Politecnico di Milano
Scuola di Ingegneria Industriale e dell'Informazione
Corso di Laurea Magistrale in Automation and Control Engineering

Nonlinear attitude and position control for a quadrotor UAV

Advisor: Prof. Marco LOVERA
Co-Advisor: Mattia GIURATO, Eng.
Davide INVERNIZZI, Eng.

Thesis by:
Davide Paolo CARELLI Matr. 837861

Academic Year 2016–2017

Alla mia Famiglia ...

Acknowledgments

Ringraziare alla conclusione di un percorso vuol dire per me ringraziare tutti quelli che ho incontrato lungo la strada e grazie ai quali sono riuscito a raggiungere le mete che mi ero prefissato superando gli ostacoli lungo il cammino.
Per questo ringrazio in primo luogo i miei genitori, che mi hanno sempre aiutato e sostenuto durante questo cammino di crescita che è stato la mia carriera universitaria, da quando sono uscito di casa la prima volta diretto alla prima lezione di Analisi 1 fino alla discussione di questa tesi. Insieme a loro non posso dimenticare le mie sorelle, mia zia e i miei nipoti, che durante questo lungo percorso mi hanno sempre stimolato e aiutato a credere in me stesso.
Un ringraziamento speciale lo devo poi ai miei compagni di avventura, che ho conosciuto appena salito a bordo e non ho più lasciato fino ad oggi. Un grazie enorme quindi a Paolo, Fabrizio, Marco, Dario, Jacopo, Emanuele e Daniele, con i quali ho condiviso le lezioni, le ore di studio, gli esami e moltissimi momenti felici. E' anche grazie a voi se oggi sono qui.
Non posso poi dimenticare i miei amici di sempre e alla mia ragazza, che pur non condividendo le mie stesse esperienze mi sono sempre stati vicino e hanno sopportato i miei discorsi da ingegnere, grazie quindi a Martina, Fabio, Ale e Simone.
Arrivando a questo ultimo anno passato nel laboratorio FLYART, non posso fare a meno di ringraziare il professor Marco Lovera per l'opportunità che mi ha concesso di sperimentarmi in questa avventura che mi ha appassionato fin dall'inizio.
Un ringraziamento speciale va poi a Mattia e Davide, che mi hanno quotidianamente supportato nello sviluppo di questo lavoro, e a Simone, il maestro dell'H_∞.
Ringrazio inoltre Pietro, Aureliano e Matteo per il supporto morale indispensabile in alcuni momenti.

Abstract

Nowadays, quadrotor UAV platforms have began to spread around and starts to catch the attention of the industry for their applications in various tasks like the monitoring of impervious or dangerous areas and the transportations of payloads. All these missions can require the UAV to cope with external disturbances from the environment, the wind for example, or to sustain aggressive manoeuvrers with large variations of attitude angles.
The linear control applied to quadrotor platforms has been deeply explored and many successful applications have been developed in the recent years. This kind of control based on standard and well known techniques provides drones that can work fine in a certain range of angles around the hovering condition but it obtains poor performances in the case of aggressive flight.
The purpose of this thesis is the development of a nonlinear control architecture for the quadrotor UAV platform of the FLYART laboratory. During the development of this work many control strategies have been developed and analysed in order to obtain a controlled system able to cope with difficult working conditions. The tuning of these control laws has been handled applying the H_∞ tuning technique to the developed control system. The starting point is the backstepping method, analysed at the beginning of the third chapter after a brief introduction to the quadcopters in the first chapter and the presentation of the quadrotor model in the second one. Then the integral backstepping technique has been implemented in two different ways in order to gain robustness for the system and compensate possible disturbances and uncertainties of the real system. After these control laws, it has been tackled the problem of singularities carried by the attitude angle parametrization with the proposal of a geometric control architecture that allows to avoid this inconvenient.
All the control laws are mathematically developed in this thesis work and the resulting simulations are shown in the last chapter.

Sommario

Al giorno d'oggi i quadricotteri UAV hanno iniziato a diffondersi e ad attirare anche l'attenzione delle industrie, principalmente per quanto riguarda la loro applicazione in alcuni contesti specifici quali il monitoraggio di aree impervie o pericolose oppure per il trasporto di carichi. Questo genere di missioni possono richiedere che l'UAV riesca a gestire disturbi provenienti dall'ambiente in cui opera, primo fra tutti il vento, e offra la capacità di manovre complesse che prevedano anche ampi angoli di inclinazione.
Il controllo lineare applicato a questo tipo di piattaforme è stato studiato in modo approfondito negli ultimi anni con lo sviluppo di molte applicazioni che hanno dato risultati più che soddisfacenti. Questo tipo di controlli basati su tecniche standard e ben note nell'ambito della teoria del controllo soffrono però di pesanti limitazioni in caso di volo lontano dalla condizione di hovering, che causano un veloce degrado delle performance.
Questa tesi si pone l'obiettivo di sviluppare dunque un'architettura di controllo non lineare per il quadricottero UAV del laboratorio FLYART. Durante lo sviluppo di questo lavoro sono state sviluppate e analizzate varie strategie di controllo in modo da ottenere un sistema in grado di affrontare condizioni di operatività non ottimali. Parallelamente alla formulazione di queste leggi di controllo è stata studiata l'applicazione della metodologia di tuning basata sulla minimizzazione della norma H_∞ per la scelta dei parametri del controllore.
Lo sviluppo di queste architetture si basa sul metodo di controllo backstepping, introdotto nel terzo capitolo dopo una breve introduzione ai quadricotteri nel primo capitolo e la presentazione del modello matematico utilizzato. Successivamente è stata implementata un'architettura di tipo integral backstepping in due differenti configurazioni in modo da migliorare la robustezza del controllo e rendere possibile la compensazione di disturbi costanti presenti nel sistema reale. Dopo queste leggi i controllo è stato affrontato il problema di come gestire le singolarità intrinseche della parametrizzazione dell'assetto con angoli di Eulero e come soluzione è stata implementata una legge di controllo geometrica.
Nel seguente lavoro di tesi sono stati quindi mostrati gli sviluppi matematici dei controlli e i risultati ottenuti in simulazione vengono presentati nell'ultimo capitolo.

Contents

List of Figures

List of Tables

Introduction

A quadrotor UAV is an aerial vehicle whose motion is controlled regulating the thrust delivered by each propeller.
Their trajectory is controlled modifying the attitude of the quadrotor itself in order to point the thrust to the desired direction. Due to the fast attitude dynamics, an on-board computer (Flight Control Unit, FCU) is needed to compute the control action based on the informations provided by an Inertial Measurement Unit (IMU) and in the case of the FLYART laboratory also provided by an Optitrack measurement system.
The control action, that is the angular rate of each propeller, has been computed by means of a linear control law in the quadrotors of the laboratory until now. This kind of controllers are designed starting from a linearised model of the UAV system that neglects the gyroscopic effects and the coupling between the attitude angles dynamics. Due to this assumptions on the model the resulting linear controllers have a limited range of angles, typically around 20°, where they can properly control the attitude dynamics. In case of larger angles, the performance worsens rapidly and the system can reach an unstable behaviour.
In the industrial field PID controllers have been preferred to nonlinear architectures because of their intuitive behaviour and the presence of standard tuning techniques, however it is the industry itself to require more complex control laws that can be able to cope with difficult flight conditions and aggressive manoeuvrers.
In this frame the analysis of four different nonlinear control approaches has been studied with the aim of allowing the FLYART UAV platform to undertake complex trajectories and reject disturbances. The first control law to be designed and implemented in the Simulink simulation environment has been the backstepping control law, this kind of controller is based on Lyapunov theory and guarantees exponential asymptotic stability of the error dynamics. However, the resulting controller performances are easily downgraded by constant disturbances and model uncertainties due to the lack of any sort of integral action.
A straightforward solution for this problem is the reformulation of the control law including an integral action. This control structure, called integral backstepping, improves the properties ensured by the pure backstepping increasing the disturbance rejection capabilities of the controller.

The third control approach that has been studied is a refinement of the integral backstepping architecture that takes into account the coupling between the axes of rotation of the quadrotor. Even if this kind of controller is affected by the presence of the gimbal lock singularity, this formulation let a design of the controller more adherent to the actual UAV dynamics.
The last control approach that has been tackled is the geometric control, this kind of nonlinear control avoids the Euler angle parametrization of the attitude representation and thus it manages to handle nearly any configuration of the quadrotor attitude at the cost of a more complex mathematical development.
One of the main problem in the design and implementation of nonlinear controllers has always been the tuning of the control parameters, since the effect of the variation of one parameter usually have effects on the behaviour of the system that are not always easy to identify. Thus in parallel with the development of the presented control laws, an application of the H_∞ tuning technique to nonlinear system has been studied. The developed approach manages to find working values for the first two controllers, while it does not manage to handle the nonlinearities introduced in the last two.
The thesis is structured as the development of the analysis as been carried on and presented here:

- In the first chapter, a brief introduction to the multirotor UAVs is presented.
- The second chapter presents in detail the model conventions and structure highlighting the assumptions made.
- The third chapter is the core of the thesis work and presents the mathematical development of the control laws, the tuning procedures and the achieved results.
- The fourth and last chapter offers a complete overview on the simulation results achieved by each controllers and compares the results obtained.
- In the last part some conclusions are derived from the presented analysis and a proposal for further developments is explained.

Chapter 1

Brief history of quadrotor UAVs

1.1 The birth of UAVs

In 1849 Austria bombarded Venice during the First Independence War (1848-1849) using balloons equipped with timed mechanisms able to release bombs. This has been the first usage in history of an Unmanned Aerial Vehicle and actually resolves in a partial success since the change of the wind makes some balloons bombing the Austrian lines themselves.
The development of the unmanned flight as it is nowadays known started in the First World War with the creation of aerial torpedoes by Hewitt-Sperry Automatic Airplane. These flying bombs used the gyroscopic stabilizer that had been just invented reaching an accuracy of 3 $[Km]$ around its target.
It was during the World War II that many UAVs were developed to provide targets for military personnel training and also for attack missions.
In the '50s the Ryan Firebee was developed, equipped with a jet engine, this is not the first jet engine drone but one of the most used. From 1959 the U.S government starts developing UAVs in order to avoid the loss of pilots and during the War of Attrition (1967-1970) the first drones with reconnaissance cameras went on duty. During the Vietnam War the U.S. airforce extensively use them for various missions.
Another step towards the present state of the art was done by Israeli government in 1973 with the development of a UAV endowed with real-time surveillance capability. Starting from that point on, with the continuous improvement of sensors, communication technologies and computing capabilities the drones continuously developed till these days.

1.2 The development of multicopters

Four years after the first flight of the Wright brothers (1903), the Frenchman Louis Breguet, who also co-founded Air France, developed together with his brother Jacques and the professor Charles Richet (Nobel Prize for Medicine) the Gyroplane.
That was the first quadcopter made of a steel-tubes frame and weights half a ton. The propulsion was ensured by a single 46 $[CV]$ Antoinette internal combustion engine that provides torque for all the four biplanar rotors with four blades each by means of a complex system of pulleys and belts. This futuristic aerial vehicle, like the today's quadcopters, was designed to compensates the countermoments generated by the rotations of the propellers making them rotating clockwise the ones on one diagonal, counter-clockwise the ones on the other.
Although the power provided by the engine was decent for that times, the Gyroplane cannot flight at more than 60 $[cm]$ from the ground. Thus this was only a first attempt in the exploration of the quadcopter field, but succeeded in proving the possibility of flight of a rotary wing aircraft.
The Gyroplane works as a basis for the development of the Gyroplane 2, endowed with a 55 $[CV]$ ICE Renault engine, with only two smaller rotors and a fixed wing. The Gyroplane 2 was thus more a combination between an airplane and a helicopter, with a vertical stabilizer and elevators. This second prototype was heavily damaged during a landing and finally wrecked by a storm in 1909.
Since that first attempt, eleven years and the First World War passed before in 1920 the French engineer Ethienne Oehmichen developed six multicopter prototypes. Among those quadcopters, the second one, the Oehmichen 2 had the features similar to the Gyroplane, being made of steel-tubes with counterrotating rotors. Thanks to the improvements on the engines of those years, the Oehmichen 2 did not need a bi-planar rotary wing and only two blades for rotor were needed. From the control point of view this multirotor was extremely inefficient, in fact while the Gyroplane was controlled varying the angular rates of the couples of propellers, the Oehmichen 2 mounted eight engines: two that powers the rotors, and six connected to other six propellers that moved the vehicle forward, backward, to the left, to the right and made it turn. Due to this extreme inefficiency the project was aborted even if it proves great stability and reliability: it was the first helicopter to run 360 $[m]$ in 1923, it completes an entire circular trajectory and some years later managed to fly for a kilometre.
Another multirotor, the de Bothezat helicopter, challenges the Oehmichen 2 in 1922. It takes its name from the Russian-American scientist De Bothezat and was renamed by the press as "The Flying Octopus ". This was a four engine with six-bladed variable pitch propellers with collective pitch control, other two three-bladed propellers were used in order to cool down the 180 $[CV]$ engine. Although the project was highly financed and uses the latest technologies at that time, it results in an underpowered, imprecise and unreliable vehicles. Due to the

impossibility for a pilot to handle the controls manually, since there were no FCU at that times, the de Bothezat helicopter cannot undertake a complex trajectory and all its more than one-hundred has been done in hovering.
The Second World War and the invention of helicopters completely stopped the development of multirotor platforms untill the '50s, when the Convertawings Model A manages to complete a flight, controlling its attitude varying the rotors angular rates. After 1958, when the quadrotor Curtiss Wright VZ-7 was developed but did not met U.S. military specifications, no more quadcopter prototypes had been created until the new millennium.

1.3 Quadrotor UAVs

The continuous reduction of power consumptions and space of the electronic components and the development of open source hardware resources like Arduino in the early '00s years paved the way for a rebirth of the quadrotor concept in the UAV field.
In fact, inheriting from aircraft modelling the engines, the propellers and reliable and precise regulators, together with the newly developed open source electronics, it has been possible to produce at an amateurish level small remotely controlled quadrotors and multirotors. From this niche field reserved to modelling enthusiasts, the integration of a camera that let the user to have an on board view from the quadcopter on his or her smartphone opens the quadrotor UAV world to the large consumer market. This diffusion of reliable, small scale quadrotors promote the development of brand new applications in various fields, both civil and military.
The possibility of a quadrotor UAV to hover can be exploited for surveillance purposes and in domestic application and in military one, moreover the possibility to endow medium size quadrotors with almost any kind of sensor make them very usefull for the so called DDD applications (Dull-Dirty-Dangerous). A UAV quadcopter can fly over steepy slopes of mountains in order to monitor a landslide, with the operator controlling it from a safe position. Also in agriculture they can be used to monitor the cultivations allowing also tailored treatment for each portion of a cultivated field. In industry there are also some ideas to adopt quadcopter drones for load handling and in these years great attention has been given to the choice of Amazon to deliver goods using UAVs.
It is clear that this research area is for all these reasons of great interest both on a control science point of view and on an industrial one. The industry calls for more and more performing controls able to cope with every day harder and more specific tasks, that require more complex control laws than the one developed in the last years.

Chapter 2

Dynamic model for the quadrotor

In the following chapter the formulation of the dynamic UAV model is described. The first issue to address is how the position and attitude of the quadrotor have been represented throughout this work, showing its derivation and highlighting the possible problems related to the adopted parametrization.
The second part faces with the formulation of the dynamic model used in designing the controllers, focusing on the considered effects and approximations.

2.1 Reference systems

2.1.1 Inertial reference system

With inertial reference system or Earth axes, it is named a coordinate system of three fixed axes that works as a fixed base for the representation of the position of a considered body or moving reference system.
The origin of the system could be set everywhere, the intersection of the equator with the prime meridian and the mean sea level for example, it is completely arbitrary but once chosen it cannot vary. The displacement along one or more set of orthogonal axes attached to the origin describes the position of anything in this reference system. It is convenient to align the axes of the reference frame with the compass, one is aligned with the axis labelled North, one with the axis labelled East and the last with the normal to the surface generated by the previous two, pointing to the centre of the Earth and labelled Down. These three axes are mutually perpendicular by construction and when referring to them in the order N,E,D form a right-handed coordinate system.

2.1.2 Body axes

For the representation of the attitude of the UAV it is convenient to set the origin of the body frame coinciding with the centre of gravity of the quadrotor. The

chosen set of axes is a right-handed system (Figure 2.1), the X axis lies in the plane of symmetry and generally points forward, the Y axis points to the right normally to the plane of symmetry and the Z axis points down.

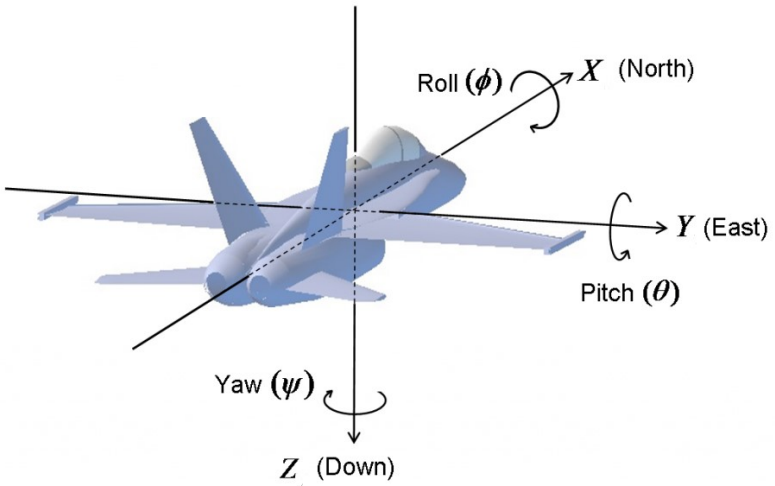


Figure 2.1: Body axes

2.2 Attitude representation

One of the main problem in modelling and controlling a 3D system has always been the correct representation of the attitude of an object with respect to an inertial frame and a body fixed frame. This is actually a key issue in this work since the errors and so the control variables are computed based on it and the singularities of the different representations strongly influence the stability and robustness properties of the controlled system.

2.2.1 Euler angles

Vectors can be rotated about any axes in any order for any number of times until the final orientation is achieved. These subsequent rotations can be represented by means of a rotation matrix from the initial orientation to the final one. Adopting the to-from notation, a rotation matrix from system E to system D would be named R_{D-E}. Thus any vector P_E in the E reference system can be resolved to system D, in the corresponding vector P_D through the matrix operation:

$$P_D = R_{D-E} P_E. \tag{2.1}$$

Rotation matrices are written for $\mathbb{R}^3$ vectors and may represent the rotation around one, two or three axes. For sake of a better comprehension it is useful to analyse the rotation around a single axis and then composing different rotations in order to achieve the final rotation matrix.
A rotation about the X axis does not change the component of the vector directed along the X axis itself, but changes the Y and Z components. All these considerations can be noticed analysing the matrix associated to this rotation of an angle Φ:

$$R_X(\Phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\Phi & \sin\Phi \\ 0 & -\sin\Phi & \cos\Phi \end{bmatrix} \tag{2.2}$$

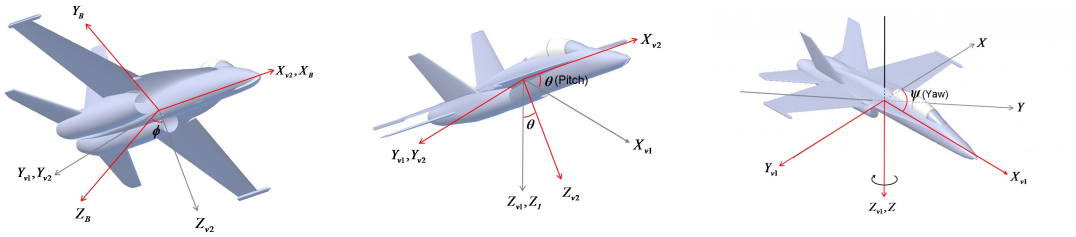
same considerations can be done for the rotation matrices representing respectively rotations of an angle Θ around Y and Ψ around Z axes.
These matrices have the form

$$R_Y(\Theta) = \begin{bmatrix} \cos\Theta & 0 & -\sin\Theta \\ 0 & 1 & 0 \\ \sin\Theta & 0 & \cos\Theta \end{bmatrix} \tag{2.3}$$

$$R_Z(\Psi) = \begin{bmatrix} \cos\Psi & \sin\Psi & 0 \\ -\sin\Psi & \cos\Psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

The positive direction of the angular displacement is the one your fingers curl when aligning your thumb with the positive direction of the axis. The 0 degrees position can be assigned arbitrarily but once fixed, it must not vary.
The three presented matrices are orthonormal since each of their columns represent a vector of unit magnitude and the scalar product of column i with column j with $i \neq j$ equals zero, that is the definition of orthogonal. Orthonormality carries also the useful property that the inverse of an orthonormal matrix is its transpose, thus in order to compute the inverse rotation matrix only a transposition is needed.
It is possible to build the overall matrix representing subsequent rotations around different axes multiplying in the same sequence the rotation matrices of each rotation around each axis. Moreover, every cascade of rotations can be reduced to a rotation about three axes only. One possible way in representing these three subsequent rotations is the Euler angle conventions, whose naming convention follows NASA standard notation and consists in a first rotation of an angle Ψ around the Z axis called yaw, then a rotation of an angle Θ around the new intermediate Y axis, the so called pitch, and finally a rotation of an angle Φ around the newer X axis and also known as roll. The three angles are shown in Figures 2.2a, 2.2b, 2.2c
This particular set of rotations is performed by the so called Euler rotation matrix defined as:

$$T_{BE}(\Phi, \Theta, \Psi) = R_X(\Phi) R_Y(\Theta) R_Z(\Psi) \tag{2.5}$$



(a) Roll angle (b) Pitch angle (c) Yaw angle

Figure 2.2: Euler angles

where subscript B stands for "body" and E stands for " Earth", respectively. Matrix T_{BE} resolves an Earth-based vector to body reference system. The overall formulation of the T_{BE} matrix is then:

$$T_{BE}(\Phi,\Theta,\Psi) = \begin{bmatrix} C_\Theta C_\Psi & C_\Theta S_\Psi & -S_\Theta \\ S_\Phi S_\Theta C_\Psi - C_\Phi S_\Psi & S_\Phi S_\Theta S_\Psi + C_\Phi C_\Psi & S_\Phi C_\Theta \\ C_\Phi S_\Theta C_\Psi + S_\Phi S_\Psi & C_\Phi S_\Theta S_\Psi - S_\Phi C_\Psi & C_\Phi C_\Theta \end{bmatrix} \tag{2.6}$$

where the shorthand notation $C_\Phi = \cos\Phi$, $S_\Phi = \sin\Phi$, $T_\Phi = \tan\Phi$ has been adopted.
The Euler angles are used to rotate a velocity or acceleration vector expressed in body frame to the Earth reference system, that is

$$\begin{aligned} P_e &= \begin{bmatrix} N \\ E \\ D \end{bmatrix}, V_e = \begin{bmatrix} \dot{N} \\ \dot{E} \\ \dot{D} \end{bmatrix} = \dot{P}_e \\ V_b &= T_{BE}(\Phi,\Theta,\Psi)V_e = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \end{aligned} \tag{2.7}$$

where P_e is the position of the UAV centre of gravity in the inertial (Earth) frame (NED), V_e is its velocity and V_b is the same velocity resolved to body axes.
Starting from these considerations the vector of angular position of the UAV body axes with respect to the Earth reference system can be defined as

$$\alpha_e = \begin{bmatrix} \Phi \\ \Theta \\ \Psi \end{bmatrix} \tag{2.8}$$

named respectively roll angle, pitch angle and yaw angle.
Of course, the three angles just defined vary with time during a maneuver and so the Euler rates are function of both the Euler angles and the body-axis angular rates.

