POLITECNICO DI TORINO

Master's Degree
In MECHATRONIC ENGINEERING

Master's Degree Thesis

Modelling and Design of a Robust Control System for an Aerial Manipulator

Advisors:
Prof. Elisa Capello
Prof. Hyeongjun Park

Candidate:
Leo Borrelli

A.A. 2019-2020

Abstract

Aerial manipulators are a class of *UAV*s that are gaining always more interest in new research studies but also military or industrial applications. In fact, the presence of a manipulator, or other types of tool, greatly improve dexterity and flexibility of the system, allowing the accomplishment of complex tasks reducing also risks and costs mainly.

The main scope of this work is to realize a model and a robust control system, for a hexacopter equipped with a four degrees of freedom manipulator, with two main characteristics: computational efficiency and ease to be modified and tuned. The research is focused on the position control of the tool centre point of the robotic manipulator, without directly taking into account its interactions with the external environment.

The presented approach is an independent one, where the two systems are modelled and controlled independently and then they are coupled together. The drone is controlled using a hierarchical architecture composed by two *sliding mode* controllers, for position and attitude. This control technique results very useful and efficient when dealing with nonlinear systems for its robustness properties.

The robotic arm is modelled using the *Simscape Toolbox* and the control architecture is implemented exploiting the *Robotics System Toolbox*. The adopted control structure is an *operational space controller* with gravity compensation, suitable to allow the manipulator to compensate position errors of the hexacopter centre of mass. The coupling is finally accomplished through equilibrium Newton-Euler equations. The whole scheme is implemented in *MATLAB*/*Simulink* environment, where the overall performances of the coupled system are evaluated through an explicative simulation of hovering and manipulation.

The final design aims to represent a reasonable trade-off between complexity and accuracy that can be improved and adapted to the specific case and its requirements. The independent approach simplifies this task, since every single component is firstly designed individually.

Sommario

I manipolatori aerei rappresentano una categoria di *APR* (aeromobile a pilotaggio remoto) che stanno guadagnando sempre più interesse in nuove aree di ricerca ma anche applicazioni militari o industriali. Infatti, la presenza di un manipolatore, o altri tipi di strumenti, migliora sensibilmente l'agilità e flessibilità del sistema, permettendo il completamento di mansioni complesse riducendo soprattutto rischi e costi operazionali.

L'obiettivo principale di questo studio è quello di realizzare un modello e un sistema di controllo robusto, per un drone esacottero equipaggiato con un braccio robotico dotato di quattro gradi di libertà, avente le seguenti principali caratteristiche: efficienza computazionale e facilità ad essere modificato e calibrato. La ricerca si concentra sul controllo di posizione del *tool centre point* del braccio robotico, senza prendere direttamente in considerazione le sue interazioni con l'ambiente esterno.

È stato adottato un tipo di approccio *indipendente* che prevede la modellizzazione e controllo indipendente dei due sistemi, per poi essere accoppiati. Il drone è controllato attraverso una struttura gerarchica composta da due controllori del tipo *sliding mode*, rispettivamente per posizione e assetto. Questa tecnica di controllo si dimostra molto utile ed efficiente nel trattare sistemi non lineari, grazie alle sue proprietà di robustezza.

Il manipolatore è modellizzato usando il *Simscape Toolbox* e la struttura di controllo è realizzata sfruttando il *Robotics System Toolbox*, entrambi forniti da *MathWorks.* La scelta del controllore ricade nella categoria di *operational space controller* con compensazione di gravità, adatto per permettere al manipolatore di compensare eventuali errori di posizione del centro di massa dell'esacottero. L'accoppiamento dei due sistemi è infine realizzato attraverso equazioni di equilibrio di Newton-Eulero. Lo schema finale è poi implementato in ambiente *MATLAB/Simulink* per valutare le performance del sistema attraverso una simulazione esplicativa di *hovering* e manipolazione.

Il design finale si pone l'obiettivo di rappresentare un ragionevole compromesso tra complessità e accuratezza, che può essere migliorato per essere adattato al caso in questione e i relativi requisiti di progetto. L'approccio indipendente semplifica questa operazione, in quanto ogni singolo componente dello schema è progettato e calibrato individualmente.

Acknowledgment

Foremost, I would like to express my sincere gratitude to my advisor Dr. Elisa Capello for the expertise and support given during this research and for the great opportunity to conclude my course of study in a foreign country, at the New Mexico State University.

Here I found a new and dynamic research environment where I had the possibility to confront myself with both a different educational system and a culture very different from mine. This experience was very valuable from both a personal and a professional point of view. In fact, a special thank of gratitude goes to my research supervisor Dr. Hyeongjun Park for his patience, guidance and constant motivation that allowed me to complete my research in an environment totally new to me. Besides my advisors, I would like to thank also Dr. Isuru Basnayake for his knowledge and constant support during the development of the project.

In addition, I extend my thanks to my friends, met in Torino and from my hometown, valuable point of reference during serene and difficult periods.

Finally yet importantly, I am extremely grateful to my whole family for the immense love and help in many different aspects of my life, provided during my entire academic career.

Table of contents

List of figures

Chapter 1. Introduction

Unmanned aerial vehicles (*UAV*) are referred to aircraft controlled without the action of a human pilot on board. The flight of *UAV*s may operate at different degrees of autonomy: either under remote control by a human operator or autonomously through on-board instrumentation. These are always more used systems in a wide variety of different applications. In fact, due to their noticeable versatility and rapidity in manoeuvres, they can be suitably designed to adapt to complex and dangerous working conditions. While they originated mostly in military applications, their use is rapidly increasing in industrial or monitoring tasks, reducing risks and cost of operations.

Since the modelling and control of *UAV* has exhaustively been object of research [as in 15, 16, 17] in the past years, it opens a new interesting scenario about aerial manipulators. They are even more flexible and dexterous architectures, used in a broader field of applications, such as commercial inspection or space as well. In fact, for their characteristics, they are gaining more interest recently. On the other hand, the need to manage the complexity of the system arises, demanding more accurate modelling and control development.

This thesis work is focused on the modelling and robust control design of a multirotor aerial manipulator, a hexacopter with a four degree-of-freedom robotic arm mounted on the upper part of the body frame. The main advantages of this system are noticeable abilities of rapid and versatile manoeuvring, through an efficient and computationally light algorithm.

Several new designs and implementations for aerial manipulation have been implemented, as stated in [14]. Different approaches are possible, mainly from a mechanical point of view and a modelling and control point of view. In fact, different shapes and bodies of the *UAV*s' exist, as well as different manipulating devices. Concerning the modelling and control, mainly two approaches are available: an independent one, that is simpler and efficient, and an overall one, that takes into account complex coupled dynamics to achieve improved performances. All these design possibilities are very important and strongly related to the proposed task.

On the other hand, this kind of system shows some difficulties, being a complex nonlinear mechatronic system, such as several external disturbances (ground and wall effect or wind) and a severe coupling interference between manipulator and drone's frame. Another aspect not to be neglected is related to weight and payload. In fact, more complexity needs more computational effort and more advanced sensors that traduces in more and heavier instrumentation. Moreover, the presence of the manipulator reduces additional possible payload.

Taking into account these problems, many different studies and real implementations have been accomplished to realize stable and accurate control strategies that allow the completion of various tasks. A common technique, used for the realization of the system model, is to use a Newton-Euler recursive algorithm, as in [3]. These equations are used to calculate recursively the velocity of the centre of mass of each rigid body of the system, to successively compute all the forces and torques exchanged between them. Non-holonomic constraints can be taken into account, as done in [1], where a feedback linearization controller is applied. In fact, concerning the control algorithms, several architectures guarantee satisfactory results, such as the decentralized *PID* control presented in [2].

More complex modelling methods exploit the Euler-Lagrange equations; this formulation is generally used for overall modelling approaches, being able to directly take into consideration dynamic coupling aspects between the two systems [4]. Here picking up and delivering tasks are considered in a real setup, accomplished through an adaptive sliding mode controller. This technique shows interesting properties in dealing with the control of drones and different implementations are available. For example, a second-order sliding mode controller with nonlinear sliding surface was used in [12] to control a quadrotor, while a sliding mode state observer is presented in [13]. Other applications where the manipulation system is controlled through a sliding mode controller were studied in [10] and [5]; here a comparison with a controller designed exploiting Lyapunov theory was performed. This is another well-known technique in control design, which directly takes into consideration stability properties of the system [6].

Another fundamental aspect to be considered in manipulation systems is the contact between manipulator and objects. This issue has to be taken into account during the design, to avoid unexpected behaviours of the system. In regards, the mechanical system can be upgraded with specific contact sensors, as presented in [8]. Here, a passivity-based PD controller is able to manage the system nonlinearities. More complex robotics algorithms based on a contact modelling can be used, such as the impedance based force control presented in [7]. In addition, many other strategies are available, concerning both the mechanical configurations and control setup; for example, in [9], the task is decomposed in sub-tasks to be tackled by the controller in different ways.

The solution to the above problems presented in this work is based on an independent approach for modelling and controlling the arm and the hexacopter systems, which are coupled successively. This allows obtaining a satisfying trade-off between performances and scheme complexity. Notice that this kind of implementation implies that the drone does not directly take into account the robot motion, thus it is considered as a disturbance. The choice of robust control for the hexacopter's position goes to the sliding mode technique. It is a suitable choice for nonlinear systems, which shows interesting properties of stability and robustness. In fact, it is founded on a strong mathematical formulation based on the system dynamics, which leads to a rejection of external disturbances, parameter uncertainties and

unmodeled dynamics. These characteristics are suitable for the considered problem, especially for the independent controllers. The manipulator scheme is implemented following a straightforward workflow, exploiting some useful features of the *Simscape Multibody* and *Robotics System* toolboxes, provided by *MathWorks*. These tools speed up the design process since they offer various robotics related functions and the possibility to exchange information between them. The dynamical parameters of both the manipulator and the drone are evaluated using the *SolidWorks* modelling software.

The two architectures, for the hexacopter and for the robotic arm, are therefore tuned and tested independently, before actually coupling them. This aspect is addressed using equilibrium Newton-Euler equation, in order to estimate the disturbance torques and forces exerted from the manipulator to the drone during its motion. These estimated values are successively compensated through a feedforward compensation technique.

It is important to point out that the principal focus is given to the position control of the tool centre point in space, while the interaction with the external environment is not directly taken into account. In regard, some force and torque sensors need to be mounted on the end effector of the manipulator. In this way, it is possible to estimate the forces and torques exerted by the environment during the manipulation phases. Many possibilities are available for these sensors and this choice is strongly related with the proposed task. In fact, the most simple and cheap option is a force sensitive resistor that is able to identify only if the end effector is having an interaction with objects; more complex devices, such as capacitive or piezoelectric sensing technologies, show more accuracy but increased costs.

Notice that the presented scheme can take into account these forces and torques for what concerns both the manipulator control algorithm and the estimation of the disturbances transmitted to the drone's frame during manipulation phases. Anyway, the manipulator controller is designed for position control purposes and the addition of external interactions may imply some refinements in the control architecture, such as the addition of a force feedback loop. Additionally, the controller can be replaced with more advanced algorithms, such as stiffness or impedance control and hybrid control for example. The choice of the adopted sensors and control scheme depends on the final purpose of the manipulation system, since controllers that are more complex need more accurate instrumentation that directly traduces in more powerful and expensive embedded systems and reduced possible payload.

The main objective of this dissertation is to define a framework, with two main characteristics: computational efficiency and ease to be modified and tuned. In this way, it can be implemented in many different scenarios and moreover it is hardware independent; in fact, different arms or drones can be deployed based on the different cases, by proper adjustment of the control parameters. Moreover, the particular structure of the control algorithm allows the possibility to refine and improve independently both models and controllers to obtain

desired trade-off performances – cost with respect to the proposed task. Consequently, the predefined goal is to achieve the realization of a computationally simple, however accurate control architecture, which can represent a solid base for practical applications and future improvements.

1.1. Thesis overview

This thesis work is divided into five chapters. A complete discussion about the modelling of the systems is exposed in chapter two. In particular, section 2.1 deals with the hexacopter mathematical model, while section 2.2 explains the realization of the model for the manipulator; finally, section 2.3 describes how the two systems are actually coupled together and the modifications required to accomplish this operation.

Chapter 3 discusses the design of the control architectures for the two systems; firstly, in section 3.1, the hexacopter robust control system is shown and in section 3.2, the one related to the manipulator. Both of them are tested and tuned individually before the coupling between them.

The fourth chapter then shows an explicative simulation of hovering and manipulation, to understand the capabilities of the overall control scheme.

Finally, the last chapter discusses conclusions about the presented work and possible future applications.

Chapter 2. Mathematical Models

2.1. Hexacopter mathematical model

In this section, a complete discussion about the modelling of the hexacopter system is presented. The mathematical model is firstly developed considering inner nonlinearities of the system to be successively simplified for control design purposes. The controller is then realized, based on these equations, exploiting the sliding mode technique.

In particular, this section expresses the derivation of the dynamical equations that describe the system. The model now can be refined through an identification process, as discussed in section 2.1.1. Starting from this point, the controller is developed and tuned with respect to real dynamical parameters, as discussed in chapter 3.

Since the aim of the control is the tracking of reference trajectories, the choice of the system states goes to the six degrees of freedom of the hexacopter and the related velocities. The variable p_0 represents the inertial position of the drone's centre of mass and $\dot{p}_0$ its velocity; Φ_0 represents the orientation of the drone's body frame in intermediate frames and $\dot{\Phi}_0$ is the rate of change of attitude angles.

The drone is modelled as a non-linear system described by the following equations, as in [17]:

$$\dot{p}_0 = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix} = R_0^i * v_0 = \begin{pmatrix} c\theta\, c\psi & s\varphi\, s\theta\, c\psi - c\varphi\, s\psi & c\varphi\, s\theta\, c\psi + s\varphi\, s\psi \\ c\theta\, s\psi & s\varphi\, s\theta\, s\psi + c\varphi\, c\psi & c\varphi\, s\theta\, s\psi\ - s\varphi\, c\psi \\ -s\theta & s\varphi\, c\theta & c\varphi\, c\theta \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix}$$

$$\dot{v}_0 = \begin{pmatrix} \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \end{pmatrix} = \begin{pmatrix} \omega_z v_y - \omega_y v_z \\ \omega_x v_z - \omega_z v_x \\ \omega_y v_x - \omega_x v_y \end{pmatrix} + \begin{pmatrix} g\, s\theta \\ -g\, c\theta\, s\varphi \\ -g\, c\theta\, c\varphi \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ T/m \end{pmatrix}$$

$$\dot{\Phi}_0 = \begin{pmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = Q * \omega_0 = \begin{pmatrix} 1 & s\varphi\, t\theta & c\varphi\, t\theta \\ 0 & c\varphi & -s\varphi \\ 0 & s\varphi/c\theta & c\varphi/c\theta \end{pmatrix} \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}$$

$$\dot{\omega}_0 = \begin{pmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{pmatrix} = \begin{pmatrix} \frac{Jy - Jz}{Jx}\, \omega_y\, \omega_z \\ \frac{Jz - Jx}{Jy}\, \omega_x\, \omega_z \\ \frac{Jx - Jy}{Jz}\, \omega_x\, \omega_y \end{pmatrix} + \begin{pmatrix} \tau_\phi / J_x \\ \tau_\theta / J_y \\ \tau_\psi / J_z \end{pmatrix}$$

where $g = 9.8067\ m/s^2$ is the gravitational acceleration constant, $m\ [Kg]$ is the hexacopter mass, $J = \left[diag(J_x\, J_y\, J_z)\right] Kg\, m^2$ represents its principal moments of inertia, $\Phi_0 = [\varphi\ \theta\ \psi]^T\ rad$ are respectively roll, pitch and yaw angles and $\omega_0 = [rad/s]$ is the angular velocity in body frame. The rotation matrix R_0^i expresses the rotation between the body and

inertial frames and it is derived using the *XYZ* Euler angles convention; Q represents the relationship between angular rates and angular velocity of the drone, since they are expressed in different reference frames. Finally, the control inputs are $T = [N]$ and $\tau_\Phi = [\tau_\varphi \ \tau_\theta \ \tau_\psi]^{\mathrm{T}} Nm$, that are respectively the vertical thrust given by the rotors and the torques around the three axes; the relations between these variables and the related motor speeds is discussed in the next section.

Notice that the *z* inertial frame axis is directed upwards with respect to the ground.

The values of both linear and angular acceleration, such as the command inputs, are expressed in body frame, while the correspondent equations in the inertial frame are the following:

$$\ddot{p}_0 = R_0^i \, \dot{v}_0 + \dot{R}_0^i \, v_0$$
$$\ddot{\Phi}_0 = Q \, \dot{\omega}_0 + \dot{Q} \, \omega_0$$

The simplified dynamical equations are computed neglecting the Coriolis terms and neglecting the derivatives of R_0^i and Q, which is a valid assumption for small angles:

$$\begin{cases} \ddot{x} = (c\varphi \, s\theta \, c\psi + s\varphi \, s\psi) \, T/m \\ \ddot{y} = (c\varphi \, s\theta \, s\psi - s\varphi \, c\psi) \, T/m \\ \ddot{z} = (c\varphi \, c\theta) \, T/m - g \end{cases} \qquad \begin{cases} \ddot{\varphi} = \tau_\varphi / J_x \\ \ddot{\theta} = \tau_\theta / J_y \\ \ddot{\psi} = \tau_\psi / J_z \end{cases}$$

Considering then that the exploited pitch and roll angles are very small ($\pm 5^\circ$), also the following assumptions can be done, making the equations linear in φ, θ:

$$\begin{cases} s\varphi \cong \varphi \\ c\varphi \cong 1 \end{cases}; \quad \begin{cases} s\theta \cong \theta \\ c\theta \cong 1 \end{cases} \quad \Rightarrow \quad \begin{cases} \ddot{x} = (\theta \, c\psi + \varphi \, s\psi) \, T/m \\ \ddot{y} = (\theta \, s\psi - \varphi \, c\psi) \, T/m \end{cases}$$

As stated before, these simplified dynamics is used in chapter 3 where the design of the control scheme is presented.

