ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

SCUOLA DI INGEGNERIA

DIPARTIMENTO di
INGEGNERIA DELL'ENERGIA ELETTRICA E DELL'INFORMAZIONE
"Guglielmo Marconi"
DEI

CORSO DI LAUREA IN INGEGNERIA DELL'AUTOMAZIONE
CURRICULUM: AUTOMATION ENGINEERING

TESI DI LAUREA
in
Controlli Automatici T-2

Modelling and Control of a Pole Balancing Drone

CANDIDATO

Edoardo Panichi

RELATORE

Chiar.mo Prof. Lorenzo Marconi

CORRELATORE/I

Dott. Lorenzo Gentilini

Anno Accademico *2020/2021*

Sessione *I*

Abstract

Modelling and Control of a Pole Balancing Drone

by Edoardo PANICHI

L'impiego di unmanned aerial vehicles, ovvero velivoli senza pilota, sta diventando sempre più utile in diversi ambiti, non solo quello militare ma anche nel settore dei trasporti e dell'agricoltura di precisione.
Lo scopo di questo lavoro è implementare un sistema di controllo che permetta ad un quadricottero, sul quale viene appoggiata un'asta in equilibrio, di eseguire manovre di *set-point tracking* e di essere in grado di inseguire una triettoria circolare rimanendo ad un'altezza costante. In ogni instante, ovviamente, il drone deve assicurarsi che l'asta non cada. Anche se non esiste un'applicazione diretta di tale progetto, questo studio aiuta a comprendere quali siano le potenzialità dei droni. Infatti, con questi dispositivi, grazie alle tecniche di controllo più avanzate, si ottengono maggiori e migliori risultati rispetto a quelli conseguibili con l'impiego di velivoli comandati da un operatore umano.
Per raggiungere l'obbiettivo si parte dalla modellazione di un sistema non lineare; in seconda battuta è necessario linearizzare il modello. Infine, è possibile progettare il regolatore. La tecnica di controllo utilizzata è nota come *frequency shaped linear quadratic regulator* e si basa sulla teoria del *controllo ottimo* e del *full state feedback* che permettono di ottenere risultati robusti anche a fronte di sistemi MIMO.
Sono questi i punti chiave della tesi in oggetto.

Keywords: Controller, Quadcopter, LQR, Modelling, Observer, Linearization, FS-LQR.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Professor Lorenzo Marconi, and to my assistant supervisor Dott. Lorenzo Gentilini, who proposed this exciting thesis project to me and helped me during the past months with the development of the project.
After that, a heartfelt thank you also goes to my friends with whom I have spent long study sessions over the years, which have allowed me to complete this course; and to my family who has supported me in every way during my university career. Last but not least, I would like to thank Albino Dallio who, especially in this last period, has been a great support to me during the writing of my thesis.

Contents

List of Figures

Chapter 1

Introduction

An unmanned aerial vehicle (UAV) or uncrewed aerial vehicle, commonly known as a drone, is an aircraft without any human pilot, crew or passengers on board. A drone designed with four rotors takes the name of quadcopter or quadrotor. The four-rotor design allows quadcopters to be relatively simple in design yet highly reliable and maneuverable. Even if the first specimens were developed for the military field, nowadays the possible applications are countless. The mains are: aerial photography, shipping and delivery, disaster management, precision agriculture and weather forecast.
Around 2005 to 2010, advances in electronics allowed the production of cheap light flight controllers, accelerometers (IMU), global positioning systems and cameras. This resulted in the quadcopter configuration becoming popular for small unmanned aerial vehicles. For these reasons, in the last ten years quadcopters have become a spread hobby.
The system taken into consideration for this project is a quadcopter over which a vertical pole is placed, but not physically locked. So, the goal of the thesis is to realize a regulator that, first of all it is capable of keeping the pole always balanced, but secondly that is also able to perform a circumference, at a constant height, without losing control of the rod. The starting point is the *modelling*, which helps us to understand the physics behind the system expressing it through a set of differential equations. Once the model is ready, we have to *linearize* it to be able to apply the known control theory that is developed for linear systems only. The controller implemented is a *frequency shaped linear quadratic regulator* that is based on the concepts of *full state feedback* and *optimal control*. Thanks to it, the drone can describe a circle in the air. One more feature present in the project is a *state observer* for the pole, since not all its states are measurable.
In literature are present different papers about a quadcopter that balances an inverted pendulum, but no one of these implements a *frequency shaped linear quadratic regulator*.

The thesis is organized with the following structure. Chapter *2* is focused on theoretical concepts such us *controllability, reachability, observability and state observer* mainly useful to understand the chapter about the controller. Chapter *3* is dedicated to the *system modelling and linearization*, fundamental for the development of the regulator. Chapter *4* faces the main part of the thesis: *the control system*. So, in Chapter *4* different types of controllers are taken into

consideration, and the necessary control theory behind the regulators is explained. In Chapter *5* are reported the schemes of the project implemented on *Simulink* and their performances, both of the FS-LQR and of the inverted pendulum state observer.

Chapter 2

Controllability and Observability

Controllability is an important property of a control system, and the controllability property plays a crucial role in many control problems, such as stabilization of unstable systems by feedback, or optimal control.
Controllability and observability are dual aspects of the same problem.

2.1 Controllability and Reachability

Let's suppose to have a system described with its state-space representation:

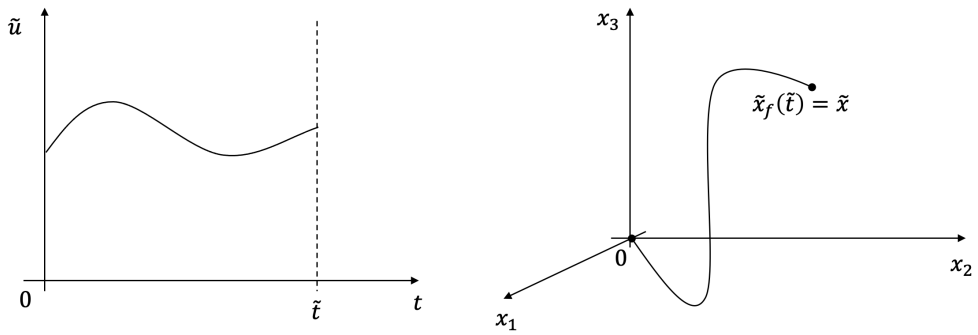
$$\dot{x} = Ax + Bu \tag{2.1}$$
$$y = Cx + Du, \tag{2.2}$$

where $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^m$.
We can now give introduce some definitions to better understand the concepts of controllability and reachability.

Definition 2.1 (Reachability of a state [2])**.**
A state $\tilde{x}$ of the system (2.1),(2.2) is said to be reachable, *if exist a finite time instant $\tilde{t} > 0$ and a input $\tilde{u}$, defined between 0 and $\tilde{t}$, such that, given $\tilde{x}_f(t)$, the force response of the state caused by $\tilde{u}$, it results $\tilde{x}_f(\tilde{t}) = \tilde{x}_f$.*

A system, where all the states are reachable, is said to be *completely reachable*.

FIGURE 2.1: Reachability of a state



In order words, a vector $\tilde{x}$ is a reachable state if it is possible, with an appropriate choice of the input, to reach it starting from the origin in an arbitrary

finite time. We will see that this property of a system is exclusively related to the equation (*2.1*), then it can be deducted by studying the couple (A, B).

Definition 2.2 (Reachability Matrix)**.**

$$M_r = [B\ AB\ A^2B\ \dots\ A^{n-1}B] \in \mathbb{R}^{n \times mn}.$$

Now it is possible to introduce a crucial theorem if we want to study the reachability of a system.

Theorem 2.1.
The system (2.1),(2.2) is completely reachable, that is the couple (A, B) is completely reachable, if and only if the rank of the Reachability Matrix M_r is equal to n, this means:

$$\rho(M_r) = n. \tag{2.3}$$

If the system has a unique input, then $m = 1$, the matrix M_r is squared and the necessary and sufficient condition (2.3) is equivalent to $det(M_r) \neq 0$.

Definition 2.3 (Reachability Set)**.**
The set of states that we can reach by applying any non-constant input, starting from all-null initial conditions is called Reachability Set *at time t, which is a subset of R^n:*

$$R^+(t) = \left\{ x \in \mathbb{R}^n \; x = \int_0^t e^{A(t-s)} B u(s)\, ds \right\}.$$

Then, considering two time instants t_1 e t_2, with $t_1 < t_2$, we will get $R^+(t_1) \subseteq R^+(t_2)$.

The concept of *reachability* is particularly important if we work with continuous-time systems, in these cases the systems are reversible, then the concept of reachability coincides with the controllability (see Theorem *2.2*).
The same is valid for the *sample data systems*.
Then, to study the controllability of continuous-time systems (or sample data systems) we can rely on the Theorem *2.1*.
In simple words, a state $\tilde{x}$ is *controllable* if it is possible to bring it from $x(0) = \tilde{x}$ to $x = 0$ in an arbitrary finite time choosing an appropriate input.

Definition 2.4 (Controllability Set)**.**
It is defined as Controllability Set *the union of all the states of a system that can be brought to the origin within an arbitrary and finite time interval:*

$$R^-(t) = \{x \in \mathbb{R}^n \; : \; x = \Phi x + \Psi u_{[0,t]} = 0\}.$$

States that do not depend on the control variable $u(t)$ cannot be controlled, and if in a system exists at least such a state, the system is called *Not Controllable*.

Definition 2.5 (Completely Controllability)**.**
A system is said to be Completely Controllable if $R^- = R^n$, that is if starting with any initial condition $x(t)$, it is possible to bring the system to the origin in an arbitrary and finite time interval thanks to the action of a specific input $u(t)$.

As already anticipated, there is a strong connection between *Reachability* and *Controllability*, as stated in the following theorem.

Theorem 2.2.
In general $R^+ \subseteq R^-$, instead if the system is reversible $R^+ = R^-$.

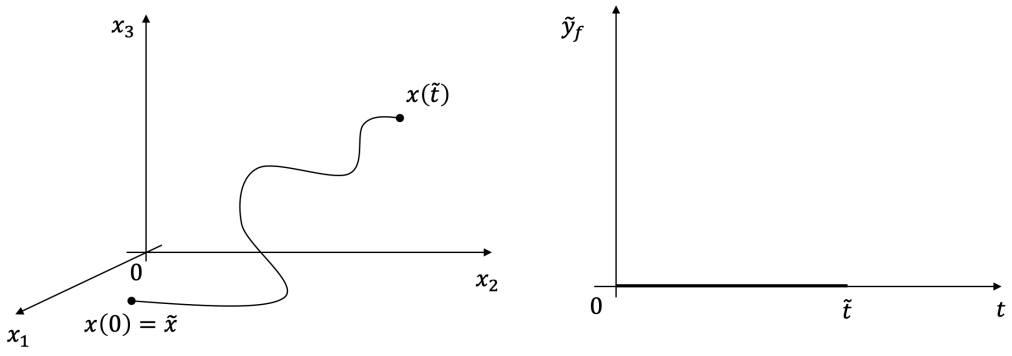
If we have a *Completely Controllable* system, this means that it is possible to arbitrarily assign the eigenvalues of the system introducing a suitable controller. A more detailed discussion about this possibility can be found in the chapter dedicated to the control.

2.2 Observability

Observability is a measure of how well internal states of a system can be inferred from knowledge of its external outputs.

Definition 2.6 (Unobservable State)**.**
A state $\tilde{x} \neq 0$ belonging to the system (2.1),(2.2) is said to be unobservable *if, for each $\tilde{t} > 0$, defining $\tilde{y}_f(t)$, $t \geq 0$, the free response of the output caused by $\tilde{x}$, it results $\tilde{y}_f(t) = 0, 0 \leq t \leq \tilde{t}$.*

FIGURE 2.2: Unobservable state



In other words, a vector $\tilde{x}$ is an *unobservable state* if, analyzing any piece of the trajectory of the free response caused by $\tilde{x}$, is not possible to distinguish it from the vector $x = 0$, that obviously generate a null free-response in the output.
A system that has no unobservable states is called *completely observable*.
The observability of a system is connected exclusively to the matrices A and C, also used to compose the *Observability Matrix*:

$$M_o = [C^T \; A^T C^T \; (A^2)^T C^T \; (A^{n-1})^T C^T] \in \mathbb{R}^{n \times np},$$

where p is the number of outputs. Now it is possible to introduce a crucial theorem if we want to study the observability of a system.

Theorem 2.3.
The system (2.1),(2.2) is completely observable, that is the couple (A, C) is completely observable, if and only if the rank of the Observability Matrix M_r is equal to n, this means:

$$\rho(M_o) = n. \tag{2.4}$$

If the system has a unique output, then $p = 1$, the matrix M_o is squared and the necessary and sufficient condition (2.4) is equivalent to $det(M_o) \neq 0$.

A slightly weaker notion than observability is *detectability*.

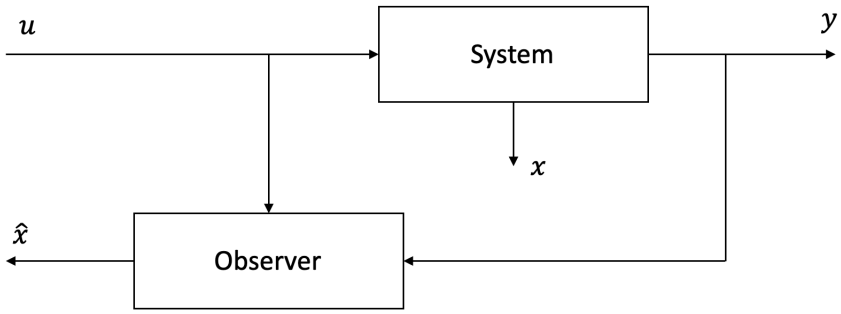
Definition 2.7 (Detectability)**.**
A system is detectable if all the unobservable states are stable.

2.3 State Observer

A dynamical system designed to estimate the state of a system from measurements of the outputs is called a *state observer* or simply an *observer* for that system.
Given a generic state-space representation of a system, like (*2.1*),(*2.2*), we usually consider knowing perfectly the state variables and their evolution in time. Actually, this is an unrealistic scenario, then we need to understand if it is possible to reconstruct the missing states with the knowledge of just inputs and outputs. The complete knowledge of the states of a system is fundamental if we want to control it through a state-feedback action (technique analyzed in the chapter dedicated to the control).
The system for which we want to reconstruct the states is the one described by the equations (*2.1*) and (*2.2*), moreover we suppose that $x(0) = x_0$ is unknown.

As depicted in the Figure *2.3*, an observer is a system (potentially even dynamic) that, taking as input $u(t)$ and $y(t)$, gives as output an estimate $\hat{x}(t)$ of the state $x(t)$ of the system. The quality of the estimate can be measured by means of the observation error $e(t) = \hat{x}(t) - x(t)$.

FIGURE 2.3: The state observer



2.3.1 Trivial State Observer

A first trivial solution for the observability problem is achieved using a copy of the original system, fed with the same input, this means:

$$\dot{\hat{x}}(t) = A\hat{x}(t) + Bu(t), \quad \hat{x}(0) = \hat{x}_0, \tag{2.5}$$

where $\hat{x}_0$ is the estimate of the initial state.
In theory, if $\hat{x}_0 = x_0$ and (A, B) are perfectly known, the output of the observer tracks exactly the states of the original system. But, the most plausible scenario is that $\hat{x}_0 \neq x_0$, then if we evaluate the difference between the equations (*2.1*) and (*2.5*) we can verify that the evolution in time of the observation error can be described by the following relationship:

$$\dot{e}(t) = Ae(t), \quad e(0) = \hat{x}_0 - x_0 \neq 0.$$

So, two cases are possible:

- The matrix A is not asymptotically stable: the estimated error $e(t)$ does not converge to zero, or it could even diverge if the matrix is unstable.
- The matrix A is asymptotically stable: the estimated error $e(t)$ converges to zero for any value of $e(0)$, but the velocity to bring the error to zero is set by the eigenvalues of A. In this second case, we obtain a convergence because both $\hat{x}$ and x converge, with different times, towards the same result.

Neither scenario, therefore, leads to a satisfactory outcome. Indeed, even in the second case we cannot decide the dynamics of the error and actually, until we have reached a zero error, we have no information about the magnitude of $e(t)$.
Then, we are forced to introduce a new type of observer to solve these problems.

2.3.2 Observer with Arbitrary Dynamics

The fundamental difference with the trivial observer is that now we exploit also the information coming from the outputs of the system. In a real physical system, the outputs are available thanks to the presence of dedicated sensors in the system.
The new state-space representation of the observer becomes:

$$\dot{\hat{x}}(t) = A\hat{x}(t) + Bu(t) + H(\hat{y}(t) - y(t)), \quad \hat{x}(0) = \hat{x}_0, \tag{2.6}$$
$$\hat{y}(t) = C\hat{x}(t) + Du(t). \tag{2.7}$$

The term introduced is proportional to the difference between the estimated output (*2.7*) and the measured output (*2.2*). This is reasonable since a greater value of this difference (in terms of absolute value) requires a stronger correcting action that brings the estimate $\hat{x}(t)$ closer to the right value $x(t)$.
The matrix H, which actually in the hypothesis done previously is a column

vector with dimension equal to n, is called *observer gain matrix* and it is a free parameter that can be decided by the designer according to the requests of the system. When we choose, through the tuning of H, the velocity of the observation error dynamics, if we require a very fast dynamics, when the measured output is not completely correct (due to the accuracy of the sensor used, or for the presence of external noises), we could increase the effect of this error in our estimate.
To understand how to choose the value of H we need calculate the difference between the equations (*2.6*) and (*2.1*), then exploiting also the relationships (*2.7*) and (*2.2*), it is possible to get:

$$\dot{e}(t) = (A + HC)e(t), \quad e(0) = \hat{x}_0 - x_0.$$

It is clear that we can choose the value of H in order to obtain the desired eigenvalues of the matrix $N = A + HC$ that is responsible for the dynamics of the observation error.
Matlab is an easy way to place the eigenvalues of N in a defined configuration, indeed we can exploit the command $H = [place(A^T, -C^T, P)]^T$, where P is a vector containing the desired position of the eigenvalues for N.