Firstly the Euler rates are defined as:

$$\omega_e = \begin{bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{bmatrix} = \dot{\alpha}_e \tag{2.9}$$

and the body-axis rates as

$$\omega_b = \begin{bmatrix} p \\ q \\ r \end{bmatrix}. \tag{2.10}$$

To get the relation from Earth-axis rates to body-axis ones each Euler rate has to be considered individually, resolved to intermediate axes and only at the end resolved to body axes. Define the Euler rate elemental vectors as:

$$\omega_{\dot{\Phi}} = \begin{bmatrix} \dot{\Phi} \\ 0 \\ 0 \end{bmatrix}, \omega_{\dot{\Theta}} = \begin{bmatrix} 0 \\ \dot{\Theta} \\ 0 \end{bmatrix}, \omega_{\dot{\Psi}} = \begin{bmatrix} 0 \\ 0 \\ \dot{\Psi} \end{bmatrix} \tag{2.11}$$

Rotate $\omega_{\dot{\Psi}}$ through the angle Θ about the Y axis, and add the result to the $\omega_{\dot{\Theta}}$ vector. Rotate that sum about the X axis through the angle Ψ and add the result to the $\omega_{\dot{\Phi}}$ vector. This last resulting vector is the body-axis angular rates vector.

$$\omega_b = \omega_{\dot{\Phi}} + R_X(\Phi)\left[\omega_{\dot{\Theta}} + R_Y(\Theta)\omega_{\dot{\Psi}}\right] \tag{2.12}$$

Rearranging the terms we get to

$$\omega_b = E(\Phi, \Theta)\omega_e = \begin{bmatrix} 1 & 0 & -S_\Theta \\ 0 & C_\Phi & S_\Phi C_\Theta \\ 0 & -S_\Phi & C_\Phi C_\Theta \end{bmatrix} \omega_e. \tag{2.13}$$

On the contrary, to get the Earth-axis rate in terms of body-axis rates the inversion of the transformation matrix E is needed. Unlike the T_{BE} matrix, matrix E is not orthonormal and in addition its inverse present two singularities in case of a pitch angle of ± 90.

$$E(\Phi, \Theta)^{-1} = \begin{bmatrix} 1 & S_\Phi T_\Theta & C_\Phi T_\Theta \\ 0 & C_\Phi & -S_\Phi \\ 0 & S_\Phi / C_\Theta & C_\Phi / C_\Theta \end{bmatrix}. \tag{2.14}$$

This singularity is called gimbal lock and is one of the main difficulties when handling aggressive flight maneuvers with Euler angles parametrization.

2.3 The model

The model is developed under the assumption of the rigid body, that is all the points belonging to the body move keeping the distances and the angles fixed.

2.3.1 Velocities and accelerations

When considering the kinematic of a moving frame system following a generic path, it has to be considered that both the modulus and the direction of the velocity vector can change. A straightforward consequence of this fact is that, defining S the position vector of a point expressed in a reference system rotating with angular rate ω_b

$$S = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \omega_b = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.15}$$

and

$$\frac{\partial S}{\partial t} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{2.16}$$

the total velocity and acceleration seen in the Earth frame are:

$$\frac{dS}{dt} = \frac{\partial S}{\partial t} + \omega_b \times S \tag{2.17}$$

$$\frac{d^2S}{dt^2} = \frac{\partial}{\partial t}\left(\frac{dS}{dt}\right) + \omega_b \times \left(\frac{dS}{dt}\right) = \frac{\partial^2 S}{\partial t^2} + 2\omega_b \times \frac{\partial S}{\partial t} + \frac{\partial \omega_b}{\partial t} \times S + \omega_b \times (\omega_b \times S). \tag{2.18}$$

2.3.2 Linear motion

The description of the linear motion of a body can be obtained starting from the second Newton's Law. Linear momentum is the product of the mass times the velocity of an object. A force acting on the object makes the related momentum change:

$$F = \frac{d(mV_b)}{dt} \tag{2.19}$$

Applying to this expression (2.17) and (2.18) and adopting dot notation for the sake of readability:

$$F = \frac{dm}{dt}V_b + m\left(\frac{\partial V_b}{\partial t} + \omega_b \times V_b\right) = \dot{m}V_b + m\dot{V_b} + \omega_b \times (mV_b) \tag{2.20}$$

The obtained expression in (2.20) represents the rate of change of the linear momentum as a function of the applied forces. In the system under analysis those forces are gravity, aerodynamics and others that can be lumped into a category called externally applied loads. In the following the vector F_{ext} represents the external forces:

$$F_{ext} = \begin{bmatrix} F_X \\ F_Y \\ F_Z \end{bmatrix}. \tag{2.21}$$

Assuming a constant mass of the UAV the resulting equation of linear motion takes the form:

$$m\dot{V_b} + \omega_b \times (mV_b) = F_{ext} \tag{2.22}$$

2.3.3 Angular motion

Expanding and rearranging (2.18), the acceleration of a point can be expressed as:

$$\begin{cases} \ddot{x} &= \dot{u} + \dot{q}z - \dot{r}y + q(w + py - qx) - r(v + rx - pz), \\ \ddot{y} &= \dot{v} + \dot{r}x - \dot{p}z + r(u + qz - ry) - p(w + py - qx), \\ \ddot{z} &= \dot{w} + \dot{p}y - \dot{q}x + p(v + rx - pz) - q(u + qz - ry). \end{cases} \tag{2.23}$$

Considering the body mass divided into infinitesimal masses, the inertial force acting on the j-th differential mass element is:

$$\begin{cases} (dF_X)_j = dm_j\ddot{x} \\ (dF_Y)_j = dm_j\ddot{y} \\ (dF_Z)_j = dm_j\ddot{z}. \end{cases} \tag{2.24}$$

If the position of the differential mass element is S_j, the mass resists an angular force (a moment) about all three axes. The differential inertial moments are:

$$dM_j = S_j \times dF_j, \tag{2.25}$$

$$\begin{cases} (dM_X)_j = (dF_Z)_j y_j - (dF_Y)_j z_j \\ (dM_Y)_j = (dF_X)_j z_j - (dF_Z)_j x_j \\ (dM_Z)_j = (dF_Y)_j x_j - (dF_X)_j y_j \end{cases} \tag{2.26}$$

By definition, the origin of the coordinate system is the centre of gravity; therefore

$$\begin{aligned} \Sigma(x)dm = 0, \\ \Sigma(y)dm = 0, \\ \Sigma(z)dm = 0. \end{aligned} \tag{2.27}$$

Combining (2.28) , (2.26) , (2.27) the linear motion equation are obtained:

$$\begin{cases} m\ddot{x} &= m(\dot{u} + qw - rv), \\ m\ddot{y} &= m(\dot{v} + ru - pw), \\ m\ddot{z} &= m(\dot{w} + pv - qu). \end{cases} \tag{2.28}$$

Introducing the definition of inertia tensor in the form

$$I_n = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix} \tag{2.29}$$

where the various terms can be defined as

$$\int (xy)dm = I_{xy}, \int (xz)dm = I_{xz}, \int (yz)dm = I_{yz},$$
$$\int (y^2 + z^2)dm = I_{xx}, \int (x^2 + z^2)dm = I_{yy}, \int (x^2 + y^2)dm = I_{zz}.$$

Considering a body frame coincident with the symmetry axes of the aircraft body, one can assume I_n to be a diagonal matrix.
From all the previous considerations the expression of the angular motion can be derived in the form:

$$\begin{aligned} L &= I_{xx}\dot{p} + (I_{zz} - I_{yy})qr, \\ M &= I_{yy}\dot{q} + (I_{xx} - I_{zz})pr, \\ N &= I_{zz}\dot{r} + (I_{yy} - I_{xx})pq, \end{aligned} \tag{2.30}$$

where L, M, N are the moments applied on the body-axes X, Y, Z respectively. Collecting all the applied moments in a vector M_{ext}

$$M_{ext} = \begin{bmatrix} L \\ M \\ N \end{bmatrix}, \tag{2.31}$$

and recalling the definition of body-axis angular velocity vector the equation of angular motion can be reorganized in the form

$$I_n\dot{\omega}_b + \omega_b \times (I_n\omega_b) = M_{ext}. \tag{2.32}$$

The similarity of this formulation with (2.22) are evident and the two together constitute the generalized equations of motion of a body.

2.3.4 States of the system

The states variables of a system are the variables that can describe the state of a system in each instant of time. Since the UAV under analysis is a mechanical system whose position and attitude must be controlled acting on the forces and moments applied, it is quite natural to choose as states the linear and angular

position and velocities of the centre of mass. The resulting the state vector is x

$$x = \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \\ \dot{N} \\ \dot{E} \\ \dot{D} \\ \dot{p} \\ \dot{q} \\ \dot{r} \\ \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{bmatrix} \tag{2.33}$$

Given (2.22) and (2.32) the resulting dynamical system to be controlled is thus in the form

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \\ \dot{N} \\ \dot{E} \\ \dot{D} \\ \dot{p} \\ \dot{q} \\ \dot{r} \\ \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{bmatrix} = \begin{bmatrix} -\omega_b \times V_b + \frac{F_{ext}}{m} \\ T_{BE}^T V_b \\ I_n^{-1}(-\omega_b \times I_n \omega_b + M_{ext}) \\ E^{-1}\omega_b \end{bmatrix}. \tag{2.34}$$

2.3.5 External forces and moments

In the study of the UAV dynamics it is necessary to analyse the external forces and moments introduced in the previous sections in order to get a correct modelling of the system. The quadrotor studied is set in X-configuration (see Figure 2.3). Each propeller produces a torque and a thrust proportional to the square of its rotational speed:

$$\begin{aligned} T_i &= K_T \Omega_i^2, \quad K_T = C_T \rho A R^2, \\ Q_i &= K_Q \Omega_i^2, \quad K_Q = C_Q \rho A R^3, \end{aligned} \tag{2.35}$$

Where C_T and C_Q are the thrust and torque coefficients, ρ the air density, A and R the propeller disk and its radius and Ω_i is the i-th propeller rotational speed.

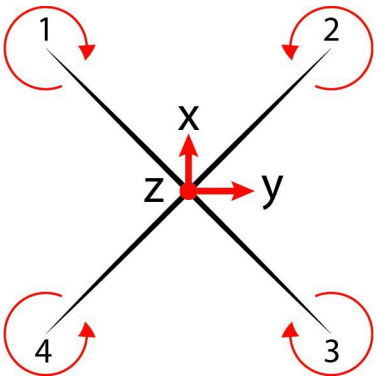


Figure 2.3: X-configuration of the UAV

The overall thrusts and moments generated by the propellers are then

$$\begin{aligned} F_{props} &= \begin{bmatrix} 0 \\ 0 \\ -K_T(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \end{bmatrix} \\ M_{props} &= \begin{bmatrix} K_T\frac{b}{\sqrt{2}}(\Omega_1^2 - \Omega_2^2 - \Omega_3^2 + \Omega_4^2 \\ K_T\frac{b}{\sqrt{2}}(\Omega_1^2 + \Omega_2^2 - \Omega_3^2 - \Omega_4^2 \\ K_Q(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2 \end{bmatrix} \end{aligned} \tag{2.36}$$

With b the distance between each propeller and the centre of gravity. Starting from the expressions of F_{props} and M_{props} a so called $MixerMatrix$ can be built for the motors. This matrix relates the desired thrust and moments required by the control with the rotational speed of each propeller needed to get those values. This matrix is so of fundamental importance in the control design process since the actual control inputs of the UAV are the Ω_i while it is far easier handling forces and moments in the dynamic equations.
The $mixermatrix$ is defined as χ

$$\begin{bmatrix} T \\ L \\ M \\ N \end{bmatrix} = \chi \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} = \begin{bmatrix} K_T & K_T & K_T & K_T \\ K_T\frac{b}{\sqrt{2}} & -K_T\frac{b}{\sqrt{2}} & -K_T\frac{b}{\sqrt{2}} & K_T\frac{b}{\sqrt{2}} \\ K_T\frac{b}{\sqrt{2}} & K_T\frac{b}{\sqrt{2}} & -K_T\frac{b}{\sqrt{2}} & -K_T\frac{b}{\sqrt{2}} \\ -K_Q & K_Q & -K_Q & K_Q \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix}. \tag{2.37}$$

Another force to be considered acting on the centre of gravity is the gravitational force. This force acts always pointing to the centre of the Earth, that is Down in the inertial frame, and so it must be rotated into body frame to get a consistent representation of its effect on the system:

$$F_g = T_{BE}(\Phi, \Theta, \Psi) \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} = \begin{bmatrix} -S_\Theta \\ S_\Phi C_\Theta \\ C_\Phi C_\Theta \end{bmatrix} mg \tag{2.38}$$

The last effect to consider is the aerodynamic damp caused by the rotating propellers moving through air. Assuming in first approximation the drag caused by linear translations to be negligible since the structure of the quadrotor is thin, the moments around each axis are only proportional to the rotational speed around that axis (decoupled moments). Moreover, if one considers only the aerodynamic damp proportional to ω_b it takes the form

$$M_{damp} = \begin{bmatrix} \frac{\partial L}{\partial p} & 0 & 0 \\ 0 & \frac{\partial M}{\partial q} & 0 \\ 0 & 0 & \frac{\partial N}{\partial r} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}. \tag{2.39}$$

The terms $\frac{\partial L}{\partial p}$, $\frac{\partial M}{\partial q}$, $\frac{\partial N}{\partial r}$ are called stability derivatives. Because of the symmetry of the UAV $\frac{\partial L}{\partial p} = \frac{\partial M}{\partial q}$, while $\frac{\partial N}{\partial r} = 0$ since the angular rate about the Z axis is far lower than the others. The stability derivatives are defined from helicopters theory as

$$\frac{\partial L}{\partial p} = -4\rho A R^2 \Omega^2 \frac{\partial C_T}{\partial p} \frac{b}{\sqrt{2}} \tag{2.40}$$

$$\frac{\partial C_T}{\partial p} = \frac{C_{L\alpha}}{8} \frac{\sigma}{R\Omega} \frac{b}{\sqrt{2}} \tag{2.41}$$

$$\sigma = \frac{A_b}{A} \tag{2.42}$$

with σ called solidity ratio obtained dividing the blade area A_b over the disk area A. $C_{L\alpha}$ is defined as the slope of the thrust coefficient curve C_L, this parameter is assumed to be 2π since the actual air-foil is not known and the propeller blade is thin.
All the values of the aerodynamic and structural parameters of the drone have been inherited from [4] and [3] where all the unknown values have been identified using grey-box estimation.
In the end the total forces and moments applied to the centre of mass of the UAV are described by:

$$\begin{aligned} F_{ext} &= F_g + F_{props} \\ M_{ext} &= M_{damp} + M_{props}. \end{aligned} \tag{2.43}$$

2.4 Conclusions

The conventions and the UAV model as been presented in detail in this chapter, highlighting all the modelling assumptions and the implication they have in the final result.
The final model to be controlled is then

$$\begin{aligned} m\dot{V}_b + \omega_b \times (mV_b) &= F_g + F_{props} \\ I_n\dot{\omega}_b + \omega_b \times (I_n\omega_b) &= M_{damp} + M_{props}. \end{aligned} \tag{2.44}$$

Starting from this model a Simulink simulator for the system has been developed in [4] and it has been used extensively in the process of control synthesis and validation. During the development of this thesis some adjustments have been added improving the coherence with the real behaviour of a quadrotor helicopter.

Chapter 3

Control design and tuning

The model presented in the previous chapter is now used to develop a control law for the position and attitude control of the UAV platform.
From the model equations (2.34) it is clear that the system under control is highly nonlinear. Until these days the main control architectures adopted on the UAV platforms of the FLYART laboratory have been linear, mainly *PID* controllers developed under the assumption that the mapping from the angular rates expressed into body frame to the inertial frame ones is an identity and neglecting the gyroscopic effects. In order to improve the results already obtained some nonlinear control architectures are introduced in this chapter that allow to drop these simplifying assumptions.
The chosen control architecture shown in Figure 3.1 and adopted throughout this work consists of two nested parts.The outer position loop computes the total thrust to be generated by the propellers and the roll and pitch angles required to get this thrust. The inner loop controls the attitude of the quadrotor using the moments generated by the propellers and the desired yaw and the computed roll and pitch as set-points instead.

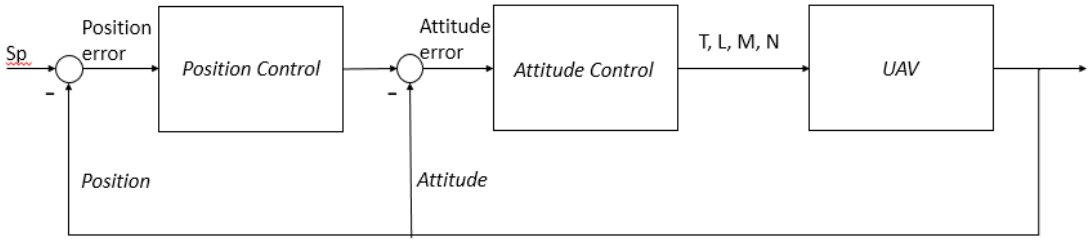


Figure 3.1: Control block diagram

Some basic concepts about nonlinear control are now introduced in order to lay the foundations for the development of the control architectures that follows.

3.1 Lyapunov stability theory

In the case a nonlinear system cannot be linearised around an equilibrium or if this approximation is too strong, one of the most used control methods is the so called Lyapunov method, from the name of Aleksandr Michajlovič Lyapunov who theorized it first.
The main idea underlying this approach is that if the total energy of the system under control is continuously dissipated, at least in the neighbourhood D of an equilibrium x_0, the system tends to move to the equilibrium itself.
In the first place an energy function must be defined. This function V has to be

- positive definite for all $x \neq 0$ with $x \in D$
- $V(x) = 0$ if and only if $x = 0$

If the derivative of $V(x)$ along the trajectories of the system is negative definite, or at least negative semi-definite and null only in $x = 0$, the energy is continuously dissipated by the system or it does not increase and can only be conserved or dissipated. This fact leads to the Lyapunov theorem for nonlinear systems: given a system $\dot{x}(t) = \phi(x(t))$ with ϕ continuous with its derivative and locally Lipschitz in a neighbourhood D of a considered equilibrium x_0, if there exists a function $V(x)$, continuous with its derivative, positive definite in x_0 and such that $\dot{V}(x)$ is semi-definite negative around x_0, that is

$$\dot{V}(x) = \frac{dV}{dx}\frac{dx}{dt} = \frac{dV}{dx}\phi(x) \leq 0 \quad x \in D \tag{3.1}$$

then x_0 is a stable equilibrium, see [8] for the complete proof.
Moreover if $\dot{V}(x) < 0$ strictly, that is $\dot{V}(x)$ negative definite, x_0 is an asymptotically stable equilibrium.
Another theorem, the Krasowsky - La Salle Lemma, proves that in addition to the previous cases if $\dot{V}(x) \leq 0$ but $\dot{V}(x) = 0$ only in $x = x_0$ then the equilibrium is asymptotically stable also in this case.
Based on the results of these theorems, many control strategies for nonlinear systems have been built.
Once a suitable energy function $V(x)$ has been found, with x the states of the system to be controlled, a stabilizing control input that forces its derivative to be negative definite is computed.

3.1.1 The backstepping method

One possible way to compute the control variable is the so called backstepping method.

The general idea behind this technique can be easily explained by means of an example. Given a system with two states x_1, x_2 in the form:

$$\begin{cases} \dot{x}_1 = f(x_1) + gx_2 \quad x_1 \in \mathbb{R}^n \quad x_2 \in \mathbb{R} \\ \dot{x}_2 = u \end{cases} \tag{3.2}$$

Where f is continuous and differentiable in a set $D \subset \mathbb{R}^n$ and $f(0) = 0$ while $g \in \mathbb{R}^{n\times 1}$ constant, and u is the control variable of our system. The objective is to stabilize the system around its equilibrium $x_1 = 0, x_2 = 0$.
To do so temporarily assume that x_2 is a free design variable and to be able to stabilize the first equation of the system with the control law:

$$x_2 = \Phi(x_1), \quad \Phi(0) = 0. \tag{3.3}$$

Adopting this control law the closed loop system is

$$\dot{x}_1 = f(x_1) + g\Phi(x_1). \tag{3.4}$$

Assume to know a Lyapunov function $V_1(x_1)$ such that:

$$\dot{V}_1(x_1) = \frac{dV_1}{dx_1}(f(x_1) + g\Phi(x_1)) = -W(x_1) \tag{3.5}$$

where $W(x_1) > 0$ for any $x_1 \in D$.
At this point it has to be considered that the state x_2 has its own dynamic and cannot be freely chosen, so it is possible to define the error between the actual and the desired behaviour of x_2, as follows

$$\eta = x_2 - \Phi(x_1), \quad \Phi(0) = 0. \tag{3.6}$$

The derivative of η is then

$$\begin{aligned} \dot{\eta} &= \dot{x}_2 - \dot{\Phi}(x_1) \\ &= u - \frac{d\Phi(x_1)}{dx_1}\dot{x}_1 \\ &= u - \frac{d\Phi(x_1)}{dx_1}\Big[f(x_1) + g\big(\eta + \Phi(x_1)\big)\Big]. \end{aligned} \tag{3.7}$$

Therefore the initial system can be reformulated as

$$\begin{cases} \dot{x}_1 = f(x_1) + g\big[\eta + \Phi(x_1)\big] \\ \dot{\eta} = u - \dfrac{d\Phi(x_1)}{dx_1}\Big[f(x_1) + g\big(\eta + \Phi(x_1)\big)\Big]. \end{cases} \tag{3.8}$$

It is now possible to define a new input

$$v = u - \dot{\Phi}(x_1). \tag{3.9}$$

It is quite evident that $\dot{\eta} = v$, and the system can be given the form

$$\begin{cases} \dot{x}_1 = f(x_1) + g\Phi(x_1) + g\eta \\ \dot{\eta} = v \end{cases} \tag{3.10}$$

where the origin is an asymptotically stable equilibrium for the first subsystem. Consider the following Lyapunov function for the overall system

$$V_2(x_1, \eta) = V_1(x_1) + \frac{1}{2}\eta^2 \tag{3.11}$$

and its derivative

$$\begin{aligned} \dot{V}_2(x_1, \eta) &= \frac{dV_1}{dx_1}\dot{x}_1 + \eta\dot{\eta} \\ &= \frac{dV_1}{dx_1}\Big(f(x_1) + g\Phi(x_1) + g\eta\Big) + \eta v \\ &\leq -W(x_1) + \frac{dV_1}{dx_1}g\eta + \eta v \end{aligned} \tag{3.12}$$

Therefore, by imposing

$$v = -\frac{dV_1}{dx_1}g - k\eta \quad k > 0 \tag{3.13}$$

it follows that

$$\dot{V}_2(x_1, \eta) = -W(x_1) - k\eta^2 < 0, \tag{3.14}$$

thus the equilibrium $(x_1 = 0, \eta = 0)$ is an asymptotically stable equilibrium of the system. Recalling that $\eta = x_2 - \Phi(x_1)$ and $\Phi(0) = 0$ it follows that $(x_1 = 0, x_2 = 0)$ is an asymptotically stable equilibrium.
One can compute the expression of the control law for u that stabilizes the origin as function of the original coordinates as

$$\begin{aligned} u &= v + \dot{\Phi}(x_1) \\ &= -\frac{dV_1}{dx_1}g - k\big(x_2 - \Phi(x_1)\big) + \dot{\Phi}(x_1) + \frac{d\Phi(x_1)}{dx_1}\big(f(x_1) + gx_2\big) \end{aligned} \tag{3.15}$$

This control law guarantees the asymptotic stability of the origin, with the associated Lyapunov function

$$V_2(x_1, x_2) = V_1(x_1) + \frac{1}{2}\big(x_2 - \Phi(x_1)\big)^2. \tag{3.16}$$

Thus it is possible to consider a state as a virtual control variable that stabilizes the other states, then back-compute the required control law in order to force that state to stabilize the others.
For the general formulation and a deeper insight on this technique see [8] .

