta)[\dot{\theta}^2] + G(\theta) \tag{2.1}$$

where $M(\theta)_{n\times n}$ describes the mass matrix of the manipulator, $B(\theta)_{n\times n(n-1)/2}$ the Coriolis terms, $C(\theta)_{n\times n}$ the centrifugal coefficients and $G(\theta)_{n\times 1}$ the gravity effects; being the number of joints $n = 2$.

Applying Equation (2.1) on TIAGo's head model:

$$\begin{aligned}\begin{bmatrix}\tau_1\\ \tau_2\end{bmatrix} &= \begin{bmatrix}I_{zz_1}+m_2d_4^2+I_{xx_2}c_2^2+I_{yy_2}s_2^2-m_2d_4^2c(2\theta_2)^2 & 0\\ 0 & I_{zz_2}+m_2d_4^2\end{bmatrix}\begin{bmatrix}\ddot{\theta}_1\\ \ddot{\theta}_2\end{bmatrix}\\ &+\begin{bmatrix}-2c_2s_2(I_{xx_2}-I_{yy_2})+m_2d_4^2s(4\theta_2)\\ 0\end{bmatrix}\dot{\theta}_1\dot{\theta}_2\\ &+\begin{bmatrix}0 & 0\\ c_2s_2(I_{xx_2}-I_{yy_2})-\frac{1}{2}m_2d_4^2s(4\theta_2) & 0\end{bmatrix}\begin{bmatrix}\dot{\theta}_1^2\\ \dot{\theta}_2^2\end{bmatrix}+g\begin{bmatrix}0\\ -m_2d_4s(2\theta_2)\end{bmatrix}.\end{aligned} \tag{2.2}$$

Terms in the form I_{a_i}, where $a = xx, yy, zz$ and $i = 1,2$, correspond to the inertial tensor diagonal values of the links.

In order to simplify the expression into a shorter more intuitive manner, all the expressions have been arranged into constant and variable-dependant terms, obtaining:

$$\begin{bmatrix}\tau_1\\ \tau_2\end{bmatrix} = \begin{bmatrix}\alpha+\beta(\theta_2) & 0\\ 0 & \xi\end{bmatrix}\begin{bmatrix}\ddot{\theta}_1\\ \ddot{\theta}_2\end{bmatrix}+\begin{bmatrix}\delta(\theta_2)\\ 0\end{bmatrix}\dot{\theta}_1\dot{\theta}_2+\begin{bmatrix}0 & 0\\ \eta(\theta_2) & 0\end{bmatrix}\begin{bmatrix}\dot{\theta}_1^2\\ \dot{\theta}_2^2\end{bmatrix}+g\begin{bmatrix}0\\ \lambda(\theta_2)\end{bmatrix}. \tag{2.3}$$

From this form, the model equations of the head subsystem can be derived for the joint accelerations and velocities:

$$\begin{cases}\ddot{\theta}_1 = -\frac{\delta(\theta_2)}{\alpha+\beta(\theta_2)}\dot{\theta}_1\dot{\theta}_2+\frac{1}{\alpha+\beta(\theta_2)}\tau_1\\ \ddot{\theta}_2 = -\frac{\eta(\theta_2)}{\xi}\dot{\theta}_1^2+\frac{1}{\xi}\tau_2-\frac{\lambda(\theta_2)}{\xi}\\ \dot{\theta}_1 = \frac{d}{dt}\theta_1\\ \dot{\theta}_2 = \frac{d}{dt}\theta_2.\end{cases} \tag{2.4}$$

Considering the complete equations and arrangements from Equations (2.2) and (2.3), terms can be further arranged as follows: (2.5), (2.6) and (2.7)

$$\delta(\theta_2) = -2\eta(\theta_2), \tag{2.5}$$

$$\gamma(\theta_2) = \alpha+\beta(\theta_2), \tag{2.6}$$

$$\varphi(\theta_2,\dot{\theta}_1) = \delta(\theta_2)\dot{\theta}_1. \tag{2.7}$$

The non-linear model from Equation (2.4) can be expressed in the state-space linear-parameter varying formulation (considering the states as scheduling parameters), according to its general expression:

$$\begin{cases}x(t+1) &= A(x)x(t)+B(x)u(t)+d(x,t)\\ y(t) &= C(x)x(t)+D(x)u(t).\end{cases} \tag{2.8}$$

Regarding TIAGo's physical system deployment, it is considered the input action u as the joints torque τ and the system output y as the joint position θ given by internal sensor measurements. The state vector x has been defined as $[\dot{\theta}_1\ \dot{\theta}_2\ \theta_1\ \theta_2]^T$, so Equations (2.4) and term arrangements from (2.5) to (2.7) can be stated according to (2.8), obtaining the following model representation:

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

$$\begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix} = \begin{bmatrix}0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \cdot \begin{bmatrix}\dot{\theta}_1\\ \dot{\theta}_2\\ \theta_1\\ \theta_2\end{bmatrix} \tag{2.9}$$

In order to ease the development of the system model, matrix $d(x,t)$ will be neglected until Section 4, where its effect will be further analysed.

It should be also pointed out that the physical model imposes limits on the joint variables, summarised in Table 2.1, on which the Takagi-Sugeno model will be developed on the next subsection.

Table 2.1 Value limits for the TIAGo head subsystem variables.

State	**Min**	**Max heads**
$\dot{\theta}_1$	0 rad/s	3 rad/s
$\dot{\theta}_2$	0 rad/s	3 rad/s
θ_1	-75 ^{o}	75 ^{o}
θ_2	-60 ^{o}	45 ^{o}

2.2.2 Takagi-Sugeno formulation

Fuzzy logic, introduced by L.A.Zadeh [13], in contrast with classical Boolean logic does not define the output of a decision as binary (*Yes*/*No*, $1/0$), but gives a degree of belonging of the output to the extreme values of the decision, according to a set of rules. This definition has been widely used in the Artificial Intelligence field as an approximation of the human decision-making process, where in most cases admits a range of possibilities between the limits.

Takagi-Sugeno Models [14] (henceforth TS Models), named after their designers, apply the concept of fuzzy logic to the description of non-linear dynamics of the systems by a set of rules, associated to linear descriptions. Overall model is obtained by *blending* this linear systems according to their rules.

In our case, the TS Model starts with the obtained system representation from Equation (2.9), which includes non-linearities that make the direct application of classical control strategies unfeasible. Firstly, non-linear terms are embedded on the so called premise variables z_i. For our particular case, these terms have been already arranged in (2.6) and (2.7), being $z_1 \equiv \gamma(\theta_2)$ and $z_2 \equiv \varphi(\theta_2, \dot{\theta}_1)$.

The concept of sector non-linearity has been used in the construction of the TS model to assure its exact representation. Its objective is to find sectors in the system state space where the non-linear behaviour lies. As these terms include parameters and variables which are bounded in the physical system, sectors can be defined in local regions, delimited by these bounds. From the limits of $\dot{\theta}_1$,$\dot{\theta}_2$,θ_1 and θ_2 presented in Table 2.1, our premise variables bounds can be found, included in Table 2.2.