Chapter 3

System Modelling

3.1 Lagrangian Approach

The modelling is a process through which is possible to describe the dynamics of a system using differential equations. To obtain the set of equations that completely describe the dynamic of the system, it is possible to exploit a powerful mathematical tool known as *Lagrangian*. No new physics is necessarily introduced in applying Lagrangian mechanics compared to Newtonian mechanics.

Lagrangian Mechanics

Definition 3.1 (Lagrangian)**.**
The Lagrangian is a function of the generalized coordinates, of their time derivatives, of the time and contains the information about the dynamics of the system. The Lagrangian $L(\dot{q}, q, t)$ of a physical system with $\boldsymbol{n}$ degrees of freedom is defined as the difference between the kinetic energy $T(q, \dot{q}, t)$ and the potential energy $U(q, t)$:

$$L(\dot{q}, q, t) = T(\dot{q}, q, t) - U(q, t),$$

where $q \in \mathbb{R}^n$ denotes the generalized coordinates, the dot represents the operation of derivation and $t \in \mathbb{R}$.

Lagrangian mechanics can only be applied to systems whose constraints, if any, are all holonomic. Non-holonomic constraints require special treatment, and one may have to revert to Newtonian mechanics, or use other methods. Once calculated the *Lagrangian* of a system, it is possible to obtain its dynamics using the *Euler–Lagrange equations of motion*:

$$\frac{\partial L}{\partial q_j} - \frac{d}{dt}\frac{\partial L}{\partial \dot{q}_j} = Q_j, \tag{3.1}$$

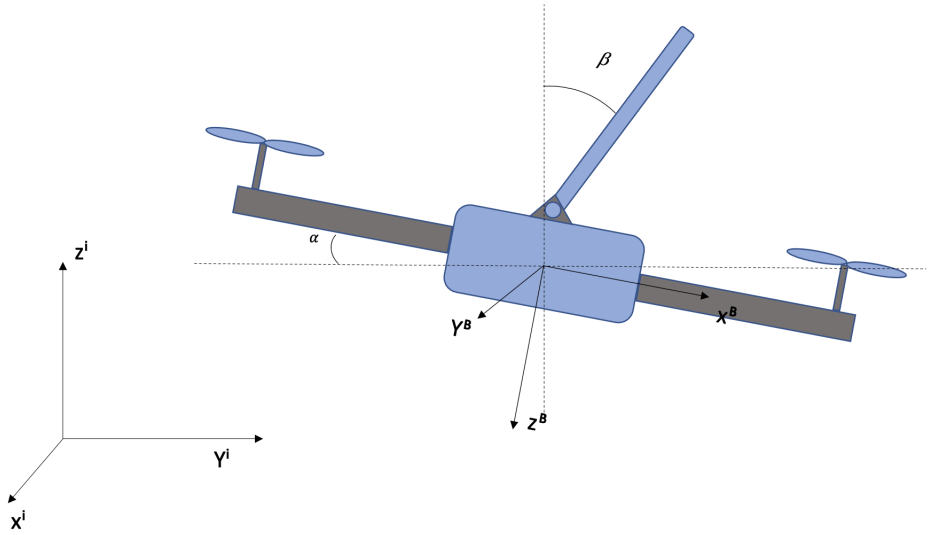
where Q_j denotes the generalized forces. In the formulation of virtual work, each generalized force is the coefficient of the variation of a generalized coordinate.

$$\delta W = Q_1 \delta q_1 + \cdots + Q_n \delta q_n.$$

3.2 2D Modelling of the Pole Balancing Drone

To better understand the modelling with the *Lagrangian*, here is reported an example also useful for the final goal of this work. The system taken into consideration for the example is a pole balancing drone confined in a 2D space, this means that, if we consider the Figure *3.1*, the drone cannot move along the x^i*-axis* and it cannot rotate around y^i or z^i.
The modelling is done considering the pole and the quadcopter as two separate systems, but including the forces introduced by mutual interaction. Even if the pole is not fixed with the frame of the drone, we can assume, without loss of generality, that the pole is connected to the drone through a revolute joint.

FIGURE 3.1: 2D scheme of the pole balancing drone



The Quadcopter: Then, starting from the drone, its *Lagrangian* is equal to:

$$L = \frac{1}{2}M(\dot{y}^2 + \dot{z}^2) + \frac{1}{2}J\dot{\alpha}^2 - Mgz,$$

where M is the mass, J the inertia, z and y are respectively the altitude and the latitude of the center of mass of the quadcopter and g the gravitational acceleration. In order to apply the *Euler–Lagrange equations of motion*, it is important to define the generalized coordinates of this system:

$$q_1 = y, \quad q_2 = z, \quad q_3 = \alpha.$$

Generalized coordinates are usually selected to provide the minimum number of independent coordinates that define the configuration of a system, indeed in this case the system under analysis has three degrees of freedom. Associated to each generalized coordinate, a *generalized force* can be determined,

in this case:

$$Q_1 = T\sin\alpha - F_y, \quad Q_2 = T\cos\alpha - F_z, \quad Q_3 = \tau_x,$$

where T and τ_x are respectively the thrust and the torque around x^i produced by the propellers, α the roll angle, F_y and F_z are the forces introduced due to the presence of the pole. Now, if the relationship (*3.1*) is applied substituting, once at the time, the generalized coordinate and the generalized force we can obtain three equations that are part of the dynamics of the drone:

$$\begin{aligned} M\ddot{y} &= T\sin\alpha - F_y, \\ M\ddot{z} &= T\cos\alpha - Mg - F_z, \\ J\ddot{\alpha} &= \tau_x. \end{aligned}$$

To achieve the description of the system in the state-space representation, the following definitions are taken into account:

$$x_1 = y, \quad x_2 = V_y, \quad x_3 = z, \quad x_4 = V_z, \quad x_5 = \alpha, \quad x_6 = \omega_\alpha, \quad u_1 = T, \quad u_2 = \tau_x.$$

Then, the full dynamics of the drone can be described through a six-equation system reported below:

$$\begin{cases} \dot{x}_1 = x_2 \\ M\dot{x}_2 = u_1\sin x_5 - F_y \\ \dot{x}_3 = x_4 \\ M\dot{x}_4 = u_1\cos x_5 - Mg - F_z \\ \dot{x}_5 = x_6 \\ J\dot{x}_6 = u_2 \end{cases}.$$

The next step is to substitute F_y and F_z with two known quantities and to do it, the dynamics of the pole must be known.

The Pole: Once defined the position of the drone as $P_d = (y, z)$, we can determine the position of the center of gravity of the pole just by adding a vector:

$$P_p = P_d + \frac{l}{2}\hat{n}$$

where $\hat{n}$ is a unit vector that points from the quadcopter towards the pole mass center and l is the length of the pole. The same quantity can also be expressed like:

$$P_p = (y + \frac{l}{2}\sin\beta,\, z + \frac{l}{2}\cos\beta).$$

Then, following the same steps done for the quadcopter we can define the *Lagrangian* of the pole:

$$L = \frac{1}{2}m(\dot{y}^2 + \dot{z}^2) + \frac{1}{6}ml^2\dot{\beta}^2 + \frac{1}{2}ml^2\dot{\beta}(\dot{y}\cos\beta - \dot{z}\sin\beta) - mg(z + \frac{l}{2}\cos\beta),$$

where m is the mass of the pole and β is the angle that identifies the inclination of the pole with respect to the z-axis. Now, as done before, it is necessary to define the generalized coordinates of the new system. Actually, two coordinates are the same of the previous system because, as seen, the position of the pole is strictly related to the drone position. Therefore:

$$q_1 = y, \quad q_2 = z, \quad q_3 = \beta.$$

Now, just applying the relationship (*3.1*) to q_3 we immediately find the rotational dynamics of the pole:

$$\frac{1}{3}ml^2\ddot{\beta} = mg\frac{l}{2}\sin\beta - \frac{1}{2}ml(\ddot{y}\cos\beta - \ddot{z}\sin\beta).$$

Once done this, it is important to develop the same calculations for q_1 and q_2 to find an expression of F_y and F_z in order to substitute them in the dynamics of the drone:

$$F_y = m\ddot{y} + \frac{1}{2}ml\ddot{\beta}\cos\beta - \frac{1}{2}ml\dot{\beta}^2\sin\beta,$$
$$F_z = mg + m\ddot{z} - \frac{1}{2}ml\ddot{\beta}\sin\beta - \frac{1}{2}ml\dot{\beta}^2\cos\beta.$$

The 2D Complete Modelling: If we combine the two dynamics we can immediately obtain the full model of the pole-drone system. As can be inferred from the below system of equations, the dynamics of the quadcopter affects the pole and vice versa. So, the set of equations obtained are:

$$x_1 = y, \quad x_2 = V_y, \quad x_3 = z, \quad x_4 = V_z, \quad x_5 = \alpha, \quad x_6 = \omega_\alpha, \quad x_7 = \beta,$$
$$x_8 = \omega_\beta, \quad u_1 = T, \quad u_2 = \tau_x,$$

$$\begin{cases} \dot{x}_1 = x_2 \\ (M+m)\dot{x}_2 = u_1\sin x_5 + \frac{1}{2}mlx_8^2\sin x_7 - \frac{1}{2}ml\dot{x}_8\cos x_7 \\ \dot{x}_3 = x_4 \\ (M+m)\dot{x}_4 = u_1\cos x_5 + \frac{1}{2}mlx_8^2\cos x_7 + \frac{1}{2}ml\dot{x}_8\sin x_7 - (M+m)g \\ \dot{x}_5 = x_6 \\ J\dot{x}_6 = u_2 \\ \dot{x}_7 = x_8 \\ \frac{1}{3}ml^2\dot{x}_8 = mg\frac{l}{2}\sin x_7 + \frac{1}{2}ml\dot{x}_4\sin x_7 - \frac{1}{2}ml\dot{x}_2\cos x_7 \end{cases}.$$

The above system fully describes the dynamics of the 2D pole balancing drone, but it presents algebraic loops that must be removed if we want to implement the equations in a simulation. To obtain more compact solutions

some parameters are introduced:

$$
\begin{aligned}
A &= u_1 \cos x_5, \\
B &= \frac{1}{2} m l x_8^2 \cos x_7, \\
C &= (M+m)g, \\
D &= \frac{3}{4} m g \sin^2 x_7, \\
E &= \frac{4u_1 \sin x_5 + 2ml x_8^2 \sin x_7 - 3mg \sin x_7}{4(M+m) - 3m\cos^2 x_7}, \\
F &= \frac{3}{4} m \cos x_7 \sin x_7, \\
G &= M + m - \frac{3}{4}\sin^2 x_7 - \frac{9m^2 \cos x_7 \sin^2 x_7}{16(M+m) - 12m\cos^2 x_7}, \\
H &= \frac{3m \sin x_7}{4(M+m) - 3m\cos^2 x_7}.
\end{aligned}
$$

Once defined these parameters, the whole system can be written without any algebraic loop, then it can be implemented in *Simulink*. The new set of equations have now this form:

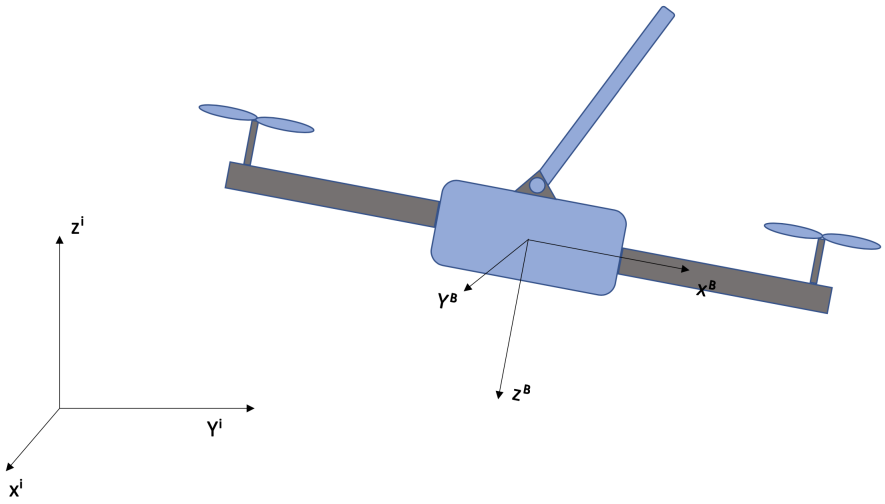
$$
\begin{cases}
\dot{x}_1 = x_2 \\
\dot{x}_2 = E - H \cdot \frac{A+B-C+D-FE}{G} \\
\dot{x}_3 = x_4 \\
\dot{x}_4 = \frac{A+B-C+D-FE}{G} \\
\dot{x}_5 = x_6 \\
J\dot{x}_6 = u_2 \\
\dot{x}_7 = x_8 \\
\dot{x}_8 = \frac{3}{2l}[g \sin x_7 + \frac{A+B-C+D-FE}{G} \sin x_7 - (E - H \cdot \frac{A+B-C+D-FE}{G}) \cos x_7]
\end{cases}
.
$$

3.3 3D Modelling of the Pole Balancing Drone

We have just seen how is possible to derive a 2D model of a system exploiting the *Lagrangian*. Even if the same approach can be used for a 3D system, there are some extra difficulties to be taken into account. A very important concept, necessary to understand the 3D model, are the rotation matrices [9] [10].
It is crucial to remind that, to obtain the complete dynamics of a system, we need a set of equations that describes the evolution in time of all the varying quantities; then for each time-dependent variable, we will need an equation where the derivative of that quantity appears.
In the following modelling only the dynamics of the pole will be derived thanks to the Lagrangian method, as the model for the drone is achieved through a more classic approach.
To generate the model, an inertial reference system (marked with the apex

"i") and a body reference system (marked with the apex "B") have been used.

FIGURE 3.2: Reference system for the 3D modelling



The quadcopter: The set of equations that describes the dynamics of the drone are the following:

$$\begin{cases} \dot{P} = V \\ m\dot{V} = -mge_3 - {}^I R_B T e_3 \\ {}^I\dot{R}_B = {}^I R_B S(\omega_B) \\ J\dot{\omega}_B = -S(\omega_B)J\omega_B + \tau \end{cases} . \tag{3.2}$$

Even if the set of equations is compact, it contains a lot of information, then is worth analyzing one equation at a time. Note that each of these relationships actually must be accounted as three separate equations, indeed in the above system are written in a vector form, where each vector includes the three components (x, y, z).
The first one is a well-known mechanics equation, indeed P is the position of the center of mass of the drone (C.o.M.) and V its velocity.
The second equation corresponds to $F = ma$, where e_3 is the unit vector of the body reference frame directed along z, T is the trust generated by the drone, g the gravitational acceleration, m the mass of the drone and ${}^I R_B$ the rotation matrix which allows us to transform a quantity from the body reference frame to the inertial one. This matrix is obtained considering the roll (ϕ), pitch (θ) and yaw (ψ) angles, respectively around x, y, z (note that RPY angles are always rotation respect to the base frame). The final result obtained is [3]:

$${}^I R_B = \begin{pmatrix} C_\psi C_\theta & C_\psi S_\theta S_\phi - S_\psi C_\phi & C_\psi S_\theta C_\phi + S_\psi S_\phi \\ S_\psi C_\theta & S_\psi S_\theta S_\phi + C_\psi C_\phi & S_\psi S_\theta C_\phi - C_\psi S_\phi \\ -S_\theta & C_\theta S_\phi & C_\theta C_\phi \end{pmatrix} .$$

Where S stands for sine and C stands for cosine.
Exploiting this matrix we can transform the trust from the body reference frame to the inertial one, with respect to which the dynamics of the system is expressed.
The third equation is needed to describe the evolution in time of the rotation matrix. This mathematical statement, where $S(\omega_B)$ is the skew-symmetric matrix composed starting from the vector ω_B (that is nothing but the angular velocity of the drone expressed in the body reference frame), is a known formula to express the derivative of a rotation matrix [5] [11].

$$S(\omega_B) = \begin{pmatrix} 0 & -\omega_{\mathrm{Bz}} & \omega_{\mathrm{By}} \\ \omega_{\mathrm{Bz}} & 0 & -\omega_{\mathrm{Bx}} \\ -\omega_{\mathrm{By}} & \omega_{\mathrm{Bx}} & 0 \end{pmatrix}.$$

As the rotation matrix depends on the RPY angles, this relationship helps us to define the three scalar equations for roll, pitch and yaw.
The last relationship of the model takes into account the rotational dynamics of the system, indeed τ is a vector of torques produced by the propellers

$$\tau = \begin{pmatrix} \tau_x \\ \tau_y \\ \tau_z \end{pmatrix},$$

and J is the rotational inertia matrix of the drone.

$$J = \begin{pmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{pmatrix}.$$

The origin of this equation can be found in the relationship between the angular momentum L, defined as $L = J\omega_B$, and the torque τ:

$$\frac{dL}{dt} = \tau. \tag{3.3}$$

Now, keeping in mind that in our context the quantity considered are matrices, we can start calculating the derivative of the *angular momentum*:

$$\dot{L} = J\dot{\omega}_B + S(\omega_B)J\omega_B. \tag{3.4}$$

Then, substituting the equation (*3.4*) in the relationship (*3.3*) and rearranging the terms, we obtain the latter equation of the system.
Even if the system (*3.2*) fully describes the behavior of our drone, the matrix form of the dynamics is not the best one to model and understand the system. So, the next step is to obtain twelve scalar equations starting from the system (*3.2*). This passage is quite straightforward as we have just to rewrite each expression putting in evidence the matrices, then performing the calculation we derive from each matrix equation three scalar relationships.