3.2 Backstepping control of a UAV quadrotor

In this section the derivation of the backstepping control algorithm for the UAV platform is carried out.
In first instance the derivation of the attitude control law is explained, all the computations are carried out considering only one degree of freedom of the system and then the final results are extended to the others. Then the derivation of the position control algorithm is introduced. All the procedure has been faced in [1].

3.2.1 Derivation of the attitude control law

Starting from (2.44) and (2.34) , one can extract the equations of the attitude dynamics:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \\ \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{bmatrix} = \begin{pmatrix} I_n^{-1}(-\omega_b \times I_n\omega_b + M_{ext}) \\ E^{-1}\omega_b \end{pmatrix}. \tag{3.17}$$

The resulting equations of the attitude dynamics turn out to be:

$$\begin{cases} \dot{p} = I_{xx}^{-1}\big[(I_{yy} - I_{zz})qr + \dfrac{\partial L}{\partial p}p + L\big] \\ \dot{q} = I_{yy}^{-1}\big[(I_{zz} - I_{xx})pr + \dfrac{\partial M}{\partial q}q + M\big] \\ \dot{r} = I_{zz}^{-1}\big[(I_{xx} - I_{yy})pq + \dfrac{\partial N}{\partial r}r + N\big]. \end{cases} \tag{3.18}$$

Assuming that the rotation axes are decoupled ($E^{-1} = I$) one has that

$$\begin{bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{bmatrix} = \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} \tag{3.19}$$

and applying to the dynamics of each Euler angle the 1-degree of freedom approximation we can directly access the roll, pitch and yaw dynamics starting from the angular rates dynamics expressed in body reference frame. All these

assumptions are widely used in the literature and lead to the model

$$\begin{cases} \ddot{\Phi} = I_{xx}^{-1}\big[(I_{yy} - I_{zz})\dot{\Theta}\dot{\Psi} + \dfrac{\partial L}{\partial p}\dot{\Phi} + L\big] \\ \ddot{\Theta} = I_{yy}^{-1}\big[(I_{zz} - I_{xx})\dot{\Phi}\dot{\Psi} + \dfrac{\partial M}{\partial q}\dot{\Theta} + M\big] \\ \ddot{\Psi} = I_{zz}^{-1}\big[(I_{xx} - I_{yy})\dot{\Phi}\dot{\Theta} + \dfrac{\partial N}{\partial r}\dot{\Psi} + N\big]. \end{cases} \tag{3.20}$$

From equation (3.20) the objective is now to find a way to control the attitude of the UAV so that it can reach any desired attitude (excluding the singularities of the representation) in a fast and stable manner.
In order to do so a new system can be built that considers the dynamics of the attitude errors. If a stabilizing control law for the origin of the states of this system is achieved, it is possible to set the quadrotor at any desired attitude. From now on all the computations are developed for one of the three coordinates and the final results for all coordinates are shown at the end of the section.
The first step is to define the attitude error

$$e_\Phi = \Phi_{des} - \Phi \tag{3.21}$$

and then compute the dynamics of the chosen Lyapunov function and find an equation for the control variable that stabilizes the function dynamics.

$$\begin{aligned} V_\Phi &= \frac{1}{2}e_\Phi^2 \\ \dot{V_\Phi}(e_\Phi) &= e_\Phi(\dot{\Phi}_{des} - \dot{\Phi}) \end{aligned} \tag{3.22}$$

In order to get Lyapunov stability for the origin, the derivative along the trajectories of the Lyapunov function has to be negative definite (or at least negative semi-definite with the only acceptable root of the function in the origin). This can be achieved setting

$$\begin{aligned} \dot{\Phi} &= \dot{\Phi}_{des} + \alpha_1 e_\Phi \\ \dot{V_\Phi}(e_\Phi) &= -\alpha_1 e_\Phi^2 \end{aligned} \tag{3.23}$$

with $\alpha_1 > 0$ free control parameter to be tuned that affects the convergence to zero of the error dynamics. The convergence of $\dot{\Phi}$ to the desired value that ensures the control of the roll angle dynamics must now be studied. To do so we extend the previous Lyapunov function including also the dynamics of the roll rate error $e_{\dot{\Phi}}$ defined as

$$e_{\dot{\Phi}} = \dot{\Phi} - \dot{\Phi}_{des} - \alpha_1 e_\Phi \tag{3.24}$$

The extended Lyapunov function takes the form

$$V_\Phi(e_\Phi, e_{\dot{\Phi}}) = \frac{1}{2}(e_\Phi^2 + e_{\dot{\Phi}}^2) \tag{3.25}$$

and after the derivation and substituting (3.23) and (3.24)

$$\begin{aligned}\dot{V_\Phi}(e_\Phi, e_{\dot\Phi}) &= e_\Phi(\dot\Phi_{des} - \dot\Phi) + e_{\dot\Phi}(\ddot\Phi - \ddot\Phi_{des} - \alpha_1(\dot\Phi_{des} - \dot\Phi)) \\ &= e_\Phi(\dot\Phi_{des} - e_{\dot\Phi} - \dot\Phi_{des} - \alpha_1 e_\Phi) + e_{\dot\Phi}[\ddot\Phi - \ddot\Phi_{des} - \alpha_1(\dot\Phi_{des} - e_{\dot\Phi} - \dot\Phi_{des} - \alpha_1 e_\Phi)] \\ &= -\alpha_1 e_\Phi^2 - e_{\dot\Phi}[e_\Phi - \ddot\Phi + \ddot\Phi_{des} - \alpha_1 e_{\dot\Phi} - \alpha_1^2 e_\Phi]\end{aligned} \tag{3.26}$$

from (3.20) we get the expression of $\ddot\Phi$. It is possible to assume $\ddot\Phi_{des} = 0$ since it turns out to be only a feed-forward term, simplifying the computations:

$$\dot{V_\Phi}(e_\Phi, e_{\dot\Phi}) = -\alpha_1 e_\Phi^2 - e_{\dot\Phi}\Bigg\{e_\Phi - \left[\frac{I_{yy} - I_{zz}}{I_{xx}}\dot\Theta\dot\Psi + \frac{\partial L}{\partial p}\frac{\dot\Phi}{I_{xx}} + \frac{L}{I_{xx}}\right] + - \alpha_1 e_{\dot\Phi} - \alpha_1^2 e_\Phi\Bigg\}.$$

The computed expression of $\dot{V_\Phi}(e_\Phi, e_{\dot\Phi})$ can be used to back-calculate the expression of L such that Lyapunov stability is achieved. Imposing

$$e_{\dot\Phi}\Bigg\{e_\Phi - \left[\frac{I_{yy} - I_{zz}}{I_{xx}}\dot\Theta\dot\Psi + \frac{\partial L}{\partial p}\frac{\dot\Phi}{I_{xx}} + \frac{L}{I_{xx}}\right] - \alpha_1 e_{\dot\Phi} - \alpha_1^2 e_\Phi\Bigg\} > 0 \tag{3.27}$$

the resulting expression for the moment L is

$$L = -(I_{yy} - I_{zz})\dot\Theta\dot\Psi - \frac{\partial L}{\partial p}\dot\Phi - I_{xx}(1 - \alpha_1^2)e_\Phi - I_{xx}(\alpha_1 + \alpha_2)e_{\dot\Phi} \tag{3.28}$$

where $\alpha_2 > 0$ is a design parameter that affects the convergence to zero of the roll rate error. Imposing this value of L one can get

$$\dot{V_\Phi}(e_\Phi, e_{\dot\Phi}) = -\alpha_1 e_\Phi^2 - \alpha_1 e_{\dot\Phi}^2 \tag{3.29}$$

ensuring Lyapunov stability for the zero equilibrium of the error dynamics. These results can be replicated for all the three degree of freedom dynamics ending up with the following expressions for the moments

$$\begin{aligned}L &= -(I_{yy} - I_{zz})\dot\Theta\dot\Psi - \frac{\partial L}{\partial p}\dot\Phi - I_{xx}(1 - \alpha_1^2)e_\Phi - I_{xx}(\alpha_1 + \alpha_2)e_{\dot\Phi} \\ M &= -(I_{zz} - I_{xx})\dot\Phi\dot\Psi - \frac{\partial M}{\partial q}\dot\Theta - I_{yy}(1 - \alpha_3^2)e_\Theta - I_{yy}(\alpha_3 + \alpha_4)e_{\dot\Theta} \\ N &= -(I_{xx} - I_{yy})\dot\Phi\dot\Theta - \frac{\partial N}{\partial r}\dot\Psi - I_{zz}(1 - \alpha_5^2)e_\Psi - I_{zz}(\alpha_5 + \alpha_6)e_{\dot\Psi}.\end{aligned} \tag{3.30}$$

3.2.2 Derivation of the position control law

The derivation of the position control law has been also carried out using the backstepping approach.

While the altitude control law derivation follows exactly the same path of the attitude one, the x, y axes controls are used to compute the desired roll and pitch angles and thus the derivation is slightly different.
The considered dynamic model can be extracted from (2.44) and (2.34) and takes the form:

$$\begin{cases} \ddot{x} = \big(c_\Phi s_\Theta c_\Psi + s_\Phi s_\Psi\big)\big(\frac{-T}{m}\big) \\ \ddot{y} = \big(c_\Phi s_\Theta s_\Psi - s_\Phi c_\Psi\big)\big(\frac{-T}{m}\big) \\ \ddot{z} = c_\Phi c_\Theta\big(\frac{-T}{m}\big) + g. \end{cases} \tag{3.31}$$

The first step is to compute the total thrust T required to get the desired altitude. Defining the altitude error e_z and the candidate Lyapunov function V_z as

$$\begin{aligned} e_z &= z_{des} - z \\ V_z &= \frac{1}{2} e_z^2 \end{aligned} \tag{3.32}$$

the derivative of V_z can be easily computed as

$$\dot{V}_z(e_z) = e_z(\dot{z}_{des} - \dot{z}). \tag{3.33}$$

Following the backstepping logic the state $\dot{z}$ can be chosen as virtual control variable and then its value should be

$$\dot{z}_v = \dot{z}_{des} + \alpha_7 e_z \tag{3.34}$$

Then the extended Lyapunov function $V(e_z, e_{\dot{z}})$ is defined as

$$\begin{aligned} e_{\dot{z}} &= \dot{z} - \dot{z}_{des} - \alpha_7 e_z \\ V_z(e_z, e_{\dot{z}}) &= \frac{1}{2}(e_z^2 + e_{\dot{z}}^2). \end{aligned} \tag{3.35}$$

It is now possible to find the total thrust to be applied in order to get the desired equation following the same procedure used in (3.26) and imposing $\dot{V}_z \leq 0$.
The resulting total thrust is then:

$$T = \frac{m}{C_\Phi C_\Theta}\Big[g + (\alpha_7 + \alpha_8)e_{\dot{z}} + (\alpha_7^2 - 1)e_z\Big]. \tag{3.36}$$

Where $\alpha_{7,8} > 0$ are control parameters to be tuned.
Once the expression of T for the altitude control is fixed, the x, y controls aim at computing the desired values of roll and pitch in order to direct the thrust correctly considering that the desired yaw is set by the user. The model as presented in (3.31) is in the form

$$\begin{cases} \ddot{x} = \big(c_\Phi s_\Theta c_\Psi + s_\Phi s_\Psi\big)\big(\frac{-T}{m}\big) \\ \ddot{y} = \big(c_\Phi s_\Theta s_\Psi - s_\Phi c_\Psi\big)\big(\frac{-T}{m}\big) \end{cases} \tag{3.37}$$

where it is possible to identify the components of the thrust in the (x, y) plane as

$$\begin{cases} u_x = \left(c_\Phi s_\Theta c_\Psi + s_\Phi s_\Psi\right) \\ u_y = \left(c_\Phi s_\Theta s_\Psi - s_\Phi c_\Psi\right) \end{cases} \tag{3.38}$$

The idea is then to compute u_x, u_y using the backstepping technique and then solve the system of two equations in two unknowns (3.38) to get the required values of Φ and Θ. The setup is the same as before and for this reason only the final results are presented. The y dynamic since the UAV is assumed to be perfectly symmetric takes the same form :

$$\begin{aligned} e_x &= x_{des} - x \\ V_x &= \frac{1}{2} e_x^2 \\ \dot{x}_v &= \dot{x}_{des} + \alpha_9 e_x \\ e_{\dot{x}} &= \dot{x} - \dot{x}_{des} - \alpha_9 e_x \\ V_{x,ext}(e_x, e_{\dot{x}}) &= \frac{1}{2}(e_x^2 + e_{\dot{x}}^2). \end{aligned} \tag{3.39}$$

The resulting u_x, u_y direction is then

$$\begin{aligned} u_x &= \frac{m}{T}\left[(\alpha_9 + \alpha_{10})e_{\dot{x}} + (\alpha_9^2 - 1)e_x\right] \\ u_y &= \frac{m}{T}\left[(\alpha_{11} + \alpha_{12})e_{\dot{y}} + (\alpha_{11}^2 - 1)e_y\right]. \end{aligned} \tag{3.40}$$

Solving (3.38) with respect to Φ and Θ the values of S_Φ and S_Θ are obtained and can be used to find the related angles:

$$\begin{cases} S_\Phi = \dfrac{u_x T_\Psi - u_y}{S_\Psi T_\Psi + C_\Psi} \\ S_\Theta = \dfrac{u_x - S_\Phi S_\Psi}{C_\Phi C_\Psi} \end{cases} \tag{3.41}$$

3.3 Tuning of the backstepping algorithm

In this section the technique adopted to tune the control parameters $(\alpha_i, i = 1..6)$ is analysed. As in the previous section only the tuning of the controller for one coordinate is presented while for all the remaining degrees of freedom the final results are presented. Taking into account (3.30) one can split the control variable equation into two parts: a nonlinear feed-forward action that counteracts the gyroscopic effects on the UAV and a PD control action based on the position error.

In fact we can rearrange the expression of L as

$$L = \underbrace{-(I_{yy} - I_{zz})\dot{\Theta}\dot{\Psi} - \frac{\partial L}{\partial p}\dot{\Phi}}_{\text{Nonlinear FF}} + \underbrace{L'}_{PD \text{ action}} \tag{3.42}$$

With L' in the form

$$L' = \underbrace{-I_{xx}(1-\alpha_1^2)e_\Phi}_{\text{Proportional action}} \underbrace{-I_{xx}(\alpha_1+\alpha_2)e_{\dot{\Phi}}}_{\text{Derivative action}} \tag{3.43}$$

Substituting in (3.43) the expressions of $e_\Phi, e_{\dot{\Phi}}$ in (3.23) and (3.24) we can rearrange its expression as

$$\begin{aligned} L' &= -I_{xx}[(\alpha_1^2 - 1 - \alpha_1\alpha_2 - \alpha_1^2)(\Phi_{des} - \Phi) - (\alpha_1+\alpha_2)(\dot{\Phi}_{des} - \dot{\Phi})] \\ &= I_{xx}[(1+\alpha_1\alpha_2)e_\Phi + (\alpha_1+\alpha_2)\dot{e_\Phi}] \\ &= K_{P\Phi}e_\Phi + K_{D\Phi}\dot{e_\Phi} \end{aligned} \tag{3.44}$$

$$\begin{aligned} K_{P\Phi} &= I_{xx}(1+\alpha_1\alpha_2) \\ K_{D\Phi} &= I_{xx}(\alpha_1+\alpha_2) \end{aligned}$$

From this results it is clear that one can deal with the control tuning problem as a PD control tuning task assuming the perfect compensation of the feed-forward terms. The aim of the procedure outlined in the following is to find optimal values for α_1, α_2 exploiting the fact that the analytical relation between them and their equivalent effect in a linear controller is available.
To do so the structured mixed sensitivity H_∞ technique has been adopted applying it to the system linearised around each rotation axes. The controller is thus parametrised as function of α_1, α_2 leaving the solution of the nonlinear relation between α_1, α_2 and $K_{P\Phi}, K_{D\Phi}$ to the optimization algorithm. The linearised system takes the form

$$\begin{cases} \ddot{\Phi} = I_{xx}^{-1}L \\ \ddot{\Theta} = I_{yy}^{-1}M \\ \ddot{\Psi} = I_{zz}^{-1}N \end{cases} \tag{3.45}$$

which can be rewritten as

$$\begin{cases} \phi = I_{xx}^{-1}\left[\frac{1}{s^2}\right]\mathcal{L} \\ \theta = I_{yy}^{-1}\left[\frac{1}{s^2}\right]\mathcal{M} \\ \psi = I_{zz}^{-1}\left[\frac{1}{s^2}\right]\mathcal{N}. \end{cases} \tag{3.46}$$

The starting point is then a double integrator since the stability derivatives are assumed to be perfectly compensated by the feedforward. In order to get a more consistent model and a more robust tuning (3.46) has been cascaded with the model of the actuators identified in [4] and analysed in 4.1.2 and a delay of $0.02[s]$ representing the sensors and the attitude determination system. The resulting closed-loop system, the control law, the sensors and actuators, the linearised quadrotor has been fed to the *Systune* function provided by the *Matlab* environment.

3.3.1 Shaping functions

One of the key aspects of the structured H_∞ tuning technique is the choice of the shaping functions to be fed to the optimization algorithm in order to achieve the tuning of the control parameters.
The objective of mixed-sensitivity synthesis is to keep the the H_∞-norm of the product between the sensitivity, complementary sensitivity and control sensitivity functions of the system with the shaping functions as close as possible to 1, that is

$$\begin{aligned} ||W_S(s)S(s)||_\infty &\leq 1 \\ ||W_T(s)T(s)||_\infty &\leq 1 \\ ||W_K(s)K(s)||_\infty &\leq 1. \end{aligned} \tag{3.47}$$

Thus each weight has a specific meaning and has to be chosen following some guidelines that have been derived from [10] .

- $W_S(s)$ weights the sensitivity of the system to external disturbances on the output. Namely it is the transfer function from disturbance on the output to the output itself. It has to be chosen high at low frequency in order to get a strong attenuation of the disturbances in the bandwidth of interest (below the frequency ω_B) and around $0dB$ at high frequency. The optimal value of $0dB$ at high frequency usually is not a good choice since there is quite often a resonance peak around ω_B, thus choosing $W_S(\omega_B) = 0$ makes the algorithm try to push the resonance down and so it strongly limits the overall performances of the resulting control, however the high frequency value must be kept low in order to damp the dominant pole of the system. The theory, see [8], suggests that the overall crossover frequency ω_c of the controlled system is higher that ω_B. The inverse of the chosen function is shown in Figure 3.2 and guarantees $\omega_c \geq 3$ $[rad/s]$.

- $W_T(s)$ weights the complementary sensitivity function, that affects the robustness properties of the system. Since the uncertainties on the model parameters are low due to various identification campaigns (see [4] and [3]) this function has been used in order to enforce a consistent upper bound to

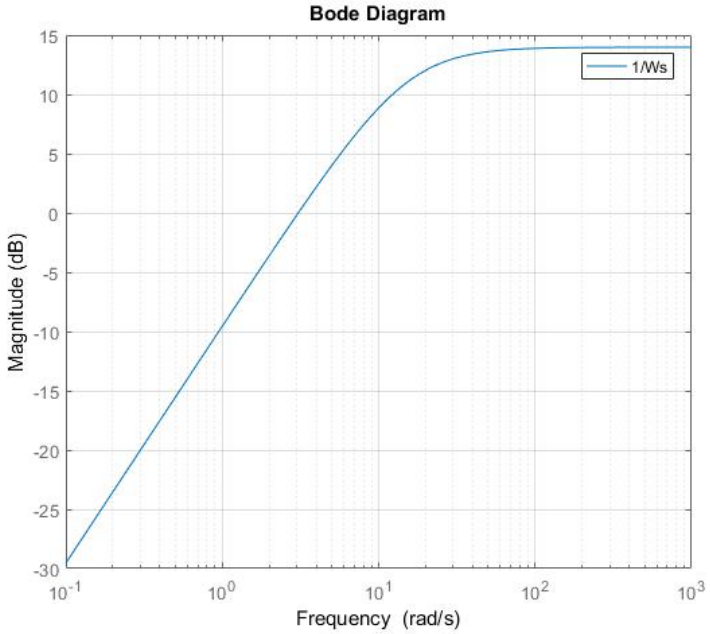


Figure 3.2: Magnitude of the frequency response of inverse of the chosen $W_S(s)$ function

the ω_c since $\omega_B \leq \omega_c \leq \omega_{BT}$ with ω_{BT} the crossover frequency of $W_T(s)$. Since the complementary sensitivity function is $1 - S(s)$ the magnitude of $W_T(j\omega)$ will be around $0dB$ at low frequency, decreases for higher frequencies and has a cutting frequency $\omega_{BT} = 7\ [rad/s]$as one can deduce from Figure 3.3 in which the magnitude of the frequency response of the inverse of $W_T(s)$ is shown.

- $W_K(s)$ weights the control sensitivity function $K(s)$. The $K(s)$ function can be seen as the transfer function
  - from reference signal to control action;
  - from disturbance on the output to control action;
  - from measurement noise to control action.

  In all these cases, it is desirable to keep the magnitude of the control sensitivity frequency response as small as possible outside the bandwidth of the system.
  $W_K(s)$ have also to keep into account physical restrictions of the system, that is the bandwidth of the actuators and the maximum deliverable moments. For all these reasons the selected weighting function has a cutting frequency of $\omega_{BK} = 18.09[rad/s]$ and a shape that can be deduced from the magnitude of the frequency response of its inverse shown in Figure 3.4 .