2.1.1. System identification

This section explains the identification process, making the model match the real hardware available in the laboratories.

The considered drone is a *DJI F550*, a hexacopter with rotors mounted in the so called *X configuration*, as shown in Figure 1. Obviously, different setups lead to different motion capabilities for the drone.

Notice that three propellers spin in a clockwise direction, while the others in a counter clockwise one, allowing the movement around all the different body axes in a controllable manner.

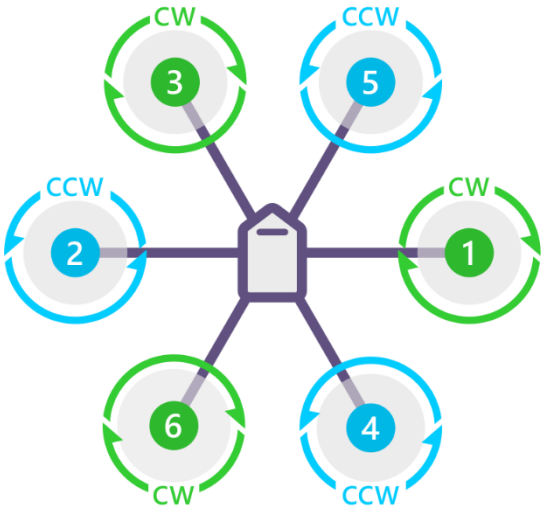

Figure 1: Schematics of hexacopter's motors configuration

The identification operation helps the model to be more reliable, making the overall scheme more suitable for real-time applications. In fact, the control tuning will be more likely close to the optimal values and simulation accuracy is improved as a result. Moreover, a conversion function between desired force/torques and input command to the motors is necessary to apply the control algorithm to the real hardware. This function is called *control mixer* and standard setups are available, depending on number of propellers, their geometry and the direction in which they rotate.

In this application, the electric motors are managed through *PWM* (pulse width modulation) signals, a widely used technique that is a digital modulation to obtain a mean voltage dependent on the duty cycle. The propellers mounted on the considered hexacopter are commanded by signals that vary from $1100\ \mu s$ to $1900\ \mu s$. It is important to constrain the six *PWM* commands in between these values, to avoid unexpected behaviours.

The computation of the control mixer is carried out as in [19]. The first step is to compute the input force and torques to the system, starting from the *PWM* values:

$$T = K_f \sum_{k=1}^{6} (PWM_k - PWM_0)$$

$$\tau_\phi = \frac{l_H\, K_f}{2}\, (PWM_2 + PWM_3 + 2\, PWM_6 - 2\, PWM_1 - PWM_4 - PWM_5)$$

$$\tau_\theta = \frac{\sqrt{3}\, l_H\, K_f}{2}\, (PWM_3 + PWM_5 - PWM_6 - PWM_4)$$

$$\tau_\psi = K_\tau\, (PWM_2 + PWM_4 + PWM_5 - PWM_1 - PWM_3 - PWM_6)$$

The variables K_f, PWM_0, K_τ represent the propulsion system constants, while PWM_k is the input to the k^{th} motor and l_H is the distance between consecutive propellers.

Collecting the equations into a single matrix, one obtains:

$$\begin{bmatrix} T \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = A_{PWM}\, V_{PWM} + B_{PWM}$$

where $V_{PWM} = [PWM_1 \dots PWM_6]$, A_{PWM} is a constant matrix and B_{PWM} is a constant vector.

This relation is used for simulation purposes, to convert *PWM*s into force/torques to be fed to the plant dynamics.

By inverting the equation, it is possible to calculate the right command signal for each motor, in order to achieve the required control inputs. Anyway, this mixer is redundant because there are infinite combinations for the six *PWM*s to generate the four force/toques. The adopted solution exploits the pseudoinverse of the matrix A_{PWM}:

$$A_{mix} = A_{PWM}^T \left(A_{PWM}\, A_{PWM}^T\right)^{-1}$$
$$B_{mix} = -A_{PWM}^T \left(A_{PWM}\, A_{PWM}^T\right)^{-1} B_{PWM}$$

Finally, the control mixer equation can be derived:

$$V_{PWM} = A_{mix} \begin{bmatrix} T \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} + B_{mix}$$

Other considerations are related to dynamical properties of the hexacopter system. In fact, the control tuning is closer to reality by using real dynamical parameters for the hexacopter. This operation is achieved exploiting the design software *SolidWorks*. A basic *CAD* model of the hexacopter is found at *www.grabcad.com* and then modified to reflect the real prototype.

Specifically, the batteries and embedded systems are added, as well as the perching system. This mechanical device is mounted with the future perspective of making the aerial manipulator work in near-wall scenarios. In fact, when the drone approaches a wall it can establish a contact with it, giving more stability during the manipulation phase thanks to the present friction.

It is clear that the inertia of the hexacopter, as well as its centre of mass, are different from the nominal values.

The model of the hexacopter is represented in Figure 2.

Figure 2: SolidWorks model of the drone available in laboratories

After setting the right material for each component, the real parameters can be extracted directly from *SolidWorks*.

2.2 Manipulator Model

This section explains the adopted workflow for the realization of the model related to the four degrees of freedom manipulator. Both the model and controller are implemented in *MATLAB*/*Simulink* environment, exploiting some available toolboxes useful to deal with the topic. The main idea is to define a fast procedure to obtain an overall scheme that can be improved and refined with respect to the proposed task and constraints. In regard, two different models of the robotic arm are realized and they are used independently for simulating and control purposes; in particular, in section 2.2.1 a *Simscape Multibody* model is presented and in 2.2.2 a *MATLAB* model is designed through the *Robotics System Toolbox*.

The considered robotic arm, available in the laboratories, is the *PhantomX Pincher*, a four joints hobby class manipulator. It is a type of anthropomorphic robot since it emulates the structure of a human body, with trunk, arm and forearm. In fact, it is composed by four revolute joints, where the first one has a vertical rotation axis and the other three are horizontal and parallel between them. In this particular type of configuration, the task space takes a shape similar to a sphere sector. This is one of the most common structures in industry, since it provides the best dexterity; consequently, it is very adapt to the aerial manipulation. This work is focused on controlling only the position of the end effector, a gripper, thus it is considered as a redundant robot. In fact, the redundancy degree is equal to

one, which is the degrees of freedom possessed by the arm minus the degrees of freedom required by the task. This type of kinematic chain improves manipulability, since the arm can reach the same pose in different ways giving the possibility to avoid some obstacles for example. Concerning the mounted motors they are *Dynamixel AX-12A* servos; they provide feedback on position, temperature and voltage with a maximum possible turn of 300°. This hardware setup shows a good flexibility and disposes of a wide documentation on the web.

In this way, it is possible to obtain a sufficiently accurate application, with satisfactory trade-off between cost and performances.

As stated before, two different models of the manipulator are realized. The most common and accurate way to model the system is using a Lagrange-Euler method. Moreover, many control approaches take advantage of this formulation, basing their control strategy on some dynamical properties of the system.

Anyway, in order to keep the complexity of the system as low as possible and to make the proposed algorithms hardware independent, a different approach is followed in this work. In particular, two different toolboxes, provided by *MathWorks*, are exploited: *Simscape Multibody* and the *Robotics System Toolbox*. They are both very versatile tools that provide the possibility to communicate and carry out co-simulations with external platforms, such as *ROS* (Robotics Operating System) or target embedded systems since they support code generation and other useful functions. Moreover, they make available fast ways to model the system and to simulate its behaviour.

Specifically, the *Robotics System Toolbox* offers several useful *MATLAB* functions and *Simulink* blocks to easily implement the main robotics related functions, such as Jacobian or inverse kinematics computation. In addition, the robot model is realized through object-oriented programming realizing an intuitive way of kinematic/dynamical modelling. These features, combined with the possibility of generating *C*-code compatible with most robotics embedded platforms, make the tool adapt for realizing the overall control scheme in a suitable way with respect to the proposed objectives.

On the other hand, the *Simscape Toolbox* is used mainly for simulation and visualization purposes. In fact, it is a very flexible tool to rapidly couple and simulate multi-domain physical systems (mechanical, electrical, hydraulic, etc.) in *Simulink* environment. This allows the designer to model the considered system at the accuracy needed by the application requirements; in regard, many *MathWorks* or user-defined add-on products are available to provide more complex components and analysis capabilities, such as a precise electric model of the motors, or task related like contact modeling for example. It is important to notice that *Simscape* blocks represent physical connections, thus they are updated all together at each sampling instant, to better simulate the behavior of a mechatronic system; for this reason, some conversion blocks are required to interface *Simscape* with *Simulink* signals. In fact, different solvers can be applied for the control scheme and the specific *Simscape* model.

Additionally, *C*-code generation is supported, allowing the possibility of rapid development and testing of reliable control systems.

Anyway, *Simscape* models are not easy to be realized since they require a deep knowledge of the kinematic and dynamical properties of a complex system such as the considered manipulator. In this regard, the toolbox offers another function to speed up the modelling phase without losing noticeable accuracy; it is the opportunity of directly importing *CAD* models by translating them into *Simscape* language, as discussed in the next subsection.

2.2.1. System identification and *Simscape* plant model

The realization of the plant is accomplished using the *Simscape Multibody Toolbox*. The reason to choose this tool is to accomplish a fast but efficient way of creating a reliable model that is hardware independent. In fact, this procedure can be applied, with the right expedients, to any manipulator. Moreover, the available visualization feature is used to directly observe the behaviour of the robotic arm.

As stated previously, a *CAD* model is the starting point of the presented workflow and it is realized using the *SolidWorks* design software. The basic model is taken at *www.grabcad.com* but some necessary operations are performed before exporting the assembly as an *.xml* file, compatible with *MATLAB*.

First off, the material of the different parts is set, giving an estimation of the dynamical properties of the system. In fact, the electric motors are made of engineering plastic while the metal parts of the arm are aluminum (*EN-AW-5052*), as specified by the manufacturer. This leads to a direct consequence, which is that the center of mass of each link is not perfectly centered but moved towards the metal components.

The next step is to fix the reference frames of each part with direction axes in a suitable way, to make them coherent within each other; the Denavit-Hartenberg convention is adopted, as discussed successively in this chapter.

Finally, each part is adequately constrained to each other in order to provide the correct degrees of freedom. The importance of this operation is that, while importing the model in *Simulink*, each degree of freedom is translated into a rotational joint block. The mapping between *SolidWorks* mates and *Simscape* joints can be found in the *MathWorks* documentation.

The last consideration regards the robot base that has been partially removed in the hardware prototype. The final *CAD* model is represented in Figure 3.

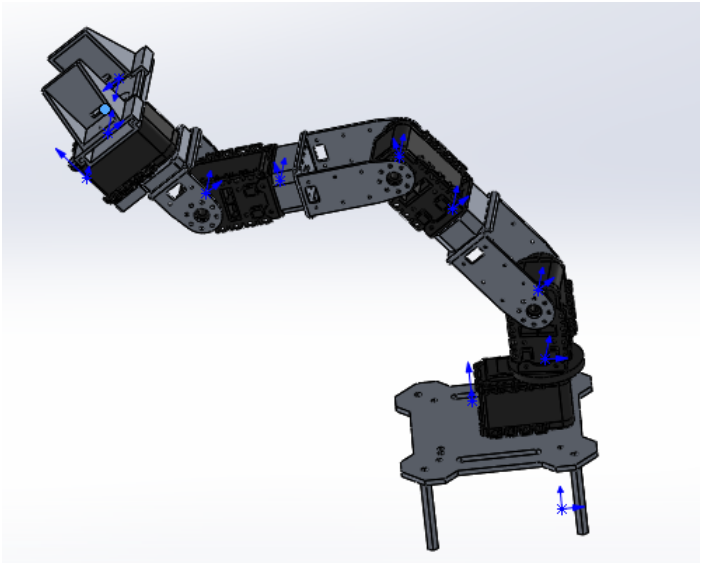

Figure 3: SolidWorks model of the manipulator

The model is now ready to be exported, after installing the dedicated plug-in, in *Simulink* in the form of *Simscape* blocks. Notice that all the kinematic and dynamical properties of the manipulator are imported as well in this process, simplifying the identification phase.

Again this scheme needs to be slightly improved for practical reasons and to make the system model more realistic; in fact, different levels of modeling accuracy and complexity can be reached at this point, such as the addition of an electromechanical model of the electric motors. The first modifications are executed in the joint blocks. Firstly, they are set to be driven by torque in order to simulate the control action. Then, a limit is set for the joint motion at $\pm 150°$, as specified by the motor manufacturer, and the position, velocity and acceleration feedback is added.

In addition, the motor damping has been computed and updated to the joints. This value is calculated from the no-load torque formula, with all the parameters available from data sheet:

$$\lambda = \frac{k_t \, i_{nl}}{w_{nl}} = 0.0227 \; Nm \; s/rad$$

where $k_t = 1 \; Nm/A$ is the torque constant, $i_{nl} = 1.5 \; A$ and $w_{nl} = 6.178 \; \text{rad/s}$ are respectively the stall current and no-load speed.

From a practical point of view, other refinements are done to avoid errors and speed up the simulation. First off, the *Simulink* solver is set as fixed-step and *ode14x*, recommended for stiff models such mechanical models in *Simscape*. Notice that the toolbox allows the user to specify different solvers for the specific *Simscape* model and *Simulink*; considering this, some rate transition blocks are added between this model and the external blocks. The plant model is now ready to be simulated and it is shown in Figure 4.

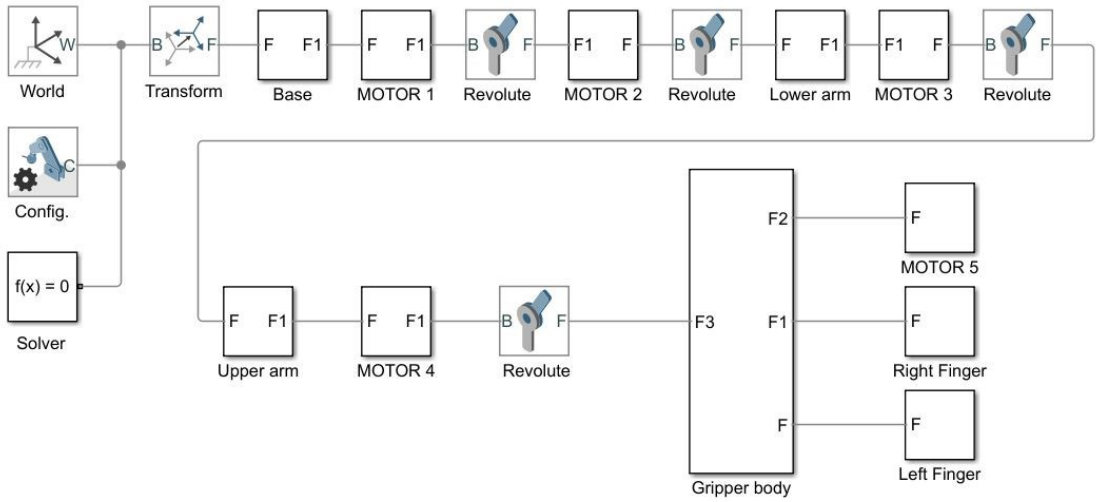

Figure 4: Simscape model of the manipulator

2.2.2. Kinematic model and *Robotics System* Toolbox model

This section deals with the *MATLAB* model of the robot, necessary for control purposes. Several tools are suitable to accomplish this task, but among the many available, the *Robotics System Toolbox* is used in this work. This choice is justified by the characteristics of the tool already discussed but mainly due to the presence of useful *Simulink* blocks that implement the main robotics functions. Moreover, the model is created as a *MATLAB* object to make it easier to understand and modify the model itself. The main fields that describe this structure are the gravity vector, intended as constant, the base reference frame that is inertial for the robot and the description of all the rigid bodies that compose the manipulator structure. In particular each body field takes into account dynamical properties of each link, such as mass, inertia and position of the centre of mass and kinematic parameters as well; among these, the rigid transformation between adjacent links are considered along with information about the joint related to each link.

The first step in modelling the kinematics of the manipulator is to fix a reference frame on each robot arm and an inertial one, typically placed in the robot base. In general, six parameters (three translations plus three rotation angles) are required to move from a frame to the following. A well-known type of conventions, called Denavit-Hartenberg conventions [24], are introduced to reduce the number of parameters needed to describe this transformation, by finding a common way to define the relative position of reference frames. According to this technique, only four parameters are used because two constraints are added to the problem. Two of these parameters are associated to translations, while the other two to rotations. Three of these values depend only on the robot geometry, thus they are constant in time; only one parameter depends on the relative motion between consecutive links and it is called joint variable $q_i(t)$. It follows that it is a translation for prismatic joints and a rotation angle for revolute ones.