Before doing this, is worth defining the states (x_i) and the inputs (u_i) of the system:

$$\begin{pmatrix} x \\ V_x \\ y \\ V_y \\ z \\ V_z \\ \phi \\ \theta \\ \psi \\ \omega_{x_B} \\ \omega_{y_B} \\ \omega_{z_B} \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \end{pmatrix}, \qquad \begin{pmatrix} T \\ \tau_x \\ \tau_y \\ \tau_z \end{pmatrix} = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix}.$$

Starting from the first equation we obtain:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_3 \\ \dot{x}_5 \end{pmatrix} = \begin{pmatrix} x_2 \\ x_4 \\ x_6 \end{pmatrix}.$$

This can be rewritten like a system as follows:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_3 = x_4 \\ \dot{x}_5 = x_6 \end{cases}. \tag{3.5}$$

From the second equation of the system (3.2) we derive the expression of the dynamics of the linear velocities:

$$m \begin{pmatrix} \dot{x}_2 \\ \dot{x}_4 \\ \dot{x}_6 \end{pmatrix} = -m \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix} - {}^{I}R_B \begin{pmatrix} 0 \\ 0 \\ u_1 \end{pmatrix}.$$

Performing the calculations, it is possible to rewrite the above equation as a system:

$$\begin{cases} \dot{x}_2 = -\frac{u_1}{m}(\cos x_7 \sin x_8 \cos x_9 + \sin x_7 \sin x_9) \\ \dot{x}_4 = -\frac{u_1}{m}(\cos x_7 \sin x_8 \sin x_9 - \sin x_7 \cos x_9) \\ \dot{x}_6 = -g - \frac{u_1}{m}(\cos x_7 \cos x_8) \end{cases}. \tag{3.6}$$

If the first six equations can be obtained without any effort starting from the first two relationships of the dynamics, the same cannot be done for the third equation of the system (3.2). As matter of fact, what we are interested in are three equations describing the evolution in time of RPY. For this case, we must introduce a relationship that connects the angular velocities with RPY [1]:

$$\begin{pmatrix} \omega_{B_x} \\ \omega_{B_y} \\ \omega_{B_z} \end{pmatrix} = \begin{pmatrix} 1 & 0 & -S_\phi \\ 0 & C_\phi & C_\theta S_\phi \\ 0 & -S_\phi & C_\theta C_\phi \end{pmatrix} \begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix}.$$

Or, actually more useful for our purpose, the vice-versa:

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} 1 & S_\phi T_\theta & C_\phi T_\theta \\ 0 & C_\phi & -S_\phi \\ 0 & S_\phi / C_\theta & C_\phi / C_\theta \end{pmatrix} \begin{pmatrix} \omega_{B_x} \\ \omega_{B_y} \\ \omega_{B_z} \end{pmatrix},$$

where S stands for sine, C stands for cosine and T stands tangent.
From this last relationship we can immediately find the equations to describe the evolution in time of the three states that represent RPY:

$$\begin{cases} \dot{x}_7 = x_{10} + \tan x_8 (x_{11} \sin x_7 + x_{12} \cos x_7) \\ \dot{x}_8 = x_{11} \cos x_7 - x_{12} \sin x_7 \\ \dot{x}_9 = x_{11} \frac{\sin x_7}{\cos x_8} + x_{12} \frac{\cos x_7}{\cos x_8} \end{cases}. \tag{3.7}$$

For the fourth equation of the system (3.2) we can follow the same method seen for the first two equations:

$$\begin{pmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{pmatrix} \begin{pmatrix} \dot{x}_{10} \\ \dot{x}_{11} \\ \dot{x}_{12} \end{pmatrix} = \begin{pmatrix} 0 & -x_{12} & x_{11} \\ x_{12} & 0 & -x_{10} \\ -x_{11} & x_{10} & 0 \end{pmatrix} \begin{pmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{pmatrix} \begin{pmatrix} x_{10} \\ x_{11} \\ x_{12} \end{pmatrix} + \begin{pmatrix} \tau_x \\ \tau_y \\ \tau_z \end{pmatrix}.$$

Then, once again, if we perform all the calculations we can rewrite this expression in the form of a system:

$$\begin{cases} \dot{x}_{10} = \frac{\tau_x + x_{11} x_{12} (J_z - J_y)}{J_x} \\ \dot{x}_{11} = \frac{\tau_y + x_{10} x_{12} (J_x - J_z)}{J_y} \\ \dot{x}_{12} = \frac{\tau_z + x_{10} x_{11} (J_y - J_x)}{J_z} \end{cases}. \tag{3.8}$$

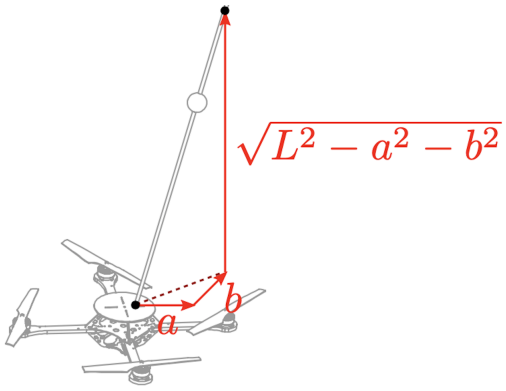
Now, that we have the twelve equations to fully describe the dynamics of the drone, we can write a unique system composed by the equations (3.5), (3.6), (3.7) and (3.8).

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -\frac{u_1}{m}(\cos x_7 \sin x_8 \cos x_9 + \sin x_7 \sin x_9) \\ \dot{x}_3 = x_4 \\ \dot{x}_4 = -\frac{u_1}{m}(\cos x_7 \sin x_8 \sin x_9 - \sin x_7 \cos x_9) \\ \dot{x}_5 = x_6 \\ \dot{x}_6 = -g - \frac{u_1}{m}(\cos x_7 \cos x_8) \\ \dot{x}_7 = x_{10} + \tan x_8 (x_{11} \sin x_7 + x_{12} \cos x_7) \\ \dot{x}_8 = x_{11} \cos x_7 - x_{12} \sin x_7 \\ \dot{x}_9 = x_{11} \frac{\sin x_7}{\cos x_8} + x_{12} \frac{\cos x_7}{\cos x_8} \\ \dot{x}_{10} = \frac{\tau_x + x_{11} x_{12} (J_z - J_y)}{J_x} \\ \dot{x}_{11} = \frac{\tau_y + x_{10} x_{12} (J_x - J_z)}{J_y} \\ \dot{x}_{12} = \frac{\tau_z + x_{10} x_{11} (J_y - J_x)}{J_z} \end{cases} . \tag{3.9}$$

The Pole: As already mentioned, the modelling of this part of the system will be derived exploiting the *Lagrangian*. We can start defining the position of the center of mass of the pole with respect to the position of the center of mass of the drone; differently from what we have seen in the 2D case, here it is better to assume the whole mass of the pole concentrated in its tip.
With this consideration it is not necessary to account for the rotational energy in the Lagrangian, indeed the pole can be seen as a mass concentrated in a single point, placed at a distance L from the C.o.M of the drone. Note that if the pole is constructed following this idea, the assumption made does not weaken our model.

$$P_P = P_D + L\hat{n},$$

FIGURE 3.3: Position of the C.o.M of the pole



where P_P denote the position of the C.o.M of the pole, P_D is the position of the C.o.M of the quadcopter, L is the length of the pole and $\hat{n}$ is a unit vector that has the same direction of the pole. We can also express P_P in Cartesian coordinates as follows:

$$P_P = (x + a,\ y + b,\ z + \zeta) \quad where \quad \zeta = \sqrt{L^2 - a^2 - b^2}.$$

In the 2D model, we have seen that the pole influences the dynamics of the drone through two forces F_y and F_z, here in the 3D context to obtain easier calculations we will neglect the contribution of these two quantities. This simplification is acceptable only if $m << M$, indeed in this scenario is clear that the contribution of the pole to the dynamics of the quadcopter is almost null. Moreover, all these simplifications, even if applied in a non-ideal way, will be compensated with the implementation of a robust controller.
Now that all the preliminary considerations have been done, we can derive the dynamics of the pole starting from the expression of the kinetic energy:

$$T_P = \frac{1}{2} m \dot{P}_P^T \dot{P}_P \quad \text{that can be rewritten as} \quad T_P = \frac{1}{2}[(\dot{x} + \dot{a})^2 + (\dot{y} + \dot{b})^2 + (\dot{z} + \dot{\zeta})^2].$$

And the potential energy:

$$U_P = mgP_P^T \ \text{ where } g = (0,\ 0,\ g) \quad \text{that can be rewritten as} \quad U_P = mg(z + \zeta).$$

So the *Lagragian* for the pole becomes:

$$L = \frac{1}{2}[(\dot{x} + \dot{a})^2 + (\dot{y} + \dot{b})^2 + (\dot{z} + \dot{\zeta})^2] - mg(z + \zeta).$$

In order to apply the relationship (*3.1*) to this Lagrangian we have to note that no generalized forces are present and we need to define two generalized coordinates, which are enough to identify the position of C.o.M of the pole since it is expressed with reference to the position of the drone:

$$q_1 = a, \quad q_2 = b.$$

The calculation to be done to obtain the final results are quite heavy, then is a good idea to introduce a new compact notation, used as intermediate step, just to be sure to correctly develop the computation:

$$\zeta_a = \frac{\partial \zeta}{\partial a} = -\frac{a}{\zeta}, \quad \zeta_b = \frac{\partial \zeta}{\partial b} = -\frac{b}{\zeta}, \quad \dot{\zeta} = -\frac{a\dot{a} + b\dot{b}}{\zeta},$$
$$\ddot{\zeta} = -\frac{(a\dot{a} + b\dot{b})^2}{\zeta^3} - \frac{a\ddot{a} + b\ddot{b} + \dot{a}^2 + \dot{b}^2}{\zeta}, \quad \dot{\zeta}_a = \frac{\partial \dot{\zeta}}{\partial a} = -\left[\frac{\dot{a}}{\zeta} + \frac{a(a\dot{a} + b\dot{b})}{\zeta^3}\right],$$
$$\dot{\zeta}_b = \frac{\partial \dot{\zeta}}{\partial b} = -\left[\frac{\dot{b}}{\zeta} + \frac{b(a\dot{a} + b\dot{b})}{\zeta^3}\right], \quad \dot{\zeta}_{\dot{a}} = -\frac{a}{\zeta}, \quad \dot{\zeta}_{\dot{b}} = -\frac{b}{\zeta},$$
$$\frac{d}{dt}\dot{\zeta}_{\dot{a}} = -\left(\frac{\dot{a}}{\zeta} - \frac{a\dot{\zeta}}{\zeta^2}\right), \quad \frac{d}{dt}\dot{\zeta}_{\dot{b}} = -\left(\frac{\dot{b}}{\zeta} - \frac{b\dot{\zeta}}{\zeta^2}\right).$$

Now applying the relationship (*3.1*) for the variable q_1, we obtain the evolution in time of a:

$$\ddot{a} = -\ddot{x} - \ddot{z}\dot{\zeta}_{\dot{a}} - \dot{z}\frac{d}{dt}\dot{\zeta}_{\dot{a}} - \ddot{\zeta}\dot{\zeta}_{\dot{a}} - \dot{\zeta}\frac{d}{dt}\dot{\zeta}_{\dot{a}} + (\dot{z} + \dot{\zeta})\dot{\zeta}_a - g\zeta_a. \tag{3.10}$$

Following the same path for the variable q_2, we obtain the evolution in time of b:

$$\ddot{b} = \ddot{y} + \ddot{z}\dot{\zeta}_{\dot{b}} + \dot{z}\frac{d}{dt}\dot{\zeta}_{\dot{b}} + \ddot{\zeta}\dot{\zeta}_{\dot{b}} + \dot{\zeta}\frac{d}{dt}\dot{\zeta}_{\dot{b}} + (\dot{z} + \dot{\zeta})\dot{\zeta}_b - g\zeta_b. \tag{3.11}$$

The result obtained describes the dynamics of the position of the rod, but it is not in the most simplified version, which can be found just substituting in the equations (*3.10*) and (*3.11*) the values associated to the compact notation previously introduced. If this further step is done, the ultimate form of these relationships is achieved. Then to be consistent with the model of the drone, we must introduce the right amount of states that allow us to summarize the dynamics of the pole:

$$\begin{pmatrix} a \\ V_a \\ b \\ V_b \end{pmatrix} = \begin{pmatrix} x_{13} \\ x_{14} \\ x_{15} \\ x_{16} \end{pmatrix}.$$

Now, it is possible to write four more equations that will be part of the final dynamics of the system:

$$\begin{cases} \dot{x}_{13} = x_{14} \\ \dot{x}_{14} = \frac{1}{(L^2 - x_{15}^2)\zeta^2}\{-x_{13}^4\dot{x}_2 - (L^2 - x_{15}^2)^2\dot{x}_2 - 2x_{13}^2\left[x_{14}x_{15}x_{16} + (-L^2 + x_{15}^2)\dot{x}_2\right] + \\ \quad + x_{13}^3\left[x_{16}^2 + x_{15}\dot{x}_{16} - \zeta(g + \dot{x}_6)\right] + x_{13}[-L^2x_{15}\dot{x}_{16} + x_{15}^3\dot{x}_{16} + x_{15}^2(x_{14}^2 - \zeta(g + \\ \quad + \dot{x}_6)) + L^2(-x_{14}^2 - x_{16}^2 + \zeta(g + \dot{x}_6))]\} \\ \dot{x}_{15} = x_{16} \\ \dot{x}_{16} = \frac{1}{(L^2 - x_{13}^2)\zeta^2}\{-x_{15}^4\dot{x}_4 - (L^2 - x_{13}^2)^2\dot{x}_4 - 2x_{15}^2\left[x_{13}x_{14}x_{16} + (-L^2 + x_{13}^2)\dot{x}_4\right] + \\ \quad + x_{15}^3\left[x_{14}^2 + x_{13}\dot{x}_{14} - \zeta(g + \dot{x}_6)\right] + x_{15}[-L^2x_{13}\dot{x}_{14} + x_{13}^3\dot{x}_{14} + x_{13}^2(x_{16}^2 - \zeta(g + \\ \quad + \dot{x}_6)) + L^2(-x_{14}^2 - x_{16}^2 + \zeta(g + \dot{x}_6))]\} \end{cases}$$

With this set of equations, the modelling of the pole is completed [6]. Even if we have an algebraic loop between $\dot{x}_{14}$ and $\dot{x}_{16}$, this can be solved in *Simulink* introducing a memory block, otherwise a manual resolution, as seen for the 2D case, would require much heavier calculations.

The 3D Complete Modelling: Summarizing now the results achieved in the two preceding paragraphs, we can write a unique system that contains a unique vector for the states, a unique vector for the inputs and the whole dynamics of the pole balancing drone.

$$\begin{pmatrix} x \\ V_x \\ y \\ V_y \\ z \\ V_z \\ \phi \\ \theta \\ \psi \\ \omega_{x_B} \\ \omega_{y_B} \\ \omega_{z_B} \\ a \\ V_a \\ b \\ V_b \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{15} \\ x_{16} \end{pmatrix}, \qquad \begin{pmatrix} T \\ \tau_x \\ \tau_y \\ \tau_z \end{pmatrix} = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix},$$

$$
\begin{cases}
\dot{x}_1 = x_2 \\
\dot{x}_2 = -\frac{u_1}{m}(\cos x_7 \sin x_8 \cos x_9 + \sin x_7 \sin x_9) \\
\dot{x}_3 = x_4 \\
\dot{x}_4 = -\frac{u_1}{m}(\cos x_7 \sin x_8 \sin x_9 - \sin x_7 \cos x_9) \\
\dot{x}_5 = x_6 \\
\dot{x}_6 = -g - \frac{u_1}{m}(\cos x_7 \cos x_8) \\
\dot{x}_7 = x_{10} + \tan x_8(x_{11} \sin x_7 + x_{12} \cos x_7) \\
\dot{x}_8 = x_{11} \cos x_7 - x_{12} \sin x_7 \\
\dot{x}_9 = x_{11} \frac{\sin x_7}{\cos x_8} + x_{12} \frac{\cos x_7}{\cos x_8} \\
\dot{x}_{10} = \frac{\tau_x + x_{11}x_{12}(J_z - J_y)}{J_x} \\
\dot{x}_{11} = \frac{\tau_y + x_{10}x_{12}(J_x - J_z)}{J_y} \\
\dot{x}_{12} = \frac{\tau_z + x_{10}x_{11}(J_y - J_x)}{J_z} \\
\dot{x}_{13} = x_{14} \\
\dot{x}_{14} = \frac{1}{(L^2 - x_{15}^2)\zeta^2}\{-x_{13}^4 \dot{x}_2 - (L^2 - x_{15}^2)^2 \dot{x}_2 - 2x_{13}^2 \left[x_{14}x_{15}x_{16} + (-L^2 + x_{15}^2)\dot{x}_2\right] + \\
\quad + x_{13}^3 \left[x_{16}^2 + x_{15}\dot{x}_{16} - \zeta(g + \dot{x}_6)\right] + x_{13}[-L^2 x_{15}\dot{x}_{16} + x_{15}^3 \dot{x}_{16} + x_{15}^2(x_{14}^2 - \zeta(g + \\
\quad + \dot{x}_6)) + L^2(-x_{14}^2 - x_{16}^2 + \zeta(g + \dot{x}_6))]\} \\
\dot{x}_{15} = x_{16} \\
\dot{x}_{16} = \frac{1}{(L^2 - x_{13}^2)\zeta^2}\{-x_{15}^4 \dot{x}_4 - (L^2 - x_{13}^2)^2 \dot{x}_4 - 2x_{15}^2 \left[x_{13}x_{14}x_{16} + (-L^2 + x_{13}^2)\dot{x}_4\right] + \\
\quad + x_{15}^3 \left[x_{14}^2 + x_{13}\dot{x}_{14} - \zeta(g + \dot{x}_6)\right] + x_{15}[-L^2 x_{13}\dot{x}_{14} + x_{13}^3 \dot{x}_{14} + x_{13}^2(x_{16}^2 - \zeta(g + \\
\quad + \dot{x}_6)) + L^2(-x_{14}^2 - x_{16}^2 + \zeta(g + \dot{x}_6))]\}
\end{cases}
\tag{3.12}
$$

If we observe the equations related to the pole, then the last four, it is possible to notice that no control inputs are included. Nevertheless, it is wrong to say that it is not possible to control the dynamics of the rod, indeed in the equations of $\dot{x}_{14}$ and $\dot{x}_{16}$ enter $\dot{x}_2$, $\dot{x}_4$ and $\dot{x}_6$ that are influenced by the control inputs.

About the last four relationships, we must make another important comment: the model obtained is valid and accurate only if the rod is kept over the drone. Indeed, as soon as the pole goes under the plane of the quadcopter, the values of a and b should decrease accordingly, but this is not taken into consideration in the model. The solution of this problem will be achieved thanks to the control, which must ensure to keep the pole always over the drone.

3.4 Linearization of the Model

The set of equations (*3.12*) is clearly a non-linear model, therefore if we want to apply the known control theory, which is developed for linear system only, we are forced to linearize the system (*3.12*). Before doing this step is worth spending some words about the theory behind the linearization procedure.