Table 2.2 Upper and lower bounds of the premise variables for the TS formulation.

Premise variables	**Min**	**Max**
$z_1 \equiv \gamma(\theta_2)$	0.0055	0.0091
$z_2 \equiv \varphi(\theta_2, \dot{\theta}_1)$	-0.0110	0.0110

Thereby, (local) sector non-linearity approach [16] is applied to reformulate the premise variables according to their limits, by means of the membership functions. These functions represent the degree of belonging to the upper or lower bounds according to a certain trend, defined in the fuzzy logic field as *fuzzy sets*. For this problem, linear membership functions have been considered, formulated for the *i*th premise variable as follows:

$$z_i = M_{i,1}(z_i)\bar{z_i} + M_{i,2}(z_i)\underline{z_i} \text{ where } \quad M_{i,1}(z_i) = \frac{\bar{z_i} - z_i}{\bar{z_i} - \underline{z_i}}, \quad M_{i,2}(z_i) = \frac{z_i - \underline{z_i}}{\bar{z_i} - \underline{z_i}}. \quad (2.10)$$

Figure 2.4 presents the graphical representation of Equation 2.10.

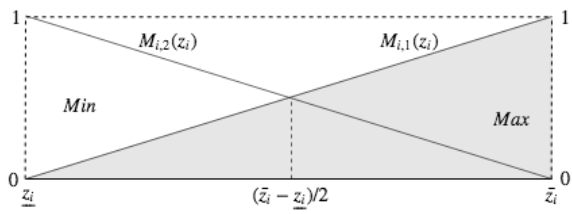

Figure 2.4 Graphical representation of the linear membership function.

Aforementioned fuzzy rules can be stated using the defined membership functions. These rules have the form of $IF-THEN$ structures where premise variables z_i are evaluated w.r.t. the membership functions. The number of rules N is equal to 2^p, being p the number of chosen premise variables, as they consider all the possible permutations between the limits of z_i. Thereby, each rule is associated with a linear system that describes the behaviour of the *freezed* original system on the corresponding limits. For the addressed system, its fuzzy rules and associated linear systems have been stated below.

Model rule 1

$$\text{IF } z_1 \text{ is "}Max\text{" and } z_2 \text{ is "}Max\text{" THEN } x(t+1) = A_1 x(t) + B_1 u(t)$$

$$A_1 = \begin{bmatrix} 0 & -\bar{\varphi}/\bar{\gamma} & 0 & 0 \\ \bar{\varphi}/2\xi & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \qquad B_1 = \begin{bmatrix} 1/\bar{\gamma} & 0 \\ 0 & 1/\xi \\ 0 & 0 \\ 0 & 0 \end{bmatrix},$$

Model rule 2

$$\text{IF } z_1 \text{ is "}Min\text{" and } z_2 \text{ is "}Max\text{" THEN } x(t+1) = A_2 x(t) + B_2 u(t)$$

$$A_2 = \begin{bmatrix} 0 & -\underline{\varphi}/\bar{\gamma} & 0 & 0 \\ \underline{\varphi}/2\xi & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \qquad B_2 = \begin{bmatrix} 1/\bar{\gamma} & 0 \\ 0 & 1/\xi \\ 0 & 0 \\ 0 & 0 \end{bmatrix},$$

Model rule 3

$$\text{IF } z_1 \text{ is "}Max\text{" and } z_2 \text{ is "}Min\text{" THEN } x(t+1) = A_3 x(t) + B_3 u(t)$$

$$A_3 = \begin{bmatrix} 0 & -\bar{\varphi}/\underline{\gamma} & 0 & 0 \\ \bar{\varphi}/2\xi & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \qquad B_3 = \begin{bmatrix} 1/\underline{\gamma} & 0 \\ 0 & 1/\xi \\ 0 & 0 \\ 0 & 0 \end{bmatrix},$$

Model rule 4

$$\text{IF } z_1 \text{ is "}Min\text{" and } z_2 \text{ is "}Min\text{" THEN } x(t+1) = A_4 x(t) + B_4 u(t)$$

$$A_4 = \begin{bmatrix} 0 & -\underline{\varphi}/\underline{\gamma} & 0 & 0 \\ \underline{\varphi}/2\xi & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \qquad B_4 = \begin{bmatrix} 1/\underline{\gamma} & 0 \\ 0 & 1/\xi \\ 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

Finally, the defuzzyfication step has to be carried out to completely represent the system according to the defined fuzzy rules and sets. This process gives a complete representation of the system behaviour according to its fuzzy rules. Thereby, the system is described by a weighted sum of all the limit systems considered on the fuzzy rules. See [16] for more details.

For the considered TIAGo's head subsystem, this defuzzification process has to be applied only for system's A and B matrices, as they both depend on premise variables:

$$x(t+1) = \sum_{n=1}^{N=4} h_n(z_p(t))[A_n \cdot x(t) + B_n u(t)] \tag{2.11}$$

$$x(t+1) = A(z_p(t))x(t) + B(z_p(t))u(t) \tag{2.12}$$

where

$$h_1(z_p(t)) = M_{1,1}(z_1) \cdot M_{2,1}(z_2) \qquad h_2(z_p(t)) = M_{1,2}(z_1) \cdot M_{2,1}(z_2)$$

$$h_3(z_p(t)) = M_{1,1}(z_1) \cdot M_{2,2}(z_2) \qquad h_4(z_p(t)) = M_{1,2}(z_1) \cdot M_{2,2}(z_2)$$

2.3 Control Design

2.3.1 Parallel Distributed Control

As it has been presented, TS Models are based upon a set of rules which enclose the behaviour of a non-linear system using linear descriptions on its *bounds*. Thereby, its performance can be described at a certain instant by a combination of the membership to these *bounds*. Following this concept, a control strategy for a system can be defined as a set of linear control laws defined at its limit operation points, being the system control at a certain point defined as a combination of these limit controllers. This concept was initially presented in [15] by Kang and Sugeno under the name of Parallel Distributed Compensation (PDC from now on).

The PDC offers a procedure to design a control strategy from a given TS model using linear techniques. For each one of the fuzzy rules defined for the model, a control rule can be stated sharing the same premise variables and their corresponding fuzzy sets (membership functions). For this chapter, a state-feedback control has been used as the linear control strategy. From the already presented TS model for the considered system in this chapter, control rules have been stated as follows:

Control rule 1

IF z_1 is "Max" and z_2 is "Max" THEN $u(t) = -K_1 x(t)$,

Control rule 2

IF z_1 is "Min" and z_2 is "Max" THEN $u(t) = -K_2 x(t)$,

Control rule 3

IF z_1 is "Max" and z_2 is "Min" THEN $u(t) = -K_3 x(t)$,

Control rule 4

IF z_1 is "Min" and z_2 is "Min" THEN $u(t) = -K_4 x(t)$.

As for the TS models, the defuzzyfication step is applied on the control action vector $u(t)$, using the same procedure and h_n functions from the TS model:

$$u(t) = -\sum_{i=1}^{N=4} h_i(z_p(t))[K_i x(t)] = -K(z_p(t))x(t) \tag{2.13}$$

The key point of PDC is to design the feedback control gains K_n assuring stability and a certain number of performance specifications. Although this strategy only implies the definition of the system in the limit operation points (bounds of our premise variables), the design has to consider global design conditions.