Concerning the revolute joints, the first constraint is that all the reference frames associated to a link has the *z*-axis set as motion axis, meaning that the joint rotates around it. Calling the starting reference frame $R_{i-1}$and the successive one R_i, the four parameters that describe the relative transformation between them are defined as follows:

- Parameter d_i: it defines the translation along the motion axis z_{i-1} between the origin of R_{i-1} and the intersection of the axis defined by z_{i-1} and the axis defined by x_i.
- Parameter θ_i: it defines the rotation angle around axis z_{i-1} such that x_{i-1} overlaps x_i. The sign follows the right hand rule.
- Parameter A_i: it defines the minimum signed distance between axis z_{i-1} and z_i along the common normal, measured along x_i.
- Parameter α_i: it defines the rotation angle around motion axis x_i such that z_{i-1} overlaps z_i. The sign follows the right hand rule.

The next table shows the choice of these variables adopted in this work.

# Transform.	d	θ	α	A
1-2	L_1	q_1	π/2	0
2-3	0	q_2	0	L_2
3-4	0	q_3	0	L_3
4-EE	0	q_4	0	L_4

Table 1: Denavit-Hartenberg parameters

where L_i represents the length of the i^{th} link and q_i is the i^{th} generalized coordinate or joint variable.

The designer can realize this kinematic model in *MATLAB* but it is time consuming; for this purpose, *MathWorks* provides a useful command to automatically generate a rigid body tree object, based on a *Simscape* model that describes the robotic arm. The procedure is really faster, since all the kinematic and dynamical parameters are directly updated to the model in this process. In addition, this function is very useful because all the reference frames are coherent between the two models, simplifying the realization of the control structure.

Again, a small modification is required, since the end effector is not considered in the initial model. In particular, another rigid transformation from the fourth joint to the end effector is added, fixing the tool centre point reference frame in the middle point of the gripper. Also the *Robotics System Toolbox* provides a visualization function, as represented in Figure 5.

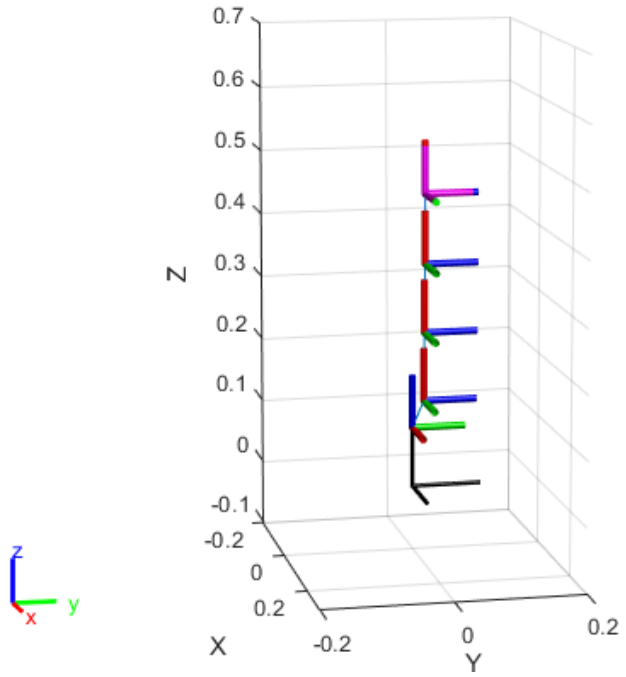

Figure 5: Robot model realized through the Robotics System Toolbox

The black reference frame is the base, inertial for the robot system. The successive four frames are associated to each joint, with rotation axis highlighted in blue. The last frame is fixed with the fourth link and it represents the tool centre point, as already stated.

The only aspect to be pointed out is that the base of the robot is treated as a fixed body without mass; this issue can be solved by considering the base of the robot as part of the hexacopter. Obviously, its dynamical properties such as mass and inertia need to be updated; this procedure is performed through a *CAD* model in *SolidWorks* environment, as seen in the previous chapter.

2.3. Mathematical Model of the Coupled Systems

After analysing and designing independently the two models for the hexacopter and manipulator systems, it directly follows the coupling between them.

This type of approach is referred as independent approach, where the robotic arm influence is not directly taken into account in the drone's control scheme but it is considered as an external disturbance. This allows obtaining a satisfactory trade-off between performances and scheme complexity. In fact, the main characteristics of this framework are its efficiency and ease to be modified and tuned. In this way, it can be implemented in many different scenarios, being also hardware independent. In fact, the already presented workflow can be followed for different arms or drones and, based on the specific case, the control parameters can be tuned to obtain desired performances. Moreover, the particular structure of the

control algorithm allows the possibility to refine and improve independently each single component of the scheme, depending on the task requirements and financial resources.

The overall control architecture is shown in Figure 6.

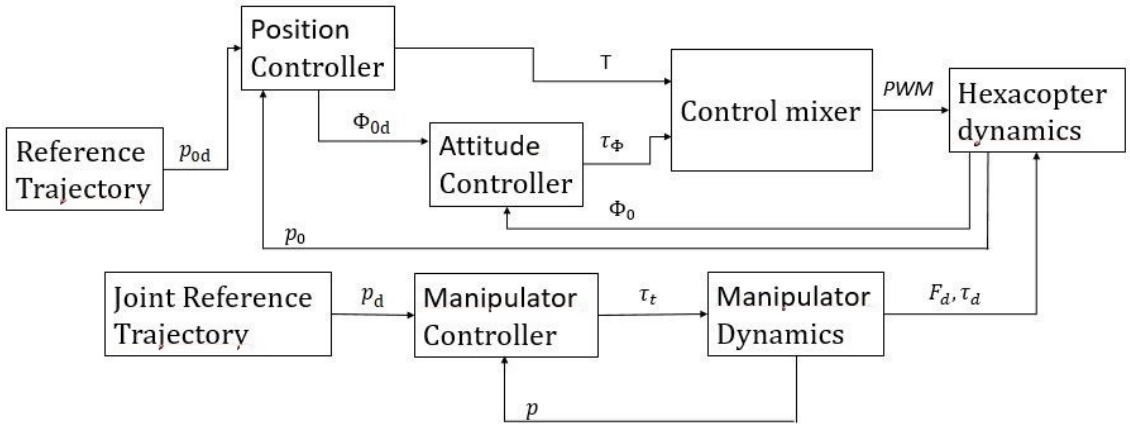

Figure 6: Overall control architecture after coupling the two systems

The first consideration regards the mechanical configuration; in fact, the position where the arm is mounted strongly influence how the centre of mass and inertia of the overall system change during flight and manipulation. Notice that mounting the manipulator superiorly to the drone's frame helps to reduce the complexity of the controller and the coupling effects between the two systems, which is therefore suitable for the independent modelling approach.

The main idea is to make all the scheme, included the manipulator logic, to work with inertial quantities. Since the goal of this work is to track predefined positions for the end effector, this solution is optimal because the manipulator can compensate eventual errors in the hexacopter position. For this reason, the trajectory of the robotic arm is referred to the inertial reference frame. This aspect is analysed in chapter 4, where the results of simulations are proposed.

The first step is to transform the end effector pose in the inertial reference frame:

$$p^i = T_0^i * T_b^0 * p^b = \begin{bmatrix} c\theta\, c\psi & s\varphi\, s\theta\, c\psi - c\varphi\, s\psi & c\varphi\, s\theta\, c\psi + s\varphi\, s\psi & x + l_{0x} \\ c\theta\, s\psi & s\varphi\, s\theta\, s\psi + c\varphi\, c\psi & c\varphi\, s\theta\, s\psi - s\varphi\, c\psi & y + l_{0y} \\ -s\theta & s\varphi\, c\theta & c\varphi\, c\theta & z + l_{0z} \\ 0 & 0 & 0 & 1 \end{bmatrix} * p^b$$

where φ, θ and ψ are respectively the drone's roll, pitch and yaw angles, *x, y* and *z* represent its centre of mass position, l_{0x}, l_{0y} and l_{0z} are respectively the *x, y* and *z* distance from the hexacopter centre of mass to the reference frame fixed with the manipulator base. In fact, the vector p^b denotes the position of the end effector in the robot base reference frame that is transformed to the inertial quantity p^i. Notice that the transformation T_b^0 represents only

a translation, since the centre of mass of the drone and the robot base associated reference frames are aligned with respect to each other.

Another modification to the scheme regards the gravity vector of the robot plant, within *Simscape* environment; in fact, this vector is a time-varying one, depending on the attitude of the hexacopter. *Simscape* provides the possibility of setting a non-constant gravity for the model, that is computed through a simple rotation between inertial and body frames:

$$g^0 = R_i^0 * g^i = \begin{bmatrix} c\theta\, c\psi & c\theta\, s\psi & -s\theta \\ s\varphi\, s\theta\, c\psi - c\varphi\, s\psi & s\varphi\, s\theta\, s\psi + c\varphi\, c\psi & s\varphi\, c\theta \\ c\varphi\, s\theta\, c\psi + s\varphi\, s\psi & c\varphi\, s\theta\, s\psi - s\varphi\, c\psi & c\varphi\, c\theta \end{bmatrix} * \begin{bmatrix} 0 \\ 0 \\ -9.8067 \end{bmatrix}$$

Concerning the actual coupling of the systems, the robot is seen as a disturbance from the hexacopter. For this reason, the disturbance forces and torques transmitted by the manipulator base to the hexacopter centre of mass have to be estimated and added to the drone's dynamics. This action is accomplished by means of Newton-Euler equilibrium equations.

The main source of disturbance are constituted by the change of inertia of the system and the accelerations of each link during their motion. For this scope, the angular accelerations of the joints are considered are measured; these values are the same for the correspondent links acceleration, under the hypothesis that both the links and the joints are totally rigid bodies.

The primary operation is to transform each link's centre of mass angular acceleration into linear accelerations. This conversion is accomplished through simple geometric relations.

It follows that all these quantities must take into account the accelerations of previous links. Likewise, every link is subject to the motion of its joint, but also all the previous links starting from the base. The procedure starts from the first link, whose linear acceleration is just the one of the associated joint. The operation is executed sequentially for all the links, by also adding the accelerations of all the previous joints; some rotation matrices are applied and the operation is repeated for the angular velocities as well. The related equations are:

$$\begin{cases} \omega_{2t} = \omega_2 + R_1^2\, \omega_{1t} \\ \omega_{3t} = \omega_3 + R_2^3\, \omega_{2t} \\ \omega_{4t} = \omega_4 + R_3^4\, \omega_{3t} \end{cases} \qquad \begin{cases} \dot{\omega}_{2t} = \dot{\omega}_2 + R_1^2\, \dot{\omega}_{1t} \\ \dot{\omega}_{3t} = \dot{\omega}_3 + R_2^3\, \dot{\omega}_{2t} \\ \dot{\omega}_{4t} = \dot{\omega}_4 + R_3^4\, \dot{\omega}_{3t} \end{cases} \qquad \begin{cases} \ddot{p}_{2t} = \ddot{p}_2 + R_1^2\, \ddot{p}_{1t} \\ \ddot{p}_{3t} = \ddot{p}_3 + R_2^3\, \ddot{p}_{2t} \\ \ddot{p}_{4t} = \ddot{p}_4 + R_3^4\, \ddot{p}_{3t} \end{cases}$$

where ω_{it} and $\dot{\omega}_{it}$ are respectively the total angular velocity and acceleration of the i^{th} link centre of mass, ω_i and $\dot{\omega}_i$ are respectively the angular velocity and acceleration associated to the i^{th} joint and $\ddot{p}_{it}$ and $\ddot{p}_i$ represent respectively the total and independent linear acceleration of the i^{th} link centre of mass.

These values are then used to compute the actual recursive Newton-Euler equations, starting from the forces/torques exerted on the end effector to finally compute the total forces and

torques transmitted to the drone's frame. No external forces/torques are considered during these computation.

The general formulation for the linear equations of equilibrium, the Newton equations, is the following, according to [22]:

$$F_{i-1,i} + F_{i+1,i} + m_i\, g_i - m_i\, \ddot{p}_i = 0$$

where $F_{i-1,i}$ and $F_{i+1,i}$ are respectively the resultant of forces applied form arm (*i-1*) and (*i+1*), m_i is the mass of each link g_i is the local gravity vector.

The general angular equations of equilibrium, the Euler equations, are:

$$M_{i-1,i} + M_{i+1,i} + d_{ci,i-1} \times F_{i-1,i} + d_{ci,i} \times F_{i+1,i} - \Gamma_i\, \dot{\omega}_i - \omega_{2t} \times \Gamma_i\, \omega_{2t} = 0$$

where $M_{i-1,i}$ and $M_{i+1,i}$ are the resultant of the torques applied from arm (*i-1*) and (*i+1*) to arm *i*, $d_{ci,i-1}$ and $d_{ci,i}$ are respectively the positions of the *(i-1)* and (*i+1*) body reference frame origin with respect to the centre of mass and Γ_i represents the inertia matrix of each link.

These equations are computed in local reference frames and then they are reported to the robot base frame to compute the actual disturbance. All the homogeneous transformations from one link to the adjacent one can be easily obtained from the robot object that describes the system; in fact, using the '*getTransform*' command, it is possible to derive the transformation between any two links of the manipulator. These matrices are successively reported to the inertial reference frame by multiplying the rotation matrix R_0^i, discussed in section 2.1, associated to the hexacopter attitude.

The last step is to compensate these forces and torques through a feedforward compensation. According to this technique, they are added to the dynamical model of the hexacopter and subtracted from the control inputs computed by the hexacopter controllers. Since the reference position adopted by the robotic arm can be known a priori, this action is performed a sampling instant before the torque is actually applied to the arm.

This operation allows a better response from the system, which results in reducing oscillations and errors in the drone's motion.

Chapter 3. Control Design

3.1. Hexacopter Robust Control Design

This section shows a short theory recall on the sliding mode technique and the actual structure of the overall control scheme. It is then analysed in its single components in sections 3.1.1 and 3.1.2. The control algorithm is finally validated by performing a simulation, as discussed in section 3.1.3.

The general control scheme that is applied both for the drone and the manipulator is shown in the following figure.

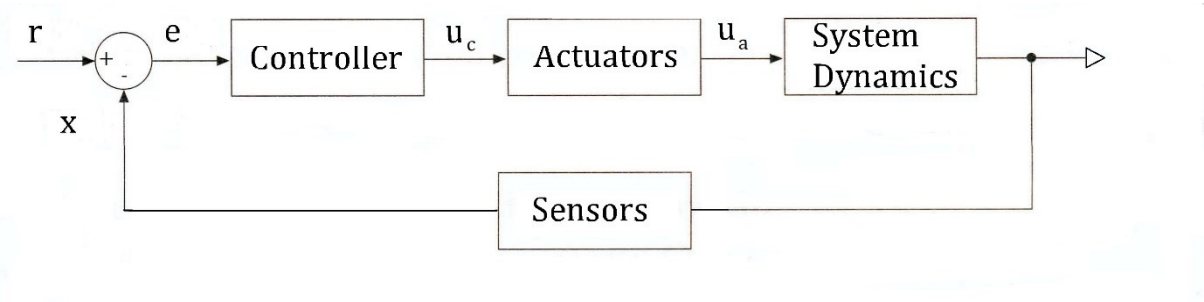

Figure 7: General control scheme

where *r, x* and *e* represent respectively the reference signal, the system states and the error, u_c is the control output and u_a is the actual command to the plant. Some assumptions are made in simulating the system performances. The first consideration is related to the states that are assumed as measured; in the case where this is not possible, an observer needs to be designed. Then the sensors and actuators are considered as ideal ones, therefore they have unitary gain and infinite bandwidth. Finally, the control problem is considered as a tracking one. This means that the reference values are time variant according to the predefined task. The trajectory generation goes outside the scopes of this work, thus it is considered as given. Concerning the simulation phase, a trajectory generator block is used in order to show the actual capabilities of the system.

The first consideration concerning the control algorithm is related to the general structure of the controller. In fact, the hexacopter is an under actuated system, meaning that four control inputs are not enough to manage the six degrees of freedom possessed by the system.

This issue has been object of many studies, and a commonly used strategy is to deploy two different controllers, one related to the position and the other to the attitude.

This leads to a hierarchical structure, exposed in Figure 8, since an outer and an inner loop are present. The first one is related to the position control and it computes the requested

vertical thrust, pitch and roll desired angles. On the other hand, the inner loop is responsible for managing the attitude of the drone. The direct consideration is that the reference is composed by four values, the three inertial positions and a yaw angle; thus, the two sacrificed degrees of freedom are the roll and pitch angles, not directly controllable anymore.

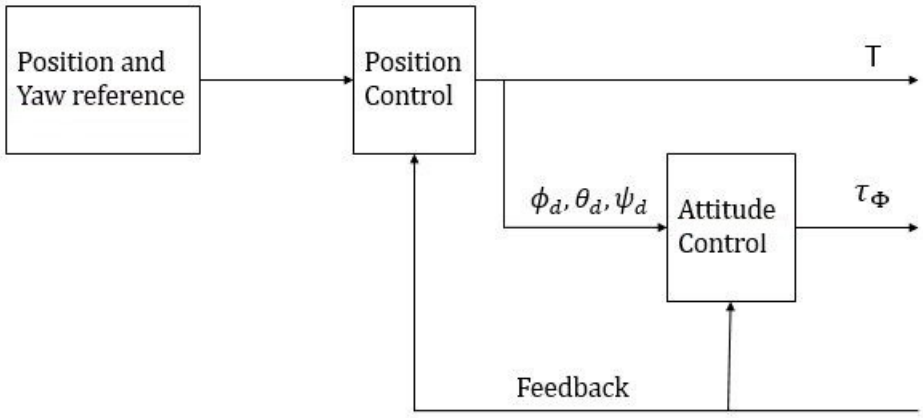

Figure 8: Hierarchical control scheme of the hexacopter

$\Phi_{0d} = [\phi_d;\ \theta_d;\ \psi_d]$ and the subscript '*d',* in this thesis work, refers to desired values.