3.4.1 Linearization of Differential Equation Models

For definition, in mathematics, linearization is finding the linear approximation to a function at a given point. The linear approximation of a function is the first order Taylor expansion around the point of interest. In the study of dynamical systems, linearization is a method for assessing the local stability of an equilibrium point of a system of nonlinear differential equations or discrete dynamical systems. Then, if we want to express these concepts through a mathematical approach we can define a general differential equation as follows:

$$\dot{x} = f(x, u),$$

where it x is a state and u a control input.
If we are interested in linearizing $f(x, u)$, an equilibrium point is needed. For definition we can assess:

$$(x^*, u^*) \text{ is an equilibrium point } \iff f(x^*, u^*) = 0.$$

This is reasonable because if (x^*, u^*) is an equilibrium point means that no variation of $\dot{x}$ should occur around that spot, then $\dot{x} = 0$. Note that (x^*, u^*) is not time-dependent.
What is time-dependent is the state $x(t)$ and the control input $u(t)$, and it is possible to rewrite them putting in evidence their relationship with the equilibrium point:

$$x(t) = x^* + \delta x(t), \quad u(t) = u^* + \delta u(t),$$

where $\delta x(t)$ is the variation of $x(t)$ from x^* and $\delta u(t)$ is the variation of $u(t)$ form u^*. Note that from now on the time dependence will be neglected in the notation for sake of simplicity.
An important consequence is highlighted if we compute the time-derivative of these quantities (remind that for definition $\dot{x}^* = \dot{u}^* = 0$):

$$\dot{x} = \dot{x}^* + \dot{\delta x} = \dot{\delta x}, \quad \dot{u} = \dot{u}^* + \dot{\delta u} = \dot{\delta u}, \quad \text{and} \quad f(x, u) = f(x^* + \delta x, u^* + \delta u).$$

Now, we have all the elements to find the linear model applying the Taylor expansion:

$$\dot{x} = \dot{\delta x} = f(x^* + \delta x, u^* + \delta u) = f(x^*, u^*) + \frac{\partial f(x,u)}{\partial x}|_{x=x^*,\, u=u^*}(x - x^*) + \\ + \frac{\partial f(x,u)}{\partial u}|_{x=x^*,\, u=u^*}(u - u^*) + h.o.t,$$

where the *h.o.t* are *higher other terms* of the Taylor expansion, which obviously can be neglected if the goal is to linearize. Therefore, to simplify the above notation:

$$f(x^*, u^*) = 0, \quad A = \frac{\partial f(x,u)}{\partial x}|_{x=x^*,\, u=u^*}, \quad \delta x = x - x^*,$$

$$B = \frac{\partial f(x,u)}{\partial u}|_{x=x^*,\, u=u^*}, \quad \delta u = u - u^*.$$

As consequence, we have achieved a linear system not anymore in terms of x and u, but as function of their variations:

$$\dot{\delta x} = A\delta x + B\delta u.$$

If we use such a system to implement our control, we must be aware that a *feed-forward action* must be added to ensure the correct functioning of the controller. Indeed, the real system is not linear and to work properly needs the full control action $u = u^* + \delta u$.
Then to be consistent with it, we will inject in the system the contribution of u^* through a feed-forward contribution. Note that u^* is the control action needed by the system to remain in the equilibrium point x^*.
A further discussion on this topic will be developed in Chapter *4*.

3.4.2 Linearization of the Pole Balancing Drone

In the previous section, we have seen the theory to linearize a model and, even if everything was explained considering a system with a unique state and a unique control variable, the same approach can be generalized for a case with n states, then n differential equations, and m control inputs. The pole balancing quadcopter is one of these cases.
The first thing to do is to identify an equilibrium point for the system, this can be done by applying the definition:

$$(x^*, u^*) \text{ is an equilibrium point } \iff f(x^*, u^*) = 0,$$

in our case $f(x,u)$ is the system (*3.12*).
Then, we have to solve the equality, coming from the definition of equilibrium point, for each equation of the system (*3.12*). Doing this, we obtain the values for x^*:

$$\begin{pmatrix} x \\ V_x \\ y \\ V_y \\ z \\ V_z \\ \phi \\ \theta \\ \psi \\ \omega_{x_B} \\ \omega_{y_B} \\ \omega_{z_B} \\ a \\ \dot{a} \\ b \\ \dot{b} \end{pmatrix} = \begin{pmatrix} p_x \\ 0 \\ p_y \\ 0 \\ p_z \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

The result obtained is crucial. As matter of fact, the vector x^* is uniquely determined for all the states, except for the position of the drone (x, y, z). For these states, any point can act as equilibrium point. This means that the linearization is independent of the position, then if we develop a controller suitable for this equilibrium point it will work whatever the space position of the drone is. The position in which the quadcopter is in equilibrium is called *hovering*.
For more complex systems can happen that the linearization depends on the position, so more advanced techniques, such as the *gain scheduling* (that is a topic not discussed in this paper), have to be implemented.
The vector x^* has been determined, now we are interested in the value of the vector u^*. This can be calculated substituting in the differential equations of $\dot{x}_6$, $\dot{x}_{10}$, $\dot{x}_{11}$ and $\dot{x}_{12}$ the value just computed of x^*:

$$\begin{aligned} \dot{x}_6 &= -g - \frac{u_1}{m}(\cos x_7 \cos x_8)|_{x=x^*} \implies u_1^* = T^* = -mg, \\ \dot{x}_{10} &= \frac{\tau_x + x_{11}x_{12}(J_z - J_y)}{J_x}|_{x=x^*} \implies u_2^* = \tau_x^* = 0, \\ \dot{x}_{11} &= \frac{\tau_y + x_{10}x_{12}(J_x - J_z)}{J_y}|_{x=x^*} \implies u_3^* = \tau_y^* = 0, \\ \dot{x}_{12} &= \frac{\tau_z + x_{10}x_{11}(J_y - J_x)}{J_z}|_{x=x^*} \implies u_4^* = \tau_z^* = 0. \end{aligned}$$

The result can be summarized as:

$$u^* = \begin{pmatrix} -mg \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

The final step of the linearization requires to find A and B.
In the previous section, we have seen their definition for a single state system, but we can generalize the result for a scenario with n states and m inputs. In this case, we obtain two matrices:

$$A = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ldots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}_{(x^*,u^*)} \quad B = \begin{pmatrix} \frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} & \cdots & \frac{\partial f_1}{\partial u_m} \\ \frac{\partial f_2}{\partial u_1} & \frac{\partial f_2}{\partial u_2} & \cdots & \frac{\partial f_2}{\partial u_m} \\ \vdots & \vdots & \ldots & \vdots \\ \frac{\partial f_n}{\partial u_1} & \frac{\partial f_n}{\partial u_2} & \cdots & \frac{\partial f_n}{\partial u_m} \end{pmatrix}_{(x^*,u^*)} \tag{3.13}$$

In our case $n = 16$ and $m = 4$, therefore we expect a matrix A_{16x16} and a matrix B_{16x4}. Applying the formulas (*3.13*) to the set of equations (*3.12*) allow us to find the matrices of the linearized model quite easily. In fact, a problem occurs when we have to compute the $14th$ and $16th$ row of the two matrices. Indeed, as already mentioned, we have an *algebraic loop* between $\dot{x}_{14}$ and $\dot{x}_{16}$, that gives as some troubles if we want to compute a derivative with respect to x_i or u_i. To resolve this problem a good approach is to exploit a piece of software, for example *Mathematica* or *Matlab*, to solve the loop and compute the derivative. Having done this, the results we obtained are the following:

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -g & 0 & 0 & 0 & 0 & \frac{g}{L} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{g}{L} & 0 \end{pmatrix}_{16x16},$$

$$B = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{1}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{1}{J_x} & 0 & 0 \\ 0 & 0 & \frac{1}{J_y} & 0 \\ 0 & 0 & 0 & \frac{1}{J_z} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}_{16x4}.$$

Once that we have computed these two matrices, the linearization of the pole balancing drone is completed and we have all the tools necessary to design a robust controller.

3.5 Analysis of Controllability and Observability

Before starting the discussion of the implemented control techniques, it is worth analyzing the controllability and the observability of the system.

Analysis of Controllability: To study the controllability, we have already all the information needed, because we have just calculated the couple (A, B). In this case, we can rely on the Theorem *2.1* as we have a continuous-time systems.
Then, using *Matlab* is quite easy, indeed we can exploit the command $M_r = ctrb(A, B)$. The output is the *Reachability Matrix*, so with the function $rank(M_r)$ we found out that the system is completely controllable as M_r has a rank equal to 16.
A completely controllable system allows us to modify its behavior with the introduction of an appropriate controller.

Analysis of Observability To introduce an observer in the system, it is necessary that this last one is completely observable. Actually, as it will be explained in Chapter *4*, our observability analysis is limited to the inverted pendulum, because the observer for the drone has already been developed by the Center for Research on Complex Automated Systems of Bologna.
As seen in Chapter *2* the study of the observability can be associated to the

couple (A, C). A is known, whereas C must be define according the measurable outputs. In the laboratory, where this project was thought to be implemented, it is available a VICON motion capture system that we can exploit to track the position of the pole on the drone. Knowing this, it is clear that the outputs available regarding the pole are the position a and b, but not the velocities associated, that will be reconstructed with the implementation of the state observer.

$$C_p = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}_{2x4}$$

The matrix C_p above has only four columns because we are focusing our analysis on the pole.
Similarly to what we have done for the controllability, the study can be developed exploiting some *Matlab* commands. First of all, we computed the *Observability Matrix* doing $M_o = obsv(A, C_p)$, then the calculation of the rank is quite straightforward: $rank(M_o)$. The result obtained is equal to 4, this means that all the states associated with the inverted pendulum are observable.

Chapter 4

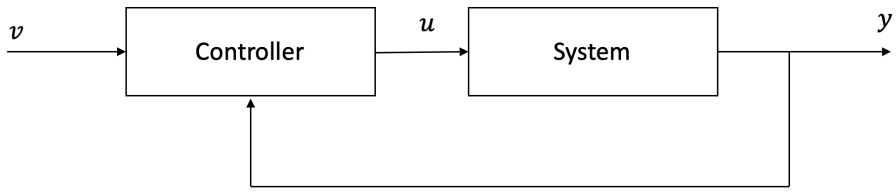
The Control System

Very roughly speaking, control system design deals with the problem of making a concrete physical system behave according to certain desired specifications. The ultimate product of a control system design problem is a physical device that, if connected to the to be controlled physical system, makes it behave according to the specifications. This device is called a controller. In this chapter, different types of control techniques will be analyzed.

4.1 Full State Feedback

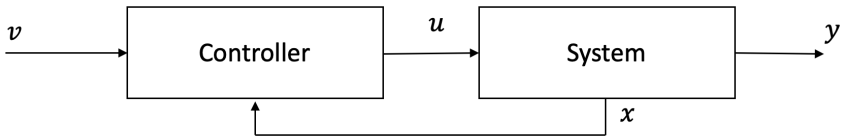
Full state feedback (FSF), or pole placement, is a method employed in feedback control system theory to place the closed-loop poles of a plant in predetermined locations in the s-plane. Placing poles is desirable because the location of the poles corresponds directly to the eigenvalues of the system, which control the characteristics of the response of the system. The system must be considered *controllable* in order to implement this method.
The classical control scheme is based on a feedback action that is applied to the output of the system y.

FIGURE 4.1: Classical control scheme



Instead, the full state feedback still implements a feedback action, but applied to the states. Assuming that we know the states, the control scheme is summarized by Figure 4.2.

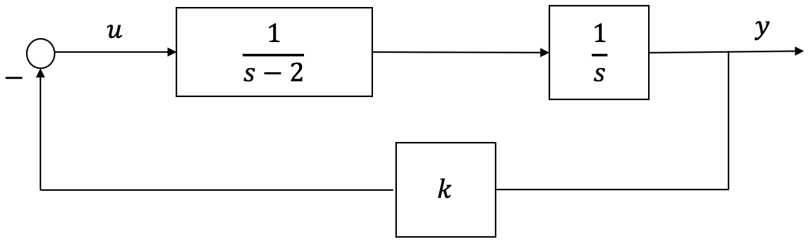
FIGURE 4.2: Full state feedback scheme



We will see now a brief example just to appreciate the differences between these two techniques.

Example 1. With reference to Figure *4.3*, let's suppose to have the possibility to tune k as we would like to stabilize the system.

FIGURE 4.3: System to stabilize

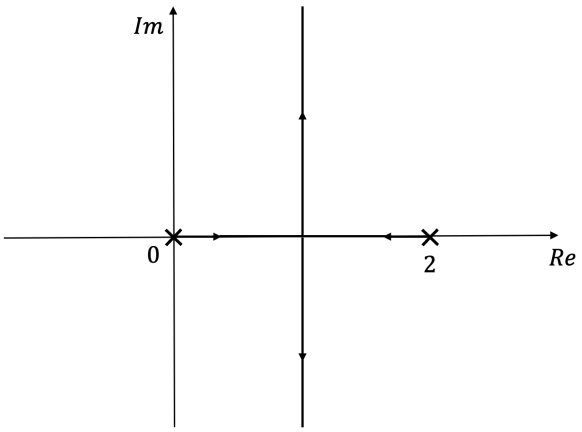


To do this analysis the easiest tool is the root locus.

$$L(s) = \frac{k}{s(s-2)}.$$

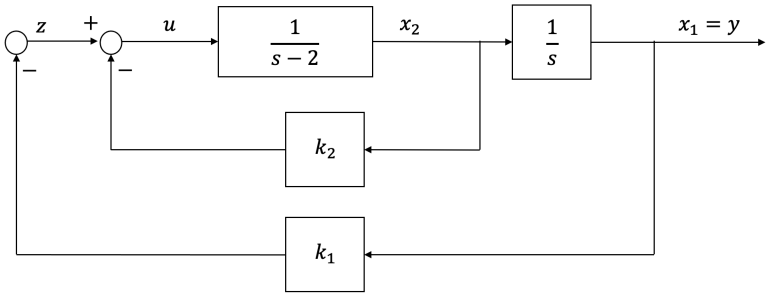
From the graph of the root locus (Figure *4.4*) is easy to understand that for any $k > 0$, there is always at least a pole of the closed-loop function that lies on right-hand side of the plane. Therefore, it is not possible to stabilize this system applying exclusively a constant gain k.

FIGURE 4.4: Root locus related to Figure *4.3*



Now, under the hypothesis that the values of both states are available, we can try to apply a full state feedback control. The scheme obtained in this case is illustrated in Figure *4.5*.

FIGURE 4.5: Full state feedback diagram



In this case the control input is $u(t) = -k_1 x_1(t) - k_2 x_2(t)$. Now, analyzing the inner loop, we can determine its transfer function:

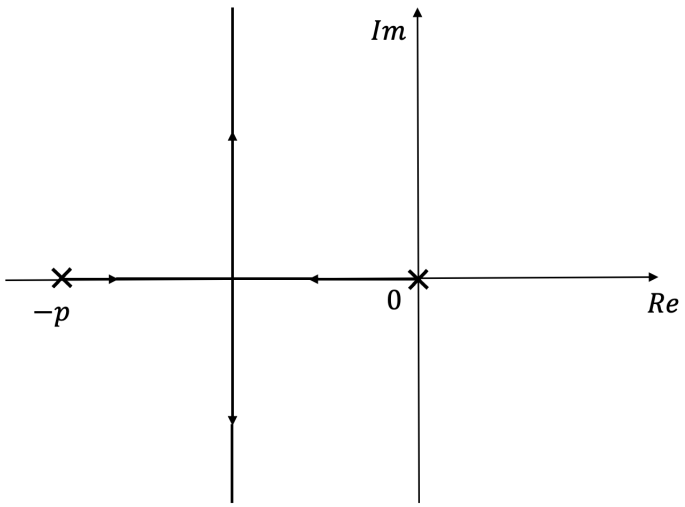
$$F_2(s) = \frac{1}{s+p},$$

where $p = k_2 - s$. Then, the first pole can be freely moved tuning k_2.
The outer loop instead has a more complex transfer function:

$$L(s) = \frac{k_1}{s} F_2(s) = \frac{k_1}{s(s+p)}.$$

We can also plot the root locus of this second case, and it is pretty clear that the results will be much better. Indeed, with a FSF controller we have two control variables that allow us to stabilize easily the system.

FIGURE 4.6: Root locus related to Figure *4.5*



Another fundamental characteristic, not highlighted by this example, of the FSF is that it can be applied to MIMO system too.

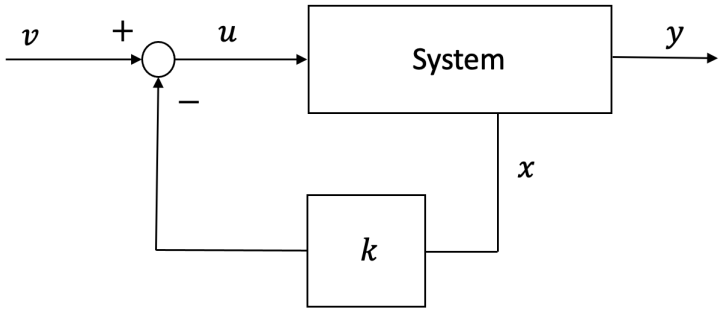
4.1.1 Pole Placement of a System with Measurable States

What we are now interested in is, given a generic system (*4.1*) (*4.2*), understanding if it is possible to arbitrarily place its pole to control the dynamics. For the following system, we will consider that all the states are available and measurable, then we will have no problem implementing an FSF controller.

$$\dot{x}(t) = Ax(t) + Bu(t), \tag{4.1}$$
$$y(t) = Cx(t) + Du(t). \tag{4.2}$$

Let's suppose to know all the states, then the control law can be chosen as $u(t) = -Kx(t) + v(t)$, where $K = [k_0 \ k_1 \ \dots k_{n-1}]$ is a row vector, and the block diagram of the system is depicted in Figure *4.7*.