According to Lyapunov's theory, global asymptotically stability exists for a set of subsystems if there exists a common positive definite matrix P for all the subsystems such that the following condition holds [16]:

$$A_i^T P A_i - P < 0 \qquad \forall i = 1,2,...N \tag{2.14}$$

Considering the *defuzzified* system expression from Equation (2.11) and the state-feedback control formulation, global asymptotically stable condition is assured if the folowing expressions hold.

$$(A_i - B_i K_i)^T P (A_i - B_i K_i) - P < 0, \qquad \forall i = 1,2,...N \tag{2.15}$$

$$G_{ij}^T P G_{ij} - P < 0, \qquad \forall i < j \leq N \quad \text{s.t.} \quad h_i \cap h_j \neq \phi \tag{2.16}$$

where

$$G_{ij} = \frac{(A_i - B_i K_j) + (A_j - B_j K_i)}{2}.$$

The solution for stated inequalities is found by means of the Linear Matrix Inequalities based approach (LMIs from now on). This method implies the formulation of the problem in a set of inequality constraints that define convex sets. Henceforth, all the design conditions that imply inequalities will be directly formulated as LMIs. The reader is referred to Reference [17] for further details on LMIs, as they are out of the scope of this chapter.

2.3.2 Optimal Control Design

Robotic applications quite often require optimal performance in order to minimise costs in terms of resources and time consumption [12] as well as present some characteristics on their output(s) related with their application. This issue has been

tackled on this chapter through the formulation of an optimal design oriented to achieve certain performance indexes.

Regarding optimality, the design is stated as a Linear Quadratic Control (LQC from now on) problem, stated in the desired discrete time domain. The gains K_i of the state feecbak control (2.13) are found such that the following quadratic performance criterion is minimized:

$$J(t) = \sum_{0}^{\infty} \left[x(t)^T Q x(t) + u(t)^T R u(t) \right], \tag{2.17}$$

where matrix Q allows to control convergence speed of the states towards their references while matrix R is selected to limit the control effort required.

For the considered PDC, the minimisation problem is subjected to a set of LMIs included in Reference [16].In these optimality conditions and on the previously stated ones, there exist a significant number of LMIs to be fulfilled, due to the inclusion of all the possible A_i and B_i permutations. This might arise some tractability and feasibility problems on finding a solution, specially when applied to high order systems [16]. To avoid these issues, the system has been re-stated such that B_i remains constant, by means of the Apkarian filter [18].

Apkarian filtering consists on pre/post applying a fast enough dynamic filter, such that it does not interfere system own dynamics, with the following form:

$$\begin{gathered} \dot{x}_f = A_f x_f + B_f u_f \\ \begin{bmatrix} \dot{\tau}_1 \\ \dot{\tau}_2 \end{bmatrix} = \begin{bmatrix} -\psi & 0 \\ 0 & -\psi \end{bmatrix} \cdot \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} + \begin{bmatrix} \psi & 0 \\ 0 & \psi \end{bmatrix} \cdot \begin{bmatrix} u_{\tau_1} \\ u_{\tau_2} \end{bmatrix}, \end{gathered} \tag{2.18}$$

where ψ represents the filter gain and u_f the new control variable vector. Pre-appliance of this filter has been chosen in the considered case, increasing the system matrices order, as follows:

$$\tilde{A}_i = \begin{bmatrix} A_i & B_i \\ 0 & A_f \end{bmatrix}, \qquad \tilde{B}_i = \begin{bmatrix} 0 \\ B_f \end{bmatrix}, \qquad \tilde{C}_i = \begin{bmatrix} C & 0 \end{bmatrix}. \tag{2.19}$$

As it can be seen, B_i term is allocated on the $\tilde{A}_i$, remaining $\tilde{B}_i$ constant according to ψ for all TS Model rules.

Applied on the considered system in this chapter, the $\tilde{A}_i$ are now formulated in such form. Fuzzy model (and rules) maintain the same structure, as they have been defined only regarding the premise variables definitions.

Considering this new formulation, the aforementioned LQC optimization problem can be stated into the following LMIs [19]:

$$\begin{bmatrix} -Y & Y\tilde{A}_i^T - W_i\tilde{B}_i^T & Y(Q^{1/2})^T & W_i^T \\ \tilde{A}_i Y - \tilde{B}_i W_i & -Y & 0 & 0 \\ Q^{1/2}Y & 0 & -I & 0 \\ W_i & 0 & 0 & -R^{-}1 \end{bmatrix} < 0 \tag{2.20}$$

$$\begin{bmatrix} \gamma I & I \\ I & Y \end{bmatrix} < 0 \tag{2.21}$$

where $Y \equiv P^{-1}$ and $W_i = K_i Y$, stating the problem to minimise γ, which represents the upper bound for the LQC criterion from (2.17).

2.4 Fault and State Estimation

2.4.1 Robust Unknown Input Observer

The main purpose of the work presented in this chapter is to apply a systematic approach for the design of a control strategy for non-linear systems, focusing on those related with robotic platforms. In these cases, besides stability and optimality conditions, some specific performance characteristics are desired. Furthermore, in those environments where robots have to perform along with human beings or even collaborate with them. Then, the robustness and fault-tolerance of the chosen control strategy is essential to avoid behaviours that do not take into account the entropy of the unknown surroundings or might even harm a person. An initial step has been taken to confront this challenge by means of a Robust Observer for Unknown Inputs (RUIO from now on) for TS Models, presented by Chadli and Karimi [20].

The RUIO presents an observer structure for TS Models which allows decoupling its state estimation from the effects of possible unknown disturbances (faults) or inputs that might affect the system. Its design is based on sufficient conditions that assure this behaviour, stated in terms of Linear Matrix Equalities (LMEs) and the already presented LMIs. For this observer, the considered TS model includes the effects of unknown disturbances and inputs as follows:

$$\begin{cases} x(t+1) & = \sum_{i=1}^{N} h_i(z_i)(A_i x(t) + B_i u(t) + R_i d(t) + H_i w(t)) \\ y(t) & = Cx(t) + Fd(t) + Jw(t) \end{cases}, \tag{2.22}$$

where $d(t)$ stands for the unknown input vector and $w(t)$ for the external disturbance vector. R_i and F represents the influence of $d(t)$ in the system behaviour, and H_i and J the influence of $w(t)$.

The existence of a solution for the problem is assured if the following necessary and sufficient condition is fulfilled:

$$rank \begin{bmatrix} CR_v & CH_v & [F\ J] \\ -F_v & -J_v & 0 \end{bmatrix} = rank \begin{bmatrix} F_v & J_v \\ R_v & H \end{bmatrix} + rank[F\ J], \tag{2.23}$$

where F_v and J_v are diagonal matrices with F and J as diagonal terms, and H_v and R_v are the horizontal concatenation of all the H_i and R_i column matrices.