The main reason why this structure is chosen is that the system is more customizable; moreover, it leads to an easier tuning of the different controllers in obtaining satisfying performances. It is important to notice that the inner loop needs to run at a faster rate for it to be able to respond quicker to disturbances than the outer loop. This effect guarantees better tracking of the reference trajectory since small angle errors can lead to big discrepancies in the actual *x* and *y* position.

Both the control algorithms are chosen as sliding mode controllers, which are useful for dealing with nonlinear systems. In fact, the main characteristics of this controller are that a particular choice of the sliding surface can modify the system dynamics to meet desired needs, as it guarantees interesting robustness properties related to its mathematical formulation. As a result, the closed loop response becomes insensitive to model parameter uncertainties, disturbances, and bounded nonlinearities. These features, together with a simple design and light computational weight, are suitable for our application where many disturbances (such as wind or wall and ground effect) and unmodeled dynamics can affect the system behaviour.

The main concept of the sliding mode controller is to design a surface in the state space, called sliding surface, which is reached by the system states in finite time. After the surface is reached, the controller aims to keep the states as close as possible to it. Therefore, the design of this type of controller is divided into two different phases. The first part is related to the definition of a stable surface so that the system behaves according to the design specification; the second part concerns the design of a control law that makes the surface to be attractive and invariant.

In addition, the system motion is divided in two parts, as it is very unlikely that the system states, in its initial conditions, are already leaning on the surface. The division of the system behaviour into two different phases, a reaching and a sliding phase, is shown in the next figure:

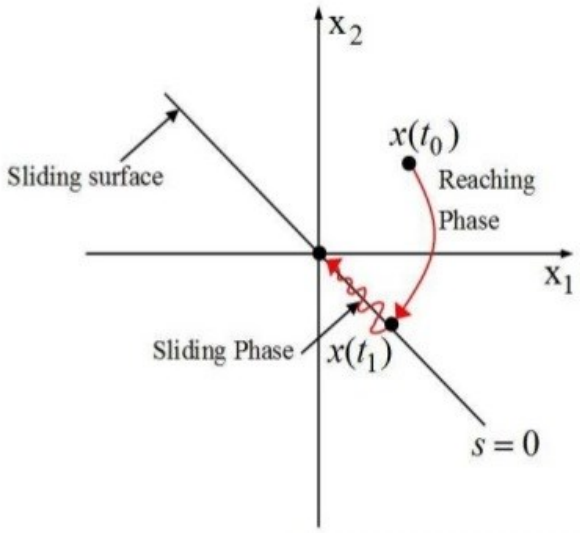

Figure 9: Behaviour of the system under the effect of a sliding mode controller

In order to accomplish this behaviour the control law has to make the surface both invariant and attractive, as discussed successively.

Consider the following *SISO* nonlinear system, as [23]:

$$\begin{cases} \dot{x} = f(x) + g(x)\,u \\ \quad y = h(x) \end{cases}$$

where $x \in \mathrm{R}^{\mathrm{n}}$ is the system state*, u is* the command input, *y* is the system output and *f*, *g* and *h* are smooth functions. A system in this form is called *affine* in *u*.

The control aim is to make the output to follow a reference input *r*. In other words, the goal is to drive the error $\tilde{y} = r - y$ to an acceptable magnitude and possibly make it converge to zero after a certain transient.

Coming back to the definition of the sliding surface, it is a scalar function of the system state: $s(x,t) : R^n \to R$. The most typical choice for the surface exploits the tracking error and a certain number of its derivatives:

$$s(x,t) = \tilde{y}^{\gamma-1} + k_\gamma\,\tilde{y}^{\gamma-2} + \cdots + k\,\tilde{y}$$

where γ is the relative degree of the system.

The choice of the different gains has to be executed in a way that, equating the surface to zero, it gives rise to a stable configuration. This is achieved when the roots of the characteristic

polynomial have negative real part, according to control theory. It is possible to demonstrate that, if the trajectory is confined to the sliding surface, then the tracking error converges to zero exponentially, according to the roots of the polynomial.

The following step is to design a control law that satisfies two main properties: invariance and attractiveness. When the surface is invariant, if the system motion is on the sliding surface, it remains on it. This implies the condition: $\dot{s}(x,t) = 0$. In fact, deriving the equation of the surface, for a number of times dependent on the relative degree of the system, the input appears. By inverting this formula, it is possible to obtain the system control law.

The total control law is completed by making the surface attractive; this means that when the state is not sliding on the surface it is pushed towards it. To accomplish this task, a discontinuous term is added to *u.* The standard choice for this parameter is the *sign* function. By defining $\dot{s}(x,t) = -k_2\ sign(s(x,t))$, it is possible to demonstrate the attractiveness of the surface. In fact, if the surface is positive its derivative will be negative and vice versa; in both cases the state reaches the surface in finite time. This property increases significantly the robustness of the system.

Anyway, the introduction of the *sign* function can cause high frequency oscillations around the surface, phenomenon called *chattering.* The solution to this problem can be found by using a smooth function instead of the discontinuous one. A typical choice is the *tanh* function, represented in Figure 10.

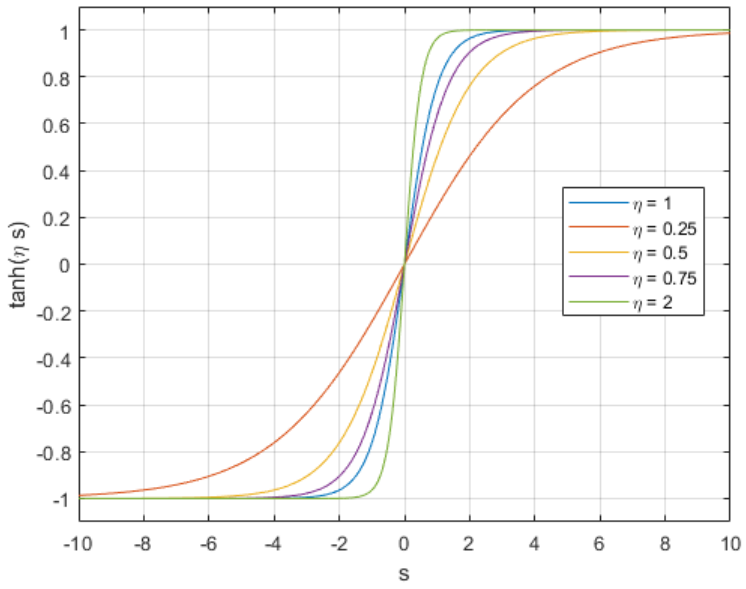

Figure 10: Hyperbolic tangent depending on the η parameter

where η is a design parameter to determine the slope of the curve. It is important to notice that this modification implies a certain deterioration of performances, depending on the slope of the curve.

Another interesting property of the sliding mode control is that, if the internal dynamics is globally asymptotically stable and the reference is bounded, then the state is also bounded for all time instants. In fact, sometimes some problems can arise regarding unstable internal dynamics and they need to be taken into account. Other disadvantage of the controller is a trade-off between performances and command activity that is difficult to manage, and in general, high command activity is present.

3.1.1 Position controller design

This section treats the realization of the position control of the hexacopter through the sliding mode technique.

As already stated, the input to this controller is the error on the states, while its output is composed by the vertical thrust and the desired values for roll and pitch angles needed to achieve desired movement in *x* and *y* directions. These values are successively given as input for the inner loop, the attitude controller.

The sliding surface is chosen in order to track desired values of the states:

$$s_l = \tilde{\dot{p}}_0 + K_l \tilde{p}_0 = (\dot{p}_{0d} - \dot{p}_0) + K_l (p_{0d} - p_0)$$

This choice allows giving desired values for positions and velocities of the hexacopter that are tracked by sending the error to zero.

The derivative of the surface is then equated to zero in order to make the surface invariant:

$$\dot{s}_l = \tilde{\ddot{p}}_0 + K_l \tilde{\dot{p}}_0 = (\ddot{p}_{0d} - \ddot{p}_0) + K_l (\dot{p}_{0d} - \dot{p}_0) = 0$$

The values of actual acceleration are substituted using the second Newton's law; the equation is then inverted to compute a vector of forces. These values represent the desired forces to drive the drone in the space and make it follow the reference inertial position and velocities $\dot{p}_{0d}$ and p_{0d}:

$$\mathrm{F} = (\ddot{\mathrm{p}}_{0\mathrm{d}} + \mathrm{K}_{\mathrm{l}}\ \tilde{\dot{\mathrm{p}}}_0)\ (\mathrm{m}\ \mathrm{I}_{3\mathrm{x}3})$$

With $F = \begin{bmatrix} F_x & F_y & F_z \end{bmatrix}^T$ for the three components of the vectorial equation.

The surface is finally made attractive by adding a *tanh* term, used to avoid the chattering effect:

$$F = (\ddot{p}_{0d} + K_l \tilde{\dot{p}}_0)\ (m\ I_{3x3}) + K_{2l} \tanh(\eta_l\ s_l)$$

where the control parameters K_l, K_{2l} and η_l are suitable three by three diagonal matrices.

The choice of these parameters is performed in order to guarantee stability of the surface and to generate smooth reference values for roll and pitch angles. This aspect is discussed in a more detailed way in chapter 4, where simulations of the system behaviour are carried out.

The values of force are finally substituted into the simplified dynamics of the system, analysed in the previous chapter, to obtain the output of the controller:

$$\begin{cases} T = \dfrac{F_z + g}{c\varphi\, c\theta} \\ \theta_d = \left(\dfrac{F_x}{T} + \dfrac{F_y}{T} t\psi\right) \dfrac{1}{c\psi\,(1 + t^2\psi)} \\ \varphi_d = \left(-\dfrac{F_y}{T} + \theta_d\, s\psi\right) \dfrac{1}{c\psi} \end{cases}$$

The use of the nonlinear dynamics is not justified, because the improvement of performances is really smaller than the increased computational weight.

3.1.2. Attitude controller design

In this section, the attitude controller is presented. The structure of the algorithm is the same as the one seen for the position control. The design of the surface is equivalent to the previous one:

$$s_a = \dot{\widetilde{\Phi}}_0 + K_a\, \widetilde{\Phi}_0 = (\dot{\Phi}_{0d} - \dot{\Phi}_0) + K_a\, (\Phi_{0d} - \Phi_0)$$

The surface is made invariant by equating its derivative to zero:

$$\dot{s}_a = \ddot{\widetilde{\Phi}}_0 + K_a\, \dot{\widetilde{\Phi}}_0 = (\ddot{\Phi}_{0d} - \ddot{\Phi}_0) + K_a\, (\dot{\Phi}_{0d} - \dot{\Phi}_0) = 0$$

Also in this case the simplified dynamics is used, obtaining the following by inverting the previous equation:

$$\tau_\Phi = J\,(\,\ddot{\Phi}_{0d} + K_a \dot{\widetilde{\Phi}}_0)$$

The surface is finally made attractive adding the *tanh* term, to avoid the chattering phenomenon:

$$\tau_\Phi = J\,(\,\ddot{\Phi}_{0d} + K_a \dot{\widetilde{\Phi}}_0) + K_{2a} \tanh(\eta_a * s_a)$$

where the control parameters K_a, K_{2a} and η_a are suitable three by three diagonal matrices.

These gains are chosen to guarantee overall stability of the surface and a fast response of the system. In fact, good performances in attitude control make possible to obtain fast and smooth movements of the drone in the three-dimensional space. These choices are discussed in a more detailed way in dealing with simulations, in chapter 4.

The last consideration concerns the desired profiles of velocity and acceleration; as noticeable from the equation of the sliding surfaces, this type of control is capable of tracking concurrently desired values of velocity and acceleration. In this application, only velocity references are given to the controller, while the acceleration references are set to zero. Again, this operation results in reasonably slower performances but great reduction of oscillations that, combined with other disturbances, can lead to instability.

3.1.3. Validation of the control architecture performances

The model and controllers are implemented in *Simulink* environment in order to test the capabilities of the control architecture. This operation is executed to tune in an efficient way the control parameters, to obtain satisfactory performances that simplify the final tuning after actually coupling the drone and manipulator systems.

The control performances are evaluated by analysing the step response of the hexacopter system in time. From a practical point of view, this simulation represents a case of hovering in a point in space.

The reference values for the inertial position and the yaw angle are represented by a constant value, with magnitude respectively equal to 0.5, 1, 1.5, and 0.6.

Regarding the general setup for the mentioned simulation, the initial conditions for the positions and velocities of the hexacopter are set equal to zero.

Concerning the sampling times, the entire scheme runs at a frequency equal to $f_s = {}^{1}/_{T_s} = {}^{1}/_{0.02}$ except for the attitude controller. In fact, the selected sampling time for the inner loop is: $T_{si} = 0.01$. In this way, it is updated two times faster than the rest of the components, allowing a better tracking of the reference inertial position.

The first choice for the parameters of the position sliding mode controller is:

$$K_l = \begin{bmatrix} 1.5 & 0 & 0 \\ 0 & 1.5 & 0 \\ 0 & 0 & 1.5 \end{bmatrix};\ K_{2l} = \begin{bmatrix} 1.5 & 0 & 0 \\ 0 & 1.5 & 0 \\ 0 & 0 & 1.5 \end{bmatrix};\ \eta_l = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{bmatrix}$$

Similarly, the control gains related to the attitude control are defined as:

$$K_a = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix};\ K_{2a} = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{bmatrix};\ \eta_a = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

These values ensure stability of the sliding surfaces and fast response of the switching function *tanh*; moreover, it is formulated in a way that eliminates the chattering effect.

This choice gives the same importance to manoeuvres in different directions. By proper tuning, it is possible to favour some movements and slow down others to improve stability in presence of disturbances.

Figure 11 shows the actual position of the drone in time, with respect to the reference values.

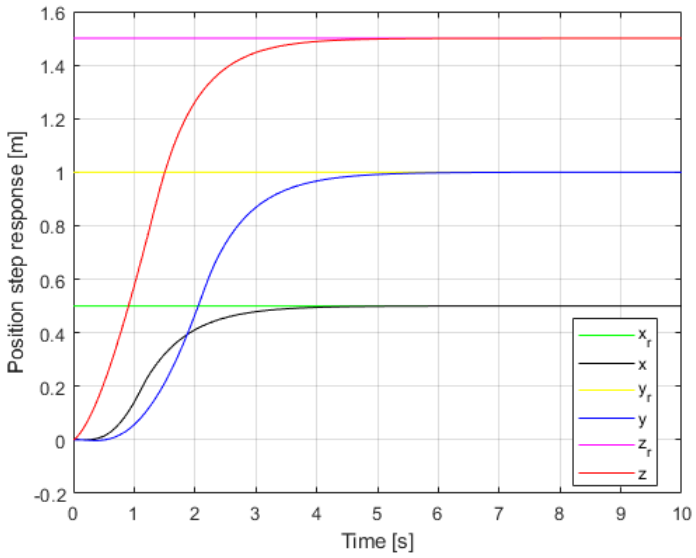

Figure 11: Inertial position step response

It is clear from the graph that no overshoot is present and the settling times are $t_{s,1\%} = 4.02, 4.89, 3.84$ respectively for *x, y* and *z* variables.

This behaviour is particularly suitable for an aerial manipulator system because no oscillations arise in the manipulator motion; in fact, it is important to make the overall performances of the systems as smoother as possible, since important disturbances are not considered in this work.

It is then possible to affirm that the drone is capable of covering a certain distance in a reasonable amount of time, hovering in a controlled manner that is not characterized by oscillations.

These data confirm that the motion along the vertical direction is faster since the position controller has the capability of directly acting on the thrust force. In fact, as already discussed, the motion in the horizontal plane *x*-*y* is controlled through the roll and pitch angles, whose step response is observable form the next two figures:

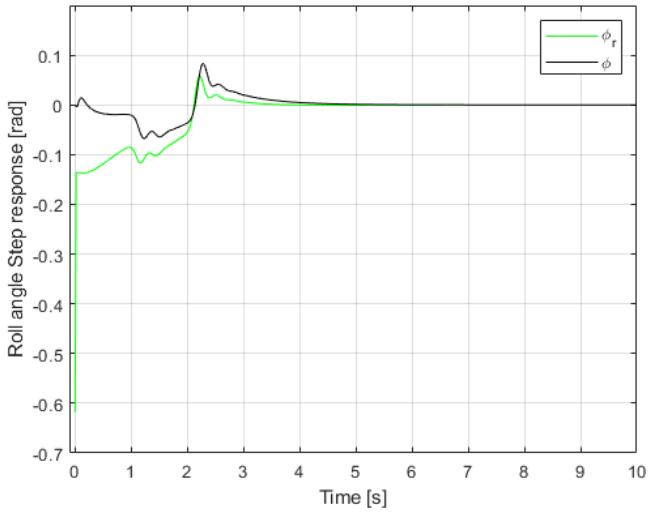

Figure 13: Pitch angle step response

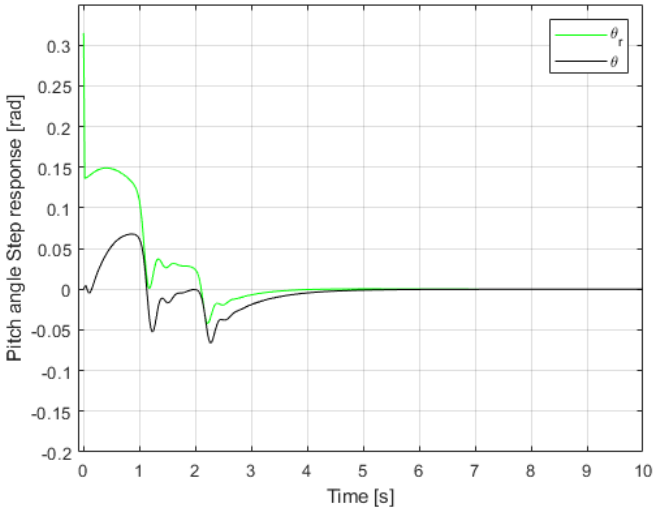

Figure 12: Roll angle step response

The reference values given from the position controller change very fast in time. Consequently, the controller is not able to follow them promptly; this is the principal explanation why the motion of the hexacopter is slower in horizontal directions. In fact, giving more weight to the switching function of the sliding mode controller it is possible to make the control response faster, at the price that some oscillations arise due to the chattering effect. Anyway, the system response is overall satisfying at this phase of the project because the presence of oscillations can represent a problem in the case of manipulation; moreover, it is enough fast with respect to the proposed objectives. This is observable also from the yaw angle response, shown in Figure 14.