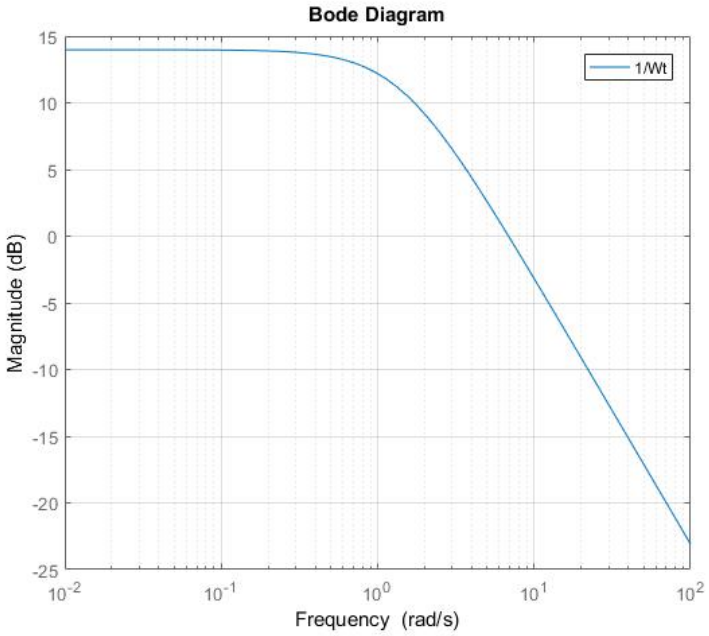


Figure 3.3: Magnitude of the frequency response of inverse of the chosen $W_T(s)$ function

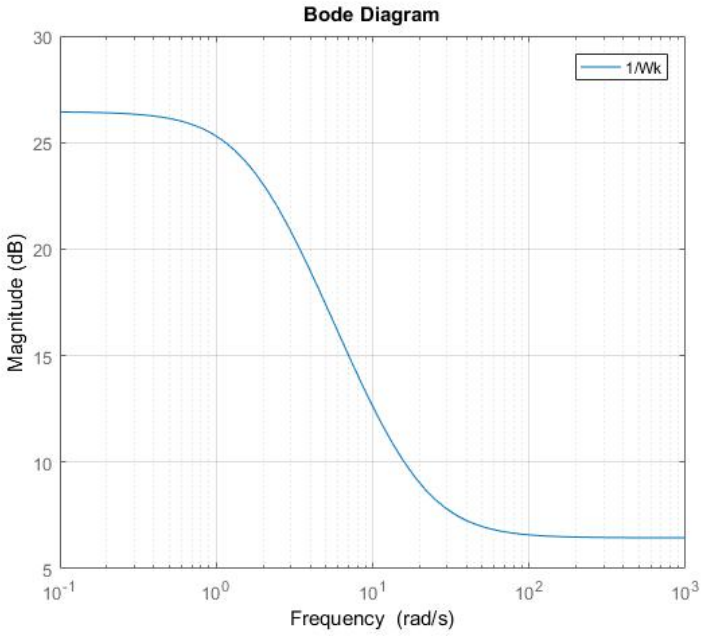


Figure 3.4: Magnitude of the frequency response of inverse of the chosen $W_K(s)$ function

Table 3.1: Outer loop shaping functions parameters, backstepping

$\omega_B[rad/s]$	$\omega_{BT}[rad/s]$	$\omega_{BK}[rad/s]$
0.1	0.2	18.09

Once established all these requirements, the shaping functions have been built up using first order functions and have been used for the tuning of each of the attitude controllers.
The same approach has been adopted for the tuning of the position control loop, following exactly the same steps and considering the inner loop to be a unitary gain since quite faster then the outer one. The inverses of the chosen weighting functions are shown in Figure 3.5 with the characteristics in Table 3.1

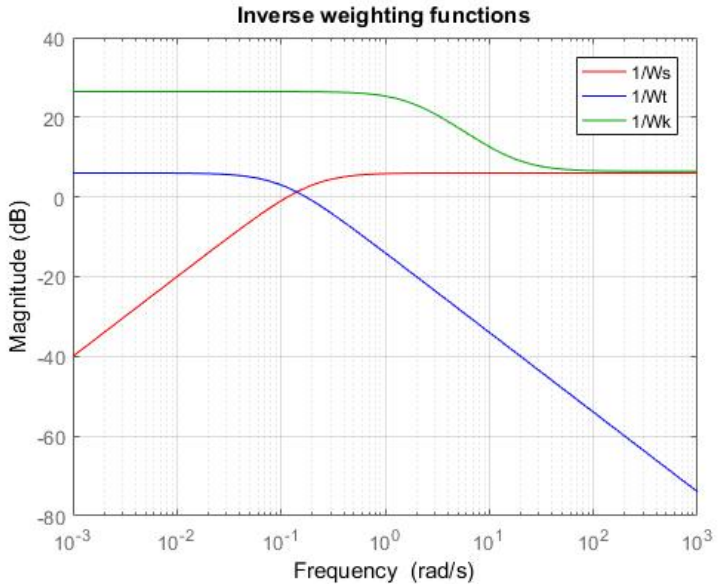


Figure 3.5: Magnitude of the frequency response of the inverse outer loop weighting functions, backstepping

3.3.2 Tuning results

The resulting tuning of the parameters for the inner and outer loops is shown in Tables 3.2, 3.3 respectively.
Figures 3.6, 3.7 show the results of the H_∞ tuning procedure for the pitch and yaw dynamics comparing the shaping functions presented before with the obtained $S(s), T(s), K(s)$ of the tuned systems. It is easy to see that the tuning procedure has led to good tuning results, at least with the linearised system. In principle it is

Table 3.2: Inner loop tuned parameters, backstepping

α_1	6.271
α_2	6.271
α_3	6.271
α_4	6.271
α_5	5.716
α_6	5.716

Table 3.3: Outer loop tuned parameters, backstepping

α_7	6.983
α_8	0.010
α_9	0.014
α_{10}	7.88
α_{11}	0.014
α_{12}	7.88

also possible to get a more aggressive control but in order to get some more phase margin since the system itself has quite strong approximations on the dynamics, a smoother control action has been preferred.

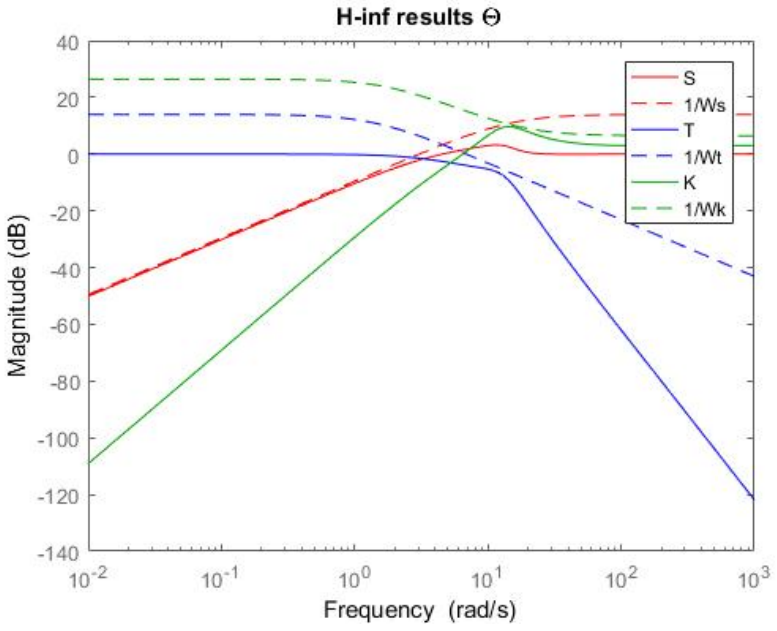


Figure 3.6: Comparison between the inverse of the weighting functions and the obtained results: pitch axis

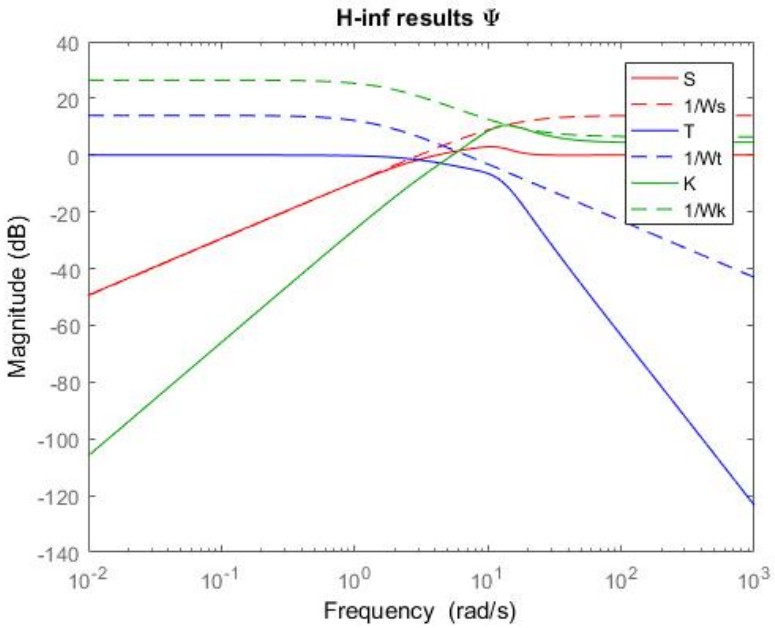


Figure 3.7: Comparison between the inverse of the weighting functions and the obtained results: yaw axis

The same consideration can be made also for the results obtained on the position dynamics whose results are presented in Figures 3.8, 3.9 respectively.

3.4 Integral backstepping control

The backstepping procedure guarantees the stability of the system but does not guarantee zero steady state error, thus in order to get a more robust and precise control the classic backstepping control is now extended with an integral action on the attitude loop. This control architecture is called *Integral Backstepping* and its formulation is derived quite easily from the backstepping one (See [1]).

3.4.1 Derivation of the attitude control law

Starting from (3.20) as in the pure backstepping case the error dynamics are considered. As in the previous case all the computations are developed for one coordinate only and then the final results for each degree of freedom are shown.
In addition to the backstepping, in the integral backstepping case the candidate Lyapunov function contains a term that weights the integral of the error, defining

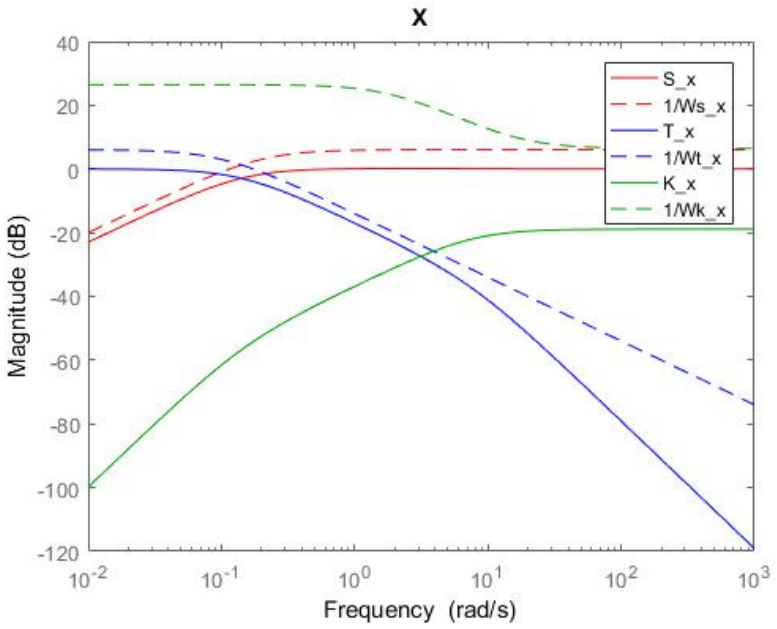


Figure 3.8: Comparison between the inverse of the weighting functions and the obtained results: x axis

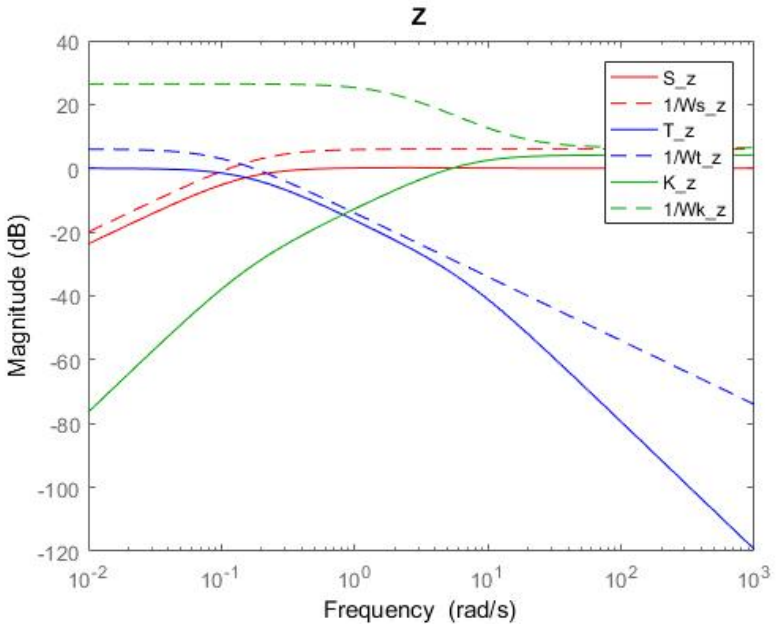


Figure 3.9: Comparison between the inverse of the weighting functions and the obtained results: z axis

the attitude error e_Φ and its integral E_Φ it takes the form

$$\begin{aligned} e_\Phi &= \Phi_{des} - \Phi \\ E_\Phi &= \int e_\Phi \\ V_\Phi &= \frac{1}{2}(e_\Phi^2 + \lambda_\Phi E_\Phi^2) \end{aligned} \tag{3.48}$$

with λ_Φ parameter to be tuned.
The virtual control $\dot{\Phi}_V$ is then computed in order to keep the derivative of $\dot{V}_\Phi < 0$

$$\begin{aligned} \dot{V}_\Phi(e_\Phi, E_\Phi) &= e_\Phi(\dot{\Phi}_{des} - \dot{\Phi} + \lambda_\Phi E_\Phi) \\ \dot{\Phi}_V &= \dot{\Phi}_{des} + \lambda_\Phi E_\Phi + \alpha_1 e_\Phi. \end{aligned} \tag{3.49}$$

The next step is to extend the Lyapunov function in order to consider that our control variable is not the virtual control one, leading to an error that must converge to zero:

$$\begin{aligned} V_\Phi(e_\Phi, e_{\dot{\Phi}}, E_\Phi) &= \frac{1}{2}(e_\Phi^2 + e_{\dot{\Phi}}^2 + \lambda_\Phi E_\Phi^2) \\ e_{\dot{\Phi}} = \dot{\Phi}_V - \dot{\Phi} &= \dot{\Phi}_{des} + \lambda_\Phi E_\Phi + \alpha_1 e_\Phi - \dot{\Phi} \\ \dot{\Phi} &= \dot{\Phi}_{des} + \lambda_\Phi E_\Phi + \alpha_1 e_\Phi - e_{\dot{\Phi}}. \end{aligned} \tag{3.50}$$

The derivative of this extended Lyapunov function is then

$$\begin{aligned} \dot{V}_\Phi(e_\Phi, e_{\dot{\Phi}}, E_\Phi) &= e_\Phi \dot{e}_\Phi + \lambda_\Phi E_\Phi e_\Phi + e_{\dot{\Phi}} \dot{e}_{\dot{\Phi}} \\ \dot{e}_\Phi = \dot{\Phi}_{des} - \dot{\Phi} &= e_{\dot{\Phi}} - \lambda_\Phi E_\Phi - \alpha_1 e_\Phi \\ \dot{e}_{\dot{\Phi}} &= \alpha_1(e_{\dot{\Phi}} - \lambda_\Phi E_\Phi - \alpha_1 e_\Phi) + \lambda_\Phi e_\Phi + \ddot{\Phi}_{des} - \ddot{\Phi}. \end{aligned} \tag{3.51}$$

Substituting all the terms it is possible to get:

$$\begin{aligned} \dot{V}_\Phi(e_\Phi, e_{\dot{\Phi}}, E_\Phi) = e_\Phi(e_{\dot{\Phi}} - \lambda_\Phi E_\Phi - \alpha_1 e_\Phi) + \lambda_\Phi E_\Phi e_\Phi \\ + e_{\dot{\Phi}}[\alpha_1(e_{\dot{\Phi}} - \lambda_\Phi E_\Phi - \alpha_1 e_\Phi) + \lambda_\Phi e_\Phi + \ddot{\Phi}_{des} - \ddot{\Phi}]. \end{aligned} \tag{3.52}$$

Imposing $\dot{V}_\Phi = -\alpha_1 e_\Phi^2 - \alpha_2 e_{\dot{\Phi}}^2$ one can extract the required value of the acceleration that is:

$$\ddot{\Phi} = (1 + \lambda_\Phi - \alpha_1^2)e_\Phi + (\alpha_1 + \alpha_2)e_{\dot{\Phi}} - \lambda_\Phi \alpha_1 E_\Phi + \ddot{\Phi}_{des}. \tag{3.53}$$

Then substituting the expression of the angular acceleration from (3.20) and assuming $\ddot{\Phi}_{des} = \ddot{\Theta}_{des} = \ddot{\Psi}_{des} = 0$ the control moments are obtained:

$$\begin{aligned} L &= -(I_{yy} - I_{zz})\dot{\Theta}\dot{\Psi} - \frac{\partial L}{\partial p}\dot{\Phi} + I_{xx}[(1 + \lambda_\Phi - \alpha_1^2)e_\Phi + (\alpha_1 + \alpha_2)e_{\dot{\Phi}} - \lambda_\Phi \alpha_1 E_\Phi] \\ M &= -(I_{zz} - I_{xx})\dot{\Phi}\dot{\Psi} - \frac{\partial M}{\partial q}\dot{\Theta} + I_{yy}[(1 + \lambda_\Theta - \alpha_3^2)e_\Theta + (\alpha_3 + \alpha_4)e_{\dot{\Theta}} - \lambda_\Theta \alpha_3 E_\Theta] \\ N &= -(I_{xx} - I_{yy})\dot{\Phi}\dot{\Theta} - \frac{\partial N}{\partial r}\dot{\Psi} + I_{zz}[(1 + \lambda_\Psi - \alpha_5^2)e_\Psi + (\alpha_5 + \alpha_6)e_{\dot{\Psi}} - \lambda_\Psi \alpha_5 E_\Psi]. \end{aligned} \tag{3.54}$$

Table 3.4: Outer loop shaping functions parameters, integral backstepping

$\omega_B[rad/s]$	$\omega_{BT}[rad/s]$	$\omega_{BK}[rad/s]$
4.5	6.5	18.09

Table 3.5: Outer loop tuned parameters, integral backstepping

α_1	7.3033
α_2	7.3007
λ_Φ	0.010
α_3	7.3033
α_4	7.3007
λ_Θ	0.010
α_5	2.0960
α_6	6.3184
λ_Φ	20.613

3.4.2 Tuning of the integral backstepping control law

Like the previous case the control expression can be divided into two parts: a nonlinear feedforward that compensates the gyroscopic effects and the stability derivatives, and a control action that depends on the errors and the control parameters to be tuned.
Expanding $e_{\dot{\Phi}}$ as in the previous case the expression turns out to be a PID controller:

$$L = -(I_{yy}-I_{zz})\dot{\Theta}\dot{\Psi} - \frac{\partial L}{\partial p}\dot{\Phi} + I_{xx}[\lambda_\Phi \alpha_2 E_\Phi + (1+\lambda_\Phi - \alpha_1\alpha_2)e_\Phi + (\alpha_1+\alpha_2)\dot{e}_\Phi] \quad (3.55)$$

So, as in the classical backstepping architecture we can consider the control action as a PID with gains that depend on the Lyapunov parameters:

$$\begin{aligned} K_I &= I_{xx}\lambda_\Phi\alpha_2 \\ K_P &= I_{xx}(1+\lambda_\Phi - \alpha_1\alpha_2) \\ K_D &= I_{xx}(\alpha_1+\alpha_2) \end{aligned} \quad (3.56)$$

The so obtained control law has been tuned using the H_∞ technique described in section 3.3 using the shaping functions whose parameters are described in Table 3.4 and depicted in Figure 3.10. Also in this case the *Systune* routine has been adopted and leads to a satisfactory tuning as can be seen in Figure 3.11 regarding the pitch dynamics and in Figure 3.12 for the yaw one.

The resulting Lyapunov parameters that have been found in this way are summarized in Table 3.5. It is quite evident that with respect to the pure backstepping, in this case ω_B has been set to 4.5 $[rad/s]$, that is quite faster, and as a

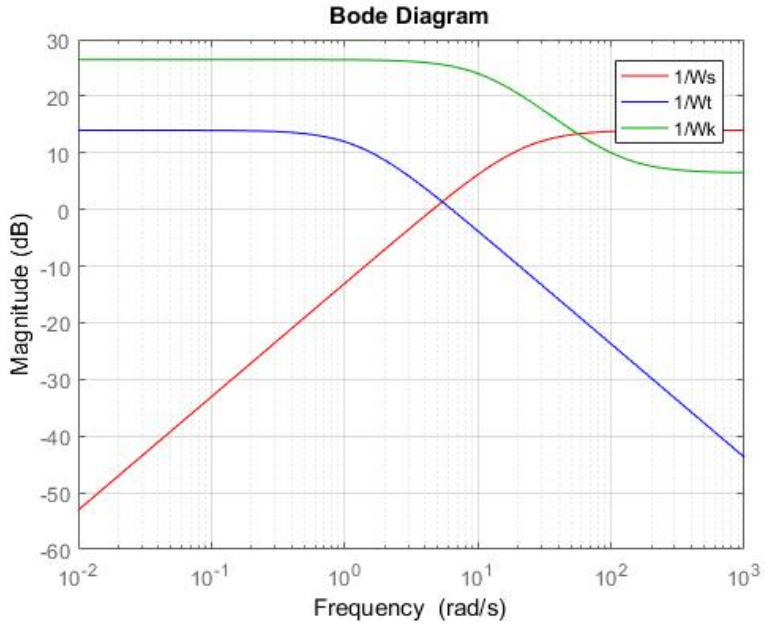


Figure 3.10: Magnitude of the frequency response of the inverse shaping functions for integral backstepping control tuning

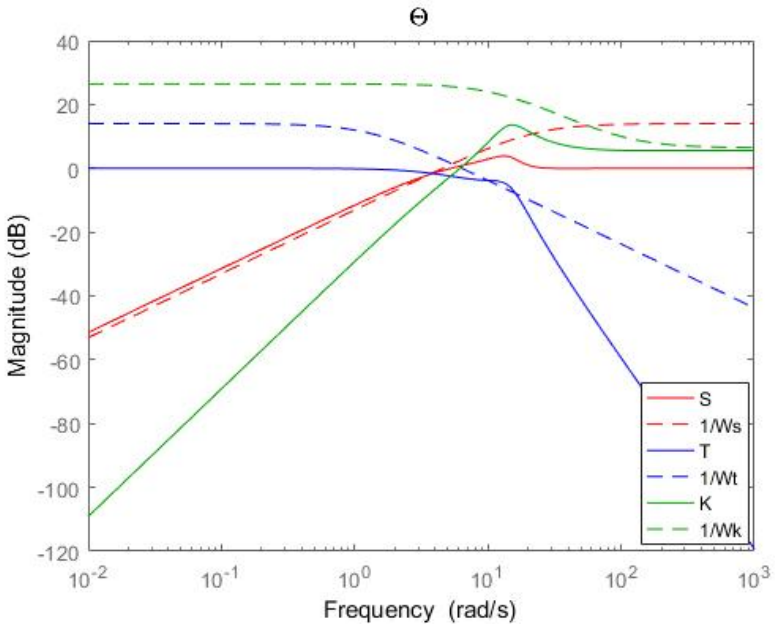


Figure 3.11: Results of the H_∞ tuning for the integral backstepping controller: pitch axis

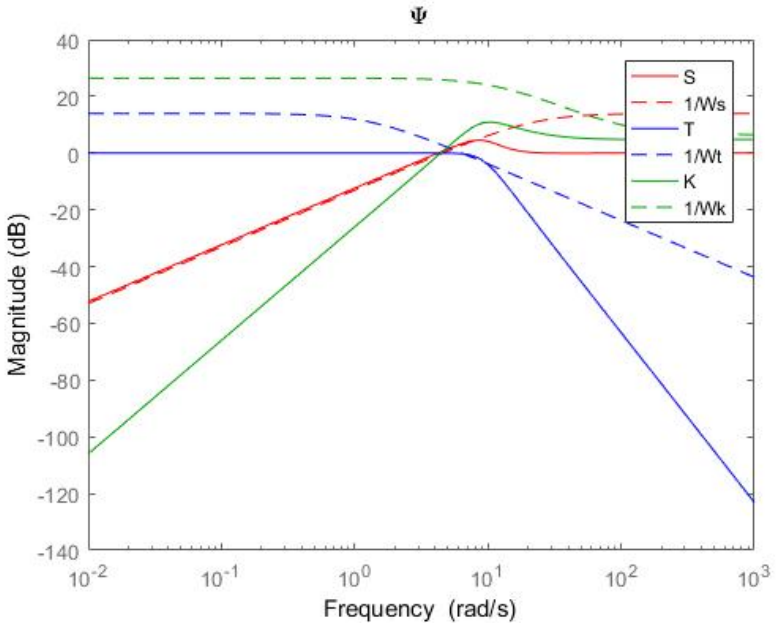


Figure 3.12: Results of the H_∞ tuning for the integral backstepping controller: yaw axis

result the value of the gains increases.