One of the main improvements of the RUIO with respect to previous similar techniques, for example the one presented in Reference [21], is the relaxation on the necessary and sufficient condition(s). As the aforementioned matrices which model the influence of the unknown terms are given from design, they can be adjusted so (2.23) holds considering the C matrix of the system. The RUIO structure has the following form

$$\begin{cases} r(t+1) = \sum_{i=1}^{N} h_i(z_i)(N_i r(t) + G_i u(t) + L_i y(t)) \\ \hat{x}(t) = r(t) - E y(t), \end{cases} \tag{2.24}$$

where $r(t)$ correspond to the RUIO internal state vector and the same set of deffuzification functions $h_i(z(t))$ of the TS model are considered. Matrices N_i, G_i, L_i and E are the observer gains to be determined. The design problem is based on assuring an asymptotic stability of the dynamics of the observer, so the estimation error converges to zero as time tends to infinite, disregarding the magnitude of the unknown inputs and disturbances. According to [21], this behaviour is satisfied if there exist matrices $X_i > 0$, S,V and W_i such that the LMI and LME conditions below hold:

$$\begin{bmatrix} \varphi_i & -V - A_i^T (V + SC)^T - C^T W_i^T \\ -V^T - (V + SC)A_i - W_i C & X_j - V - V^T \end{bmatrix} < 0, \tag{2.25}$$

$$\text{where } \varphi = -X_i + (V + SC)A_i - W_i C + A_i^T (V + SC)^T - C^T W_i^T,$$

$$(V + SC)R_i = W_i F, \tag{2.26}$$

$$(V + SC)H_i = W_i J, \tag{2.27}$$

$$S[F\ J] = 0. \tag{2.28}$$

Observer gains from Equation (2.24) are determined from the found solution according to the following expressions:

$$E = V^{-1} S, \tag{2.29}$$

$$G_i = (I + V^{-1} SC)B_i, \tag{2.30}$$

$$N_i = (I + V^{-1} SC)A_i - V^{-1} W_i C, \tag{2.31}$$

$$L_i = V^{-1} W_i - N_i E. \tag{2.32}$$

It should be noted from Equation (2.22) that the system has not been formulated using the Apkarian filter for the RUIO design. In this case, LMIs only considering system matrices A_i and C, so there is not an increase on the problem complexity due to variant B_i.

2.4.2 Fault concept and design implications

Regarding this robotic platform and the literature insights into faults classification and sources [22], two main types of faults that could affect the TIAGo head subsystem and that could be overcome using the presented scheme have been distinguished:

1. **Sensor measurement error,** existing a difference between the given measures for θ_1 and θ_2 and their real values.
2. **External forces affecting the system** in both rotation axis, that will appear in terms of torques in the rotational joints.

Recalling the obtained expression of the system behaviour from Equation (2.9), there exists a term included in $d(x,t)$ that is described as function of θ_2. From Equation (2.1), it can be noticed that it is derived from the effect of the mass of the second body on the second rotational joint, which produces a torque variable with the distance between its Center Of Gravity (COG) and the joint axis. In Figure 2.5, this phenomenon is graphically depicted.

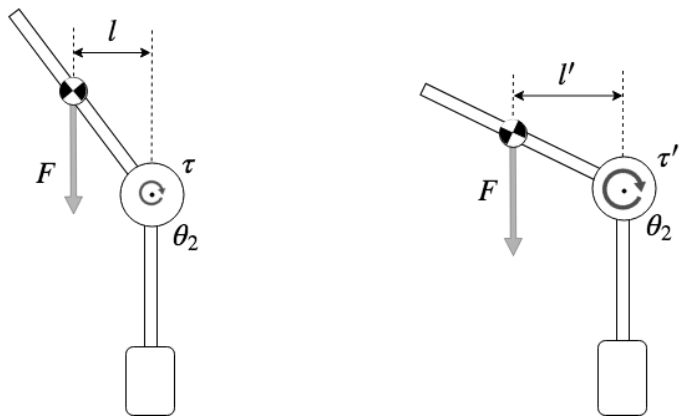

Figure 2.5 Induced torque on θ_2 by the effect of the distance between the COG and the rotation axis.

Thereby, this term can be understood as a second type fault, and only this definition has been considered on the presented work on this chapter. Future work could be carried out on the first one using a similar scheme.

Using this definition, aforementioned R_i, H_i, F and J matrices from the RUIO structure can be designed such that they agree with the effects of the faults. From stated concepts in the RUIO, the fault will affect the system as an unknown input (torque) $d(t)$, being $w(t)$ null (and consequently H_i and J). Matrix F will be also null as the fault does not occur on the system output.

This last consideration does not imply that joint angles will not be affected by faults. By construction, our state variables and its derivatives are joint accelerations, velocities and positions. If $d(t)$ is a torque, affects in the computation of $\ddot{\theta}$, so $\dot{\theta}$ and θ will *perceive* this effect even if it has not been injected directly on their calculation. Thereby, R_i variables have been set as unitary column vectors for all

the N subsystems, as the fault will present the same behaviour disregarding the operational point of the system.

2.4.3 Fault estimation and compensation

The presented RUIO scheme is able to decouple the estimation of the system's states from unknown inputs or disturbances that might degrade the performance of the system, i.e. faults in this work. Using this decoupled estimation, faults are overcame internally by the classical state-feedback mechanism.

This behaviour is highly desired for autonomous robotic platforms as it has been discussed, but also information about fault characteristics could be useful in order to evaluate further measures to be taken according to it. Taking as an illustrative example a robotic arm affected by a sensor fault, information about its effects could lead to an isolation of the faulty sensor or maybe other further measures that allow the robot achieve its goal disregarding the fault (if it isn't critical). But also if the robotic arm is collaborating along with a human and a direct contact, undesired or desired, happens unexpectedly for the robot, information about the detected effects could be crucial to avoid harming the person or to gather data that specifies the task to be performed together.

This latter example where unknown forces might occur agrees with the aforementioned concept of fault considered in this chapter, and justifies the importance of estimating its magnitude. Thus, in this work the estimation of its value and isolation has been studied and included to the developed scheme.

For this purpose a reference control structure is introduced within the presented approach, which consists on the aforementioned state-feedback scheme where the head subsystem has been substituted by a continuous time model implemented according to the system description described on Equation (2.9), but without the $d(x,t)$ term nor other faults. Desired positions of θ_1 and θ_2 for the real system are also set to the reference structure, so the corresponding states for a non-faulty behaviour are obtained. Using the estimation of the states $\hat{x}_{sys}$ from the RUIO and the computation of the system matrices, the value of the disturbance can be obtained as the difference between the reference and the real system:

$$\hat{d}(t) = O_d[x^a_{ref}(t+1) - \hat{x}^a_{sys}(t+1)], \tag{2.33}$$

where O_d matrix is defined so $\hat{d}$ has the components of the disturbance in both axis $\hat{d}_{\theta_1}$ and $\hat{d}_{\theta_2}$, corresponding to the differences on the first and second components of $x(t+1)$, and

$$\begin{aligned} \hat{x}^a_{sys}(t+1) &= A(\hat{x}_{sys})\,\hat{x}_{sys}(t) + B(\hat{x}_{sys})u_{sys}(t) + \hat{d}, \\ x^a_{ref}(t+1) &= A(x_{ref})x_{ref}(t) + B(x_{ref})u_{ref}(t), \end{aligned} \tag{2.34}$$

standing '*sys*' and '*ref*' subindexes for variables of the current system and of the reference structure, respectively. The superindex '*a*' points out that the values at $t+1$ are obtained by means of the analytic model of the TIAGo head subsystem, which is considered as a close enough approximation of the evolution of the states.