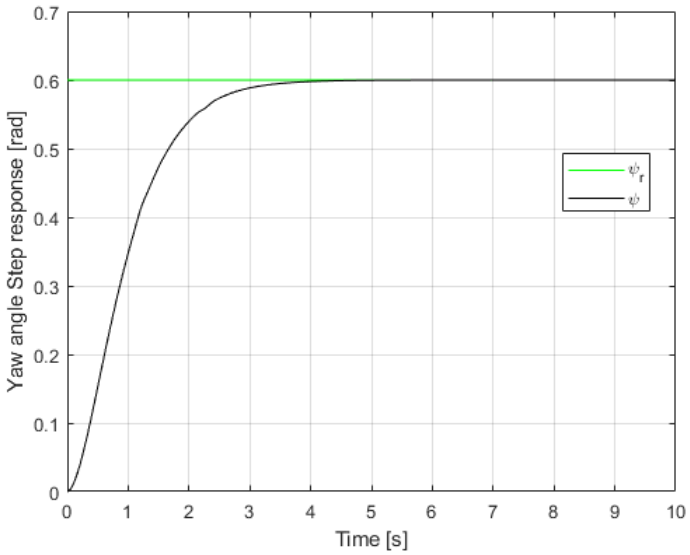

Figure 14: Yaw angle step response

Also here no overshoot is present and the settling time is equal to 3.4 seconds.

The total trajectory followed by the drone is visible is Figure 15:

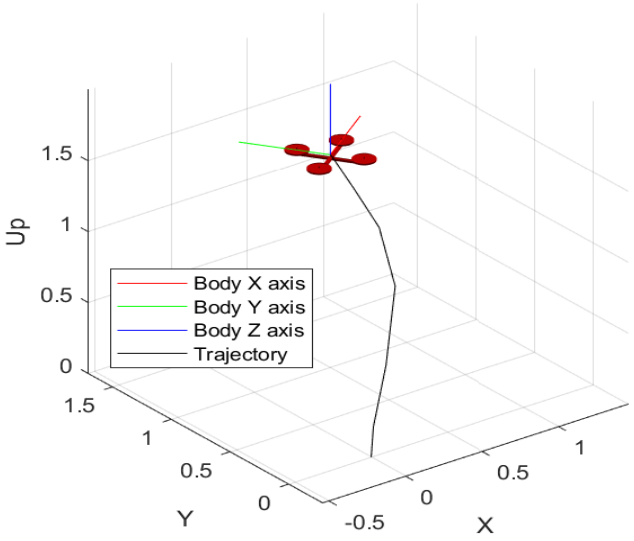

Figure 15: Trajectory of the drone in 3D space

This graphical representation is obtained using the *UAV* animation block, available from the *Robotics System Toolbox*. This is a very complete tool in the robotics field; in fact, it is used for the realization of the manipulator control architecture, as discussed in the next section. The inputs to the function are the two parameters that respectively represent the position of the hexacopter centre of mass and its attitude angles, under the form of quaternions. The function can be slightly customized regarding the shape and size of the drone.

In conclusion, the presented control architecture represents a valid solution in the control of the hexacopter, intended as an independent system. The control parameters are adjusted after the coupling with the manipulator architecture, in order to furtherly improve the system performances.

3.2. Manipulator Control Design

This section deals with the realization of the control algorithm related to the four degrees of freedom manipulator.

Firstly, in section 3.2.1, two possible choices of controllers are presented, comparing their advantages and characteristics. Then, an operational control architecture is chosen and implemented in *Simulink* environment. Its behaviour is successively simulated and its performances analysed and validated, as discussed in section 3.2.2.

3.2.1. Possible control strategies and choice of operational space control with gravity compensation

This section considers the realization of the control structure. Several different algorithms can be found in literature, with respect to the proposed task. In fact, the control architecture has to meet specific requirements of the considered application, allowing an advantageous trade-off between performances and cost. For this reason, some aspects of the system are emphasised, neglecting others with reasonable assumptions.

The control structures commonly used in robotics applications, as mentioned in [25], can be divided into two major families: task space and joint space controllers, considering that the task description is often specified in the task space while control action are defined in the joint space. These categories, in turn, are divided in many other possible implementations, which focus on particular aspects of the system.

For example, the joint control architectures can be identified as decentralized (independent) and centralized control. The first one is a *SISO* type and it is implemented through a local controller for each motor that takes into account only local variables (joint position and velocity); this scheme is very common in industrial applications due to its simplicity and robustness.

On the other hand, the centralized structure offers a more complete understanding of the overall system behaviour, being composed by only a single *MIMO* controller that commands all the motors. It is used when the task requires fast velocities since disturbance torques are not neglectable anymore and they can cause big errors. In fact, this kind of architecture takes directly into account the complete dynamical model of the robot, considering all the coupling effects between the links. This increased accuracy implies a significant increase on the computational weight of the algorithm.

Concerning the task space control, several structures have been studied in literature, and they are strongly related to the proposed task, directly taking into account the interaction of the robot end effector with the external environment.

To keep the complexity as low as possible, without deterioration of performances, a basic control structure composed by a feedback and a feedforward component is adopted in this work and it is shown in the following figure:

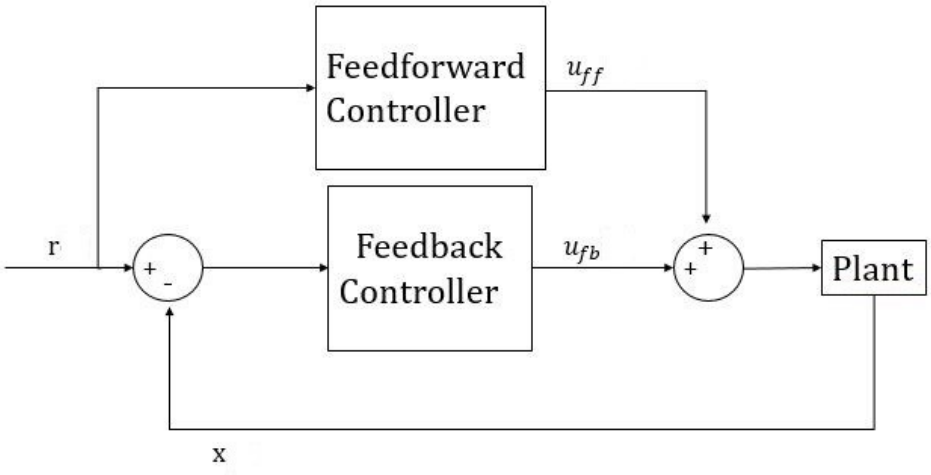

Figure 16: General control architecture based on feedforward and feedback component

where u_{ff} and u_{fb} represent respectively the feedforward and feedback components of the control action.

Two different control architectures, based on the presented scheme, are implemented and compared.

The first one is an independent joint space control, whose general structure is reported in the Figure 17.

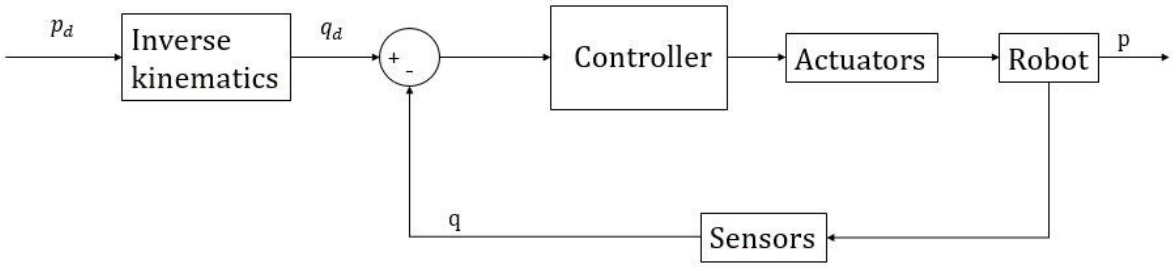

Figure 17: Joint space control architecture

The inverse kinematics block transforms the desired task space positions and velocities into desired joint space reference values. This function is implemented through the related *Simulink* block available from the *Robotics System Toolbox*; moreover, it allows the user to choose a tolerance on each component of position and orientation of the end effector and to customize the algorithm, setting a trade-off between accuracy and time of response. The sensors, considered as ideal, measure the joint angles to be compared with the desired values coming from the inverse kinematics.

The controller is composed by two different components. It exploits an inverse dynamics block as feedforward component. The inputs are the joint positions, velocities and accelerations; the latter two variables are computed through a discrete derivative block from

the desired joint positions obtained from the inverse kinematics. This may imply the presence of noise and unwanted peaks that must be taken into account.

The inaccuracies of the model are compensated by the feedback part that is realized through four different *PD* controllers, each one acting on the correspondent joint angle error. The two components of torque are finally summed up and saturated at the nominal maximum value obtainable from the motors: $\pm 1.5\ Nm$.

Passing to the task space control, it falls into the category of operational space control with gravity compensation, as discussed in [26]; its general structure is shown in the next figure:

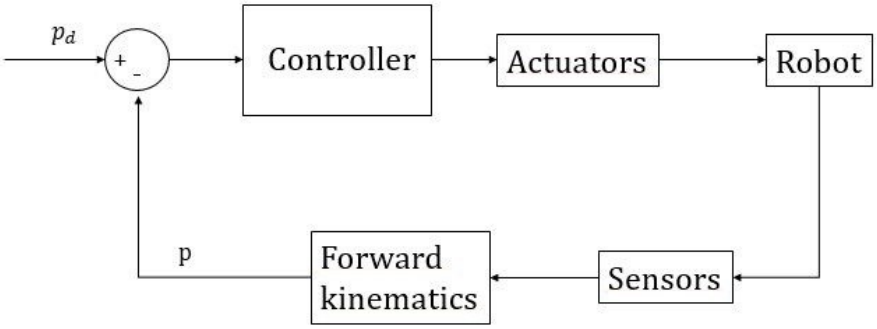

Figure 18: Task space control architecture

It is very similar the one already presented, with the difference that the inverse kinematics function is not needed anymore, since the logic of the controller is realized using task space quantities. For the same reason a forward kinematics function is added after the plant dynamics, in order to convert the joint angles into three-dimensional pose of the end effector. In fact, these variables are used as feedback to be compared to the desired tool centre point position.

The trajectory design goes outside the scope of this work and a trajectory generation block is used for simulation purposes, as discussed in chapter 4.

The position error is the input to a *PID* controller, while its output represents the required force in space to drive the end effector to the desired position; these values are concatenated with three zeros representing the needed torques. In fact, we are not interested in orientation in this part of the control design, since the four degrees of freedom of the manipulator cannot act on the six ones of the end effector. This issue must be addressed in the trajectory generation phase, to make the drone move in a way that facilitates the robot motion with respect to the proposed task.

The forces are then multiplied to the transpose of the Jacobian matrix, according to the relation between joint and task space static forces/torques under the hypothesis of small pose errors:

$$\tau_j = J^T * \begin{bmatrix} 0 \\ 0 \\ 0 \\ F_f \end{bmatrix}$$

The scheme of the controller is presented in Figure 19:

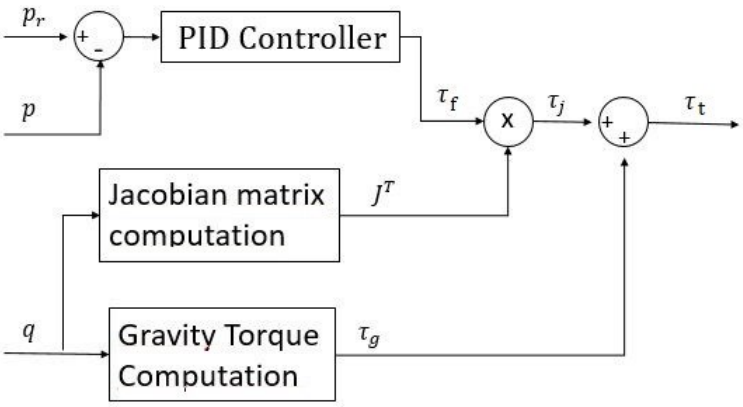

Figure 19: Operational space controller with gravity compensation

The feedforward component is realized compensating the joint space gravity torque, the torque needed by each joint to counter the gravity effects. This solution leads to a feedback action more independent on the actual configuration of the manipulator that is affected by fast gravity changes depending on the orientation of the hexacopter.

It follows that the tuning process for the *PID* controller is faster and can be done independently.

An issue is related to the gravity compensation block provided by the toolbox; in fact, it cannot take into account a time varying gravity vector. This problem is solved by implementing a *MATLAB* function that computes the potential energy of each link's centre of mass and derives these values with respect to each generalized coordinate. The total potential energy is:

$$P = \sum_{i=1}^{4} P_i = \sum_{i=1}^{4} (m_i \, g^T \, cm_i)$$

where m_i is the mass of each link, g is the gravity acceleration vector and cm_i represents the position in space of each link centre of mass. These values are extracted from the *MATLAB* model of the robot, which contains all the kinematic and dynamical information related to the manipulator.

The equation of the potential energy is then derived with respect to each joint variable, to compute the torque necessary to keep the arm in the current configuration:

$$\tau_g = \left[\frac{dP}{\delta q_1} \quad \frac{dP}{\delta q_2} \quad \frac{dP}{\delta q_3} \quad \frac{dP}{\delta q_4}\right]^T$$

The overall command torque is finally computed as the sum of the open and closed loop components and it is saturated at a value equal to $\pm 1.5\ Nm$.

Passing to the comparison between the presented algorithms the main consideration regards their computational weight. The task space architecture is significantly computationally lighter; in fact, in the joint structure the inverse kinematics is solved every time step and this is very resource consuming. Moreover, the task space control directly acts on 3D quantities, thus it is more sensitive to the environment and it is easier to follow trajectories in space. On the other hand, the joint behaviour is difficult to predict and needs to be taken into account to avoid collisions with the hexacopter frame.

Differently, the joint control is decentralized (four *PD* controllers) thus it is more versatile and can be tuned to obtain desired performances, giving more weight to some manoeuvres for example; however small errors in the inverse kinematics procedure can be translated into big errors in space and need to be compensated. In addition, the joint structure needs to deal with singularities problem (inverse kinematics issue).

Taking these statements in consideration, and noticing that the two different control architectures show similar performances, the direct choice goes to the task space architecture. For these reasons, only this control system's performances are presented in this work.

3.2.2. Validation of the control architecture performances

The model and overall control architecture are implemented in *MATLAB* and *Simulink* environment following the above presented workflow. This section discusses a general case of manipulation; this simulation is intended as illustrative for what concerns the tracking capabilities of the inertial position of the tool centre point of the gripper.

The robot, in its initial conditions, is in a fully extended position almost parallel to the *x-y* horizontal plane; the related joint variables are the following: $q_1, q_3, q_4 = 0\ rad$ and $q_2 = 1\ rad$. The second main point of the trajectory results in a fully extended arm but in the *y-z* plane, resulting in $q_1, q_2, q_3, q_4 = 0\ rad$.

A cubic polynomial is used to interpolate the initial and intermediate desired positions to be tracked from the end effector. The simulation ends with the manipulator coming back to its starting position.

The different configurations adopted by the arm are exposed in the following images:

Figure 20: Simscape visualization of manipulator motion

These graphical representations of the manipulator are obtained through the *Simscape Mechanics Explorer*, the tool's visualization feature.

Concerning the general setup of the simulation the same solver, a fixed step *ode14x* algorithm with sample time equal to 0.01, is used for both the *Simscape* plant and the overall control architecture.

The control gains of the *PID* feedback controller are chosen as $K_p = 300;\ K_i = 50$ and $K_d = 2$.

This choice guarantees a response that is not too aggressive, aiming to reduce the possible oscillations that can arise.

Notice that the presence of the integral terms ensures that eventual steady state errors are eliminated in time.