3.5 Second formulation of integral backstepping

At the beginning of the chapter the assumption $E \approx I$ has been introduced, that allows to consider the effect of the body angular rate affecting only one coordinate in NED reference frame.
This strong assumption limits the effectiveness of the control to certain range of angles and can lead to unstable behaviours of linear controllers in the case of very aggressive manoeuvres. In order to get a better control action, the previous assumption has been dropped and a new controller based the integral backstepping technique has been developed.

3.5.1 Derivation of the attitude control law

The idea behind this modification of the integral backstepping is quite simple, the small angle approximation that lead to $E \approx I$ is dropped, and so while controlling the attitude error dynamics the control action is considered on the body angular rate ω_b and not directly on the attitude angles rates.

The first steps and the derivation is analogous to the integral backstepping one shown in equations (3.48) and (3.49), from those expressions the virtual control Φ_V is computed

$$\dot{\Phi}_V = \dot{\Phi}_{des} + \lambda_\Phi E_\Phi + \alpha_1 e_\Phi. \tag{3.57}$$

Now the convergence of the extended Lyapunov function has to be studied. As in equation (3.50) the error on the angular rate has to be defined:

$$\begin{aligned} e_{\dot{\Phi}} &= \dot{\Phi}_V - \dot{\Phi} = \dot{\Phi}_{des} + \lambda_\Phi E_\Phi + \alpha_1 e_\Phi - \dot{\Phi} \\ V_\Phi(e_\Phi, e_{\dot{\Phi}}, E_\Phi) &= \frac{1}{2}(e_\Phi^2 + e_{\dot{\Phi}}^2 + \lambda_\Phi E_\Phi^2) \\ \dot{V}_\Phi(e_\Phi, e_{\dot{\Phi}}, E_\Phi) &= e_\Phi \dot{e}_\Phi + \lambda_\Phi E_\Phi e_\Phi + e_{\dot{\Phi}} \dot{e}_{\dot{\Phi}} \end{aligned} \tag{3.58}$$

and after some computation the expression of $\dot{V}_\Phi$ can be obtained in the form

$$\dot{V}_\Phi = -\alpha_1 e_\Phi^2 + e_{\dot{\Phi}} \left(e_\Phi + \lambda_\Phi e_\Phi - \lambda_\Phi \alpha_1 E_\Phi - \alpha_1^2 e_\Phi + \alpha_1 e_{\dot{\Phi}} - \ddot{\Phi} \right). \tag{3.59}$$

The expression of $\ddot{\Phi}$ can be substituted taking into account that

$$\begin{aligned} \begin{bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{bmatrix} &= E^{-1} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \\ \begin{bmatrix} \ddot{\Phi} \\ \ddot{\Theta} \\ \ddot{\Psi} \end{bmatrix} &= \dot{E}^{-1} \begin{bmatrix} p \\ q \\ r \end{bmatrix} + E^{-1} \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} \end{aligned} \tag{3.60}$$

and imposing $\dot{V}_\Phi = -\alpha_1 e_\Phi^2 - \alpha_2 e_{\dot{\Phi}}^2$ it is possible to work out the following equation in three unknowns

$$e_\Phi + \lambda_\Phi e_\Phi - \lambda_\Phi \alpha_1 E_\Phi - \alpha_1^2 e_\Phi + \alpha_1 e_{\dot{\Phi}} + \alpha_2 e_{\dot{\Phi}} = \dot{E}^{-1}_{(1,:)} \begin{bmatrix} p \\ q \\ r \end{bmatrix} + E^{-1}_{(1,:)} \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix}, \tag{3.61}$$

where $\dot{E}^{-1}_{(1,:)}, E^{-1}_{(1,:)}$ are the first rows of the two matrices. It is quite clear that in this case the control action can be computed only taking into account all the three coordinates, intuitively because this algorithm takes into account also the coupling between the axes. For all the other coordinates the computation until this point are the same and the resulting system can take the form

$$\begin{aligned} E^{-1} \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} &= K_1 \begin{bmatrix} E_\Phi \\ E_\Theta \\ E_\Psi \end{bmatrix} + K_2 \begin{bmatrix} e_\Phi \\ e_\Theta \\ e_\Psi \end{bmatrix} + K_3 \begin{bmatrix} \dot{e}_\Phi \\ \dot{e}_\Theta \\ \dot{e}_\Psi \end{bmatrix} - \dot{E}^{-1} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \\ K_1 &= diag(\lambda_\Phi \alpha_1, \lambda_\Theta \alpha_3, \lambda_\Psi \alpha_5); \\ K_2 &= diag((1 + \lambda_\Phi - \alpha_1^2), (1 + \lambda_\Theta - \alpha_3^2), (1 + \lambda_\Psi - \alpha_5^2)); \\ K_1 &= diag((\alpha_1 + \alpha_2), (\alpha_3 + \alpha_4), (\alpha_5 + \alpha_6)); \end{aligned} \tag{3.62}$$

Table 3.6: Second formulation of integral backstepping, control parameters

α_1	7
α_2	12
λ_Φ	8
α_3	7
α_4	12
λ_Θ	8
α_5	10
α_6	13
λ_Φ	5

If the dynamics of p, q, r from (3.18) are substituted in (3.62) the moments can be extracted and the resulting control takes the form:

$$\begin{bmatrix} L \\ M \\ N \end{bmatrix} = E\left[K_1 E_\alpha + K_2 e_\alpha + K_3 e_{\dot{\alpha}}\right] - \left[E\dot{E}^{-1} + D_{stab}\right] \begin{bmatrix} p \\ q \\ r \end{bmatrix} - G \tag{3.63}$$

With

$$D_{stab} = \begin{bmatrix} \frac{\partial L}{\partial p} & 0 & 0 \\ 0 & \frac{\partial M}{\partial q} & 0 \\ 0 & 0 & \frac{\partial N}{\partial r} \end{bmatrix} G = \begin{bmatrix} I_{xx}^{-1}\left[(I_{yy} - I_{zz})qr\right] \\ I_{yy}^{-1}\left[(I_{zz} - I_{xx})pr\right] \\ I_{zz}^{-1}\left[(I_{xx} - I_{yy})pq\right] \end{bmatrix}. \tag{3.64}$$

It can be noticed that the structure of the control variable is consistent with what can be expected from a control law that takes into account the rotation between body and inertial frame: the control action computed in inertial reference frame is mapped in body frame while the gyroscopic effects and the stability derivatives are directly compensated.
The H_∞ tuning technique has not been adopted for the tuning of this control since the high nonlinearities introduced makes it ineffective. In this case the parameter have been chosen via subsequent refinements trying to get the best result. The chosen values are shown in Table 3.6.

This kind of control, even if it is mathematically more adherent to the nonlinear quadrotor model, nevertheless is subject to the limitations imposed by the Euler angles parametrization, thus it is infeasible when $\Theta = \pm 90°$. In order to avoid the singularities intrinsic in the angle parametrization a geometric control architecture has been studied, both for the position and the attitude control.

3.6 Geometric control

In the previous developments the Euler angle parametrization of the special orthogonal group has been extensively adopted. This parametrization allows a

more intuitive design of the controller, but is prone to singularity: every minimal parametrization is never further away than 90° from a singular configuration. In order to avoid this singularities and synthesize a controller that can manage any orientation, the parametrization of the attitude can be dropped and the design can be carried out working in the Special Orthogonal Group $SO(3)$ with the rotation matrix R. At the price of a quite mathematically intensive development, the resulting control law has a simple expression and it is computationally tractable while offering a large operating region..
This kind of control has been previously developed in [7], [6] and [5]. The control law has been derived from [5] by assuming that the thrust vectoring capability is locked, in which case the co-planar UAVs configuration is recovered.

3.6.1 Model and definition of the errors

The starting point of the control design is an enlarged system with respect to the one presented in (2.44) that include also the dynamic of the rotation matrix R. The resulting system represented in inertial frame with the centre of mass coincident with the origin of the body frame is then

$$\begin{aligned} \dot{x} &= v \\ \dot{R} &= R\hat{\omega} \\ m\dot{v} &= -mge_3 + RF_{props} \\ I_n\dot{\omega} &= -\omega \times (I_n\omega) + M_{props}. \end{aligned} \tag{3.65}$$

Where

$$v = RV_b \in \mathbb{R}^3 \tag{3.66}$$

represents the inertial velocity. In this way the translational model evolves in the inertial frame, whereas the rotational motion in the body frame. This choice breaks the Special Euclidean group $SE(3)$ structure but it allows to obtain a simpler expression of the final controller.
If a smooth tracking command $G_d(t) = \big(x_d(t), R_d(t)\big) \in \text{SE(3)}$ is assigned as a function of time, the corresponding desired velocity command is computed as $\xi_d(t) = \big(v_d, \omega_d\big)$ with $\omega_d = (R_d^T\dot{R}_d)^\vee$. Where $[.]^\vee$ is the mapping from SO(3) to $\mathbb{R}^3$ while v_d and ω_d are the desired linear and angular velocities namely. From this set-points and the (3.65) it is possible to define the linear position and velocity error in the inertial frame as:

$$\begin{aligned} e_x &= x - x_d \\ e_v &= v - v_d \end{aligned} \tag{3.67}$$

The attitude error is defined as the left error, that allows to get a simpler control expression

$$R_e = RR_d^T \in SO(3) \tag{3.68}$$

This expression of the left attitude error represents the transport map τ_l that is used to compare the desired tangent vector $\dot{R}_d$ in the tangent space of the current orientation, that is:

$$\dot{R} - \tau_l\big(\dot{R}_d\big) = \dot{R} - R_e\dot{R}_d = R(\hat{\omega} - \hat{\omega}_d) \tag{3.69}$$

and the expression of the left velocity error is straightforwardly derived as

$$e_\omega = \omega - \omega_d \tag{3.70}$$

Since the desired angular velocity ω_d in the non-tilting quadrotor case is directly computed by the controller and not imposed by the user, from now on it will be referred as ω_c, keeping the notation used in [5] . For the same reason, also R_d is denoted as R_c from now on , since only the yaw angle can be set. Finally, the navigation error function Ψ has been defined as

$$\begin{aligned} \Psi &= \frac{1}{2}\big(K_R(I - R_e)\big) \\ K_R &= diag(k_{R1}, k_{R2}, k_{R3}) \end{aligned} \tag{3.71}$$

And the left trivialised derivative of Ψ can be expressed as

$$T_I^* L_{R_e}(d_{R_e}\Psi) = skew(K_R R_e)^\vee = e_R. \tag{3.72}$$

Once the attitude tracking errors are defined, it is possible to set up a control architecture for the position and attitude dynamics.

3.6.2 Control law

The position control law computes the forces along each axis in order to reach the desired position. Of course in the non-titling UAV case, the force can be only directed in the opposite direction of the z body axis e_3. The main idea is to exploit the fully actuated rotational motion in order to orient the thrust axis direction along the direction of the force required to track the position trajectory. This is the only feasible approach due to the inherent underactuation of co-planar platforms [9]. Therefore, it is not possible to follow a decoupled attitude and position motion but only the rotation around the third body axis can be freely assigned, as shown next. In formulas

$$\begin{aligned} f_c &= c(\Psi)||f_c^d||e_3 \\ f_c^d &= -K_x e_x - K_v e_v - mge_3 + m\dot{v}_d \\ c(\Psi) &= \frac{\Psi_{Max} - \Psi}{\Psi_{Max}} \end{aligned} \tag{3.73}$$

Where K_x, K_v are diagonal matrices to be tuned and a feed-forward compensation of the weight and inertial forces can be noticed. $c(\Psi)$ is a weighting function for

the required control action, it is equal to one when the quardotor is correctly oriented and thus the thrust generated by the propellers pushes the UAV in the correct direction, while when the orientation is different from the desired one, the attitude control is preferred and so the thrust is scaled in order to increase the position error pushing the quadrotor along a wrong direction.
Once defined the desired force, it can also be computed a desired attitude in order to have the thrust directed in the right direction: the z body axis has to be in the same direction of the desired force, while the x and y ones should be consistently oriented in order to get the desired yaw angle ψ_d. The R_c matrix is then built up assembling all the column vectors created from these requirements:

$$b_d = \begin{bmatrix} \cos\psi_d \\ \sin\psi_d \\ 0 \end{bmatrix} \tag{3.74}$$

$$R_c = \begin{bmatrix} b_{c1} & b_{c2} & b_{c3} \end{bmatrix} \tag{3.75}$$

$$b_{c3} = \frac{f_c^d}{||f_c^d||} \tag{3.76}$$

$$b_{c2} = \frac{b_{c3} \times b_d}{||b_{c3} \times b_d||} \tag{3.77}$$

$$b_{c1} = b_{c2} \times b_{c3} \tag{3.78}$$

The desired orientation of the UAV can be fed to an attitude controller in order to compute the required control torques.
It is possible to consider the attitude control law as a PD controller plus some feed-forward terms, since the proportional action is included in the computation of e_R [2]

$$\tau_c = -R_c^T e_R - K_\omega e_\omega + I_n \dot{\omega}_c + \omega_c \times (I_n \omega). \tag{3.79}$$

In this case the feed-forward terms have a not-negligible weight and have to be computed in real time by the controller since the matrix R_c is computed on board. As stated before

$$\begin{aligned} \omega_c &= (R_c^T \dot{R}_c)^\vee \\ \dot{\omega}_c &= (R_c^T \ddot{R}_c - \hat{\omega}_c)^\vee \end{aligned} \tag{3.80}$$

so the first and second derivative of the computed desired attitude have to be worked out

$$\dot{R}_c = \begin{bmatrix} \dot{b}_{c1} & \dot{b}_{c2} & \dot{b}_{c3} \end{bmatrix}. \tag{3.81}$$

The three columns are the derivatives of the previously defined vectors and so an analytical expression for each of them can be obtained by derivation. The first term to be computed is the third column as before, where the effect of the control

action affects the value of the derived vector

$$\begin{aligned} \dot{b}_{c3} &= \frac{\dot{f}_c^d}{||f_c^d||} - (f_c^{d,T} \times \dot{f}_c^d)\frac{f_c^d}{||f_c^d||^3} \\ \dot{f}_c^d &= \frac{K_x K_v}{m} e_x + \left(\frac{K_v^2}{m} - K_x\right) e_v - \frac{K_v}{m} D_f + m\ddot{v}_d \\ D_f &= R f_c - f_c^d \end{aligned} \tag{3.82}$$

From the third vector the second column can be computed recalling (3.77)

$$\begin{aligned} \dot{b}_{c2} &= \frac{\dot{c}}{||c||} - (c^T \times \dot{c})\frac{c}{||c||^3} \\ c &= b_{c3} \times b_d \\ \dot{c} &= \dot{b}_{c3} \times b_d + b_{c3} \times \dot{b}_d \\ \dot{b}_d &= \begin{bmatrix} -\dot{\psi}_d \sin\psi_d \\ \dot{\psi}_d \cos\psi_d \\ 0 \end{bmatrix} \end{aligned} \tag{3.83}$$

and as last step deriving (3.78)

$$\dot{b}_{c1} = \dot{b}_{c2} \times b_{c3} + b_{c2} \times \dot{b}_{c3}. \tag{3.84}$$

The second derivative of the rotation matrix is used for the computation of the inertial effects compensation and its the mathematical development follows the same step just shown

$$\ddot{R}_c = \begin{bmatrix} \ddot{b}_{c1} & \ddot{b}_{c2} & \ddot{b}_{c3} \end{bmatrix} \tag{3.85}$$

$$\ddot{b}_{c3} = \frac{\ddot{f}_c^d}{||f_c^d||} - 2(f_c^{d,T}\dot{f}_c^d)\frac{\dot{f}_c^d}{||f_c^d||^3} - (||\dot{f}_c^d||^2 + (f_c^{d,T}\ddot{f}_c^d))\frac{f_c^d}{||f_c^d||^3} + 3(f_c^{d,T}\ddot{f}_c^d)^2\frac{f_c^d}{||f_c^d||^5} \tag{3.86}$$

$$\ddot{b}_{c2} = \frac{\ddot{c}}{||c||} - 2(c^T\dot{c})\frac{\dot{c}}{||c||^3} + 3(c^T\ddot{c})^2\frac{c}{||c||^5} \tag{3.87}$$

$$\ddot{b}_{c1} = \ddot{b}_{c2} \times b_{c3} + 2(\dot{b}_{c2} \times \dot{b}_{c3}) + b_{c2} \times \ddot{b}_{c3}. \tag{3.88}$$

Where

$$\ddot{f}_c^d = \frac{K_x K_v}{m} e_v + \left(\frac{K_v^2}{m} - K_x\right)\left(-\frac{K_x}{m} e_x - \frac{K_v}{m} e_v + \frac{D_f}{m}\right) - \frac{K_v}{m}\dot{D}_f \tag{3.89}$$

$$\dot{D}_f = c(\Psi) R_e (e_\omega)^\wedge f_c^d + c(\Psi) R_e (R_c e_\omega)^\wedge f_c^d - e_R R_c \frac{e_\omega}{\Psi_{Max}} R_e f_c^d + c(\Psi) R_e \dot{f}_c^d - \dot{f}_c^d \tag{3.90}$$

$$\ddot{c} = \ddot{b}_{c3} \times b_d + 2(\dot{b}_{c3} \times \dot{b}_d) + b_{c3} \times \ddot{b}_d \tag{3.91}$$

$$\tag{3.92}$$

Table 3.7: Position control parameters, geometric control law

K_x	K_y	K_z	$K_{v,x}$	$K_{v,y}$	$K_{v,z}$
3	3	3	2	2	2

Table 3.8: Attitude control parameters, geometric control law

K_{R1}	K_{R2}	K_{R3}	K_{w1}	K_{w2}	K_{w3}
1	1	1	0.1	0.1	0.1

Using this control approach it is also possible to compute and an estimate for the region of attraction of the equilibrium $(e_x, e_v, R_e, e_\omega) = (0, 0, I, 0)$, see [5].
The tuning of this control law has been tackled manually by a trial and error procedure because it has been impossible to set an optimization problem like the ones seen before. For the sake of simplicity, the diagonal terms of each control parameters matrix have been assumed to be equal and the resulting control tuning is shown in Tables 3.7, 3.8 setting the maximum value of $\Psi_{max} = 20$.

Chapter 4

Simulation results

In this chapter the simulated results obtained using the previously developed nonlinear control laws are presented.
A first introduction to the Simulink simulator built in the FLYART laboratory is provided at the beginning. Then all the presented control laws are tested.
Each simulation campaign consists in a test on the attitude loop only and then in the cascade of the attitude control with the position control. A comparison between the results of the different architectures is proposed that highlights the results achieved by each control law.
Since it is not possible to define a cutting frequency for a nonlinear system, the attitude control tracking capabilities have been tested on a step response in first issue and then on a sinusoidal setpoint.
In a similar way, the position tracking has been validated using an 8-shape trajectory.

4.1 UAV simulator

The simulator adopted is the one described in [4] with some minor modifications and implemented using the Simulink environment as shown in Figure 4.1.

4.1.1 Quadrotor and sensors blocks

The main block of the simulator is of course the quadrotor model. It is based on the equations of (2.44) plus some additions that can be seen in Figure 4.2 . In fact it has been modified in order to consider the fact that in the real world there is a lower limit, the Earth surface, that prevents the UAV to fall at time zero, when the actuators are not delivering enough thrust in order to keep it in flight.
The green blocks in Figure 4.1 are the sensors models. Since there is no information on the sensors dynamics and noises, they are modelled as simple delays of $0.01[s]$ in the transmission of the state signals from the quadrotor to the control

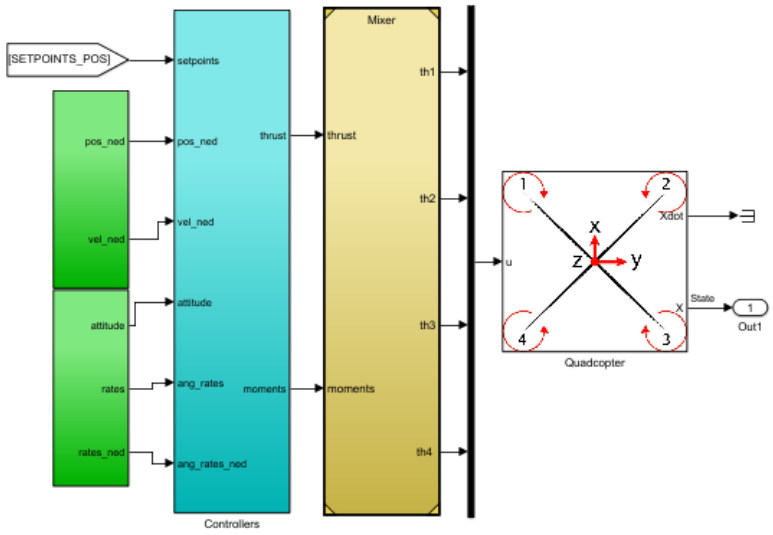


Figure 4.1: UAV simulator implemented in Simulink

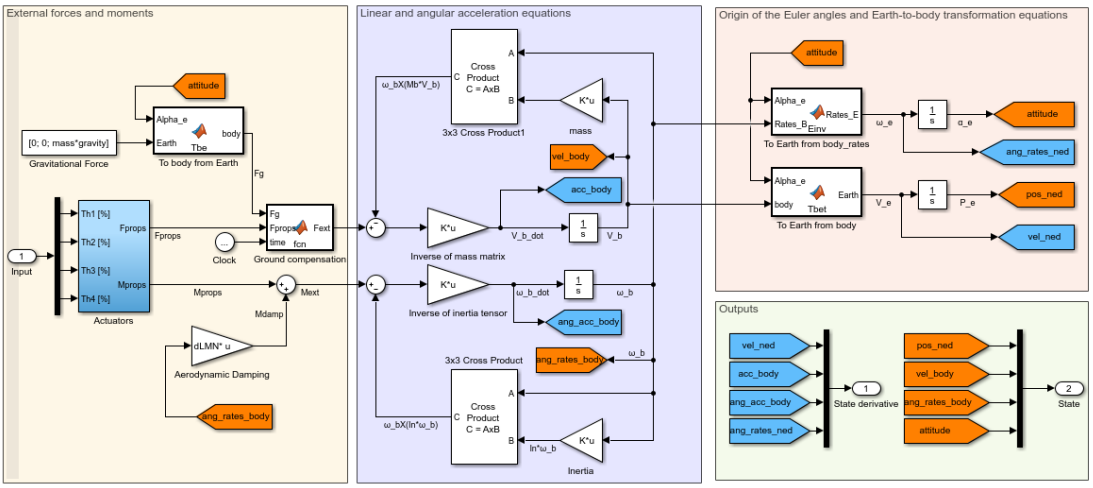


Figure 4.2: Simulator of the UAV dynamics

system.
The measured signals are sent to the controllers that computes the required thrust and moments in order to follow the desired trajectory. These control signals are sent to the *Mixer* that inverting the relation derived in (2.37) computes the desired angular rates for the actuators.