The active compensation mechanism consists on the direct injection of the opposite value of $\hat{d}$ into the control action transferred to the head subsystem. Presumably, with this procedure the state-feedback plus RUIO control scheme will regard only for the part of control actions related with the nominal operation (i.e. movement between different setpoints), as the disturbance will be compensated by the injection of its estimation.

2.5 Fault-tolerant Scheme

In this section, all the different components of the presented approach along this chapter are integrated into the general fault-tolerant scheme presented in Figure 2.6.

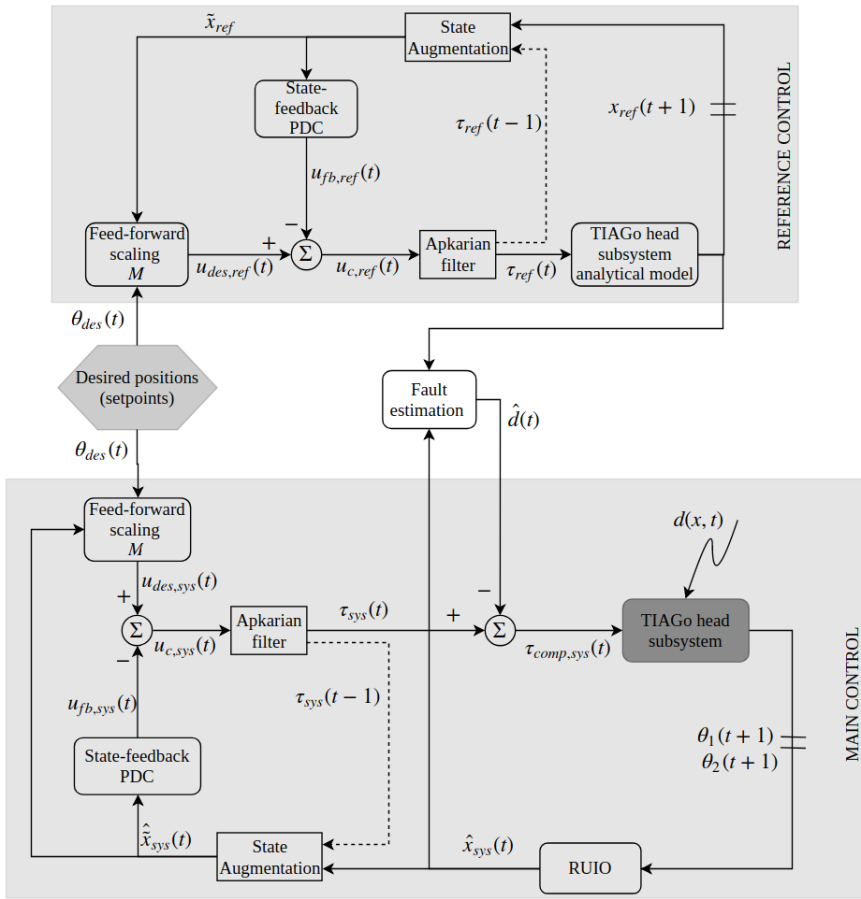

Figure 2.6 Schematic representation of the complete fault-tolerant control approach on discrete time.

At each time instant, the desired positions for both axis θ_{des} are externally given (e.g. by a task planner layer) to both the reference and the main control structures. A corresponding control action is computed for the desired positions by means of a feedforward scaling matrix M [19], which depends on the current states (of the respective structure) according to the following expression:

$$M(z_p(t)) = [\tilde{C}(I + \tilde{B}K(z_p(t)) - \tilde{A}(z_p(t)))^{-1}\tilde{B}]^{-1} \tag{2.35}$$

recalling that $z_p(t)$ corresponds to the set of defined premise variables (computed from $x(t)$) and $K(z_p(t))$ to the state-feedback gain from PDC.

State-feedback PDC computes the feedback control action $u_{fb}(t)$ from the augmented state space taking into account the introduction of the Apkarian filter:

$$\tilde{x}(t) = \begin{bmatrix} x(t) \\ x_f(t-1) \end{bmatrix} = \begin{bmatrix} \dot{\theta}_1(t) \\ \dot{\theta}_2(t) \\ \theta_1(t) \\ \theta_2(t) \\ \tau_1(t-1) \\ \tau_2(t-1) \end{bmatrix}. \tag{2.36}$$

It should be noted that for the main control structure, a decoupled estimation of the states $\hat{x}(t)$ is given by the RUIO, which is the one to be used on the augmented states, as the TIAGo head subsystem only provides as output the angular positions.

The value of the control action $u_c(t)$ is computed as the difference between $u_{des}(t)$ and $u_{fb}(t)$, and the control torque $\tau(t)$ is obtained by applying the discrete time formulation of the Apkarian filter. In parallel, the magnitude of the faults on both axis are estimated according to Equations (2.33) and (2.34) from $\hat{x}_{sys}(t)$ and $x_{ref}(t)$. Their counter values are injected on $\tau_{sys}(t)$, obtaining the compensated control torque $\tau_{comp,sys}(t)$ that is sent to the TIAGo Head Subsystem, affected by $d(x,t)$ (which is assumed to include all possible faults in both axis in this scheme).

Finally, both the TIAGo head real subsytem and its model from the reference structure, update their outputs on the respective injected control torques for the next time instant $t+1$. As already mention before, the real subsystem only provides the value of $\theta_1(t+1)$ and $\theta_2(t+1)$, but for the reference structure all states are known as its part of the implementation.

2.6 Application results

To evaluate the proposed fault-tolerant control approach for the addressed problem, a simulator has been implemented using MATLAB programming environment [23]. The TIAGo head subsystem has been included into the implementation as model in continuous-time, using the analytic expressions and physical parameters and limits of the real components. The Dormand-Prince method [24] has been applied to solve these differential analytic expressions.

For the discrete-time implementation, classical Euler discretization method has been used, for a sampling time $T_s = 10\ [ms]$. Apkarian filter has been designed considering this implementation, and it has been determined that its gain ψ has to be less or equal to $1/T_s$ in order to avoid consistency related problems between samples from consecutive time instants on its application.

The design of the PDC and the RUIO, according to the aforementioned LMIs and LMEs, has been solved using YALMIP toolbox [25] [26] and SEDUMI optimisation software [27]. Additional LMIs regarding pole placement have been included on the design stage, in order to achieve fast response time and zero overshooting on the PDC, and to avoid couplings between RUIO and the subsystem dynamics (10 times faster). The reader is referred to Reference [17] for more details on them.

According to the given description of the faults considered in this chapter, a single scenario is contemplated for all the results to be shown. On the Pan (θ_1) axis, a constant positive torque of $d_{\theta_1} = 4\,N \cdot m$ is injected at $t = 20\,s$ (half of the complete simulation time). For the Tilt (θ_2) one, only the aforementioned variable-dependant effect of the mass of the second link on θ_2 is present during all the simulation. The desired positions for θ_1 and θ_2 are their upper and lower limits, respectively (included in Table 2.1). At $t = 20\,s$, position references change to half of these value.