The positions and related desired values of the tool centre point are presented in Figure 21, 22 and 23 while the correspondent joint variables are shown in Figure 24:

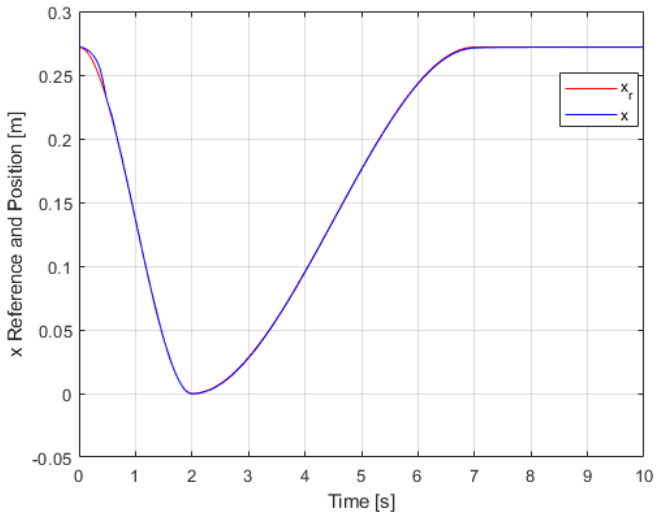

Figure 22: x tracking of the reference trajectory

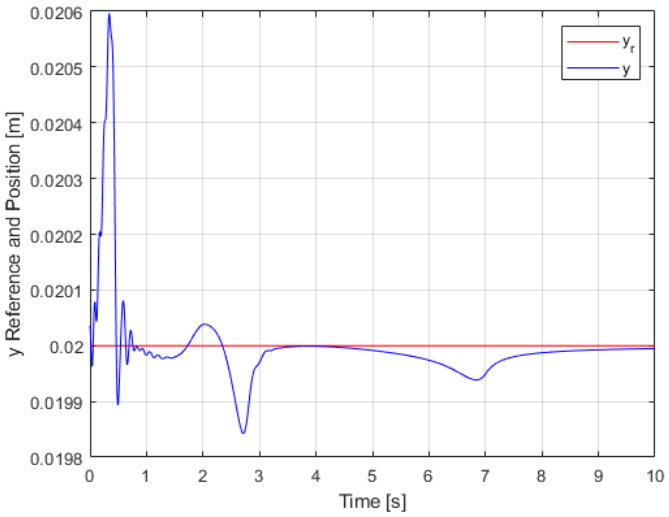

Figure 21: y tracking of the reference trajectory

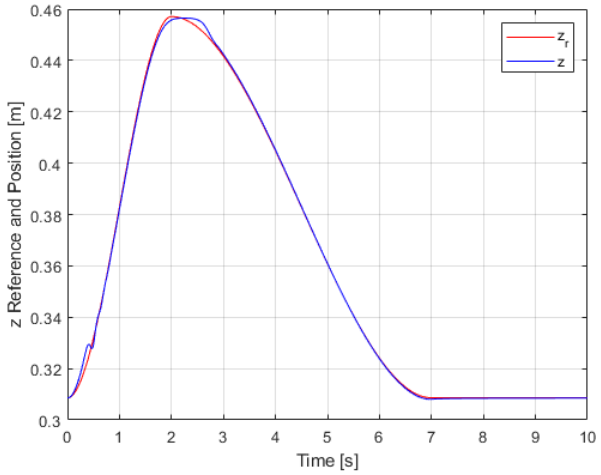

Figure 24: z tracking of the reference trajectory

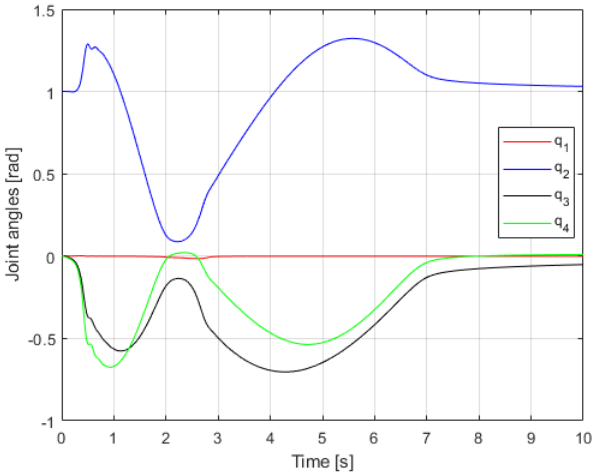

Figure 23: Measured values of joint variables

The first part of the system response shows a certain overshoot. This condition depends from the fact that the first manoeuvre executed by the robot is quite fast; in fact, the transition from the initial configuration to the intermediate one happens in a time interval equal to two seconds.

On the contrary, the second transient, to be completed in other five seconds, occurs in a smoother way.

Anyway, a small error is noticeable in the *z* position tracking, when the robot is ready to come back to its original position; this behaviour happens due to the fact that the arm is working in maximum extension conditions. In fact, this is a singular configuration for anthropomorphic manipulators and can lead to unexpected problems.

This issue is taken into account in simulating the complete control architecture of the coupled systems, avoiding that the robot gets too close to these critical configurations.

Concerning the tracking of *y* positions, it is possible to notice that it is a constant value. The errors in this case are acceptable, since their magnitude is of the order of 10^{-4}; it is also observable the difference between the first and the second manoeuvre, that generates oscillations with smaller amplitude and frequency.

The integral action is visible as well since all the errors, after the time instant equal to 7 seconds, converge to zero.

The evolution in time of the joint variables is fairly smooth and surely realizable from the actuators responsible for the joint's motion.

The control inputs are decomposed in their constitutive part of open and closed loop, and they are represented in Figure 25 and 26.

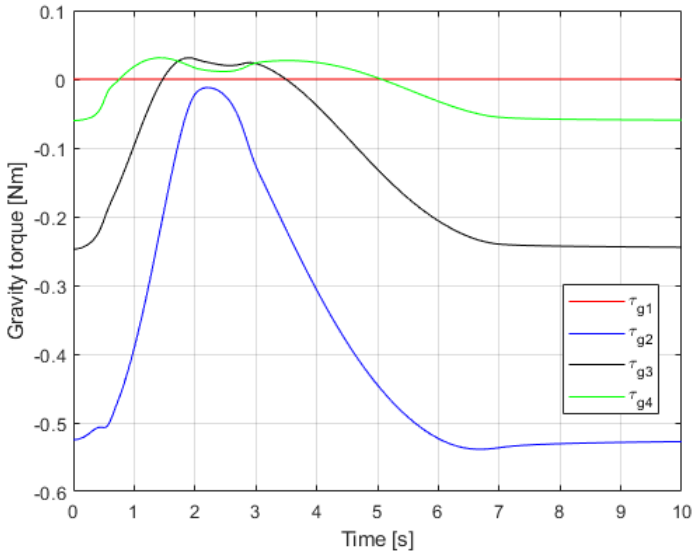

Figure 25: Gravity compensation torque

The torques designated for compensating gravity effects are coherent with the robot motion. In fact, they start from a certain value that is bigger for the second joint and decreases for successive joints; then they sensibly drop when the robot is almost in “vertical” position to come again to the starting values in the end of the simulation. The component associated to the first joint is zero for the whole simulation, since the joint’s motion axis is parallel with the gravity acceleration vector.

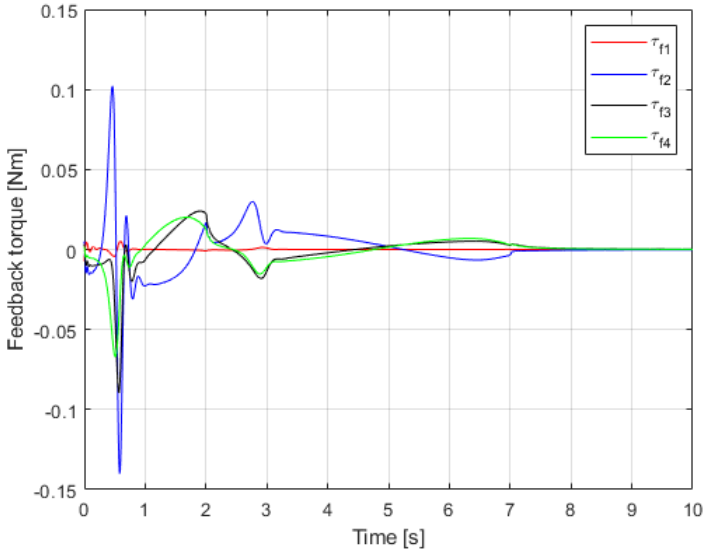

Figure 26: PID torque component

Some oscillations are present in the feedback control action. Also in this case their amplitude depends on the rapidity of the manoeuvre and it can be managed by increasing the control action or by deploying suitable reference trajectories. Other useful operation is represented by filtering the inputs and outputs of the controller, to remove high frequency components.

These aspects are taken into account in chapter 4, where the behaviour of the overall system composed by the drone and manipulator is analysed.

As conclusion, the presented performances in time are adequate with respect to the requirements and they show good perspectives for the final control architecture.

Chapter 4. Simulation Results

The overall control architecture is implemented in *MATLAB* and *Simulink* by combining the hexacopter and manipulator systems, as described in the previous chapters. Anyway, the control parameters are modified to guarantee improved performances with respect to the proposed task.

In this chapter, the setup of the simulations is firstly presented, whose results are successively analysed and compared between them.

4.1. Trajectory generation and general setup

The first considerations are related to the reference trajectory. The realization of suitable positions and velocity profiles is very important and related to the proposed task. These trajectories must take into account the fact that the robot gripper is controlled only in its position but not in its orientation; regarding this, the desired position for the drone has to be designed in order to avoid unwanted orientations of the end effector or critical configurations of the manipulator.

Different solutions are addressed to accomplish this task in an efficient way. In particular, two different blocks provided by the *Robotics System Toolbox* are considered; they are the polynomial and trapezoidal velocity profile trajectory generators. They are both meant to interpolate different positions in space creating custom time laws for the specified time intervals; the results are smooth behaviours with different profiles for positions and velocities.

Trapezoidal velocity profiles are a well-known topic in robotics application; they allow to exploit the full capabilities of the robot and also to add particular constraints in the robot motion. On the other hand, the polynomial generator allows meeting predefined points at determined time instants; moreover, a continuity of motion can be achieved, specifying the velocity conditions to be met at point boundaries. For this reason, this trajectory generator is preferred to the previous one.

The first parameter to consider is the order of the polynomial expression used to interpolate the positions. Available choices are third (cubic) and fifth order (quintic); the adopted one is the cubic order polynomial, sufficiently precise for its purposes. The other inputs are the simulation time, the actual waypoints that have to be met and the correspondent time intervals that define the time they are encountered.

First off, the trajectory of the drone is computed. These values are successively suitably rotated and summed up to the manipulator's end effector trajectory in space. In fact, as discussed before, in this way the robot control exploits only inertial variables, being in this way able to compensate drone's position errors.

The chosen waypoints for the hexacopter trajectory and related time instants are:

$$P = \begin{bmatrix} 0 & 0.5 & 1 & 1.5 & 1.5 & 1.5 & 1.5 & 1.5 & 1 & 0 \\ 0 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 1 & 0 \\ 0 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 0.5 & 0 \end{bmatrix} m$$

$$T = [0 \;\; 6 \;\; 12 \;\; 24 \;\; 32 \;\; 40 \;\; 44 \;\; 48 \;\; 54 \;\; 60]\; s$$

While the specified velocity boundary conditions for the drone are the following:

$$V_{b_d} = \begin{bmatrix} 0 & 0.1 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0 \\ 0 & 0.1 & 0.1 & 0 & 0 & 0. & 0 & 0 & 0.05 & 0 \\ 0 & 0.1 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0 \end{bmatrix} {}^{m}/_{s}$$

This choice makes the overall trajectory to be divided in different phases as follows:

- A hovering manoeuvre for the time interval that goes from 0 to 24 seconds; in this interval the position of the manipulator is kept constant, at its initial conditions.
- A manipulating phase from 24 to 48 seconds, where the drone aims at keeping its position and attitude as constant. During this time interval the arm moves, occupying four different positions in the *y-z* plane and influencing the hexacopter motion.
- A conclusion phase to the end of the simulation, where the hexacopter comes back to the origin of the inertial reference frame.

In this formulation, the simulation represents a valid case concerning the robot and hexacopter motion. The manipulator trajectory is designed using the polynomial trajectory as well and the expected motion of the robot is represented in the following images:

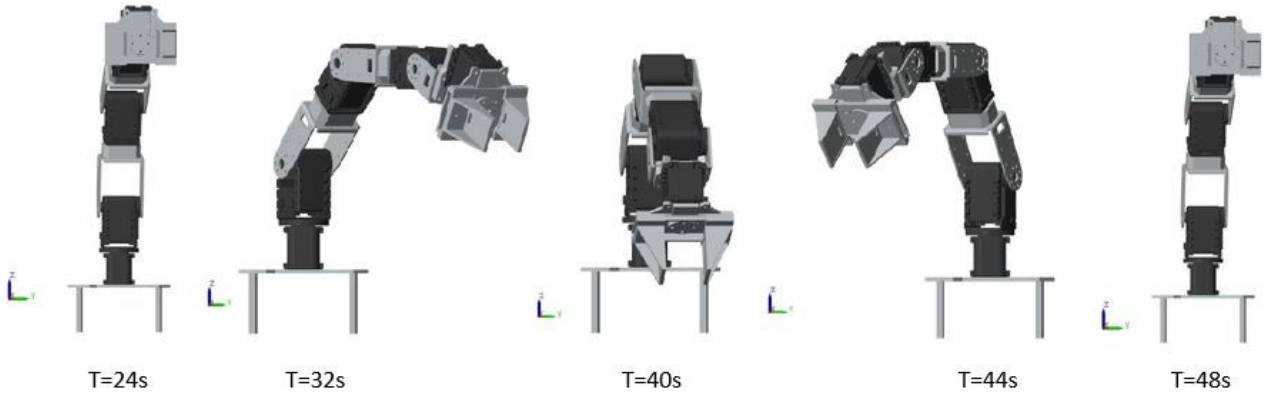

Figure 27: Robot configurations at specified time instants

Regarding the yaw angle reference generation, it is implemented as a custom function. In particular, it increases from 0 to $0.5\,rad$ and it is kept constant successively during the manipulation interval; finally, it goes back to the original value of zero.

Concerning the other parameters of the model, the chosen solver is a fixed-step *ode14x*, suitable for stiff models. The adopted sampling times are the following:

- $T_s = 0.02\ s$ for the drone position controller and dynamics, and for the manipulator controller and dynamics
- $T_{si} = 0.01s$ for the drone attitude controller that needs to run faster

The following considerations regard the control parameters that need to be adjusted after the coupling of the two systems. In particular, the sliding mode parameters for the linear and angular control are chosen as follows:

$$K_l = \begin{bmatrix} 0.05 & 0 & 0 \\ 0 & 0.05 & 0 \\ 0 & 0 & 0.05 \end{bmatrix};\ K_{2l} = \begin{bmatrix} 0.8 & 0 & 0 \\ 0 & 0.8 & 0 \\ 0 & 0 & 0.8 \end{bmatrix};\ \eta_l = \begin{bmatrix} 8 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 8 \end{bmatrix}$$

$$K_a = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix};\ K_{2a} = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{bmatrix};\ \eta_a = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

It is possible to notice that the attitude gains are the same as seen in chapter 3, while the ones related to the position control have been modified. This operation results in a slower response from the position outer loop. This choice is justified from the fact that the main source of oscillations for the robot comes from fast changes in attitude; these rapid variations imply a fast change in the local gravity vector that need to be compensated by the control action.

It is important to remark that these parameters give the same importance to manoeuvres in different directions and they can be tuned to favour some movements with respect to others.

The initial conditions for the drone's positions and velocities are set to zero, while for the arm's joints they are: $q_0 = [0\ \ 0.2\ \ 0.8\ 0.55]^T\ rad$, that represents a stable configuration leading to a reduced disturbance torque. Moreover, it is assumed that the following variables are measured: hexacopter centre of mass inertial position and velocity, joints' positions, velocities and accelerations.

Concerning the manipulator *PID* control parameters, they are set as: $K_p = 600;\ K_i = 50$ and $K_d = 2$; only the proportional gain has been modified, in order to increase the feedback control action on the system behaviour. This choice guarantees a fast response, and at the same time reducing the amplitude of oscillations and magnitude of steady-state errors.

The presented setup is adopted in the simulation exposed in the next section.

4.2. Simulation results and analysis

The results of the simulation are discussed here. The next figures expose the time evolution of the inertial position and reference for the hexacopter's trajectory. In particular, Figure 29 shows the position of the hexacopter with respect to the reference values and in Figure 28, a detail of the manipulation transient is presented.