FIGURE 4.7: General system



Having defined this, if the control law is substituted in the equation (*4.1*), the system is described by the following equation:

$$\dot{x} = (A - BK)x(t) + Bv(t) = Fx(t) + Bv(t), \tag{4.3}$$

defining $F = A - BK$.
Now, the goal is to understand if it is possible to assign arbitrary eigenvalues to the matrix F, tuning the values of K. Indeed, it is known that the eigenvalues of the matrix A of a system like (*4.1*) corresponds to the position of the poles of the system itself. The solution to this problem is given by the following theorem:

Theorem 4.1.
Given two real matrices A and B and an arbitrary set $\Lambda^o = [\lambda_1^0, \lambda_2^0, \dots, \lambda_n^0]$ of complex numbers (reals or couple of complex conjugates). Necessary and sufficient condition to be able to find a real matrix K, such that the eigenvalues of $F = A - BK$ coincide with the set Λ^o is that the couple (A, B) is completely reachable [2].

Obviously, if in a system a state is not reachable, it means that we cannot influence it with the control law $u(t)$, then we do not have the possibility of moving its associated eigenvalue.
So, if we are working with a system that is not completely reachable there are two possible scenarios:

- The eigenvalues that are *not reachable* are stable, so even if we cannot move them as we would like, the system can be stabilized.
- The eigenvalues that are *not reachable* are not stable. This is the worst case because is not even possible to stabilize the system.

Let's see now an example of a possible technique that is useful if we want to perform a pole placement.

Example 2. Consider a system given by the following state-space equations:

$$\dot{x} = \begin{pmatrix} 0 & 1 \\ -2 & -3 \end{pmatrix} x + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u.$$

The uncontrolled system has open-loop poles at $s = -1$ and $s = -2$. These poles are the eigenvalues of the matrix A and they are the roots of $|sI - A|$. Let's suppose that we wish the controlled system eigenvalues to be located at $s = -1$ and $s = -5$, which are not the poles we currently have. The desired characteristic equation is then $s^2 + 6s + 5 = 0$, from $(s+1)(s+5)$. Following the procedure given above, the FSF controlled system characteristic equation is:

$$|sI - (A - BK)| = det \begin{bmatrix} s & -1 \\ 2 + k_1 & s + 3 + k_2 \end{bmatrix} = s^2 + (3 + k_2)s + (2 + k_1),$$

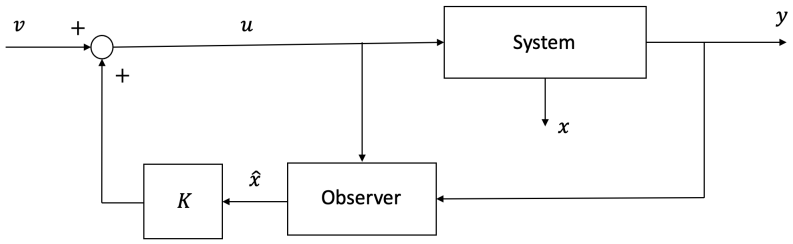
where $K = [k_1,\ k_2]$.
Upon setting this characteristic equation equal to the desired characteristic equation, we find $K = [3,\ 3]$. Therefore, setting $u = -Kx$ forces the closed-loop poles to the desired locations, affecting the response as desired.
This only works for Single-Input systems. Multiple input systems will have a K matrix that is not unique. Choosing, therefore, the best K values is not trivial. A linear-quadratic regulator (explained in the following sections) might be used for such applications.

Another possibility that can be exploited to locate arbitrarily the poles of a system is through the *Matlab* command *K = place(A, B, P)*, where P is a polynomial whose solutions coincide with the desired positions for the poles.

4.1.2 Pole Placement of a System with Non-Measurable States

The analysis developed in the previous paragraph is now extended for a more general case. Indeed, also for the pole balancing drone, we will see that not all the states are measurable, then the smartest solution is to implement an observer that reconstructs the missing states. (Note that in Figure *4.8* the feedback has a positive sign).

FIGURE 4.8: System with FSF controller and observer



In Figure *4.8* we have combined a full state feedback controller with an observer, so we have to understand which properties such a system has. Let's quickly recap the equations that describe the observer, in particular the observer with arbitrary dynamics:

$$\dot{\hat{x}}(t) = A\hat{x}(t) + Bu(t) + H(\hat{y}(t) - y(t)), \quad \hat{x}(0) = \hat{x}_0, \tag{4.4}$$
$$\hat{y}(t) = C\hat{x}(t) + Du(t). \tag{4.5}$$

The control law is slightly different than before as now it relies on the estimate states $\hat{x}$:

$$u(t) = K\hat{x}(t) + v(t). \tag{4.6}$$

If we substitute the equation (*4.6*) in the relationships (*4.1*) and (*4.4*), exploiting also the equations (*4.2*) and (*4.5*), we get:

$$\dot{x}(t) = Ax(t) + BK\hat{x}(t) + Bv(t), \tag{4.7}$$
$$\dot{\hat{x}}(t) = (A + BK + HC)\hat{x}(t) - HCx(t) + Bv(t). \tag{4.8}$$

Subtracting from the second the first one, and keeping in mind the definition of $e(t)$ seen in Chapter *2*, we obtain:

$$\dot{e}(t) = \dot{\hat{x}}(t) - \dot{x}(t) = (A + HC)\hat{x}(t) - (A + HC)x(t) = (A + HC)e(t). \tag{4.9}$$

The next step is to rewrite the relationship (*4.7*) taking into consideration that $\hat{x} = x(t) + e(t)$.

$$\dot{x} = (A + BK)x(t) + BKe(t) + Bv(t). \tag{4.10}$$

To conclude, according to equations (*4.9*) and (*4.10*), and defining a new extended state

$$z(t) = \begin{bmatrix} x(t) \\ e(t) \end{bmatrix},$$

the system in Figure *4.8* can be represented by the following equation:

$$\dot{z}(t) = \begin{bmatrix} A + BK & BK \\ 0 & A + HC \end{bmatrix} z(t) + \begin{bmatrix} B \\ 0 \end{bmatrix} v(t) = \bar{A}z(t) + \bar{B}v(t).$$

Note that the matrix $\bar{A}$ is a *block triangular matrix*, then its eigenvalues are the union of the eigenvalues of $F = A + BK$ and of $N = A + HC$. So, it is correct

to state that:

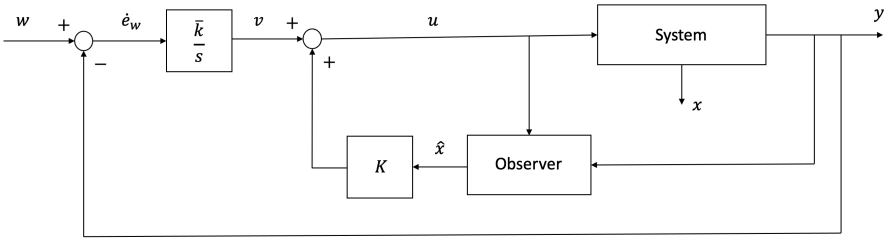
- The overall system is asymptotically stable if and only if F and N are asymptotically stable;
- If the initial system is completely reachable and observable, the eigenvalues of the overall system can be placed arbitrarily.
- The design of the gains K and H can be done separately. In particular, when we choose K it is possible to consider the whole state $x(t)$ as measurable, and H can be chosen considering the system to be estimated as an open-loop system. This important result is called *separation principle*.

Usually, a good design tries to make the dynamics of the observation error faster than the dynamics of the system. In this way, the error converges to zero immediately and the controller of the system can rely on accurate estimates of the states.

4.1.3 Controller with Integral Action

If we consider the variable $v(t)$ as reference signal, the system depicted in Figure *4.8* does not guarantee any static precision. This means that in static conditions the perfect tracking of the reference signal is not ensured. To get this important result, we can modify the control scheme seen for the pole placement adding an integral action, as shown in Figure *4.9*.

FIGURE 4.9: System with FSF controller and integral action



Thanks to the integrator, if the system has to follow a step $w(t)$, the tracking will have zero error when the transient is finished. To properly describe this system we need to use two equations (supposing for simplicity that $y = Cx$).

$$\dot{x}(t) = Ax(t) + Bu(t), \tag{4.11}$$
$$\dot{e}_w(t) = w(t) - y(t) = w(t) - Cx(t). \tag{4.12}$$

If we want to apply the FSF control for such a system, it is necessary to define an extended state that includes both $x(t)$ and $e_w(t)$:

$$x_e(t) = \begin{bmatrix} x(t) \\ e_w(t) \end{bmatrix}.$$

So, the relationships (*4.11*) and (*4.12*) can be compressed into a single equation

$$\dot{x}_e = \begin{bmatrix} A & 0 \\ C & 0 \end{bmatrix} x_e + \begin{bmatrix} B \\ 0 \end{bmatrix} u = A_e x_e + B_e u, \tag{4.13}$$

where the matrix C is used to select the states for which we are interest to have a feedback of the tracking error. Now, as seen for the classic FSF, we have to choose K to stabilize $(A_e - B_e K)$.

$$u(t) = \begin{bmatrix} K & \bar{k} \end{bmatrix} \begin{bmatrix} x(t) \\ e_w(t) \end{bmatrix}.$$

As soon as $(A_e - B_e K)$ becomes stable, means that all the states of the system, including $e(t)$, are constant. If $e(t)$ is constant, means that its derivative $\dot{e}(t) = w(t) - y(t)$ is null, then we are perfectly tracking the reference signal $w(t)$.

4.2 Control of Linearized Systems

At the end of Chapter *3*, we have seen the linearization of the drone model, this passage is essential because all the control theory we have studied is based on linear models.

$$\dot{\delta x} = A\delta x + B\delta u \quad \textit{(Equation of a Linearized system).}$$

To control linearized systems a couple of important considerations must be done. The first one is related to the concept of *domain of attraction*.

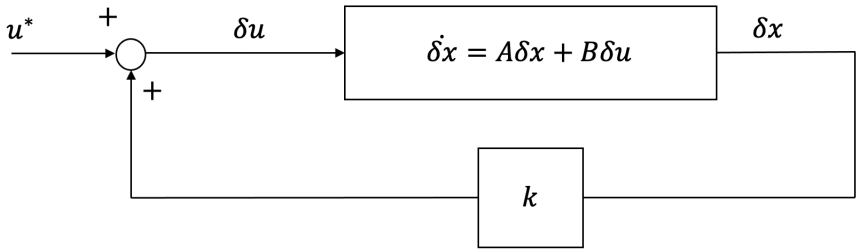
Definition 4.1 (Domain of Attraction)**.**
Given an equilibrium point x^*, it is called domain of attraction, the set of initial positions, around the equilibrium point, from which it is possible to stabilize the system. That is, if the system is out of the domain of attraction, the system cannot be stabilized in x^* and any linear approximation of the system obtained around x^* is no longer valid.

According to this definition, we will develop our controller. Indeed, the first goal of it is to maintain the drone inside the domain of attraction. Only in this case the linearization done in Chapter *3* is valid and can be used as model for the controller. How to ensure that the controller has this property will be discussed in the following sections of this chapter.
The second point to consider when we control a linearized system is related to the *feed-forward action* to ensure the right control action to the system.
Indeed, we have already discussed that, even if the linearized system requires only u^*, the real system needs also the contribution to maintain it in the equilibrium point:

$$u = u^* + \delta u.$$

Even if it is true that generally, a *feed-forward action* equal to u^* is necessary, it can be neglected whenever the controllers has also an integral action. As a matter of fact, the integral action is capable of canceling any type of constant error, including the one introduced to do the lack of u^*.

FIGURE 4.10: Feed-forward action u^*



4.3 Inverted Pendulum State Observer

In the last section of Chapter 2, we have seen the theory behind a state observer with arbitrary dynamics. Now, in this section, we will develop one for the inverted pendulum. Once again, we will recall the equations describing the observer, but this time without loss of generality we consider $D = 0$. This simplification is allowed because in our case $\hat{y}_p$ is nothing but the measurable states of the pendulum, selected through the matrix C_p.

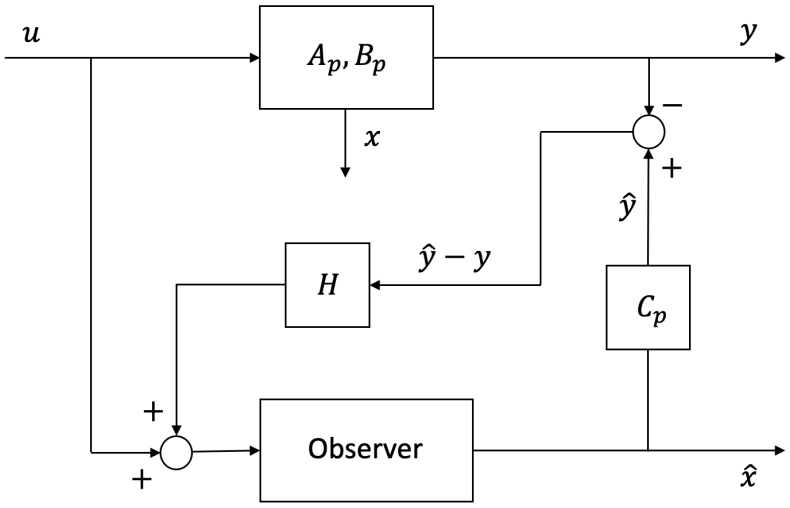
$$\dot{\hat{x}}_p(t) = A_p\hat{x}(t) + B_p u(t) + H(\hat{y}_p(t) - y_p(t)), \tag{4.14}$$
$$\hat{y}_p(t) = C_p\hat{x}(t), \tag{4.15}$$
$$\dot{e}_p(t) = (A_p + HC_p)e(t). \tag{4.16}$$

The subscript p stands for pendulum, as this analysis is concerned only about it.

$$A_p = \begin{pmatrix} 0 & 1 & 0 & 0 \\ \frac{g}{L} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{g}{L} & 0 \end{pmatrix}, \qquad B_p = \begin{pmatrix} 0 & 0 \\ 0 & -g \\ 0 & 0 \\ g & 0 \end{pmatrix}, \qquad C_p = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

The structure of C_p has already been explained; A_p is just the portion, concerning the pendulum, of the matrix A calculated in Chapter 3, whereas B_p has a more complex origin. Indeed, if we watch the matrix B calculated in Chapter 3, in the last four rows, the ones related to the inverted pendulum, there are only zeros. This means, that the control input does not affect directly the behavior of the pole. Indeed, as already mentioned, the states of the inverted pendulum depend on the state of the drone, then controlling the quadrotor we are also able to control the pole. If we consider the linearized system, then the rod depends on the *roll* and *pitch* angle, as can be deducted from the formulation of matrix A. For this reason, to obtain a more accurate observer, instead of using a null matrix B_p, we introduce in B_p the contribution of *roll* and *pitch* that, in this particular case, are working as u signal.
An actual *Simulink* scheme of the observer implemented is present in Chapter 5.

FIGURE 4.11: Pendulum observer



As explained in the theory section, our goal is to tune the eigenvalues of the matrix $N = A + HC$ and the fastest way to do it is through the *Matlab* command *place*. But, to use it, we must define a polynomial whose solutions are equal to the desired eigenvalues for the matrix N. The choice must be done carefully. For sure a faster dynamics is a good property because we will have the right values of the estimated states sooner. On the other hand, having poles further from the complex axis, that is a faster dynamics, means that if there are errors or noises on $\hat{y} - y$, these will be amplified.
For the inverted pendulum, two different observers have been developed, in this way the different performances are clear. The two observers are based on two different polynomials:

$$P_1 = (\lambda + 2)(\lambda + 5)(\lambda + 12)(\lambda + 1), \tag{4.17}$$
$$P_2 = (\lambda + 20)(\lambda + 50)(\lambda + 120)(\lambda + 10). \tag{4.18}$$

The performances and the implementation of these two state observers are shown in the following chapter.

4.4 Optimal Control, LQR and Pontryagin's Principle

Optimal control deals with the problem of finding a control law for a given system such that a certain optimality criterion is achieved. A control problem includes a cost functional that is a function of state and control variables. An optimal control is a set of differential equations describing the paths of the control variables that minimize the cost function. The optimal control can be derived using Pontryagin's maximum principle (a necessary condition also known as Pontryagin's minimum principle or simply Pontryagin's

Principle), or by solving the Hamilton–Jacobi–Bellman equation (a sufficient condition).
Let's now suppose to take a general (linear or non-linear) stationary system:

$$\dot{x} = f(x, u). \tag{4.19}$$

First of all, to talk about *optimal control* we must introduce the objective function J that has to be minimized by choosing the right value of $u(t)$.

$$J = \beta(x_f, t_f) + \int_{t_0}^{t_f} f_0(x(t), u(t))\, dt, \tag{4.20}$$

where $\beta(x_f, t_f)$ is a function of the final state and instant of the system.
The peculiarity of this control is that, through the cost function J, we can weigh differently the states and inputs of the system. A state (or an input) with a greater weight will be more penalized by the controller. This means, that the variation of a state (or an input) with a greater weight is less accepted if compared to the variation of a state (or an input) with a lower weight. Therefore, our controller will try to minimize the variation of well-weighted states (or inputs), whereas it will be more tolerant regarding variations of less-weighted states (or inputs).

We are interested in finding $u_{opt}(t)$ that denotes the optimal choice to minimize J. Actually, the same analysis can be set as a maximum problem just inverting the signs. For the further discussion, we will consider t_0 and $x(t_0)$ as known, whereas t_f and $x(t_f)$ as unknowns (this is the most general case). The process to find the expression of the optimal control action u_{opt} is quite long and complex, so only the main steps will be reported in this work; moreover to simplify the notation the time dependence of x, u and $\dot{x}$ will be neglected. The first step concern a modification in the expression of J:

$$\tilde{J} = \beta(x_f, t_f) + \int_{t_0}^{t_f} f_0(x, u) + \lambda^T(t)[f(x, u) - \dot{x}]\, dt. \tag{4.21}$$

Even if in $\tilde{J}$ there is a new term, $\tilde{J}$ can be considered equal to J because the quantity introduced is null if the condition of the equation (*4.19*) is respected. With this new expression of J we can introduce the *Hamiltonian function*:

$$H(x, \lambda, u) = f_0(x, u) + \lambda^T f(x, u).$$

Note that also for λ we are neglecting its time dependence.
Then, the objective function can be rewritten as:

$$\tilde{J} = \beta(x_f, t_f) + \int_{t_0}^{t_f} [H(x, \lambda, u) - \lambda^T \dot{x}]\, dt.$$

Once that the *Hamiltonian* has been introduced, we can now proceed with framing the problem to find the minimum of $\tilde{J}$. In this part, we will use δ to indicate a function variation and Δ for variations of numerical quantities.