To demonstrate the effectiveness of the proposed solution, results have been obtained for methods that incrementally incorporate all the presented techniques towards the final fault-tolerant control scheme. Thus, the approach is proved against the results of partial solutions in a *incremental improvement* fashion. Initially, from the basic control structure only including the state-feedback PDC, the feedforward scaling matrix and the Apkarian filter, a classical Luenberger observer [17] is used to estimate the system states. Then, it is substituted by the RUIO, and finally the reference control structure and the fault estimation is included. In this latter case, the fault estimation mechanism is evaluated with and without its compensation.

2.6.1 Basic control structure with Luenberger observer

The design of the Luenberger observer has been performed applying the LQC problem as in the State-Feedback PDC. Also the same aforementioned additional pole placement LMI conditions for the RUIO have been applied. In Figure 2.7, the evolution of the system states along the simulation on an scenario without faults (but maintaining the position references) are included, to be used for determining differences with respect to it.

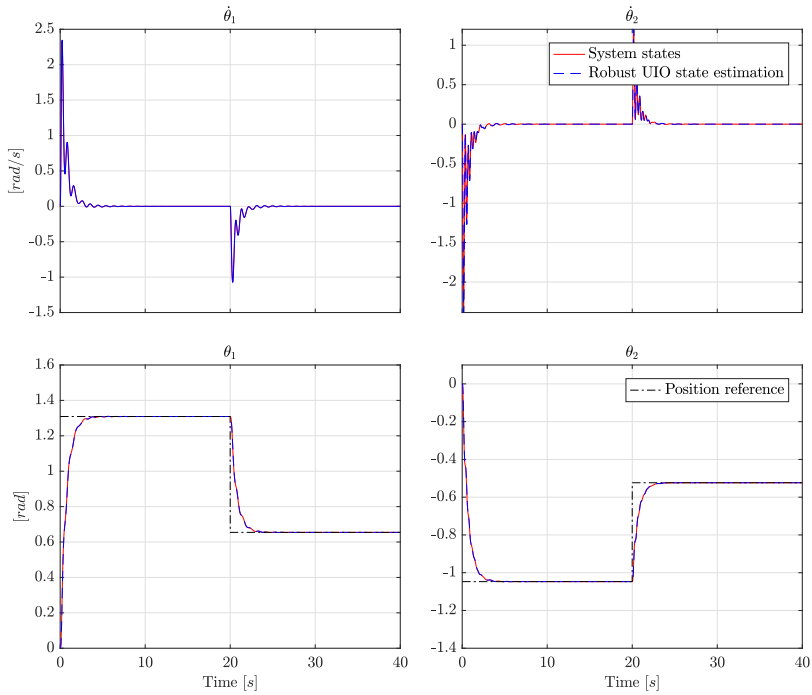

Figure 2.7 System states evolution on a non-fault scenario using the basic control structure with Luenberger observer.

Figure 2.8 shows the same results for the common fault scenario. As expected, the classical Luenberger observer does not decouple the effects of the fault(s) affecting the system in any of the axis, producing a noticeable error in the estimation of the state variables. Also, oscillatory behaviours during the transitory appear in both axis with respect to results on Figure 2.7, due to the injection of unseen faults within the control structure. It should be pointed out that as a result from the coupling effects between both axis, even when there is no fault torque on θ_1, the oscillatory behaviour still remains. Regarding the angular velocities, besides the oscillatory behaviour, there exist a difference between their estimated and real value also produced due to the effect of the fault.

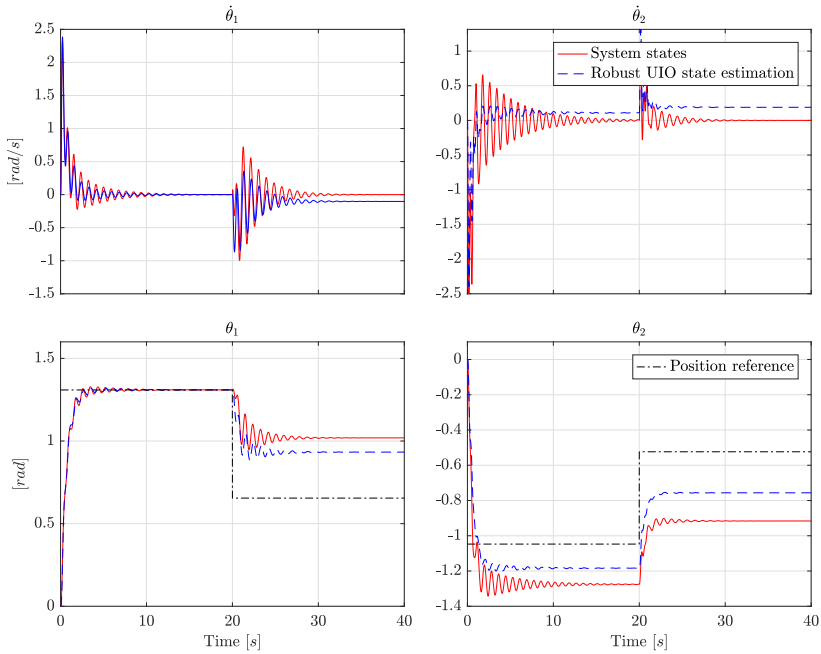

Figure 2.8 System states evolution on fault scenario using the basic control structure with Luenberger observer.

2.6.2 Basic control structure with RUIO

From previous control structure, the substitution of the classical Luenberger observer by the RUIO leads to a overall reduction of the estimation errors along with the cancellation of the oscillatory phenomena on both axis, as it is shown on Figure 2.9.

Although there is a remarkable improvement with respect to the previous control structure, the presented behaviour does not correspond with the expected one as the position errors are not null. The RUIO was supposed to completely decouple the effect of the disturbance from the system states by design, as the LMI conditions assure that the estimation error converges to zero. As in the results on Figure 2.8, a nearly equal estimation error appears on both angular velocities. Further analysis has been performed on this issue, finding that it might be related with the design, as SeDuMi presents limitied capabilities to solve strict LME.

2.6.3 Complete fault-tolerant control scheme

With the complete fault-tolerant control scheme, magnitude of faults can be estimated using the reference control structure. In Figure 2.10, the magnitude of the estimated injected torques are presented without including its compensation mechanism. They converge to the real values of the injected faults for both axis, achieving their complete isolation disregarding the existing coupling effects between axis. Oscillatory effects appear during the transitory phase on both estimations, being more significant on the Tilt axis due to the dependency of the affecting fault on θ_2.

Results in Figure 2.12 correspond to the evolution of the system states for the final control scheme compensating the estimated faults. The evolution of the estimations for this case are included in Figure 2.11.