4.1.2 Actuators model

The brushless DC motors dynamics has been identified in [4].
It forces a limit on the performance of the control system and have to be considered both in tuning and in validation phases. The identified model is a first order low pass filter that relates the throttle computed by the FCU using the mixer and the obtained rotational speed. The transfer function is then

$$G(s) = \frac{\Omega(s)}{Th(s)} = \frac{5.2}{1 + s(92 \times 10^{-3})}. \tag{4.1}$$

In Figure 4.3 it is possible to see that the actuators bandwidth is approximately equal to $55.5[rad/s]$.

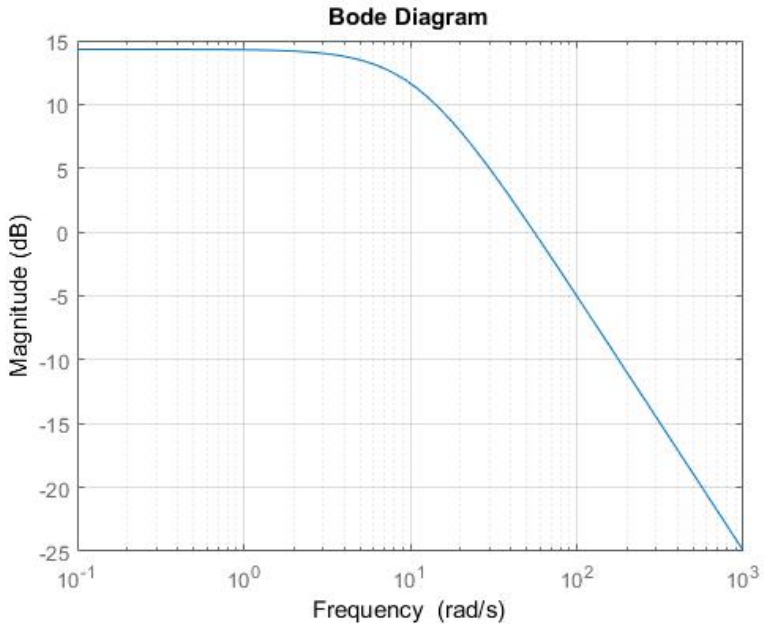


Figure 4.3: Bode magnitude plot of the frequency response of the actuators transfer function

4.2 Backstepping simulations

4.2.1 Attitude control simulation

The simulations for the attitude control law have been carried out closing the attitude loop only. In order to do this a constant thrust of magnitude equal to the UAV weight has been imposed to the actuators and the controller has been in charge of the required moments computation.
The result of the first simulation in shown in Figure 4.4 , a step of 30° on the pitch angle has been required while roll and yaw desired values are set to zero. It is possible to see that the system reaches 90% of the desired value in about $0.5[s]$ and then to the final value in about $1[s]$. This result is consistent with the tuning since the required bandwidth for the system was in the range $(0.48\text{-}1.1)Hz$.
The same simulation settings can be used to test the yaw control law. The related results are proposed in Figure 4.5 and the resulting dynamic is slower than the pitch one since the inertia around the Z axis is bigger than the other two.

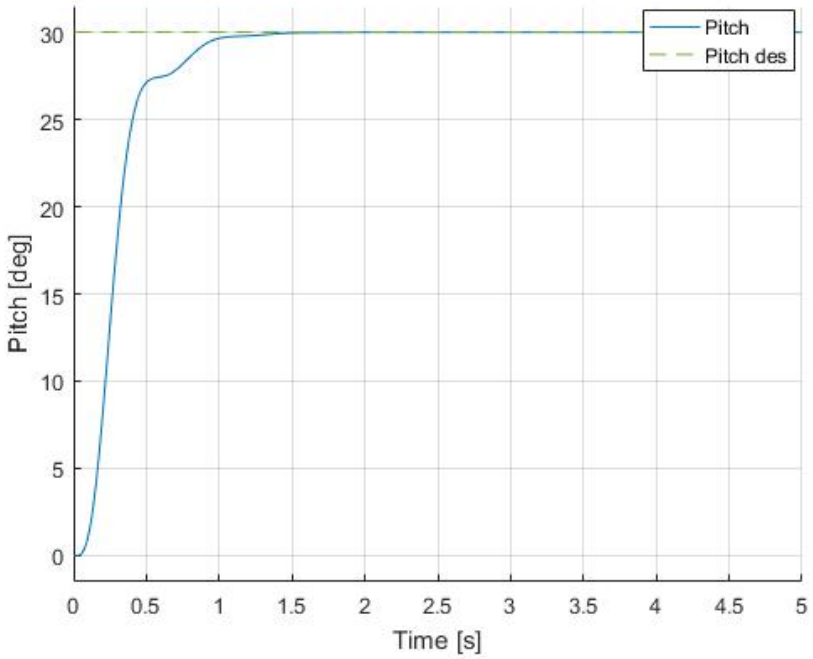


Figure 4.4: Response of the UAV to a 30° step on the desired pitch, backstepping

A second simulation campaign has been carried out applying a sinusoidal input with 30° amplitude at $2[rad/s]$ frequency.
In Figures 4.6 and 4.7 it is possible to see the response of the system. It is clear that there is a delay of about $0.2[sec]$ that strongly affects the performance of the system due to the delay in sensors and the dynamics of the actuators.

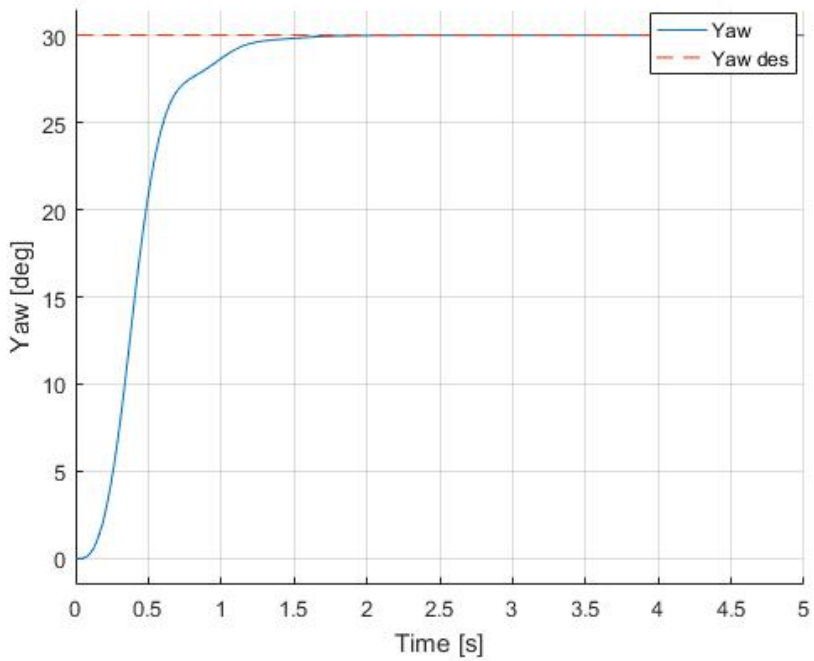


Figure 4.5: Response of the UAV to a 30° step on the desired yaw, backstepping

In Figure 4.8 the errors in yaw and pitch tracking related to the presented simulations are shown. Since the frequency of the sine wave is at $2[rad/s]$ the result is consistent with the model because the linearised system has been tuned for a cut off frequency of about $3[rad/s]$ thus the attenuation of the signal in the actual system starts a bit before.

Last simulation in figure 4.9 shows that the system has quite good rejection of the disturbances caused by the effects of the coupling between axes, excited requiring a $2[rad/s]$, 10° sinusoid on both yaw and pitch. In fact both the dynamics stays stable and the error (Figure 4.10) is limited, thus it has been possible to close the outer position loop and the overall results are presented in the following. In any case the performance of the system can be improved and for this reason the implementation of an integral backstepping architecture has been studied.

4.2.2 Position control simulation

In this paragraph the results obtained cascading the attitude control law with the position backstepping control law are shown.
In order to test effectively the control laws it has been decided to feed the system with an 8-shape trajectory, this kind of path has the characteristic to excite all the degree of freedom of the quadrotor while maintaining continuous derivatives.

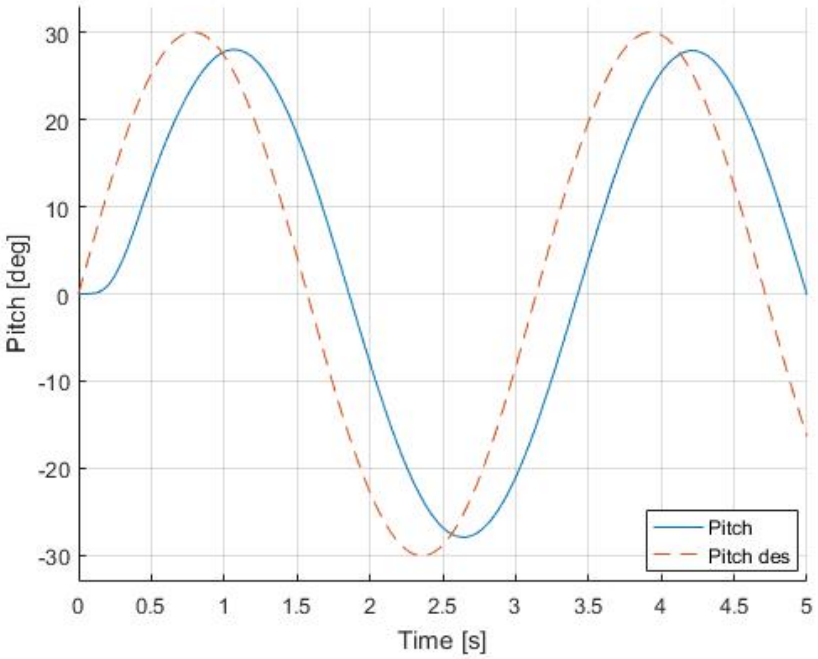


Figure 4.6: Response of the UAV to a 30° sinusoidal variation of the pitch setpoint at 2 $[rad/s]$, backstepping

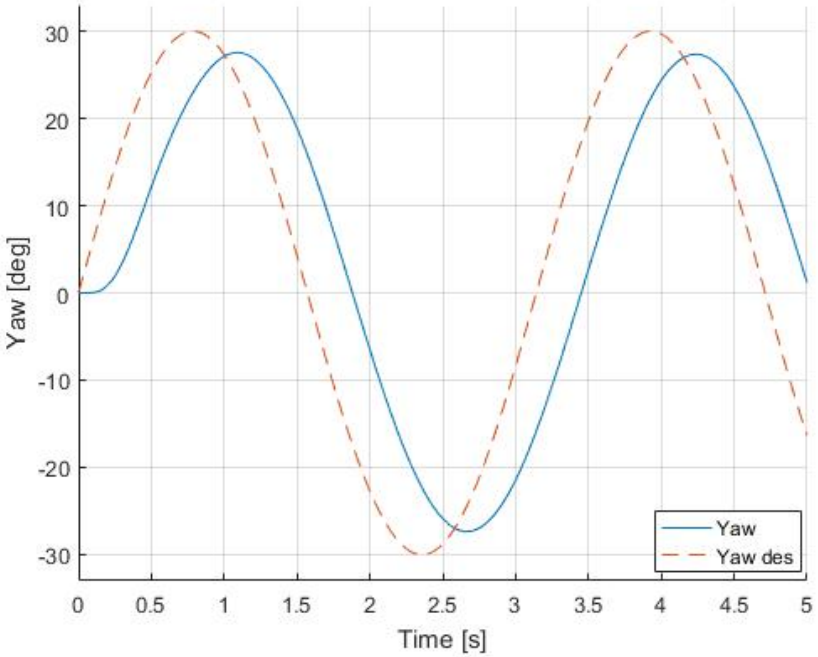


Figure 4.7: Response of the UAV to a 30° sinusoidal variation of the yaw setpoint at 2 $[rad/s]$, backstepping

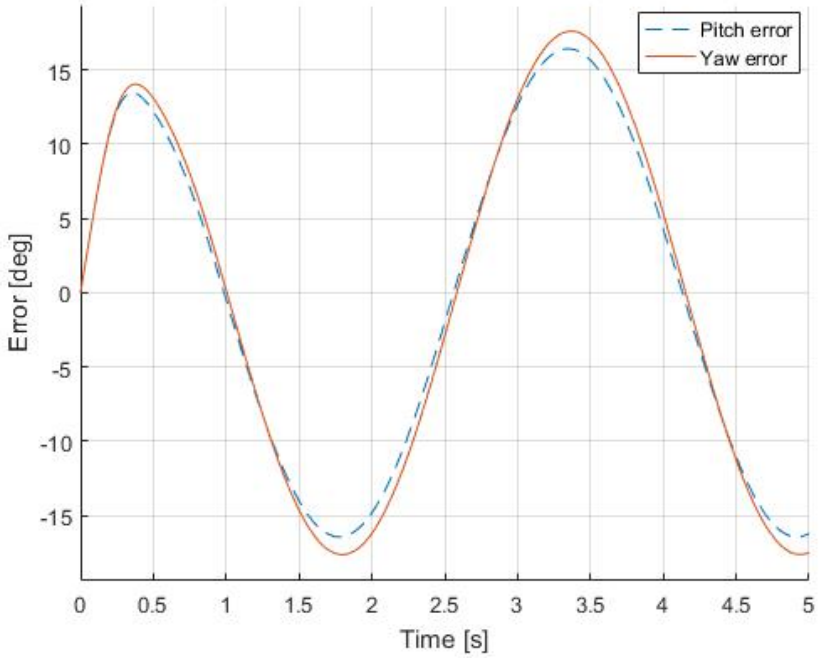


Figure 4.8: Pitch and yaw tracking errors, backstepping

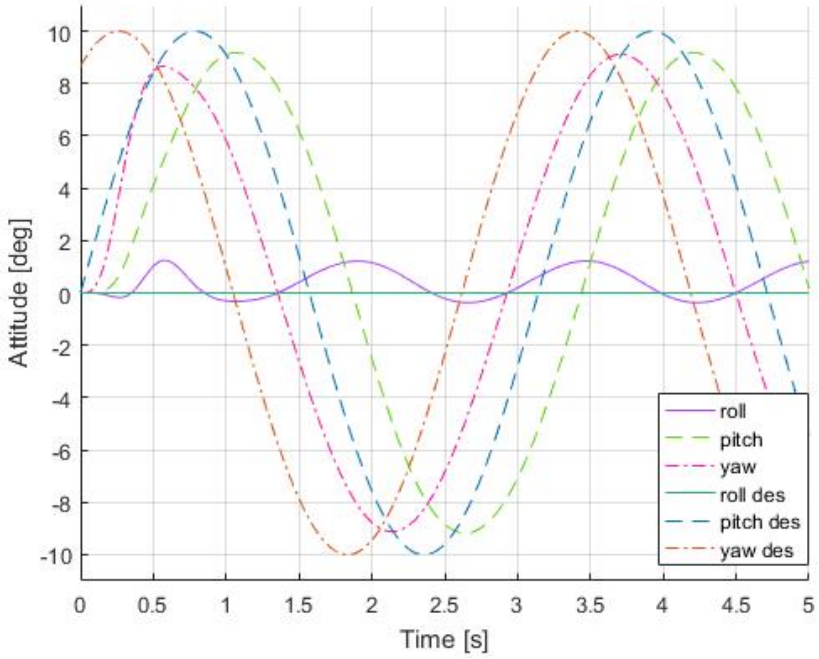


Figure 4.9: Attitude response on a sinusoidal reference on both pitch and yaw, backstepping

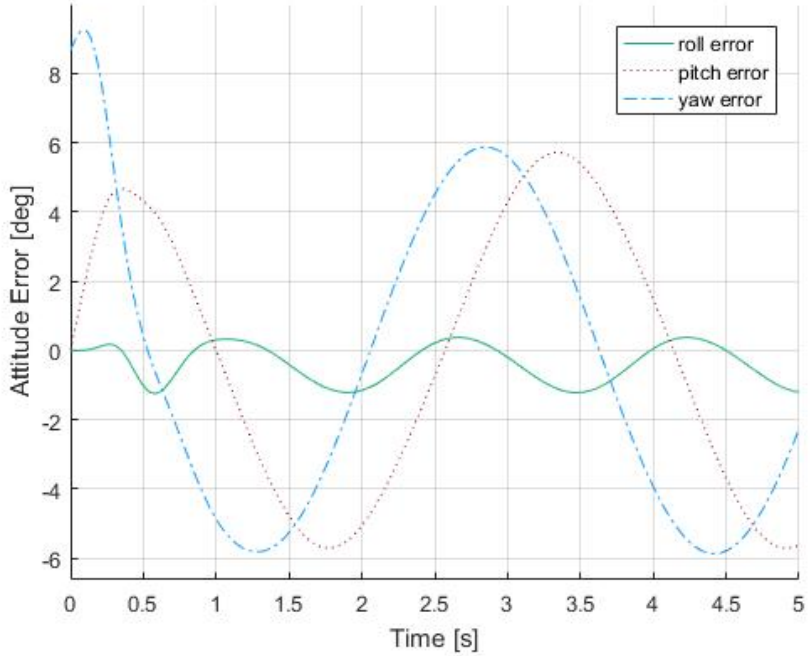


Figure 4.10: Attitude error on a sinusoidal reference on both pitch and yaw, backstepping

In figure 4.11 it is possible to see the response on the x, y axes with a frequency of the 8-shape equal to 1.35 $[rad/s]$. On those axes the results are clearly satisfactory, with small overshoots and an almost perfect tracking.

Some different considerations can be done regarding the behaviour of the z axis dynamics in Figure 4.12.

It can be noticed that the response is quite noisy, this fact is due to the fact that the required rate of the 8-trajectory push the quadrotor to its limits.
Nonetheless the control works fine as it can be seen from the plot of tracking errors in Figure 4.13.

4.3 Integral backstepping simulations

4.3.1 Attitude control law simulation

The integral backstepping control law has been developed in order to gain the robustness properties given by the integral action and to bring the steady state error to zero.
From Figure 4.14 the improvement on the step response performances of about one tenth of second can be noticed.