The following equation expresses a variation of the objective function:

$$\Delta\tilde{J} = \beta(x_f + \Delta x_f, t_f + \Delta t_f) + \int_{t_0}^{t_f+\Delta t_f} [H(x+\delta x, \lambda+\delta\lambda, u+\delta u) + \\ - (\lambda^T + \delta\lambda^T)(\dot{x}+\delta\dot{x})]\, dt - \beta(x_f, t_f) - \int_{t_0}^{t_f} [H(x,\lambda,u) - \lambda^T\dot{x}]\, dt.$$

Then, applying the Taylor expansion, neglecting the *h.o.t terms* and making some simplifications that will not be discussed in detail, we get:

$$\Delta\tilde{J} = \frac{\partial\beta}{\partial x_f}\bigg|^T \Delta x_f + \frac{\partial\beta}{\partial t_f}\Delta t_f - \lambda_f^T\delta x_f + H_f\Delta t_f - \lambda_f^T\dot{x}_f\Delta t_f + \int_{t_0}^{t_f} \left[\frac{\partial H}{\partial x}\bigg|^T \delta x + \right. \\ \left. + \frac{\partial H}{\partial\lambda}\bigg|^T \delta\lambda + \frac{\partial H}{\partial u}\bigg|^T \delta u + \dot{\lambda}^T\delta x - \dot{x}^T\delta\lambda\right] dt.$$

Now, as our goal is to find the minimum, we impose $\Delta\tilde{J} = 0$. The best thing to do is to analyze separately the pieces that depend on different terms.

- Terms that are function of x_f:

$$\frac{\partial\beta}{\partial x_f}\bigg|^T \Delta x_f - \lambda_f^T(\delta x_f + \dot{x}_f\Delta t_f) = 0,$$

  from maths reasoning, it is possible to infer that $\Delta x_f = \delta x_f + \dot{x}_f\Delta t_f$, that allows us to rewrite the previous equation.

$$\frac{\partial\beta}{\partial x_f}\bigg|^T \Delta x_f - \lambda_f^T\Delta x_f = 0.$$

  Due to the fact that Δx_f is an arbitrary value that can assume every value, we can conclude that:

$$\frac{\partial\beta}{\partial x_f} = \lambda_f \qquad \textit{(1st Condition).} \tag{4.22}$$

- Terms that are function of Δt_f:

$$\left(\frac{\partial\beta}{\partial t_f} + H_f\right)\Delta t_f = 0.$$

  Due to the fact that Δt_f is an arbitrary value that can assume every value, we can conclude that:

$$\frac{\partial\beta}{\partial t_f} = -H_f \qquad \textit{(2nd Condition).} \tag{4.23}$$

- Terms that are function of δx:

$$\left.\frac{\partial H}{\partial x}\right|^T \delta x + \dot{\lambda}^T \delta x = 0.$$

Due to the fact that δx is an arbitrary value that can assume every value, we can conclude that:

$$\dot{\lambda} = -\frac{\partial H}{\partial x} \qquad \textit{(3rd Condition).} \tag{4.24}$$

- Terms that are function of $\delta\lambda$:

$$\left.\frac{\partial H}{\partial \lambda}\right|^T \delta\lambda - \dot{x}^T \delta\lambda = 0.$$

Due to the fact that $\delta\lambda$ is an arbitrary value that can assume every value, we can conclude that:

$$\dot{x} = \frac{\partial H}{\partial \lambda} \qquad \textit{(4th Condition).} \tag{4.25}$$

Note that if we develop the above derivative for H, we will obtain $f(x,u)$, this means that this condition is consistent with the initial relationship (*4.19*).

- Terms that are function of δu:

$$\left.\frac{\partial H}{\partial u}\right|^T \delta u = 0.$$

Due to the fact that δu is an arbitrary value that can assume every value, we can conclude that:

$$\frac{\partial H}{\partial u} = 0 \qquad \textit{(5th Condition).} \tag{4.26}$$

These five are the necessary conditions to have the optimal u_{opt}, according to these we will develop our control system.
The relationships (*4.24*) and (*4.25*) are also called *Euler–Lagrange equations*. Whereas the condition (*4.26*) is also known as *Pontryagin's Principle*.

Pontryagin's Principle: *The value of the Hamiltonian along the optimal trajectory has always the minimum value with respect to u. That is, u_{opt} is a minimum point for the Hamiltonian.*

One more condition can be determined by evaluating the time-derivative of the Hamiltonian.

$$\frac{dH}{dt} = \left.\frac{\partial H}{\partial x}\right|^T \dot{x} + \left.\frac{\partial H}{\partial \lambda}\right|^T \dot{\lambda} + \left.\frac{\partial H}{\partial u}\right|^T \dot{u}.$$

If we apply the conditions (*4.24*), (*4.25*) and (*4.26*) we can simplify the above expression.

$$\frac{dH}{dt} = -\dot{\lambda}^T \dot{x} + \dot{x}^T \dot{\lambda} = 0 \tag{4.27}$$

From here we can conclude that the Hamiltonian does not change in time.

4.4.1 Linear Quadratic Regulator (LQR)

The case where the system dynamics are described by a set of linear differential equations and the cost is described by a quadratic function is called the LQ problem. If the system under analysis is not linear, as we have already seen, we can linearize it and apply the following theory. The general cost function (*4.20*) can be rewritten for this particular case:

$$J = \frac{1}{2} x_f^T S_f x_f + \frac{1}{2} \int_0^{t_f} [x^T(\tau) Q x(\tau) + u^T(\tau) R u(\tau)] \, dt. \tag{4.28}$$

In this case, the first term is not a function of t_f. Another consideration about the above equation is related to coefficient $\frac{1}{2}$: due to the fact that we are solving a minimum problem, the presence of a constant coefficient does not change the final result, but it can be helpful to simplify calculations.
The most important observation to make, concerns the matrices Q, R and S_f:

- S_f is a positive-definite matrix.
- Q is a symmetric and positive semi-definite matrix.
- R is a symmetric and positive-definite matrix.

Having defined this, the resolution of the LQ problem can start:

$$H = \frac{1}{2} x^T Q x + \frac{1}{2} u^T R u + \lambda^T [Ax + Bu].$$

And from the expression of the Hamiltonian, it is possible to derive the *Euler–Lagrange equations*.

$$\dot{x} = \frac{\partial H}{\partial \lambda} = Ax + Bu, \tag{4.29}$$

$$\dot{\lambda} = -\frac{\partial H}{\partial x} = -(Qx + A^T \lambda). \tag{4.30}$$

The value of the optimal u can be obtained thanks to the relationship (*4.26*):

$$\frac{\partial H}{\partial u} = Ru + B^T \lambda = 0 \quad \Longrightarrow \quad u_{opt} = -R^{-1} B^T \lambda. \tag{4.31}$$

One step further is obtained combining the equation (*4.29*) with the equation (*4.31*):

$$\dot{x} = Ax - BR^{-1} B^T \lambda. \tag{4.32}$$

And we can also reorganize the Euler–Lagrange equations in matrix form.

$$\begin{bmatrix} \dot{x} \\ \dot{\lambda} \end{bmatrix} = \begin{bmatrix} A & -BR^{-1}B^T \\ -Q & A^T \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix}.$$

To solve these two sets of differential equations with need the boundary conditions: x_0 is known for hypothesis, x_f could be either known or not. For our analysis, we will consider the final state as unknown. Exploiting the condition (4.22), noting that in this context $\beta(x_f) = \frac{1}{2}x_f^T S_f x_f$ we can find a relationship for λ_f:

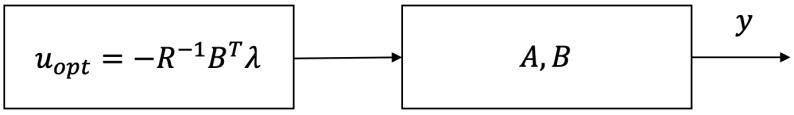
$$\lambda_f = S_f x_f,$$

and the above equation can be used as boundary condition.
If until now we have considered t_f as a finite quantity, for our purpose is better to consider it as infinite. Indeed, the whole project has been developed imposing the asymptotic stability, that is a property obtained for $t \to \infty$. With this new condition the objective function loses a term:

$$J = \frac{1}{2}\int_0^\infty (x^T Q x + u^T R u)\, dt$$

The term $\beta(x_f)$ has been canceled because for $t \to \infty$ a stable system converges, then $x \to 0$ (for the moment we are considering exclusively systems that converge to zero, later this analysis will be generalized). Therefore $x_f = 0$.
An LQR where $t \to \infty$, it is called *infinite horizon linear quadratic regulator*.
The main problem we are facing now is that the expression for u_{opt} we got is not a function of $x(t)$, this means that the control we are applying is an open-loop regulator. The disadvantages of an open-loop control are well known, so it is important to find a way to express u_{opt} in a closed-loop form.

FIGURE 4.12: u_{opt} in open-loop



To ensure us that a description of the optimal control action in state-feedback is possible is the *Bellman's principle of optimality*.

Bellman's Principle of Optimality: *The principle says that if we have found the optimal trajectory on the interval from $[t_0, t_f]$, by solving the optimal control problem on that interval, the resulting trajectory is also optimal on all sub-intervals of this interval of the form $[t, t_f]$ with $t > t_0$, provided that the initial condition at time t was obtained from running the system forward along the optimal trajectory from time t_0 [8].*

What we are searching is something like $u_{opt} = f(x)$. In this way, the control law would be a function of the states. But, it is important to mention that the solution to an optimization problem is unique, then even if a new relationship is found, this represents the same solution contained in the relationship (*4.31*) but expressed in a different way.
Thanks to the *Bellman's principle of optimality* we can assume as true the equation (*4.33*):

$$\lambda(t) = S(t)x(t) \quad then \tag{4.33}$$

$$\dot{\lambda} = \dot{S}(t)x(t) + S(t)\dot{x}(t), \tag{4.34}$$

where obviously $S(t)$ is a matrix still to be determined (in the next step its time-dependence will be sometimes neglected). Now equating the relationships (*4.30*) and (*4.34*), we get:

$$\dot{\lambda} = \dot{S}x(t) + S\dot{x}(t) = -(Qx + A^T\lambda).$$

Let's now combine the equation (*4.32*) with the equation (*4.33*) and we will manipulate a little the expression.

$$(\dot{S} + SA + A^TS - SBR^{-1}B^TS + Q)x(t) = 0.$$

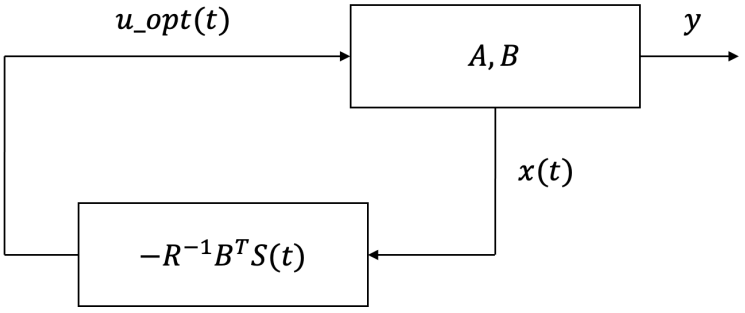
This expression must be valid for each $x(t)$ because according to different initial conditions the $x(t)$ obtained as result of u_{opt} can change. As consequence:

$$\dot{S}(t) + S(t)A + A^TS(t) - S(t)BR^{-1}B^TS(t) + Q = 0. \tag{4.35}$$

The above expression is known as *Dynamic Riccati Equation*, and it a nonlinear differential equation in matrix form.
One more consequence of the relationship (*4.33*) is the possibility to express the control action in a closed-loop form, as previously anticipated.

$$u_{opt} = -R^{-1}B^T\lambda = -R^{-1}B^TS(t)x(t). \tag{4.36}$$

FIGURE 4.13: $u_{opt}(t)$ in closed-loop



This closed-loop form works if we know the matrix $S(t)$. To find it we have to

find the solution of the *Dynamic Riccati Equation*. To solve it, as it is a differential equation, we need the boundary conditions. For the moment a boundary condition is $\lambda_f = S_f x_f$ obtained for the time instant t_f. The problem is that, in this case, to solve the equation we should integrate backward in time. So if the boundary condition is referred to the final instant, it means that we cannot solve the problem in real-time, during the evolution of the system, but everything must be calculated a priori. This is obviously complex as we do not know which is the evolution of the system in advance. Moreover, $S(t)$ is not a constant quantity.
The solution of these issues is obtained considering, as already done previously, an infinite time horizon, that is $t \to \infty$. In this scenario is possible to demonstrate that:

$$\lim_{t\to\infty} S(t) = S_\infty.$$

So, we can rewrite the *Dynamic Riccati Equation* for this particular case.

$$S_\infty A + A^T S_\infty - S_\infty B R^{-1} B^T S_\infty + Q = 0 \tag{4.37}$$

This new form of the equation (*4.35*) is known as *Algebraic Riccati Equation* (ARE).
Then, for an infinite time horizon problem, the block diagram of the control action (Figure *4.13*) have to be modified substituting $S(t)$ with S_∞.
It is possible to compute S_∞ through analytic passages, but it is not an easy task, then the best solution is to exploit once again *Matlab* and its command *are* that returns the stabilizing solution (if it exists) to the (algebraic) continuous-time Riccati equation.
Before concluding with the LQR, we have to resolve one more question. Indeed, the analysis shown in the previous pages is based on a fundamental assumption:

$$\textit{if} \quad t \to \infty \quad \textit{then} \quad u(t), x(t) \to 0.$$

Actually, when we have to control a system, this is not the only possible scenario. For example, when we control the position of the drone is unlikely to ask it always to reach the point $(0, 0, 0)$. The solution is quite easy. It starts from a review of some definitions:

$$x(t) = x^* + \xi(t),$$
$$u(t) = u^* + \nu(t),$$

Where x^* is an equilibrium point, u^* the control input to keep the system in x^*, $\xi(t)$ and $\nu(t)$ are respectively the variation of the state and of the input from x^* and u^*.

$$J = \frac{1}{2}\int_0^\infty [\xi^T(t) Q \xi(t) + \nu^T(t) R \nu(t)]\, dt.$$

Realizing an LQR for such an objective function generates a controller that is able to bring the system in any desired state, provided that it can also work as an equilibrium point. This is fundamental for the controller of our drone.

Indeed, as we have already seen, the linear model of the drone does not depend on the position, this means that we can move the drone wherever we like, always keeping it in an equilibrium point. So, the main task of the control will be to keep the drone in such a configuration that it does not go out of linearization.

A good design for an LQR is obtained by building the matrices Q and R as diagonal matrices. In this way, each element of the matrix will be responsible for the weight of a specific state or input.
The actual control implement in the pole balancing drone designed in this work is not an LQR, but an upgraded version explained in the next section.

4.5 Frequency Shaped Control (FS-LQR)

The Linear Quadratic Regulator (LQR) method, while known for its idealized guaranteed stability and relative robustness, does not on its own permit the application of frequency dependent cost functions. However, frequency weighted variables can be worked into the LQR cost function by augmenting the system plant model. This section develops the theory behind the frequency domain LQR method, also called *frequency shaped LQR* [4]. The main advantage of an FS-LQR, if compared with an LQR, is that we are now able to weigh differently the frequencies of each state. So, for example, we can decide to give greater weight to the high frequencies of a state, and a lower weight to its low-frequency content. The application of such a controller are plenty.
Let's start taking into consideration a Linear Time-Invariant (LTI) dynamic system

$$\begin{cases} \dot{x} = Ax + Bu, & x(t_0) = x_0, \\ y = Cx \end{cases} .$$

in which $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$ and $y \in \mathbb{R}^p$. Suppose, without loss of generality, $t_0 = 0$. Note that the classical concept of output associated to y loses its meaning in the full state feedback controllers, indeed the best interpretation of y in this context is to consider it as a combination of the states that for some reasons we are interested in.
Typically the infinite horizon LQR cost function is expressed as:

$$J = \int_0^\infty (x^T Q x + u^T R u)\, dt \qquad \left(\textit{We can neglect the coefficient } \frac{1}{2} \right).$$

But, with the new interpretation of y makes sense that the same objective function can be also rewritten as follows:

$$J = \int_0^\infty (y^T \hat{Q} y + u^T R u)\, dt, \tag{4.38}$$

where now $\hat{Q} = C^T Q C$.
This cost function can be converted into the frequency domain by invoking

Parseval's Theorem.

$$J = \frac{1}{4\pi}\int_{-\infty}^{\infty}[Y^*(j\omega)\hat{Q}(\omega^2)Y(j\omega) + U^*(j\omega)R(\omega^2)U(j\omega)]\,d\omega. \tag{4.39}$$

By setting $\hat{Q}(\omega^2) > 0$ and $R(\omega^2) > 0\ \forall\omega$, J results to be compliant with classical LQR framework.
The star $(\cdot)^*$ denotes a conjugate transpose operation.

Theorem 4.2 (Parseval's Theorem).

$$\int_{-\infty}^{\infty}|x(t)|^2\,dt = \frac{1}{2\pi}\int_{-\infty}^{\infty}|X(\omega)|^2\,d\omega = \int_{-\infty}^{\infty}|X(2\pi f)|^2\,df,$$

where $X(\omega) = F_\omega\{x(t)\}$ represents the continuous Fourier transform (in normalized, unitary form) of $x(t)$, and $\omega = 2\pi f$ is frequency in radians per second.
The interpretation of this form of the theorem is that the total energy of a signal can be calculated by summing power-per-sample across time or spectral power across frequency.