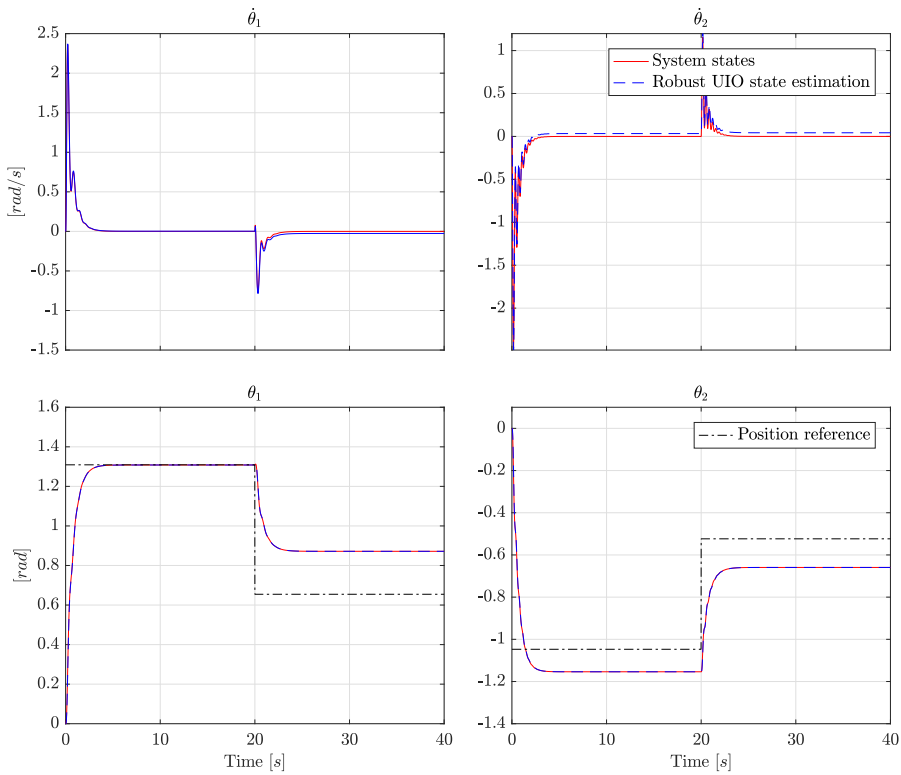

Figure 2.9 System's states evolution on fault scenario using the basic control structure with RUIO.

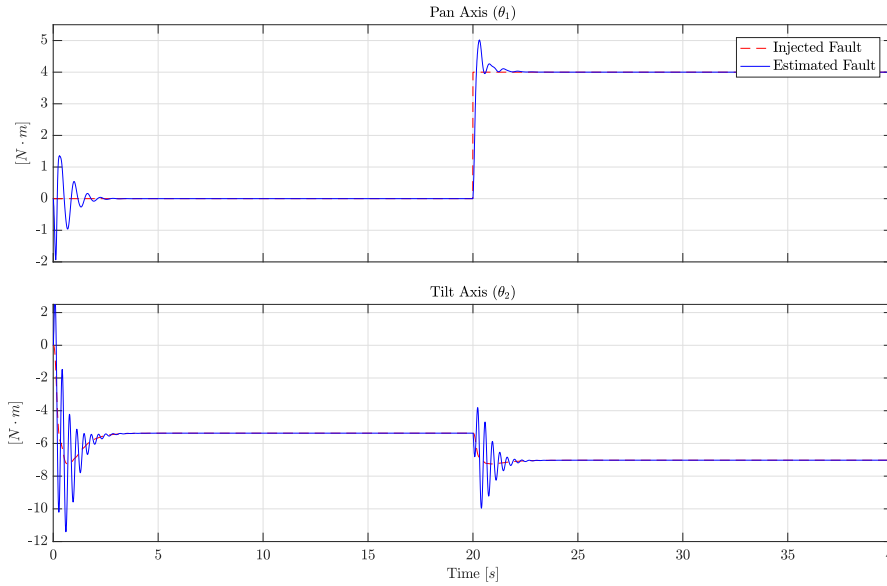

Figure 2.10 *Fault estimation in Pan and Tilt axis without including its compensation mechanism.*

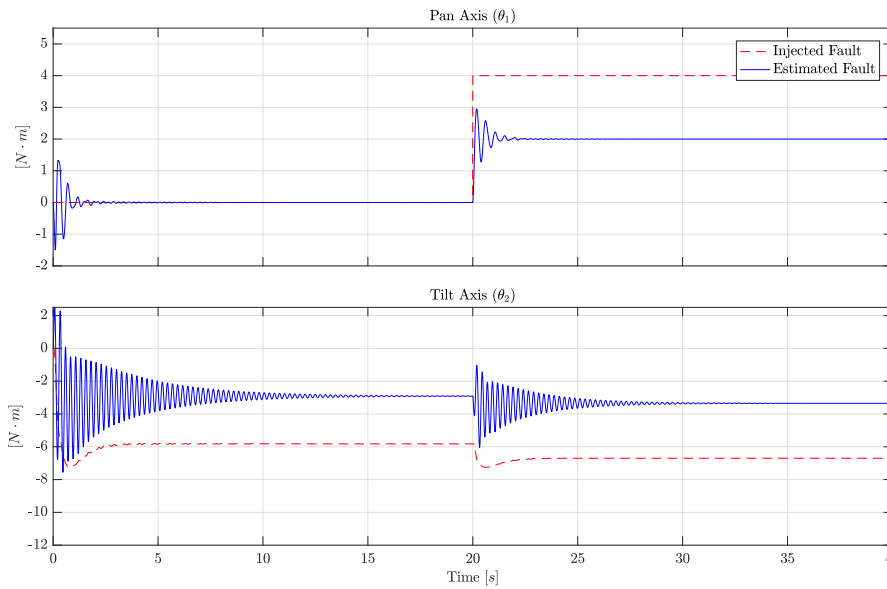

Figure 2.11 *Fault estimation in Pan and Tilt axis including its compensation mechanism.*

Obtained results present an improvement with respect to the previous control structure, but still the position error isn't null as expected, and there exist a (slightly reduced) difference between estimated and real position velocities. The cause behind this issue for this case is related with the error on the faults' estimation: magnitudes converge to approximately half the values of the current faults being injected. Further analysis has been performed, revealing that the estimation within the compensation mechanism has not fast enough dynamics to avoid that part of the disturbance is assumed by the RUIO. Thus, the aforementioned problem derived from the design is induced into the solution, and errors do not converge to zero.