On the other hand the gain in robustness and the zero steady state error makes

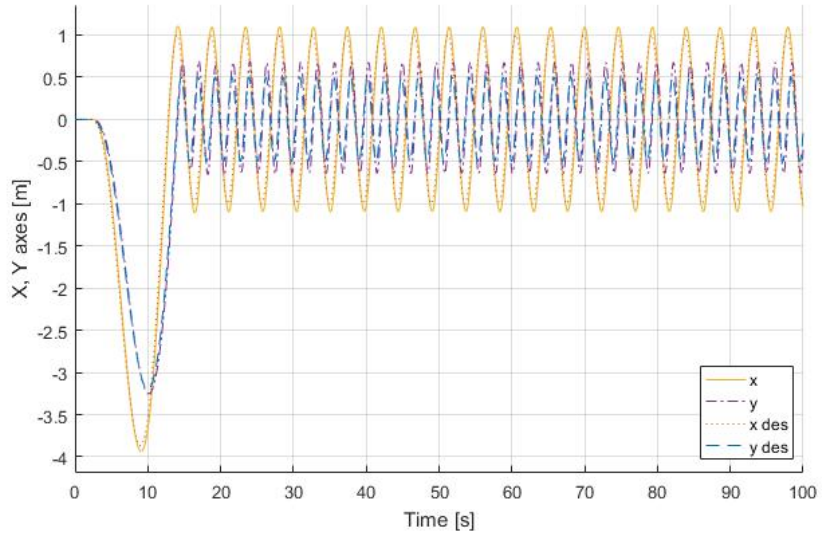


Figure 4.11: Response of the x, y coordinate to an 8-shape trajectory, backstepping

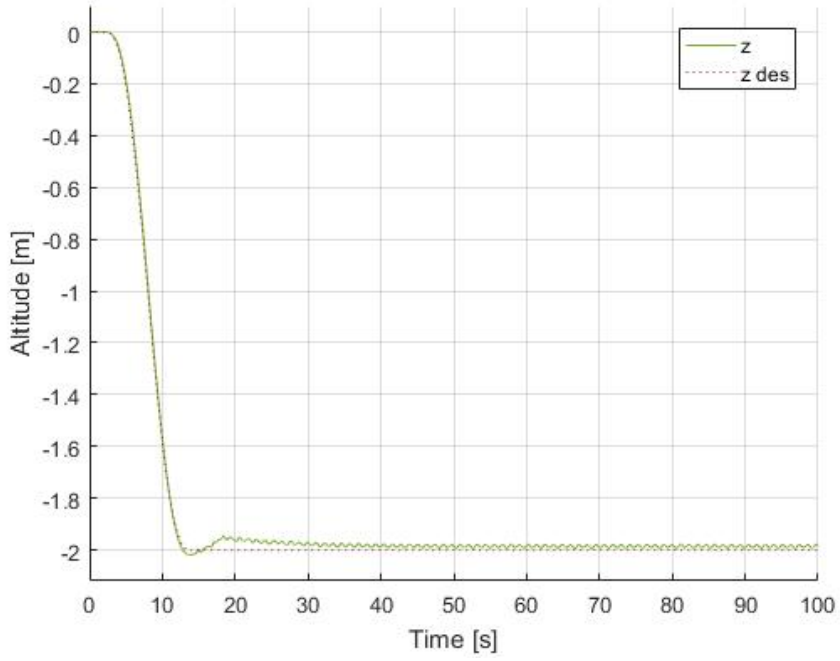


Figure 4.12: Response of the z coordinate to an 8-shape trajectory, backstepping

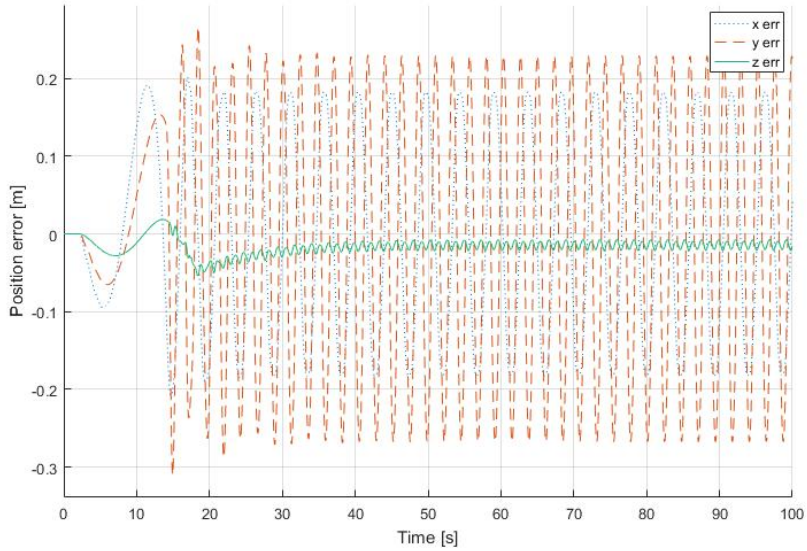


Figure 4.13: Tracking error of the response to an 8-shape trajectory, backstepping

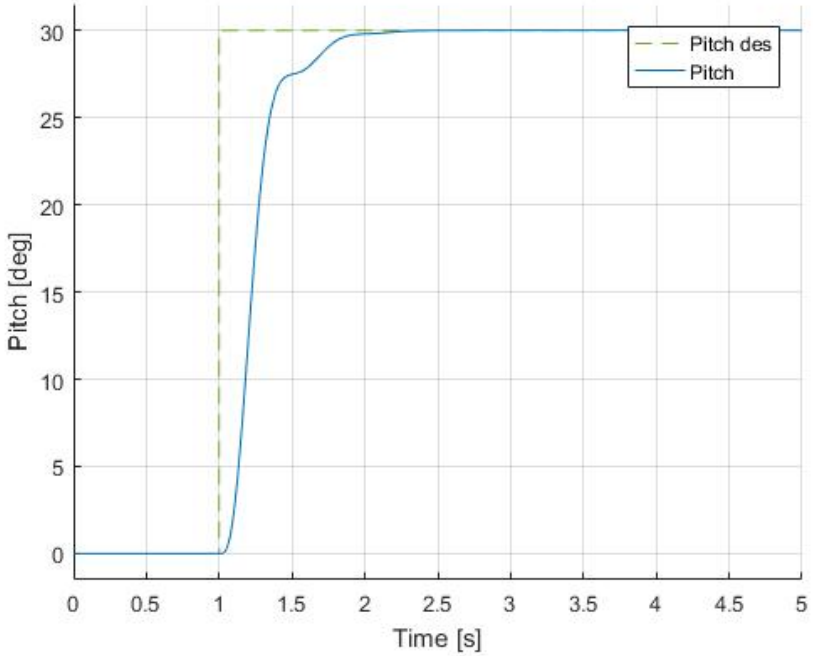


Figure 4.14: Response of the UAV to a 30° step on the desired pitch, integral backstepping

the yaw control more aggressive once tuned with the H_∞ causing an overshoot on the step response as can be seen in Figure 4.15 This sensibility of the system to step inputs suggests the choice of feeding the system with smooth setpoints.
The overshoot shown is anyway fully acceptable for the system under control since a 30° step on any coordinate is to consider aggressive and unlikely to happen.

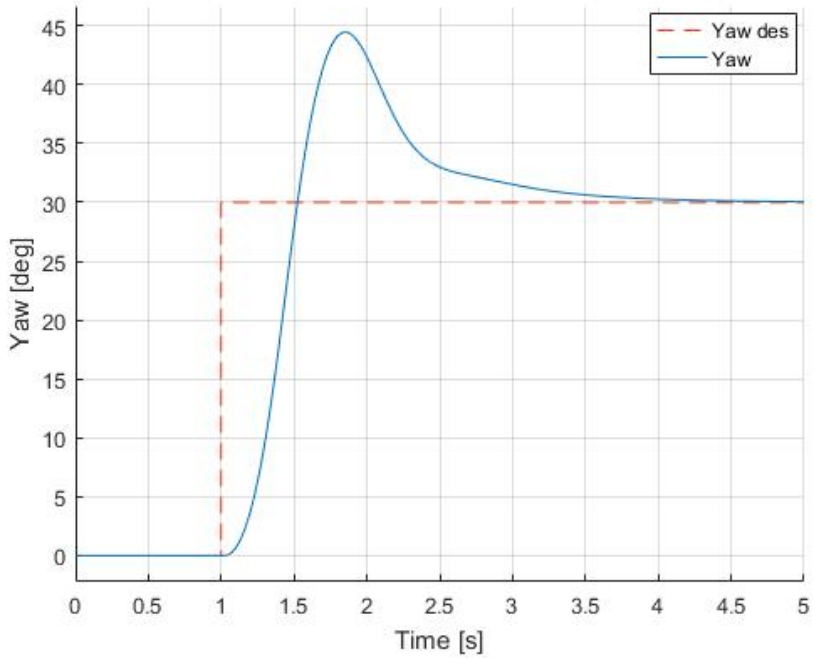


Figure 4.15: Response of the UAV to a 30° step on the desired yaw, integral backstepping

The responses to sinusoidal inputs of 30° amplitude at $2[rad/s]$ on the pitch and yaw setpoints are shown in Figure 4.16, 4.17. Consistently with the adopted tuning, the bandwidth of the linearised system is around 5 $[rad/s]$ and thus the resulting response is slightly damped. As for the step response, the yaw control law results to be quite aggressive. However, as said before, it is completely acceptable.

Analysing the behaviour of the error due to the sinusoidal input in Figure 4.18, it can be seen that the improved speed of the response and the gained robustness have as counterpart a significant increment of the error.

4.3.2 Position control law simulation

An 8-shape at 1.35 $[rad/s]$ has been fed to the system as setpoint in order to get a some comparable results between the pure backstepping control law and

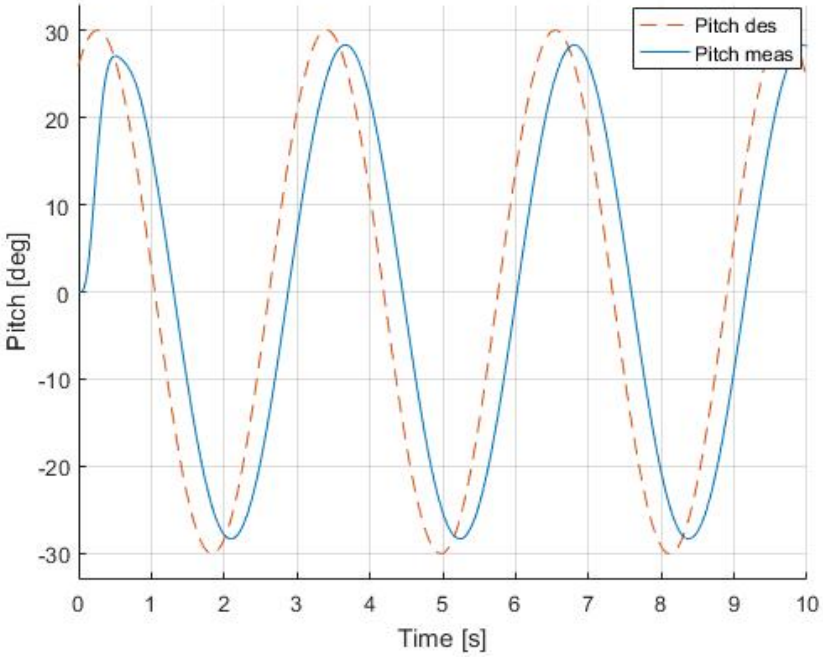


Figure 4.16: Response of the UAV to a sinusoid of amplitude 30° on the desired pitch at 2$[rad/s]$, integral backstepping

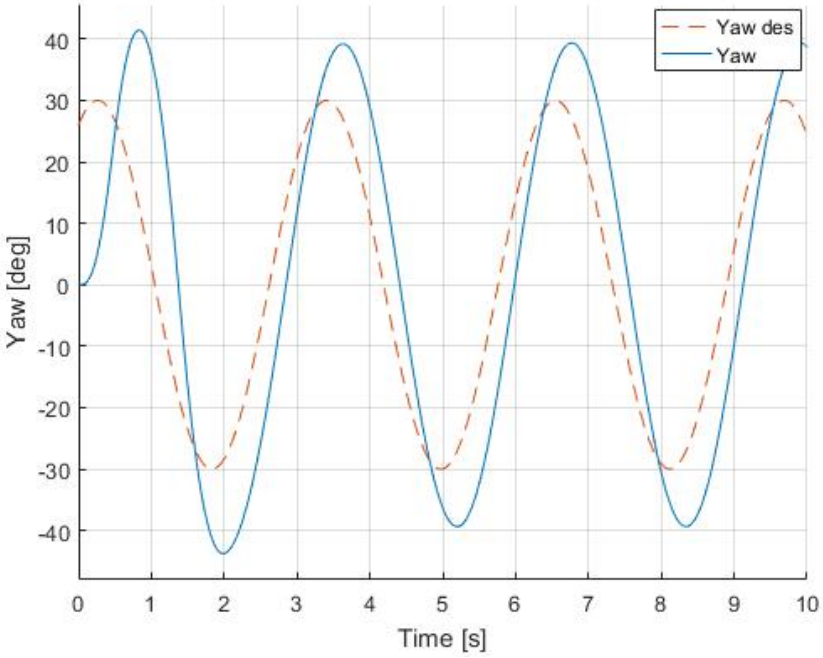


Figure 4.17: Response of the UAV to a sinusoid of amplitude 30° on the desired yaw at 2$[rad/s]$, integral backstepping

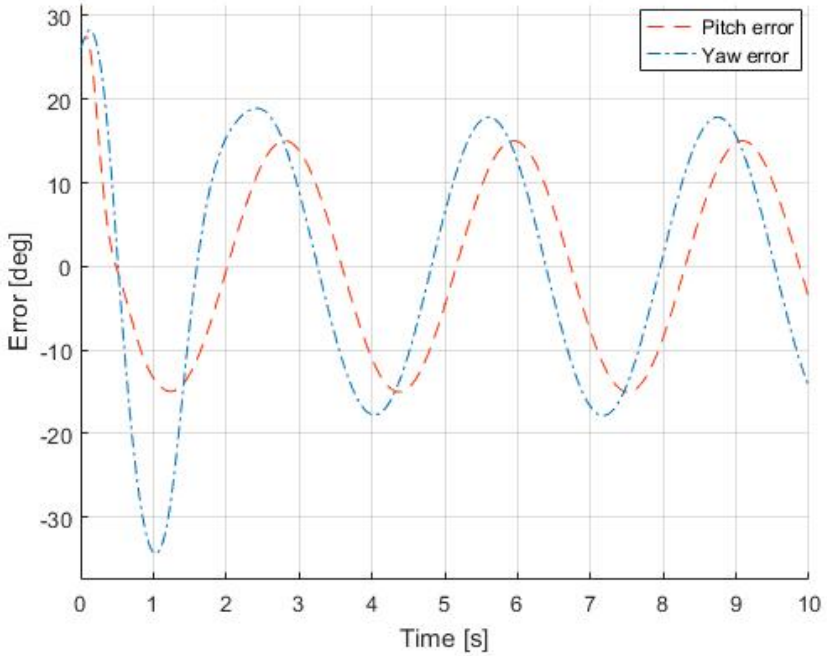


Figure 4.18: Pitch and yaw tracking errors due to a sinusoid of amplitude 30° at $2[rad/s]$ respectively on pitch and yaw, integral backstepping

the integral backstepping one. Figure 4.19 shows the obtained results and it is possible to see comparing the errors shown in Figure 4.20 with the one in 4.13 that the altitude error has been reduced. This is possible because the disturbance on the attitude can be ascribed to the effect of the coupling between the yaw and the other attitude coordinates, thus a better control action on the attitude improves also the performances of the position control loop.

4.4 Simulations of integral backstepping second formulation

4.4.1 Attitude control law simulation

The response to a step on pitch and yaw setpoints is quite satisfactory as one can see in Figure 4.21, 4.22. The system has a slower response than the other seen but it has been considered acceptable since the improvement got on the position control.

The response to a sinusoidal reference shown in Figures 4.23, 4.24 is on the contrary way better than the one obtained with the integral backstepping, with a reduction of the attitude error up to 40% as it can be noticed from Figure 4.25. This fact is a clear example of the possibilities of this control law to improve the

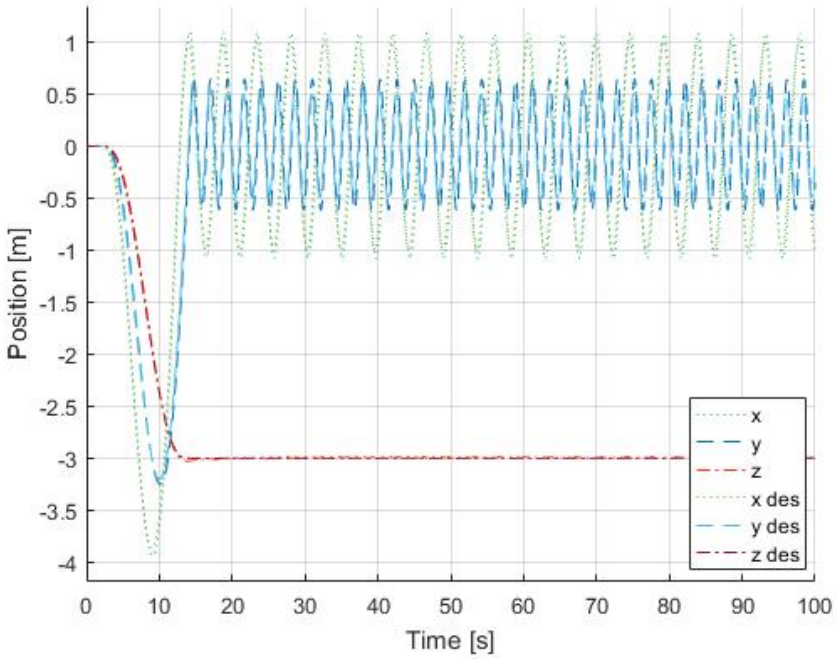


Figure 4.19: Position of the UAV quadrotor during an 8-shaped trajectory at 1.35 $[rad/s]$, integral backstepping

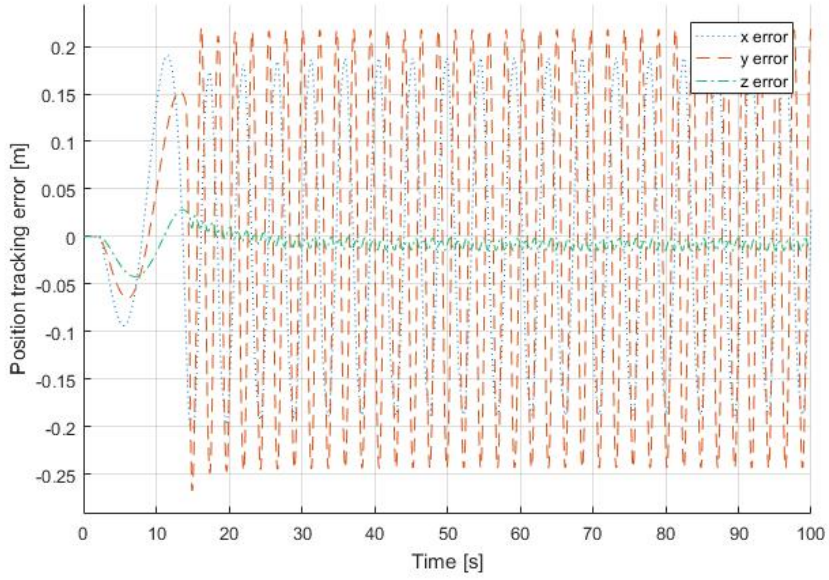


Figure 4.20: Tracking error of the response to an 8-shape trajectory, integral backstepping

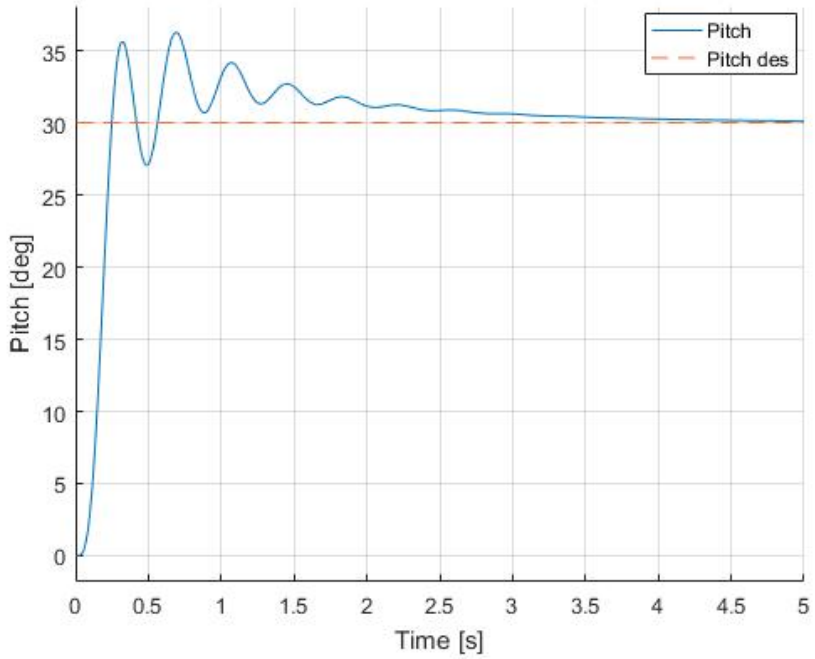


Figure 4.21: Response of the UAV to a 30° step on the desired pitch,second formulation of the integral backstepping

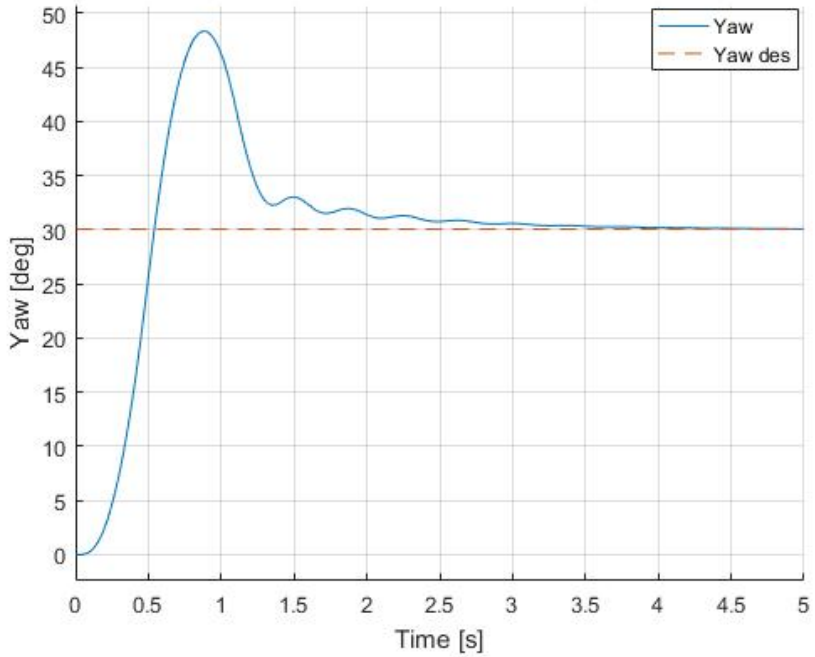


Figure 4.22: Response of the UAV to a 30° step on the desired yaw, second formulation of the integral backstepping

behaviour of the overall system.

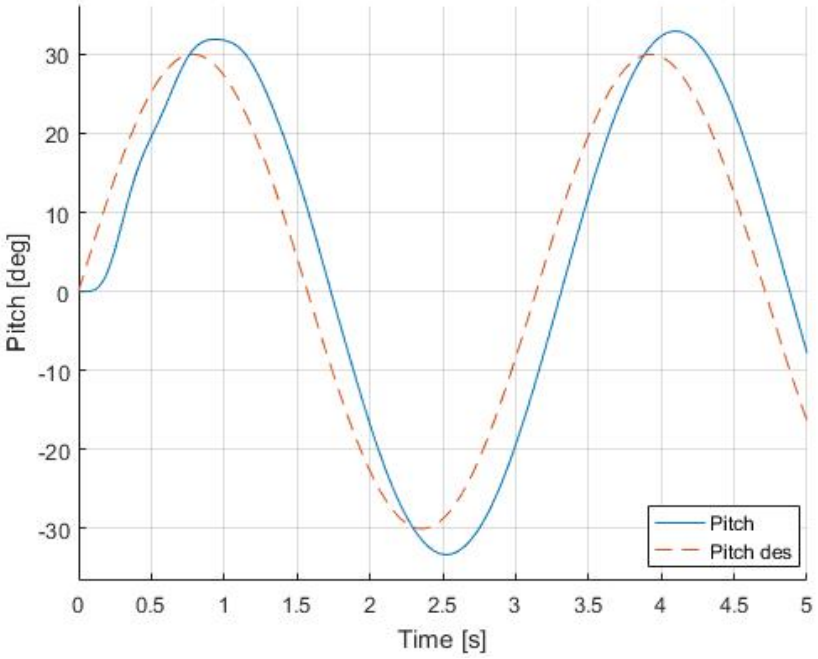


Figure 4.23: Response of the UAV to a 30° amplitude, 2 $[rad/s]$ sinusoidal wave as desired pitch, second formulation of the integral backstepping

4.4.2 Position control law simulation

The better performances of the attitude control affects also the position tracking performances. In Figure 4.26 the results of the simulation of an 8-shaped trajectory. While the overshoots are present like in all other simulations results, in this case the overall performances are better than in the previous controllers as it can be expected from this more complex control law. This is clear looking at the tracking errors plotted in Figure 4.27 where a better tracking on both y and z axes can be noticed.