Note that if we want to apply Parseval's theorem to the cost function (*4.38*) we have to modify the interval of definition. As J is a quadratic function its value for the negative interval $t \in [-\infty, 0]$ is equal to the value in the positive interval $t \in [0, \infty]$:

$$J = \int_0^{\infty}(y^T\hat{Q}y + u^TRu)\,dt = \frac{1}{2}\int_{-\infty}^{\infty}(y^T\hat{Q}y + u^TRu)\,dt.$$

Having done this, we can define the structure of $\hat{Q}(\omega^2)$ and $R(\omega^2)$.

$$\hat{Q}(\omega^2) = P_q^*(j\omega)P_q(j\omega), \qquad P_q(j\omega) = diag(P_{q_1}(j\omega), \ldots, P_{q_p}(j\omega)),$$
$$R(\omega^2) = P_r^*(j\omega)P_r(j\omega), \qquad P_r(j\omega) = diag(P_{r_1}(j\omega), \ldots, P_{r_m}(j\omega)).$$

As seen for the LQR, also here is a good idea to exploit a diagonal structure, in this way each component of $P_q(j\omega)$ and $P_r(j\omega)$ will work as filter on a specific state or input.
$P_q(j\omega)$ and $P_r(j\omega)$ can be interpreted as the harmonic responses of two transfer functions.

$$Y_p(j\omega) = P_q(j\omega)Y(j\omega), \tag{4.40}$$
$$U_p(j\omega) = P_r(j\omega)U(j\omega), \tag{4.41}$$

where $Y_p(j\omega)$ and $U_p(j\omega)$ are respectively the filtered combination of states and inputs. By substituting the equations (*4.40*) and (*4.41*) in the cost function and applying the Parseval's theorem

$$J = \frac{1}{4\pi}\int_{-\infty}^{\infty}[Y_p^*(j\omega)Y_p(j\omega) + U_p^*(j\omega)U_p(j\omega)]\,d\omega \tag{4.42}$$
$$= \int_0^{\infty}[y_p^T(t)y_p(t) + u_p^T(t)u_p(t)]\,dt. \tag{4.43}$$

As already mentioned $P_q(j\omega)$ and $P_r(j\omega)$ are two filters applied to $u(t)$ and $y(t)$, so we can represent these two types of filters in their state-space representation.

$$P_q : \begin{cases} \dot{z}_q = A_q z_q + B_q y \\ y_p = C_q z_q + D_q y \end{cases}, \tag{4.44}$$

$$P_r : \begin{cases} \dot{z}_r = A_r z_r + B_r u \\ u_p = C_r z_r + D_r u \end{cases}. \tag{4.45}$$

If we want to resolve the FS-LQR, we must extend the system including the states now associated with the filters.

$$x_a = \begin{pmatrix} x \\ z_q \\ z_r \end{pmatrix}, \quad y_a = \begin{pmatrix} y \\ z_q \\ z_r \end{pmatrix}. \tag{4.46}$$

With the new expression for x_a and y_a, also a new state-space representation of the model can be written:

$$\begin{cases} \dot{x}_a = A_a x_a + B_a u \\ y_a = C_a x_a \end{cases}. \tag{4.47}$$

C_a has a structure that depends on the desired combination of the states. For example, if we want to filter all the states individually, without any cross-coupling among them, C_a will be a simple *identity matrix* with dimension equal to the rows of x_a. Instead, the structure of A_a and B_a is fixed.

$$A_a = \begin{pmatrix} A & 0 & 0 \\ B_q & A_q & 0 \\ 0 & 0 & A_r \end{pmatrix}, \quad B_a = \begin{pmatrix} B \\ 0 \\ B_r \end{pmatrix}. \tag{4.48}$$

The frequency shaped LQR problem can be seen as a cross-coupled LQR problem applied on the extended system, in which the corresponding cost function can be found by substituting the relationships (*4.44*) and (*4.45*) into the objective function (*4.43*).

$$J = \int_0^\infty \begin{pmatrix} y_a^T & u^T \end{pmatrix} \begin{pmatrix} Q_a & N_a \\ N_a^T & R_a \end{pmatrix} \begin{pmatrix} y_a \\ u \end{pmatrix}, \tag{4.49}$$

$$Q_a = \begin{pmatrix} D_q^T D_q & D_q^T C_q & 0 \\ C_q^T D_q & C_q^T C_q & 0 \\ 0 & 0 & C_r^T C_r \end{pmatrix}, \quad N_a = \begin{pmatrix} 0 \\ 0 \\ C_r^T D_r \end{pmatrix}, \quad R_a = D_r^T D_r. \tag{4.50}$$

To decouple the system, it is possible exploiting the results present in literature regarding the cross-coupled LQR [7]. The system obtained once the

cross-coupling is removed is expressed by the following equations:

$$\begin{cases} \dot{x}_a = \bar{A}x_a + B_a\bar{u} \\ y_a = C_a x_a \end{cases}.$$

With the associated cost function:

$$J = \int_0^\infty (y_a^T \bar{Q} y_a + \bar{u}^T R_a \bar{u})\, dt.$$

in which the quantities $\bar{A}$, $\bar{Q}$ and $\bar{u}$ are defined as:

$$\bar{A} = A_a - B_a R_a^{-1} N_a^T C_a \quad \bar{Q} = Q_a - N_a R_a^{-1} N_a^T \quad \bar{u} = u + R_a^{-1} N_a^T C_a x_a. \tag{4.51}$$

The obtained problem is a well-known classical stationary LQR problem. We already know how to solve it.
Then, to achieve the above results we have done the following assumptions:

- $\bar{Q} > 0$,
- $R_a > 0$,
- $(\bar{A},\ B_a)$ stabilizable,
- $(\bar{A},\ C_a)$ detectable.

If these assumptions are verified, according also to the results seen for the LQR, we have:

$$\bar{u} = -\bar{K}x_a, \quad \bar{K} = R_a^{-1} B_a^T S.$$

in which S (previously called S_∞) comes out from the Algebraic Riccati Equation, that in this case is defined as:

$$S\bar{A} + \bar{A}^T S - S B_a R_a^{-1} B_a^T S + C_a^T \bar{Q} C_a = 0.$$

But, it is important to notice that in any case the real system must be fed with u and not with $\bar{u}$, then we can calculate quite easily the real control action used in the system starting from the third relationship contained in the row (*4.51*):

$$u = \bar{u} - R_a^{-1} N_a^T C_a x_a = -\bar{K}x_a - R_a^{-1} N_a^T C_a x_a = -(\bar{K} + R_a^{-1} N_a^T C_a) x_a = -\bar{K}_a x_a.$$

In theory, such a control works beautifully. But, the problem lies in one of the four assumptions. Indeed, the matrix $\bar{Q}$ is never positive define. Then, the first assumption does not hold. But, $\bar{Q} > 0$ is a very strong condition, which can be substituted by a weaker one. Before seeing the new condition, let's define $\bar{Q}$ in a new way:

$$\bar{Q} = H^T H \ \textit{ then with this new definition } \ y_a^T \bar{Q} y_a = (HC_a x_a)^T (HC_a x_a) = y_h^T y_h.$$

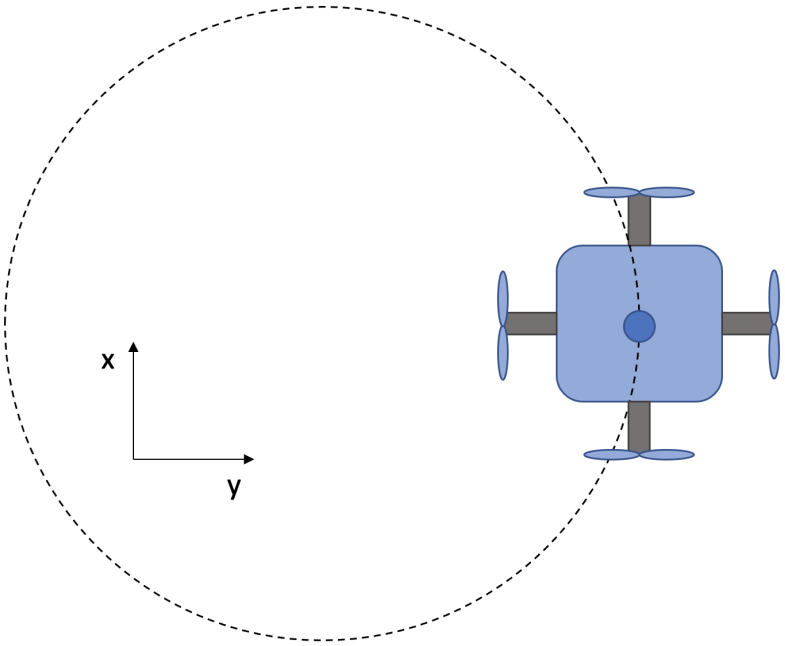
Q now is an identity matrix (> 0) and the new condition becomes:

- $(A_a,\ HC_a)$ detectable.

4.6 The Controller for the Pole Balancing Drone

The controller implemented for this project is a *frequency shaped linear quadratic regulator*, design to have a drone capable of performing a circumference on the *x-y* plane remaining at the same height on the *z-axis*. The trajectory must be followed keeping balanced the pole over the quadcopter. Obviously, with this controller the drone will be able also to perform a *set-point tracking*.

FIGURE 4.14: Aerial view of the desired trajectory



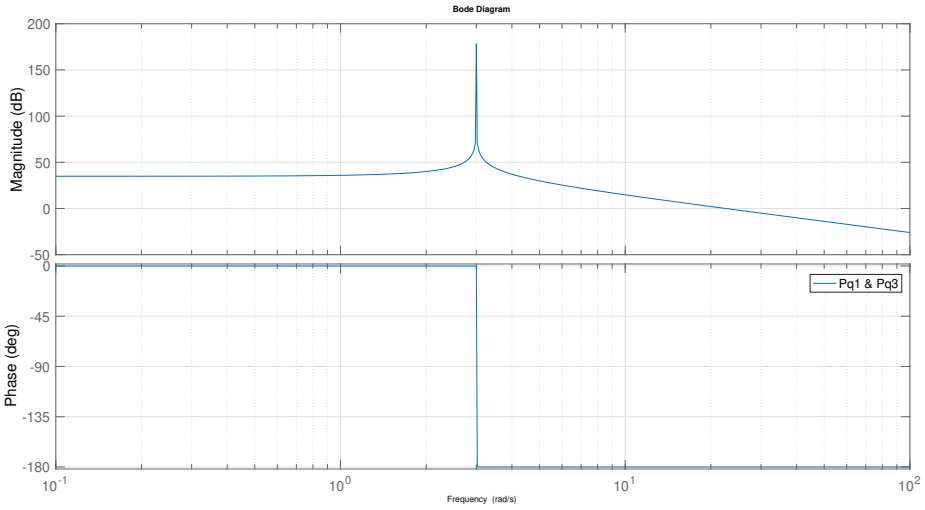
To build an FS-LQR we start from the design of the filters P_{q_i} and P_{r_i}. To understand the logic behind the design of the filters we have to make three considerations.
The first thing to notice is that the system we are dealing with is an error system. Indeed, the signal that goes in the filters is the difference between the state of the drone and the reference signal. Thanks to the fact that we have an error system, when the system converges to zero means that the states are equal to the reference signals.
The second observation regards the gains of the filters. The gain of the filter corresponds to the weight we want to give to the associated state. A greater (compared to the other filters) constant gain on the whole frequency domain means that the cost function will try to minimize the error of that state to the detriment of the others. Whereas a filter, whose gain is not constant along the frequency domain, will be more tolerant with errors for the frequencies where the gain is smaller.
The last remark is on the priorities of our controller. As matter of fact, the magnitude of the gains is chosen according to a hierarchy of importance: the top position is occupied by *a, b, roll, pitch* and *yaw* because if their values increase too much, the system goes out of the linearization domain, then the model on which the control is design is no longer valid; for a similar reason

the second place goes to the angular velocities around x, y and z. The other can be easily inferred from the graphs below.
Let's see now, for each state and input, which are the best design choices for the task we want to perform:

- On x and y we need a notch filter. That is, for a certain frequency the filter has an infinite (or very high) gain. In this way, the error associated with x and y cannot oscillate at that frequency. This means that, if the reference signal has exactly that frequency, we will get perfect tracking. It is important to remember that a circumference on the *x-y plane* can be described as a sine curve on x and a cosine curve on y (or vice-versa). The two filters have been designed to have as resonant frequency $\omega = 3\,rad/s$. Then, if the reference signals on x and y are a sine and a cosine with angular frequency $\omega = 3\,rad/s$ our drone will perform a circumference whose radius depends on the amplitude of the reference signals.

$$P_{q_1}(s) = P_{q_3}(s) = \frac{500}{s^2 + 9}.$$

FIGURE 4.15: Bode diagram of $P_{q_1}(s)$ and $P_{q_3}(s)$



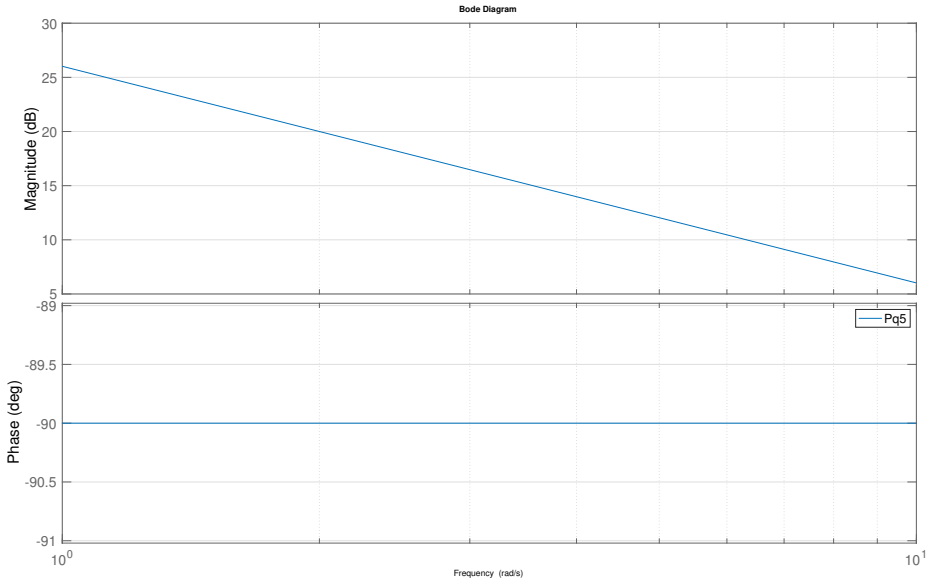
A notch filter of this type can be also implemented to filter out a disturbance that exists for $\omega = 3\,rad/s$. The difference from what we have done for our project, where the final result can be seen as opposed because we are perfectly tracking a signal with $\omega = 3\,rad/s$, is that the disturbance is added to $\dot{x}$. In *Simulink* that means before the integrator used to pass from $\dot{x}$ to x. The reference signal instead is summed to x.

- On z we have to implement a filter that behaves as an integrator. Indeed, we are interested to bring the quadcopter to a certain height, to perform the circle, without any error. Moreover, an integrator on z allows us to remove the feed-forward action needed to balance the gravitational force. If the drone were to rely solely on feed-forward action,

an inaccurate measurement of drone mass would cause errors in positioning along the z axis. With the integrator, this problem is resolved. Basically, the concept behind the integrator is very similar to the notch filter, where the resonant frequency is $\omega = 0\,rad/s$.

$$P_{q_5}(s) = \frac{20}{s}.$$

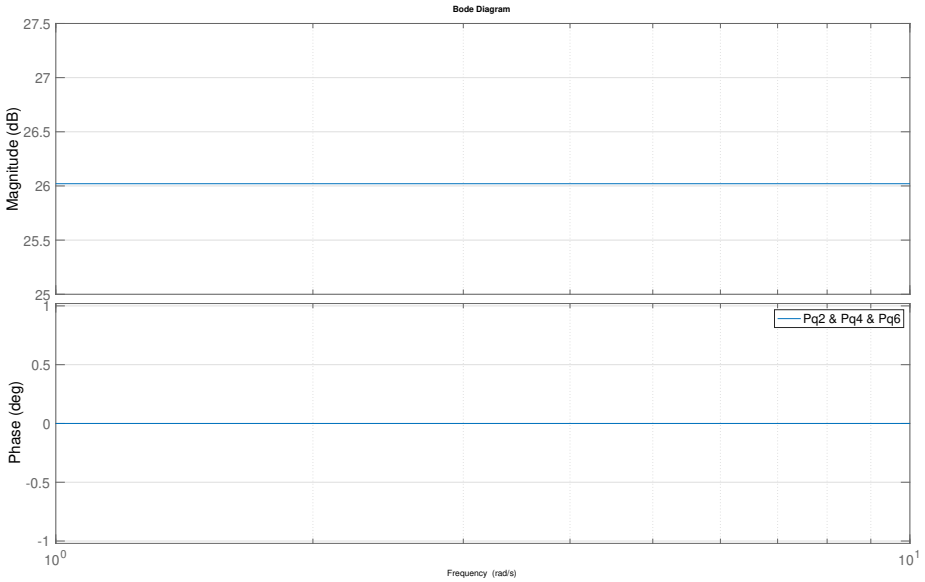
FIGURE 4.16: Bode diagram of $P_{q_5}(s)$



- On V_x, V_y and V_z we do not have any special requirements, then the filter is just a constant gain along the whole frequency domain.

$$P_{q_2} = P_{q_4} = P_{q_6} = 20.$$

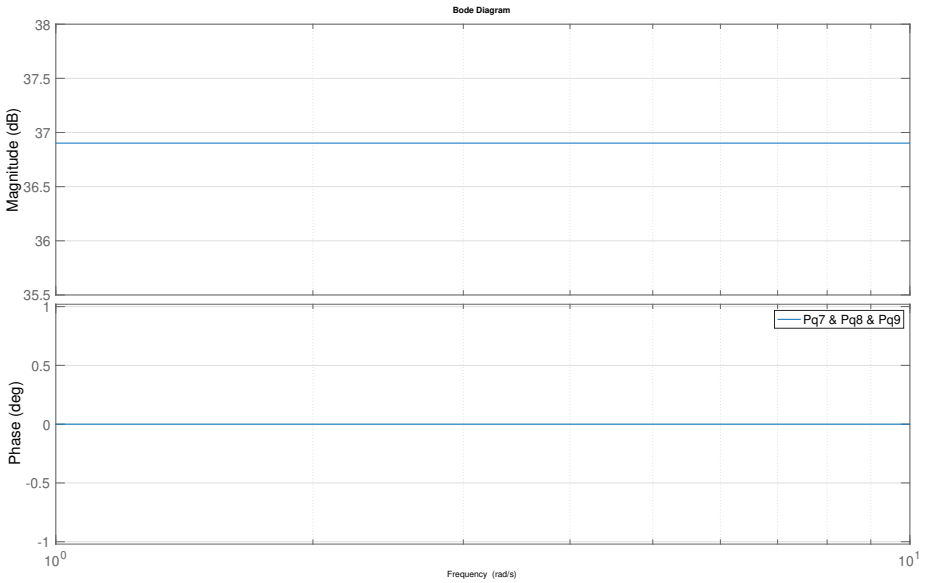
FIGURE 4.17: Bode diagram of P_{q_2}, P_{q_4} and P_{q_6}