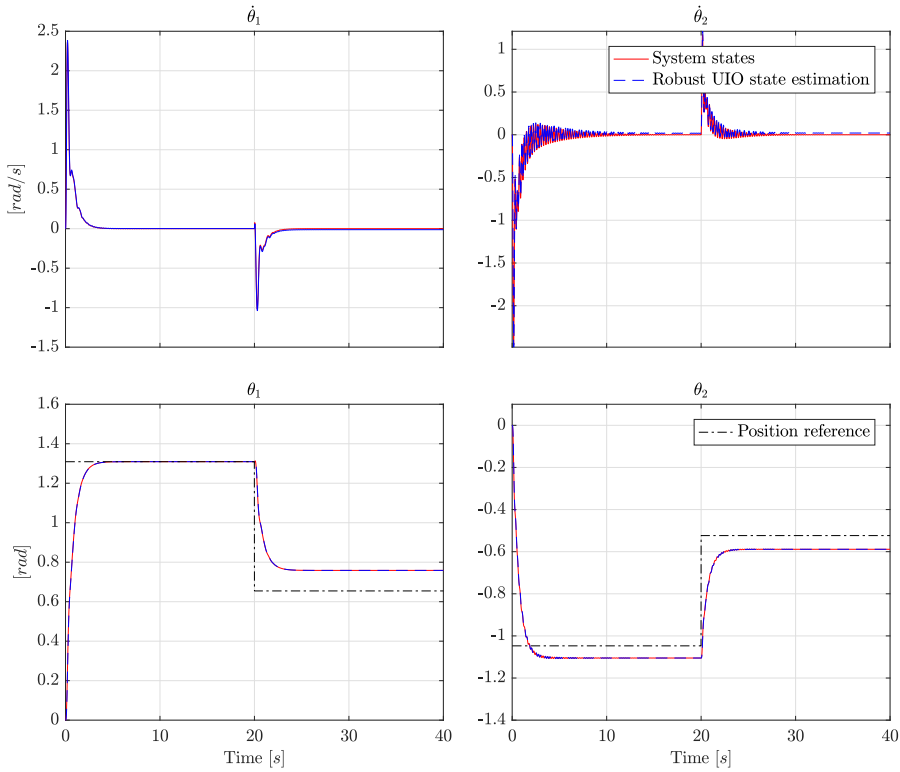

Figure 2.12 System states evolution on fault scenario using the complete fault-tolerant control scheme.

Finally, it should be pointed out that some strategies have been evaluated to overcome this problem, for example weighting the injected estimations by gains greater than the unit in order to initially over compensate it as in Reference [28]. This strategy led to an improvement in the results, but oscillatory effects (as those present on the Tilt axis estimation on Figure 2.11) compromised the stability of the control scheme, so it had to be discarded.

2.7 Conclusions

In this chapter, the proposed fault-tolerant Model-based control approach for the head subsystem of a TIAGo humanoid robot has been proved to tackle the stated problem. Although results show that the problem isn't completely answered, used methodology presents a first line of action towards a final fulfilling solution, upon which improvements can be made.

As future work, the proposed scheme will be implemented in the real system available at our labs. Other research opportunities regarding this problem can be related with gaining more information on the fault itself (for example, in the case of an unexpected collisions, its point of contact) and its integration within a complete autonomous architecture.

References

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

[2] Gertler J. *Fault detection and diagnosis in engineering systems*. Virginia (USA): CRC; 1998.

[3] Zhuo-hua D., Zi-xing C., Jin-xia Y. 'Fault diagnosis and fault tolerant control for wheeled mobile robots under unknown environments: A survey'. *Proceedings of the International Conference on Robotics and Automation (ICRA)*. IEEE; 2005.

[4] Khalastchi E, Kalech M. 'On Fault Detection and Diagnosis in Robotic Systems'. *ACM Computing Surveys.* 2018;51(1):1-24.

[5] Van M., Kang H. 'Robust fault-tolerant control for uncertain robot manipulators based on adaptive quasi-continuous high-order sliding mode and neural network'. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science.* 2015;229(8):1425-1446.

[6] Haddadin S., Albu-Schaffer A., De Luca A., Hirzinger G. 'Collision detection and reaction: A contribution to safe physical human-robot interaction'. *Proceedings of the International Conference on In Intelligent Robots and Systems (IROS)* IEEE; 2008; pp. 3356-3363.

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

[8] Hornung R., Urbanek H., Klodmann J., Osendorfer C., van der Smagt P. 'Model-free robot anomaly detection'.*Proceedings of the International Conference on Intelligent Robots and Systems (IROS)* IEEE; 2014; pp. 3676-3683.

[9] De Luca A., Ferrajoli L. 'A modified Newton-Euler method for dynamic computations in robot fault detection and control'. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* IEEE; 2009; pp. 3359-3364.

[10] Prassler E., Kazuhiro K. 'Domestic Robotics'. *Springer Handbook of Robotics* Springer; 2008; pp. 1253-1281.

[11] TIAGo Robotic Plaform, PAL Robotics. Available from: http://tiago.pal-robotics.com [Accesed 31 January 2019]

[12] Craig. J.J. *Introduction to Robotics. Mechanics and control.* 3rd edn. Pearson Education International; 2005. p. 173.

[13] Zadeh L.A. 'Fuzzy Sets'. *Information and Control* 1965; 8(3):338-353.

[14] Takagi T., Sugeno M. 'Fuzzy Identification of Systems and Its Applications to Modelling and control'.*IEEE Transactions on Systems, Man and cybernetics*. 1985, vol. 1, pp. 116-132.

[15] Sugeno M., Kang G. T. 'Fuzzy Modeling and Control of Multilayer Incinerator' *Fuzzy Sets and Systems* 1986, vol. 18(3), pp. 329-364.

[16] Tanaka K., Wang H. *Fuzzy control systems design and analysis*. New York (USA): Wiley; 2010.

[17] Duan G. R., Yu H. H. *LMIs in control systems: analysis, design and applications*. Boca Raton (USA): CRC press; 2013.

[18] Apkarian P., Gahinet P., Becker, G. 'Self-scheduled H_∞ Control of Linear Parameter-varying Systems: a Design Example' 1995; 31(9):1251-1261.

[19] Ostertag E., *Mono- and multivariable control and estimation: linear, quadratic and LMI methods*. Springer Science and Business Media; 2011.

[20] Chadli M., Karimi H.R. 'Robust Observer Design for Unknown Inputs Takagi–Sugeno Models'.*IEEE Transactions on Fuzzy Systems*. 2013, vol. 21(1), pp. 158-164.

[21] Chadli M. 'An LMI Approach to Design Observer for Unknown Inputs Takagi-Sugeno Fuzzy Models'.*Asian Journal of Control*. 2010, vol. 12(4), pp. 524-530.

[22] Crestani D., Godary-Dejean K. 'Fault tolerance in control architectures for mobile robots: Fantasy or reality?'. Presented in CAR: Control Architectures of Robots; Nancy, France, 2012.

[23] MATLAB, MathWorks. Available from: https://www.mathworks.com/ [Accesed 31 January 2019]

[24] Dormand J.R. , Prince P.J. 'A family of embedded Runge-Kutta formulae' *Journal of Computational and Applied Mathematics*. 1980, vol. 6(1), pp. 19-26.

[25] Lofberg J., 'YALMIP: a toolbox for modeling and optimization in MATLAB', *IEEE International Conference on Robotics and Automation*; Taipei, Taiwan, 2004.

[26] YALMIP toolbox. Available from: https://yalmip.github.io/ [Accesed 31 January 2019]

[27] SeDuMi: Optimization over symmeric cones. Available from: https://github.com/sqlp/sedumi [Accesed 31 January 2019]

[28] Witczak M. *Fault diagnosis and fault-tolerant control strategies for non-linear systems*. Lecture Notes in Electrical Engineering (LNEE), no. 266. Switzerland: Springer; 2014, p. 119-142.