4.5 Geometric control law simulations

For the geometric control law simulations it has been decided not to split the position and attitude control laws but to handle them as a whole system.
This choice is mainly caused by the formulation of the control law that does not allows a clear distinction between an inner attitude control loop and an outer position loop. For this reason the presented results refer to a simulation of the

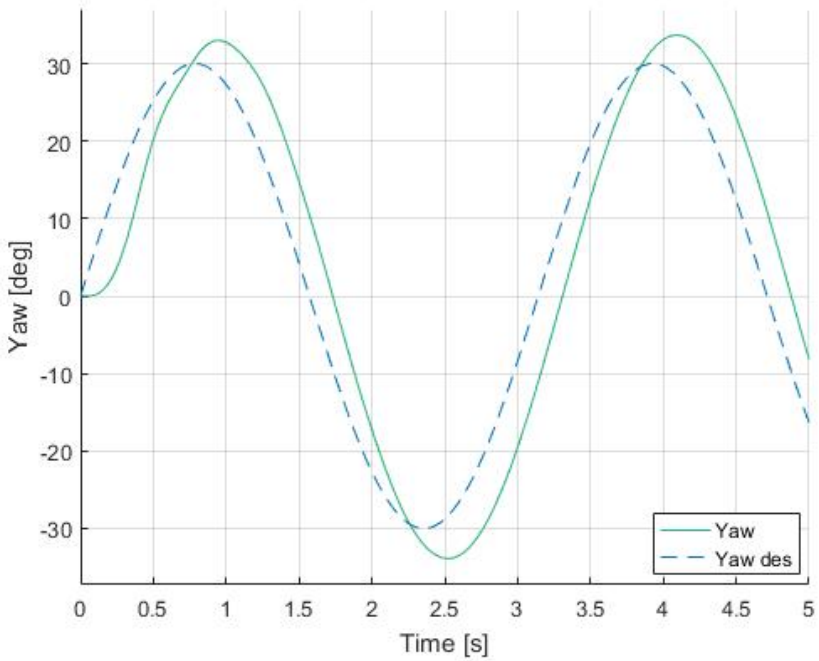


Figure 4.24: Response of the UAV to a 30° amplitude, 2 $[rad/s]$ sinusoidal wave as desired yaw, second formulation of the integral backstepping

quadrotor following an 8-shaped path with both the position and attitude controls active.

4.5.1 Simulation results

The geometric control turns out to be the a very reliable control, with small errors and good tracking capabilities even if there is no integral action implemented on it. On the other hand, the overall bandwidth is smaller than the backstepping based controllers presented before, since the control law manage to follow trajectories only up to a frequency of the 8-shape of 0.5 $[rad/s]$.

The trajectory tracking is presented in Figure 4.28. In can be seen that the altitude control works perfectly fine and the overall errors are far smaller than the ones observed in the other controllers as it is possible to see in Figure 4.29.

On the attitude control loop side, the performances are really satisfactory, with a small error even in the case of a quite nervous setpoint as shown in Figures 4.30, 4.31, 4.32. In this case as one can see from the plots of the attitude errors in Figure 4.33, compared with the ones shown before in 4.25 and 4.18 that are related to more regular setpoints, that the overall disturbance rejection is very effective.

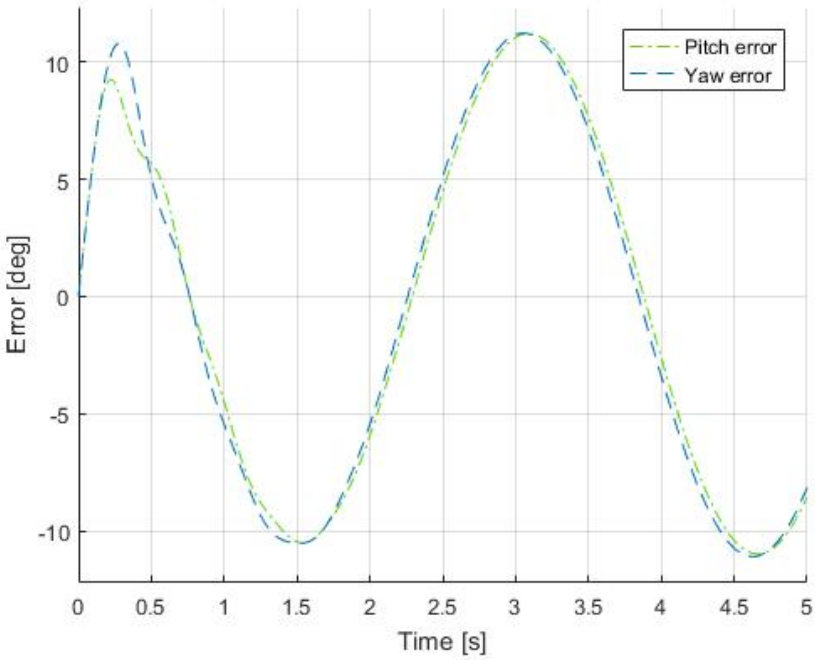


Figure 4.25: Pitch and yaw tracking errors due to a sinusoid of amplitude 30° at $2[rad/s]$ respectively on pitch and yaw, second formulation of the integral backstepping

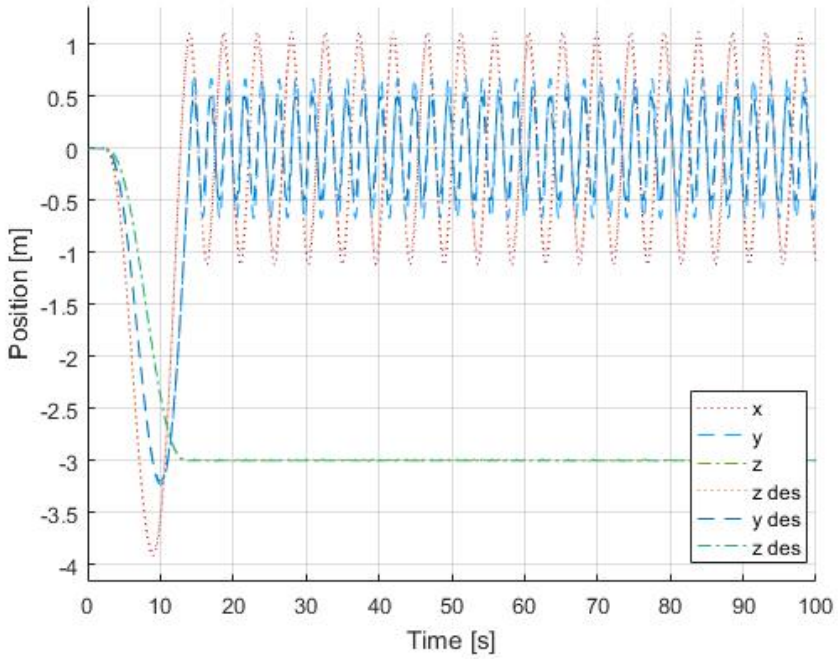


Figure 4.26: Position of the UAV quadrotor during an 8-shaped trajectory at 1.35 $[rad/s]$, second formulation of the integral backstepping

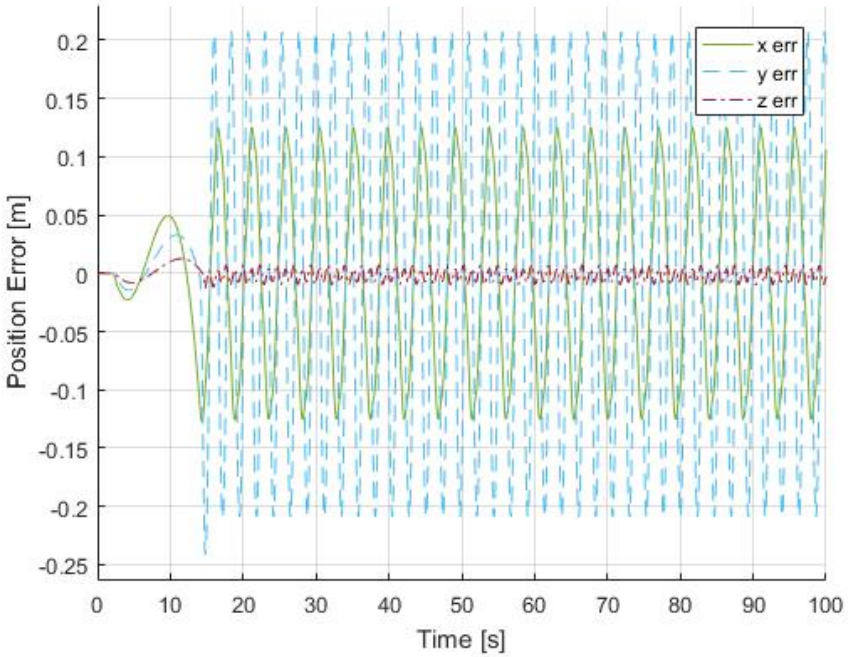


Figure 4.27: Tracking error of the response to an 8-shape trajectory, second formulation of the integral backstepping

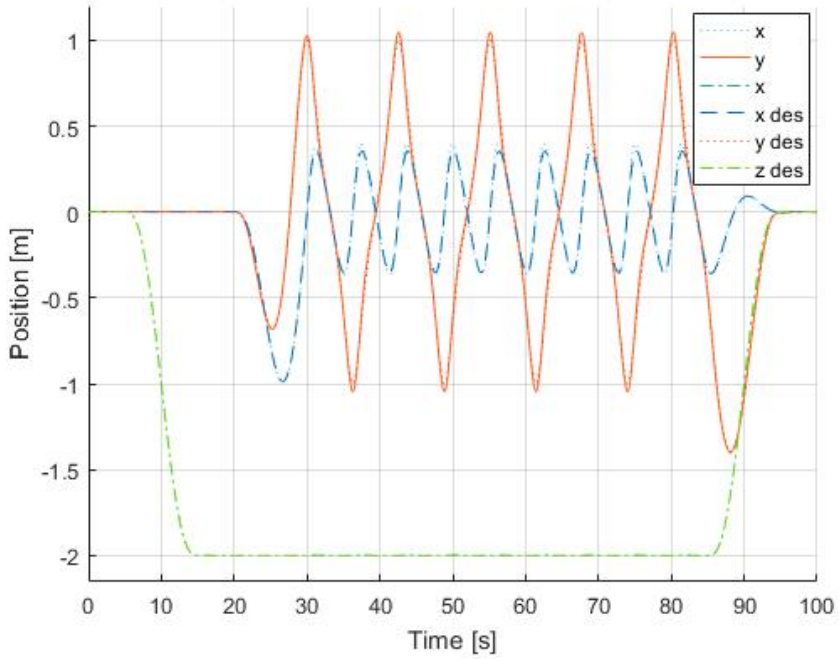


Figure 4.28: Position of the UAV quadrotor during an 8-shaped trajectory at 0.5 $[rad/s]$, geometric control

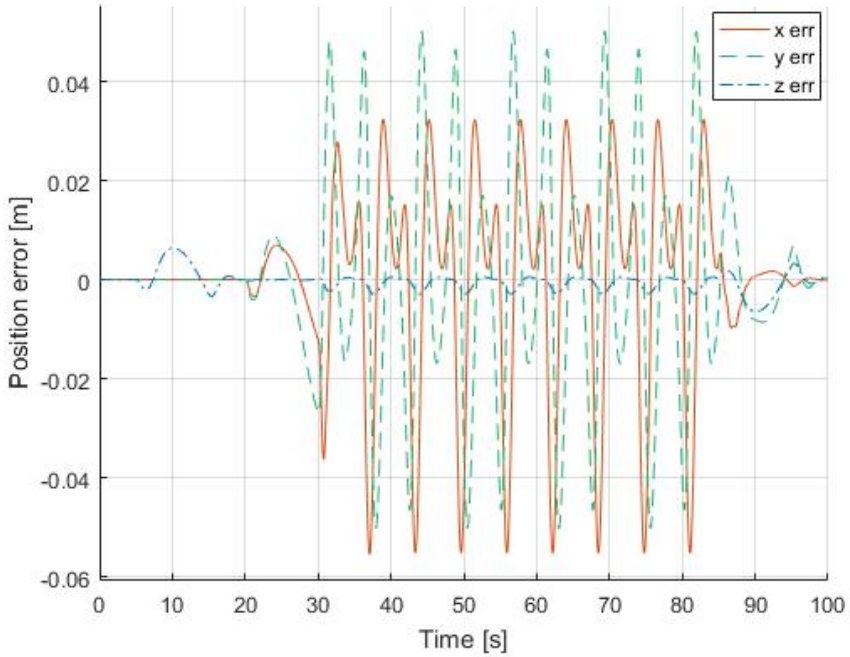


Figure 4.29: Tracking error of the response to an 8-shape trajectory, geometric control

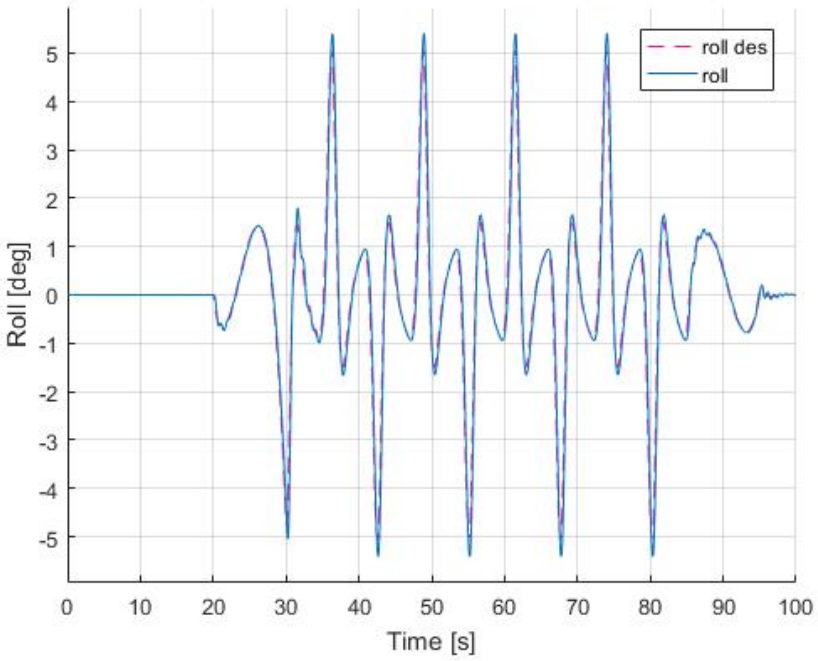


Figure 4.30: Tracking of a roll setpoint, geometric control

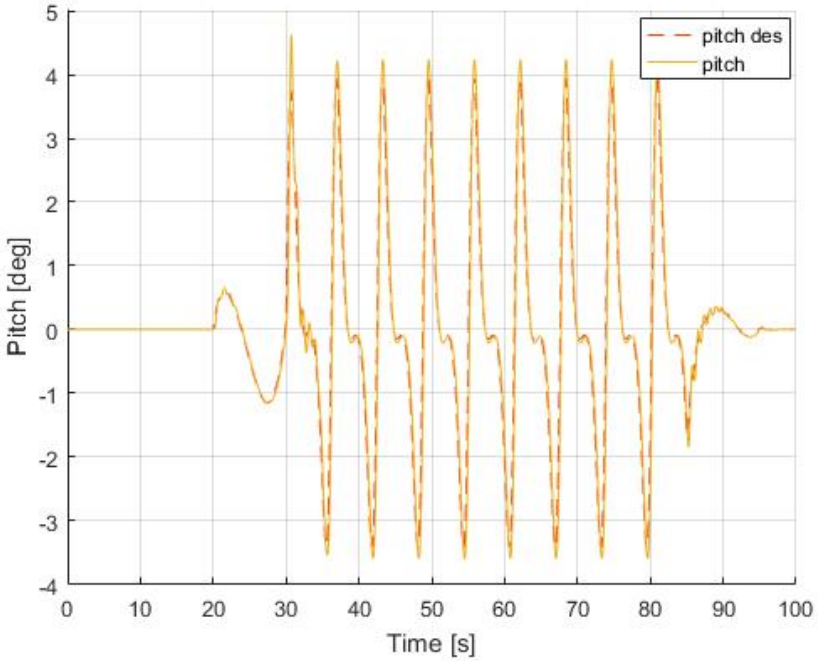


Figure 4.31: Tracking of a pitch setpoint, geometric control

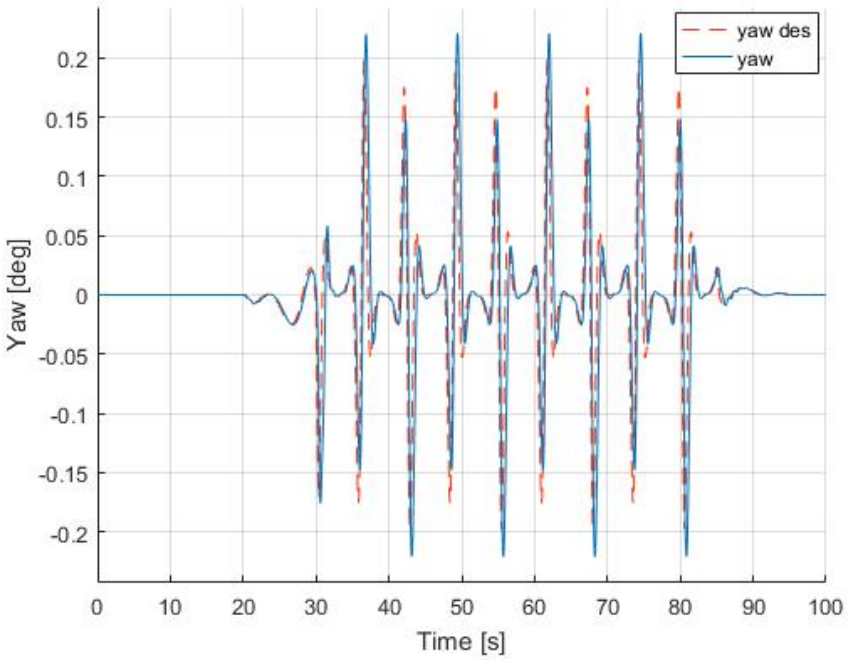


Figure 4.32: Tracking of a yaw setpoint, geometric control

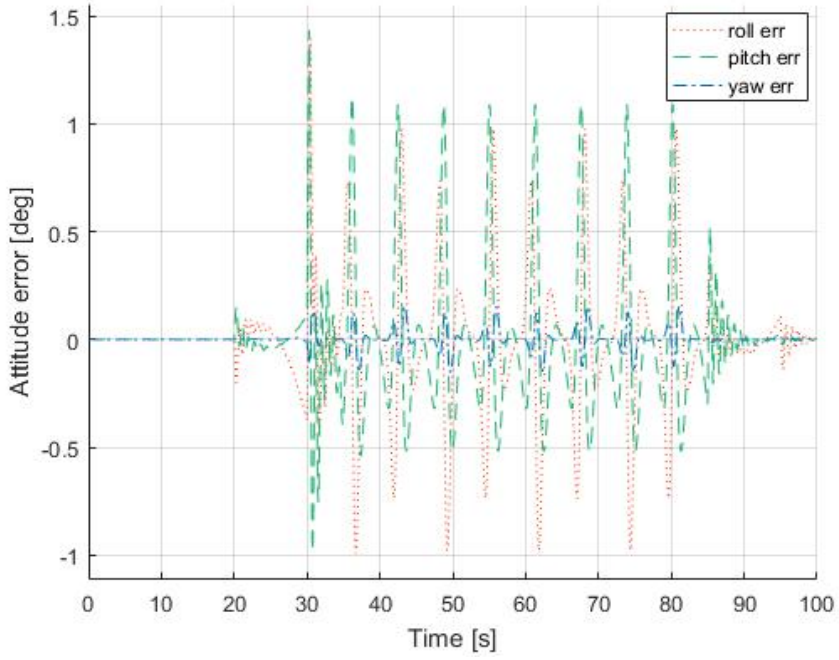


Figure 4.33: Attitude error, geometric control

4.6 Final considerations

In this chapter the results obtained simulating each control law has been presented and analysed.
From this results it is clear that there are still margins for the development of a better tuning procedure that can achieve better performances in tracking capabilities both for the position and the attitude control laws.
It turns out that the H_∞ tuning technique in this case give consistent results in terms of the setted bandwidth for the closed loop system, while it is clear that the overall tracking performances can be improved.
Eventually the Lyapunov based controllers turns out to have good performances even if they are limited by the assumptions presented in the previous chapters, while the geometric control has the potential to get better results but has the problem not to have a standard tuning procedure in order to choose the values of the parameters, resulting in a slower control.

Conclusions

In the presented work of thesis the design, implementation and simulation of four different nonlinear control laws for a quadcopter UAV has been presented. The controllers synthesised have been tested in different simulations and the results have been compared in the last chapter.
In detail, the mathematical development of the control laws has been extensively analysed in order to give a complete overview of the adopted techniques.
For the first two control laws the tuning method adopted has been particularly emphasised in order to give a complete example of how the minimization of the H_∞ norm of the sensitivity can be adopted also for particular nonlinear systems.
Eventually, the obtained results have been analysed and compared in order to stress the capabilities and limits of each controller.
The goal to obtain a control law that can manage complex trajectories has been reached, with a controlled system that can afford to undertake quite fast manoeuvrers.
Moreover, one of the main difficulties faced in nonlinear control design has always been the choice of the tunable parameters values, since their effects on the system performance are not always clear. Thus the adoption of a standard tuning technique in this field results is a remarkable improvement in the possibilities to obtain a better choice of the control parameters.
Due to hardware problems, it has not been possible to test properly the developed control on the real UAV platform to provide meaningful data to be analysed. The implementation on the quadrotor of this control laws is thus the main development that will be undertaken in the near future. This work is actually already in progress and have highlighted some limitations of the simulator developed in the laboratory in modelling the noises and disturbances that acts on the real system.
Another field of development for this work is a refinement of the H_∞ tuning procedure, adopted in order to better cope with the nonlinearities of the tuning problem. As can be noticed from the simulation results, it works satisfactorily but results sometimes in too aggressive control actions. This is mainly due to the nonlinear correlation shown in Chapter 2 between the Lyapunov parameters and the related gains of the linearised controller.
One last possible development would be the reformulation of the geometric control in order to get an integral action both on the position and the attitude control.

This because the lack of this terms strongly affects the performances on the real system due to unavoidable constant disturbances like bias on the propellers thrust.

Bibliography

[1] Samir Bouabdallah and Roland Siegwart. Backstepping and sliding-mode techniques applied to an indoor micro quadrotor. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2247–2252. IEEE, 2005.

[2] Francesco Bullo and Richard M Murray. Tracking for fully actuated mechanical systems: a geometric framework. *Automatica*, 35(1):17–34, 1999.

[3] Matteo Ferronato. Identificazione e controllo della dinamica verticale di un elicottero quadrirotore. 2016.

[4] Mattia Giurato. Design, integration and control of a multirotor uav platform. 2015.

[5] Davide Invernizzi and Marco Lovera. Geometric tracking control of thrust vectoring uavs. *arXiv preprint arXiv:1703.06443*, 2017.

[6] Taeyoung Lee, Melvin Leok, and N Harris McClamroch. Nonlinear robust tracking control of a quadrotor uav on se (3). *Asian Journal of Control*, 15(2):391–408, 2013.

[7] Taeyoung Lee, Melvin Leoky, and N Harris McClamroch. Geometric tracking control of a quadrotor uav on se (3). In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 5420–5425. IEEE, 2010.

[8] Lalo Magni and Riccardo Scattolini. *ADVANCED and MULTIVARIABLE CONTROL*. Pitagora Editrice Bologna, 2014.

[9] Robert Mahony, Vijay Kumar, and Peter Corke. Multirotor aerial vehicles. *IEEE Robotics and Automation magazine*, 20(32), 2012.

[10] Simone Panza. Hinfguide. `https://flyart.gitlab.io/hinfguide/`, 2017.