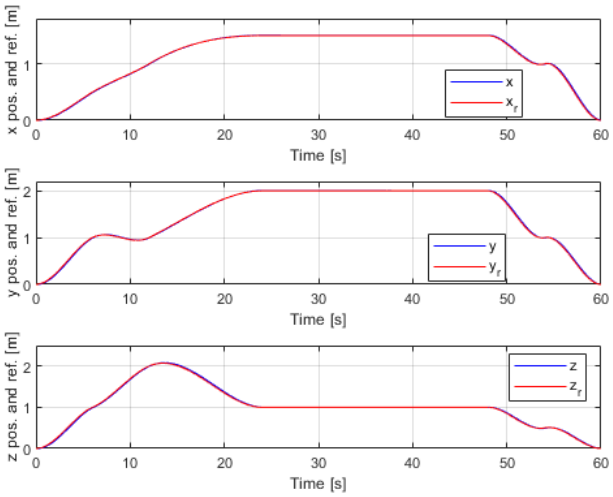

Figure 29: Position values and references of the hexacopter

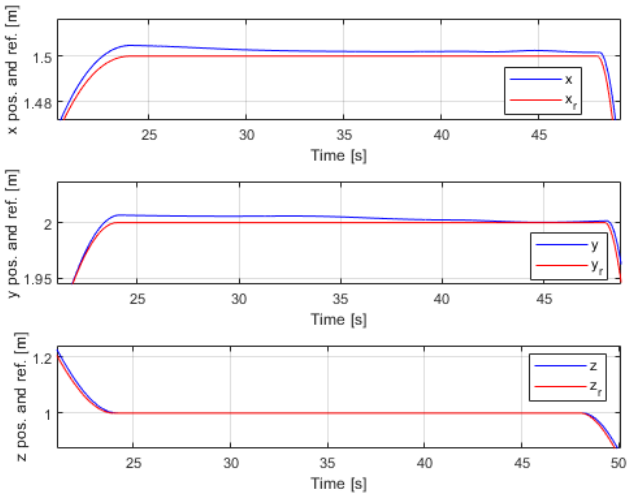

Figure 28: Detail of position values and references of the hexacopter

The first consideration regards the tracking of time-varying signals; in fact, it is possible to observe that the controller shows a small lag, of the order of millimetres, in tracking the reference trajectory. This is because the position sliding mode control parameters have been modified. These modifications, as said, have the effect of generating smoother reference values for the desired roll and pitch angles; this operation translates in the fact that the hexacopter motion is slower in *x* and *y* direction.

On the other hand, the vertical control is directly manageable from the motor thrust command input, thus it is slightly more accurate; moreover, it is not subjected to noticeable disturbances that affect mainly the attitude and consequently the *x* and *y* position.

According to these considerations, the drone's overall position performances are satisfying.

Figure 29 shows the detail of the time interval from 24 to 48 seconds, where the robotic arm moves and the drone has to keep its position as constant. Here a small overshoot is present, but the error converges to zero in time; moreover, it is possible to observe the action of the first disturbances caused by the manipulator motion, resulting in small oscillations in the drone centre of mass position.

This effect depends mainly on the disturbance torque that influence the attitude of the drone, as noticeable from the attitude angles in Figure 31:

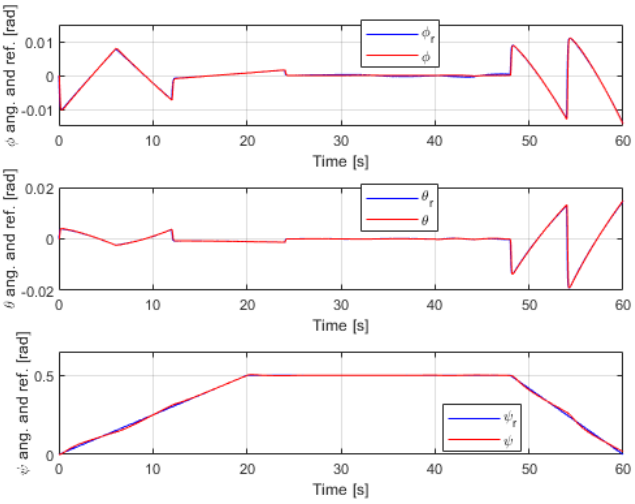

Figure 30: Attitude angle values and references of the hexacopter

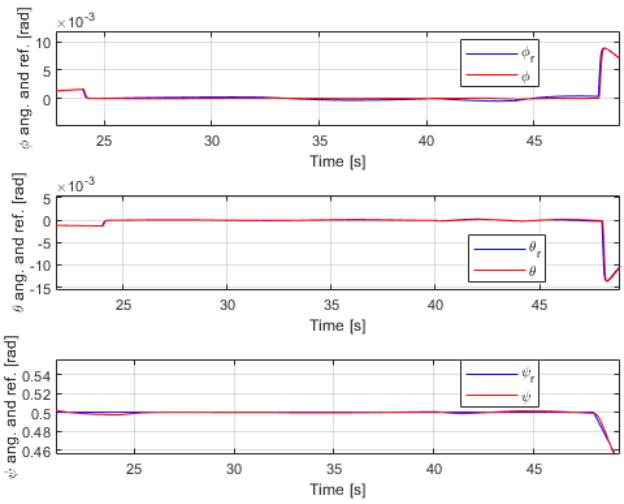

Figure 31: Detail of attitude angle values and references of the hexacopter

The disturbance torques, represented in Figure 32, act mainly on the roll and yaw angles because the robot motion is confined to the *y-z* plane in this particular simulation. A detail of the attitude measurements and references during the manipulation time interval is exposed in Figure 30, where this effect is noticeable. Nevertheless, it is possible to affirm that also the attitude sliding mode control shows good performances in time.

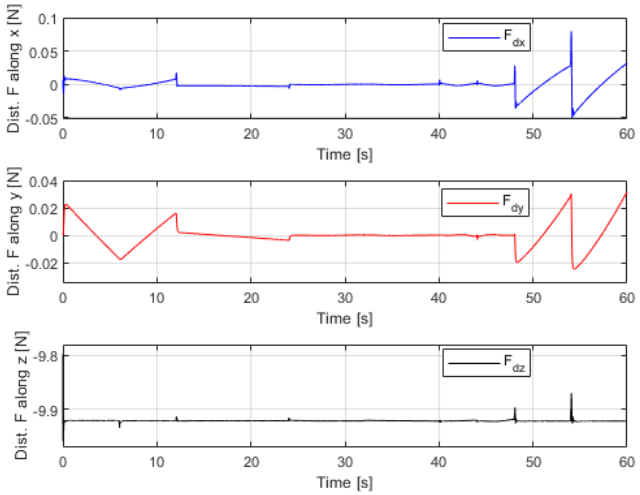

Figure 33: Disturbance forces transmitted from the robot to the hexacopter

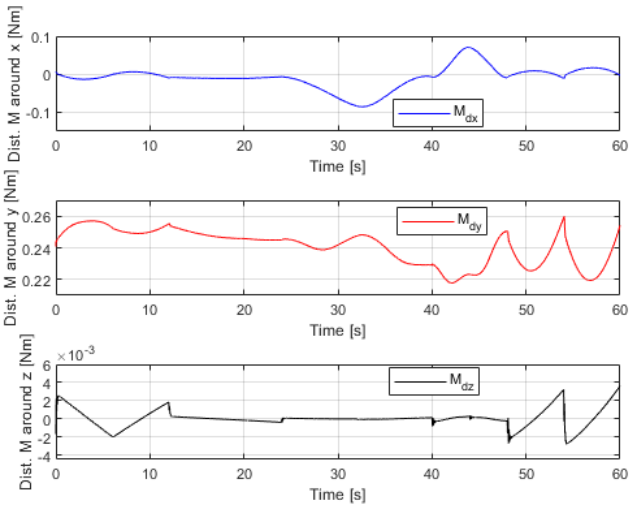

Figure 32: Disturbance torques transmitted from the robot to the hexacopter

Concerning the disturbance forces, represented in Figure 33, their magnitude is very small, thus they do not affect very much the system behaviour. This does not hold for the force

along the vertical axis, but it is successfully compensated since the position control directly acts on the vertical thrust. On the other hand, the disturbance torques are quite big if compared to the control inputs, shown in Figure 34. This consideration reflects in the small attitude oscillations observed previously. Regarding the general evolution in time of both disturbance forces and torques, it is possible to affirm that they constantly vary according to the robot motion but also some peaks are present at the time instants where the drone performs a rapid change in attitude. This happens because fast variations of attitude result in errors in the manipulator position due to rapid changes in the gravity vector.

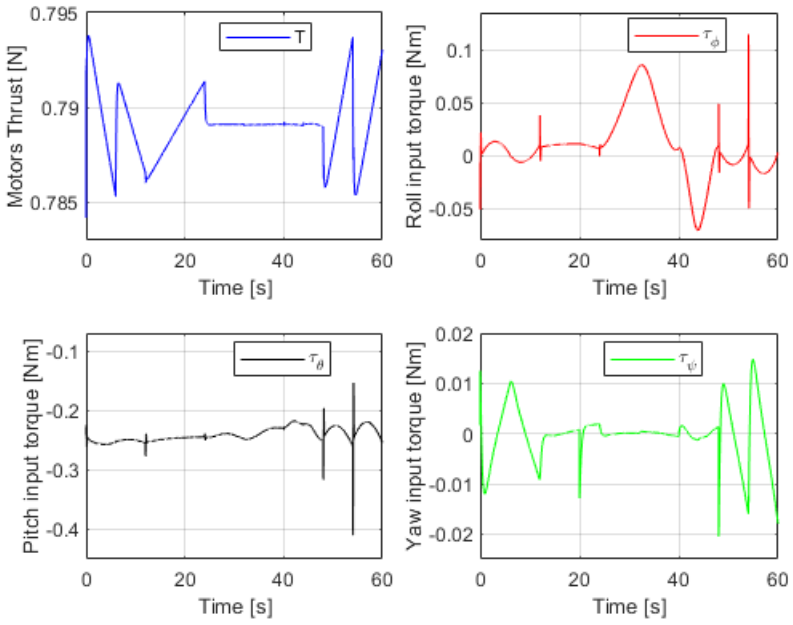

Figure 34: Control inputs to the hexacopter system

The above mentioned problem about peaks is visible also from the input representation; in fact, after the drone meets a waypoint, the reference values change in a rapid way to guarantee that it can move correctly towards the next point.

Moreover, during the manipulation time interval, the control action keeps almost constant; anyway, the fact that they slightly oscillate depend on variations of the robot disturbances. This effect is clear in the roll input torque; in fact, it reaches big values during the manipulation interval, due to the fact that here a big roll disturbance torque arises, as previously presented.

The number of peaks and oscillations can be reduced by increasing the number of waypoints to be met or, in any case, by generating suitable reference trajectories.

Notice that the values of the control inputs are normalized, thus the motors constantly work around the 79% of their full capabilities in generating the vertical thrust.

The last values to be analysed, concerning the hexacopter control performances, are the tracking of the desired velocities, shown in Figure 35 and 36.

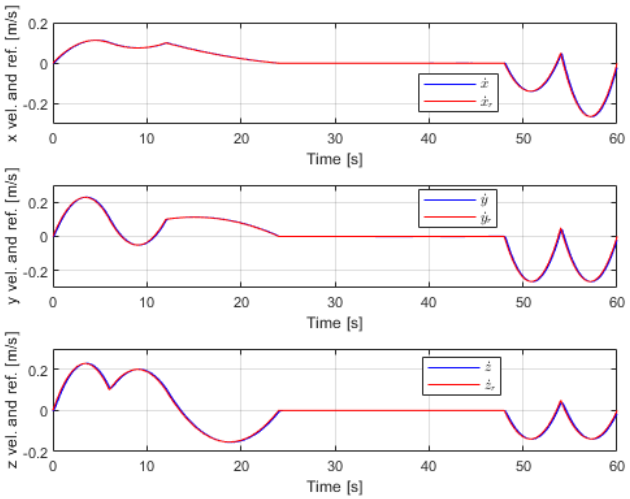

Figure 36: Linear velocity values and references of the hexacopter

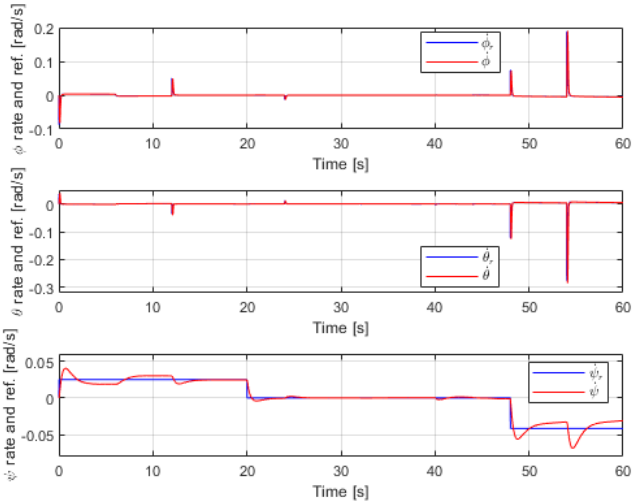

Figure 35: Angular velocity values and references of the hexacopter

Even if some errors are present due to the interaction with the robotic arm, it is possible to affirm that also here the velocities are tracked in a sufficiently precise way. In fact, it is possible to observe from Figure 37 that the velocity boundary conditions are met. This graph represent a detail of the linear velocities in the first part of the simulation, where the drone moves towards the goal point. The specified velocity of $0.1\ m/s$ is reached at the correct time instants, 6 and 12 seconds.

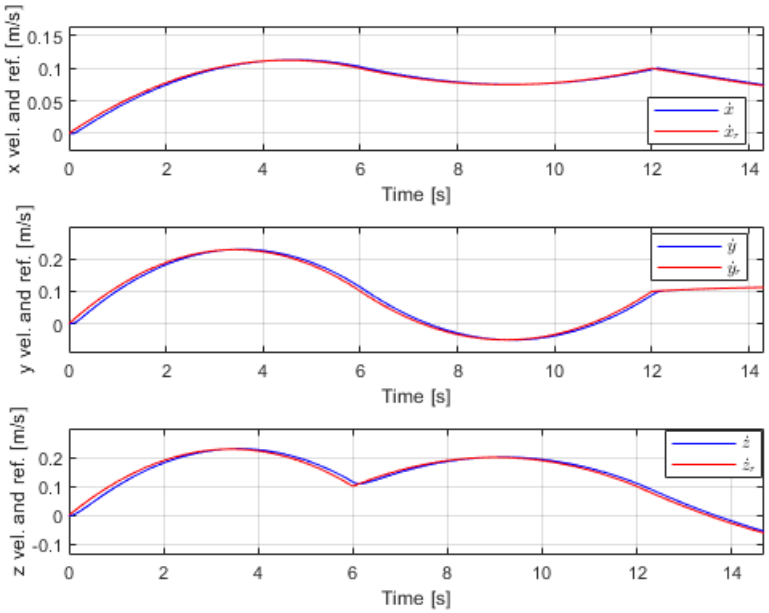

Figure 37: Detail of angular velocity values and references of the hexacopter

Concerning the performances of the manipulator control scheme, it is possible to analyse them starting from the comparison between measured positions and desired values of the trajectory, exposed in Figure 38.

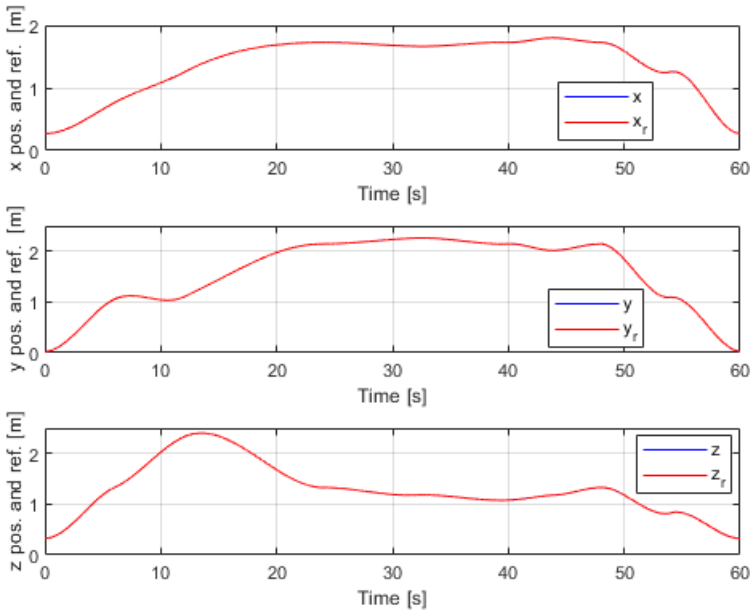

Figure 38: Position values and references of the manipulator

As said, these values represent the total inertial trajectory followed by the end effector of the manipulator. The magnitude of the errors is not visible from this figure, thus the following graph represents the errors of the tool centre point position in the inertial reference frame:

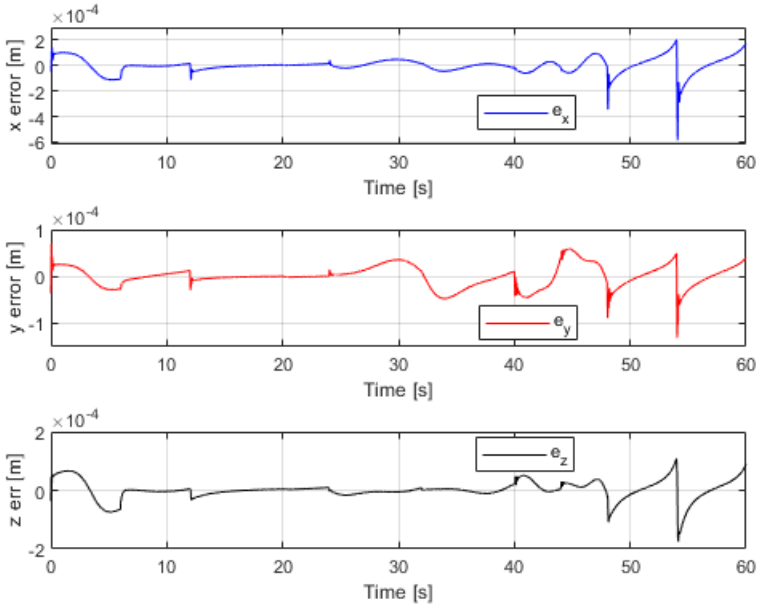

Figure 39: Errors of the end effector position in 3D space

The order of magnitude of the errors is small if compared to the one of the *x, y* and *z* positions; therefore, the manipulator scheme can successfully manage the tracking of reference values.

The major source of errors, as already said, is represented by the change of attitude of the hexacopter and it is visible at the related time instants.

They can be sensibly reduced by specifying a greater number of waypoints for the desired trajectory.

The joint variables associated to the presented motion, are shown in Figure 40.

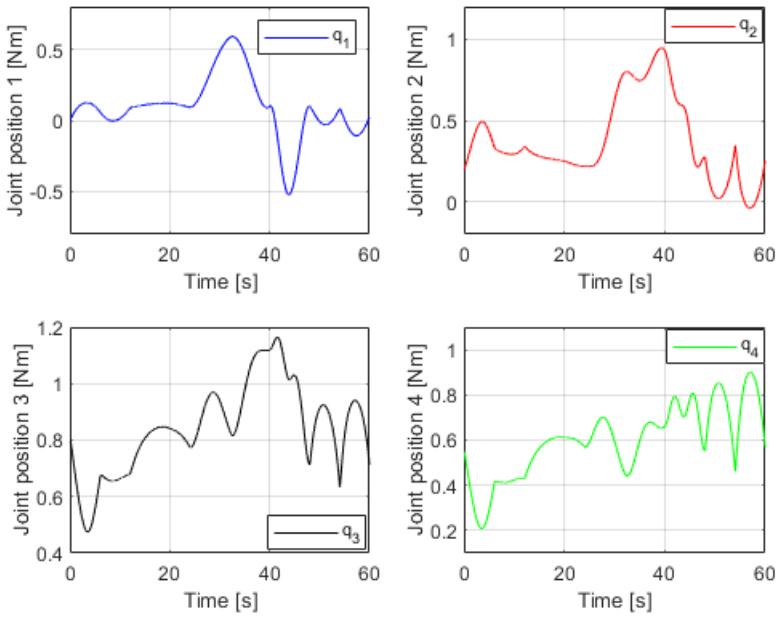

Figure 40: Joint variables

The evolution in time of the joint positions are important to show the capabilities of the system; in fact, from the graph is clear that the robot carries out a certain amount of movements to compensate errors in the drone position. These values of generalized coordinates is only a portion of the total possible movement, thus they can be executed from the motors.

The only aspect to be underlined is to take care that this motion does not make the robot collide with the hexacopter frame; in regard, some constraints in the robot task space can be applied.

Concerning the motion of the joints, during the manipulation time interval, they are consistent with the three-dimensional movement planned for the manipulator.

The next figures show the torque commands computed by the controller.

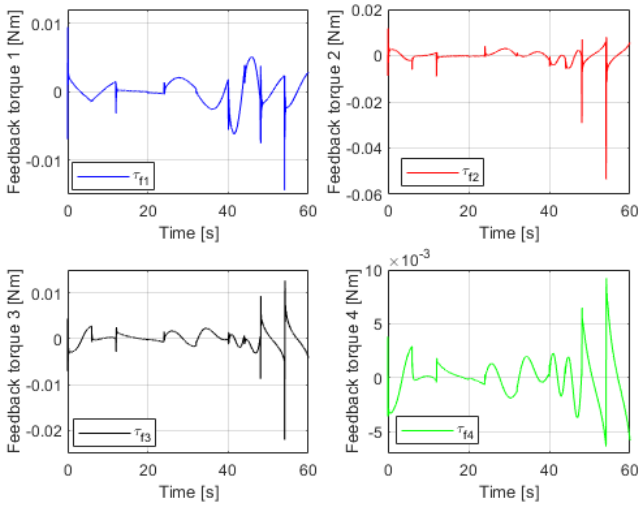

Figure 42: Joints feedback component of the control action

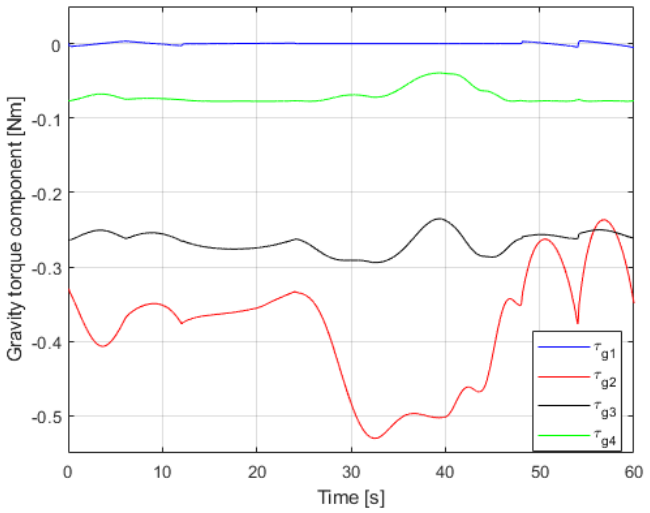

Figure 41: Joints gravity torque components of the control action

The torque input signals to the plant model are represented in Figure 41 and 42. Here, the control input has been divided into its components. On the left, it is possible to observe the feedback torque, computed by the *PID* controller. As already said, the biggest peaks for these values are encountered in drone's changes of attitude. On the right, the torque designated for gravity compensation is coherent with the robot motion; in fact, the biggest variations happen during the manipulation phase, where the manipulator totally changes its configuration.

All the elements discussed in this section allow to conclude that the overall performances are enough accurate with respect to the proposed goals.

The complete trajectory, actually followed by the hexacopter, is visualized through the UAV animation block, as presented in Figure 43:

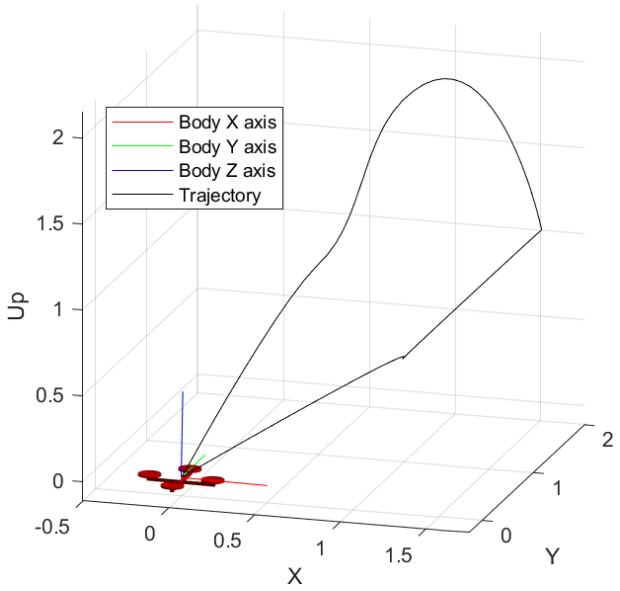

Figure 43: Trajectory followed by the hexacopter during the simulation

Chapter 5. Conclusion and future work

The proposed work shows the realization of both a model and related control architecture of a hexacopter equipped with a four degree-of-freedom manipulator. The research is focused on the position control of the tool centre point of the robotic arm considering coupling effects between this system and the hexacopter, without giving particular attention to the interaction with the environment.

The presented implementation exploits an independent approach and it aims to obtain an efficient and robust structure, which can be adapted to various scenarios. In fact, the model of the drone, as well as its controller, is realized independently from the manipulator's ones. This allows realizing a flexible scheme, which can be modified and improved in any of its constitutive parts, with respect to the proposed objectives.

In particular, the workflow starts with the realization of the model and controller of the hexacopter. Chapter 2 shows how the drone is modelled through nonlinear dynamical equations that describe the behaviour of the system; these equations are successively simplified in order to obtain an efficient and reliable control architecture. The adopted control technique, analysed in chapter 3, is the sliding mode control and it is based on a hierarchical structure able to deal with the six degrees of freedom of the hexacopter. In fact, two controllers are designed, one responsible for managing its position and the other for its attitude. The sliding mode control technique is very useful when dealing with nonlinear systems, being able to reject several effects that disturb the system behaviour, thanks to its robustness properties.

Concerning the manipulator, a similar procedure is adopted; in fact, the related model is firstly realized, as seen in chapter 2, to be used for the realization of the control architecture, discussed in chapter 3. In particular, the modelling phase takes advantage of two useful toolboxes, provided by *MathWorks*, to realize two models of the robotic arm in a fast and efficient manner. The first model is created using the *Simscape Toolbox*, very powerful tool for modelling complex mechatronic systems, and it is used to simulate the evolution in time of the plant under the effect of the control action. A very convenient feature of the toolbox is the possibility to directly transform a *CAD* model of the arm into *Simscape* blocks in *Simulink* environment. This model is successively transformed again, using the *Robotics System Toolbox*, into a rigid body robot model. This description of the robot takes into account both kinematic and dynamical properties of the system and it is used for the design of the control architecture. This operation is achieved exploiting the toolbox capabilities, which provides useful functions and blocks to deal with robotic systems, simplifying the realization of the controller. This is an operational space controller with gravity compensation, composed by a

feedforward and a feedback part that, summed up, represent the overall control action on the manipulator. In particular, the feedback component is realized through a *PID* controller that acts on errors of the end effector position in space, while the feedforward part provides the torques needed to counter gravity effects, making the *PID* tuning independent on the position. This solution shows reasonable performances, considering the computational weight of the algorithm.

The independent architectures are tested and tuned independently, to be successively coupled, as described in section 2.3. Here, the main idea is to make the overall control logic work in the inertial system, to allow the manipulator to compensate position errors in the hexacopter position. Moreover, particular attention is given to the dynamic coupling of the systems, since the robotic arm is seen as a disturbance from the drone. This issue is addressed through the computation of Newton-Euler equilibrium equations, that give an estimation of the forces and torques transmitted from the manipulator to the hexacopter centre of mass. These disturbances are then rejected through a feedforward compensation technique.

The consecutive operation consists in retuning the independent control algorithms, in order to improve the performances of the overall system; in particular, the sliding mode position control has to generate smooth reference values for the reference pitch and roll angles, to avoid too fast changes in the gravity vector seen from the manipulator. In regard, the *PID* proportional gain is increased to obtain a more aggressive control action, able to face these variations.

Finally, the capabilities of the overall system are tested in simulation through *MATLAB* and *Simulink* environment. These aspects are discussed in chapter 4, where the simulation setup is firstly presented and the results are successively discussed. A general case of hovering and manipulation is considered, where the reference trajectories are generated using polynomial interpolation of desired waypoints.

The results show satisfying behaviour in tracking the reference trajectory for the end effector. In addition, the influence of the disturbance torques is visible from the graphs that represent the position and attitude of the drone. These effects are successfully countered, since the manipulator control can compensate eventual errors in the hexacopter's position, as observable from the order of magnitude of the end effector position errors in the three dimensional space.

On these bases, it is possible to affirm that the presented work represents an efficient control architecture, being the sliding mode and *PID* controller computationally lighter than many other commonly used algorithms. Moreover, it is important to remark again that the independent modelling approach represents a sufficiently accurate and versatile method, allowing the designer to customize the system with respect to the requirements of the considered application.

To conclude, the discussed application represents a solid base for practical implementations and future improvements.

In fact, the following development of the project are related to practical aspects, to test the system behaviour in a real case scenario as well. The inertial position and velocities of the drone can be measured respectively through camera sensors and inertial measurements units, integrated sensors that usually include three-axis accelerometers, gyroscopes and compasses. In case that these instruments are not available, an observer needs to be designed, in order to estimate the evolution in time of the states of the system. Many techniques are available and, in addition, some source of disturbances can be estimated through observers, improving the performances of the control architecture.

Other important effects and disturbances, which are neglected in the present work, such as contact modelling or ground and wall effect can be considered to improve accuracy and reliability of the overall system. In fact, the interaction of the manipulator with environment is not directly taken into account in this thesis work. To address this issue, some specific force/torque sensors have to be mounted in the end effector of the manipulator, estimating the interactions exchanged with objects in space. Many options are available for this family of sensors, such as strain gauges and capacitive or passive compliant components; anyway, this choice strongly depends on the requirements of the considered task and on economic considerations.

Lastly, some experimental test can be performed in order to obtain a more accurate estimation of the dynamical properties of the hexacopter and manipulator systems; in fact in this work, these data are extracted from CAD models of the systems that give only an approximation of such parameters.

Bibliography

[1] Khalifa, A., Fanni, M., Ramadan, A., & Abo-Ismail, A. (2012, December). Modeling and control of a new quadrotor manipulation system. In *2012 First International Conference on Innovative Engineering Systems* (pp. 109-114). IEEE.

[2] Wuthier, D., Kominiak, D., Kanellakis, C., Andrikopoulos, G., Fumagalli, M., Schipper, G., & Nikolakopoulos, G. (2016, June). On the design, modeling and control of a novel compact aerial manipulator. In *2016 24th Mediterranean Conference on Control and Automation (MED)* (pp. 665-670). IEEE.

[3] Dube, C., & Pedro, J. O. (2018, May). Modelling and Closed-Loop System Identification of a Quadrotor-Based Aerial Manipulator. In *Journal of Physics: Conference Series* (Vol. 1016, No. 1, p. 012007). IOP Publishing.

[4] Kim, S., Choi, S., & Kim, H. J. (2013, November). Aerial manipulation using a quadrotor with a two dof robotic arm. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 4990-4995). IEEE.

[5] Guayasamín, A., Leica, P., Herrera, M., & Camacho, O. (2018, November). Trajectory Tracking Control for Aerial Manipulator Based on Lyapunov and Sliding Mode Control. In *2018 International Conference on Information Systems and Computer Science (INCISCOS)* (pp. 36-41). IEEE.

[6] Orsag, M., Korpela, C., Bogdan, S., & Oh, P. (2013, May). Lyapunov based model reference adaptive control for aerial manipulation. In *2013 International Conference on Unmanned Aircraft Systems (ICUAS)* (pp. 966-973). IEEE.

[7] Car, M., Ivanovic, A., Orsag, M., & Bogdan, S. (2018, October). Impedance based force control for aerial robot peg-in-hole insertion tasks. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 6734-6739). IEEE.

[8] Nayak, V., Papachristos, C., & Alexis, K. (2018). Design and control of an aerial manipulator for contact-based inspection. *arXiv preprint arXiv:1804.03756*.

[9] Kondak, K., Krieger, K., Albu-Schaeffer, A., Schwarzbach, M., Laiacker, M., Maza, I., ... & Ollero, A. (2013). Closed-loop behavior of an autonomous helicopter equipped with a robotic arm for aerial manipulation tasks. *International Journal of Advanced Robotic Systems*, *10*(2), 145.

[10] Kannan, S., Olivares-Mendez, M. A., & Voos, H. (2013). Modeling and control of aerial manipulation vehicle with visual sensor. *IFAC Proceedings Volumes*, *46*(30), 303-309.

[11] Xu, R., & Ozguner, U. (2006, December). Sliding mode control of a quadrotor helicopter. In *Proceedings of the 45th IEEE Conference on Decision and Control* (pp. 4957-4962). IEEE.

[12] Sumantri, B., Uchiyama, N., & Sano, S. (2014, October). Second order sliding mode control for a quad-rotor helicopter with a nonlinear sliding surface. In *2014 IEEE Conference on Control Applications (CCA)* (pp. 742-746). IEEE.

[13] Ahmed, N., & Chen, M. (2018). Sliding mode control for quadrotor with disturbance observer. *Advances in Mechanical Engineering*, *10*(7), 1687814018782330.

[14] Xilun, D. I. N. G., Pin, G. U. O., Kun, X. U., & Yushu, Y. U. (2019). A review of aerial manipulation of small-scale rotorcraft unmanned robotic systems. *Chinese Journal of Aeronautics*, *32*(1), 200-214.

[15] Gibiansky, A. (2012). Quadcopter dynamics, simulation, and control. *Andrew. gibiansky. com*.

[16] Mahony, R., Kumar, V., & Corke, P. (2012). Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *IEEE Robotics and Automation magazine*, *19*(3), 20-32.

[17] Beard, R. W. (2008). Quadrotor dynamics and control. *Brigham Young University*, *19*(3), 46-56.

[18] Caldera, F. (2019). *Sliding Mode Attitude Control for Small Satellites with Manipulator* (Master Thesis, Politecnico di Torino).

[19] Tavora, B. F. (2017). *Feasibility study of an aerial manipulator interacting with a vertical wall* (PhD Dissertation, Naval Postgraduate School Monterey United States).

[20] d'Armi, P. (2016). A quick introduction to sliding mode control and its applications. *Universita'Degli Studi Di Cagliari*, 1-22.

[21] Song, P. (1998). *Robotic manipulator control, fundamentals of task space design* (Doctoral dissertation, National Library of Canada= Bibliothèque nationale du Canada).

[22] A. Rizzo, 2018, Robotics lecture notes, *Dynamics*, slides 5-11, Politecnico di Torino, Torino, Italy.

[23] C. Novara, 2019, *Nonlinear control and aerospace applications* lecture notes, *Sliding mode control*, slides 10-18, Politecnico di Torino, Torino, Italy.

[24] A. Rizzo, 2018, *Robotics* lecture notes, *Kinematic functions*, slides 8-26, Politecnico di Torino, Torino, Italy.

[25] A. Rizzo, 2018, *Robotics* lecture notes, *Control 1*, slides 3-7, Politecnico di Torino, Torino, Italy.

[26] A. Rizzo, 2018, *Robotics* lecture notes, *Control 4*, slides 3-14, Politecnico di Torino, Torino, Italy.