- On *roll* (ϕ), *pitch* (θ) and *yaw* (ψ), as already mentioned we need the greatest gain, constant in the whole frequency domain.

$$P_{q_7} = P_{q_8} = P_{q_9} = 70.$$

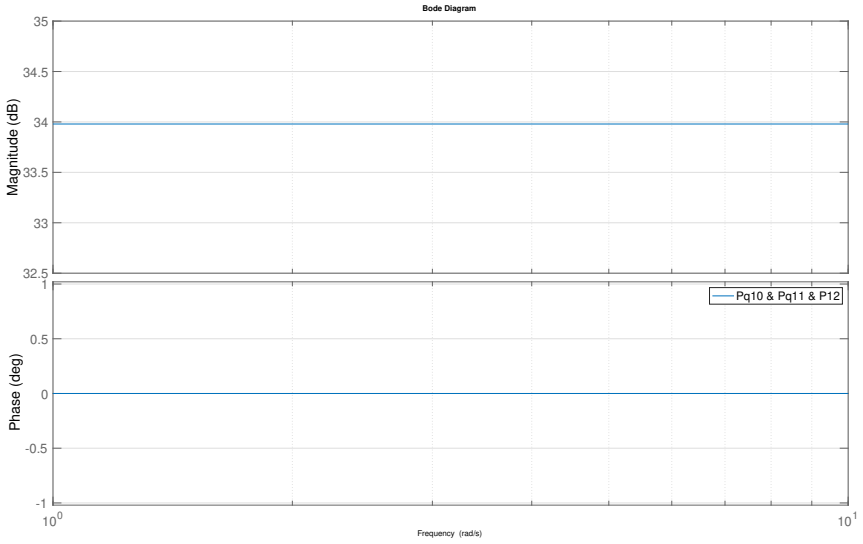
FIGURE 4.18: Bode diagram of P_{q_7}, P_{q_8} and P_{q_9}



- On ω_{x_B}, ω_{y_B} and ω_{z_B} still a constant gain has been implemented, but its magnitude is lower than the one in RPY.

$$P_{q_{10}} = P_{q_{11}} = P_{q_{12}} = 50.$$

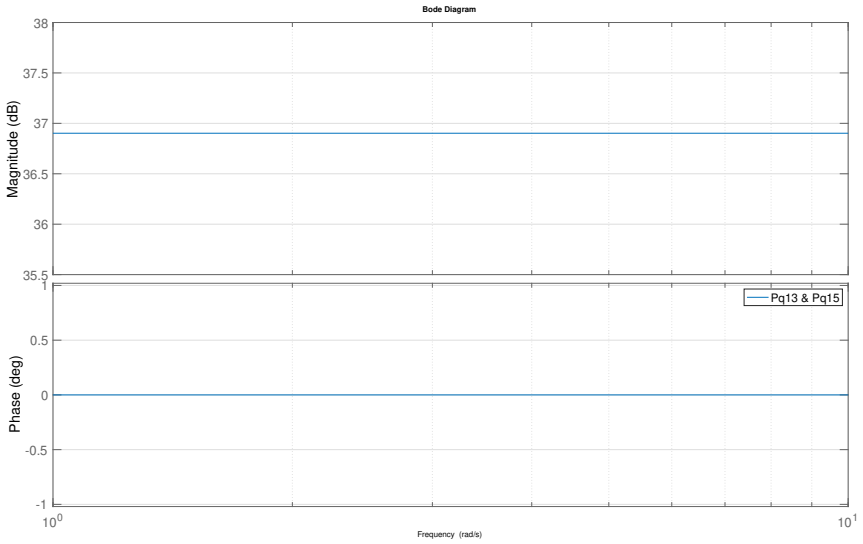
FIGURE 4.19: Bode diagram of $P_{q_{10}}$, $P_{q_{11}}$ and $P_{q_{12}}$



- On a and b, that is the position along x and y of the C.o.M of the pole, we still need a constant gain. Here the gain needs to be quite high because if the pole tilts, it falls.

$$P_{q_{13}} = P_{q_{15}} = 70.$$

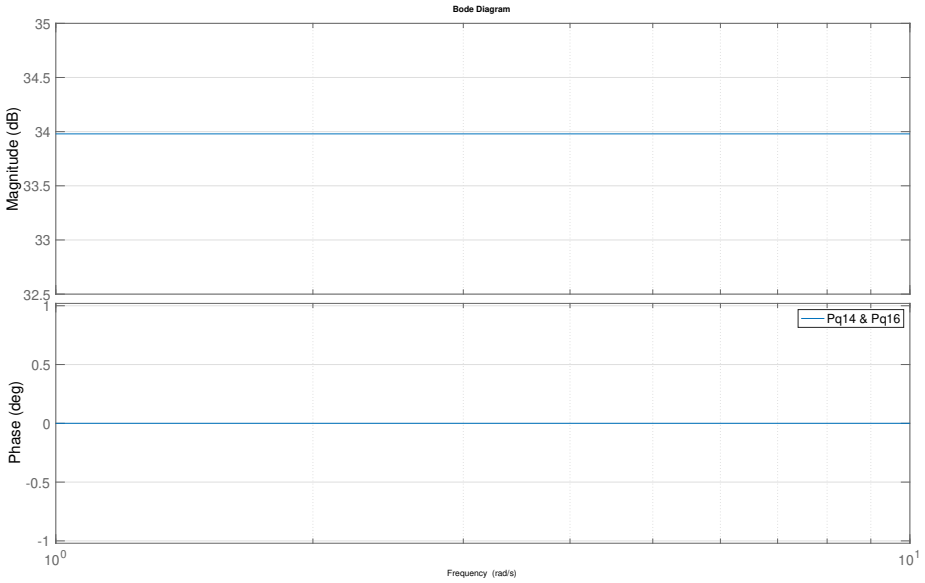
FIGURE 4.20: Bode diagram of $P_{q_{13}}$ and $P_{q_{15}}$



- On V_a and V_b we have again a constant gain along the whole frequency domain.

$$P_{q_{14}} = P_{q_{16}} = 70.$$

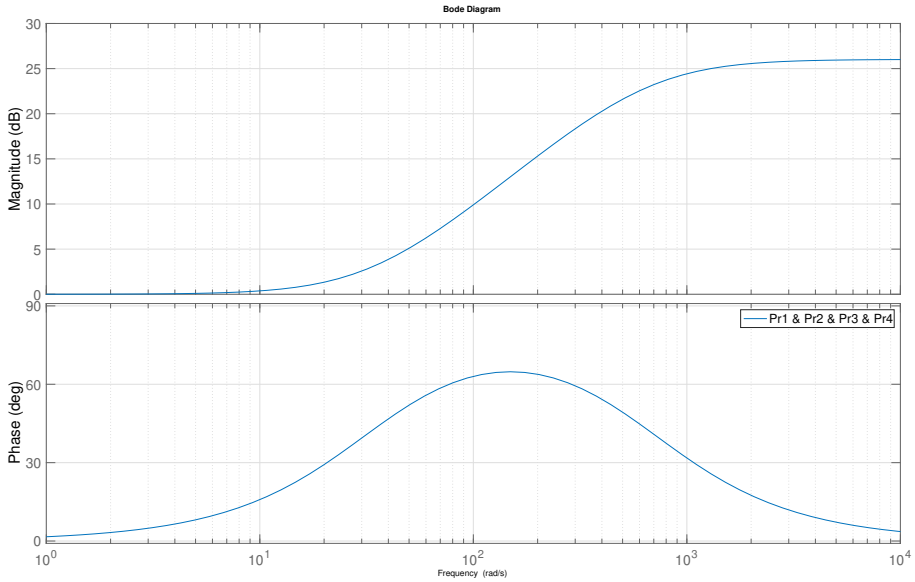
FIGURE 4.21: Bode diagram of $P_{q_{14}}$ and $P_{q_{16}}$



- On the inputs T, τ_x, τ_y and τ_z we have implemented a high-pass filter. In this way, the high-frequency content of the control action is filtered out. Indeed, with the filter we have associated a higher gain to the high frequencies, so the controller will penalize them more. This choice is done because usually external disturbances are at high frequencies.

$$P_{r_1}(s) = P_{r_2}(s) = P_{r_3}(s) = P_{r_4}(s) = \frac{0.03s + 1}{0.0015s + 1}.$$

FIGURE 4.22: Bode diagram of $P_{r_1}(s)$, $P_{r_2}(s)$, $P_{r_3}(s)$ and $P_{q_4}(s)$



Associated to each state filter we have the matrices A_{q_i}, B_{q_i}, C_{q_i} and D_{q_i}; and associated to each input filter we have the matrices A_{r_i}, B_{r_i}, C_{r_i} and D_{r_i}. In the theory related to the FS-LQR we have considered the presence of only one matrix for A_q, B_q, C_q, D_q, A_r, B_r, C_r and D_r. So to exploit the theory studied, we have to rearrange the matrices as follows:

$$A_q = blkdiag(A_{q_1}, \dots, A_{q_{16}}), \quad B_q = blkdiag(B_{q_1}, \dots, B_{q_{16}}),$$
$$D_q = blkdiag(D_{q_1}, \dots, D_{q_{16}}), \quad A_r = blkdiag(A_{r_1}, \dots, A_{r_4}),$$
$$B_r = blkdiag(B_{r_1}, \dots, B_{r_4}), \quad C_r = blkdiag(C_{r_1}, \dots, C_{r_4}),$$
$$D_r = blkdiag(D_{r_1}, \dots, D_{r_4}),$$

where $blkdiag(\cdot)$ is a *Matlab* command able to generate a diagonal matrix, whose diagonal is composed by the matrices given as input to the command. Note that, actually, not for each state filter the matrices A_{q_i}, B_{q_i} and C_{q_i} exist, as it depends on the space-state representation associated with the filter. When the command $blkdiag(\cdot)$ meets one of these non-existing matrices, it just neglects them. The only matrix that does not follow this path is C_q, and we will see its structure in a while, but first is worth understanding how many states the extended system has. The filters on x and y add two states whereas the filters on z, T, τ_x, τ_y and τ_z have a single state. Then, according to the definition (*4.46*), our system has *25 states*. Therefore, as we are not selecting any particular combination of the states, the matrix C_a defined for the system (*4.47*) is just an identity matrix $I_{25\times 25}$.
Now, the structure of C_q can be inferred from the second equation of the system (*4.44*). We will report here the same expression, but highlighting the

matrices dimension.

$$(y_p)_{16\times 1} = C_q(Z_q)_{5x1} + (D_q)_{16\times 16}(y)_{16\times 1}.$$

It is clear that the only possible structure for C_q is a (16×5) matrix. To ensure the right interfacing between the filter states z_q and the associate matrices C_{q_i}, the matrix C_q must be build as follows (express in *Matlab* code for simplicity):

$$\begin{aligned} Cq = [&Cq_1,\ zeros\,(1,\,3);\\ &zeros\,(1,\,5);\\ &zeros\,(1,\,2),\ C_{q_3},\ 0;\\ &zeros\,(1,\,5);\\ &zeros\,(1,\,4),\ C_{q_5};\\ &zeros\,(11,\,5)], \end{aligned}$$

where *zeros (i, j)* represent a matrix $(i \times j)$ of zeros.
If we pay attention to the filter realized, it is possible to notice that for all the filters that are a simply constant gain, A_{q_i}, B_{q_i} and C_{q_i} do not exist. Indeed, the type of control applied on these states is more similar to an LQR rather than an FS-LQR. So, what we are doing can be seen as an FS-LQR applied only to some states. As consequence, we need to define a matrix $(CC)_{3\times 16}$ that is used to select the states that are actually filtered and not just weighed through the matrix D_{q_i}.

$$CC = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Once defined that, the work is done, as all the definitions seen in the theory section can be applied without problems. Only two things must be taken into consideration: firstly the zeros in the formulas (*4.48*) and (*4.50*) must be adjusted to the right dimension according to the other element present on the same column and on the same row; secondly now in the structure of A_a enters also the new matrix CC.

$$A_a = \begin{pmatrix} A & 0_s & 0_s \\ B_q CC & A_q & 0_s \\ 0_s & 0_s & A_r \end{pmatrix}. \tag{4.52}$$

The subscript of the zeros is to point out that more than one zero are needed in that spot, as previously mentioned.
In Chapter *5* are shown the necessary schemes to implement the controller on *Simulink*.
The last check that is worth doing is about the controllability of the extended system, this means that the couple to be check is $(\bar{A},\ B_a)$. To do it, the fastest way is *Matlab*, but if we directly calculate the rank of the controllability matrix we would get a misleading result.

```
disp(rank(ctrb(ABar, Ba)))
```

The output of the above line of code is three. So, the system appears to be not controllable. Actually, the system is controllable and the problem relies on the default digit precision of *Matlab* set to 16. After some tests, it is possible to find out that the minimum precision to correctly perform the calculation of the rank is 40. The following code is able to evaluate correctly the rank of the controllability matrix:

```
digitsOld = digits(40); %This increases the number of si-
gnificant digits after the coma

COa = ctrb(vpa(ABar), vpa(Ba)); %vpa is used to performe
calculations with the number of digits previously selec-
ted

disp(rank(vpa(COa)));
```

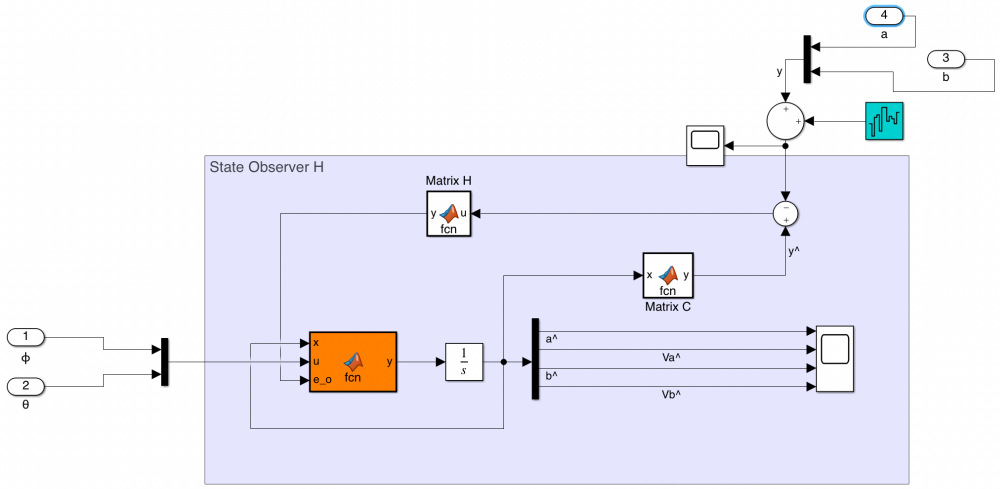
Chapter 5

Performances and Simulink Implementations

5.1 Inverted Pendulum State Observer

As anticipated in Chapter *4* we have implemented two observers for the pole.

FIGURE 5.1: Inverted pendulum state observer



In the orange box, the equation (*4.14*) is implemented, where actually the term $H(\hat{y}_p(t) - y_p(t))$ is computed outside the block, as shown in Figure *5.1*. The cyan square is a white noise added to show the behavior of the observer if external disturbances or errors in the measures are present. The noise power, if turned on, is set on 0.0001 Hz.

5.1.1 Performances with no External Disturbances

First of all, let's have a look at the performances in an ideal scenario, i.e. no external disturbances. We will ask the drone to reach the position $(1, 1, 1)$ and we can compare the output of the real state with the two estimates from the observers designed.
The results are summarized in Figure *5.2* and *5.3*, where the elements with the number 1 (in black) are the estimates of the observer defined through the

FIGURE 5.2: State a and V_a with no external disturbances

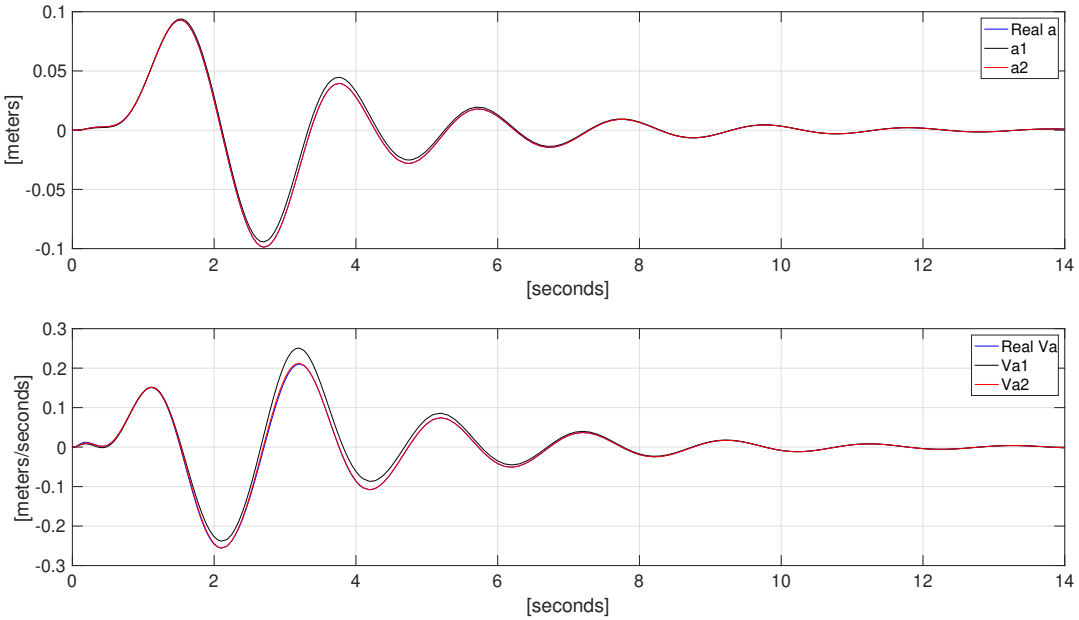
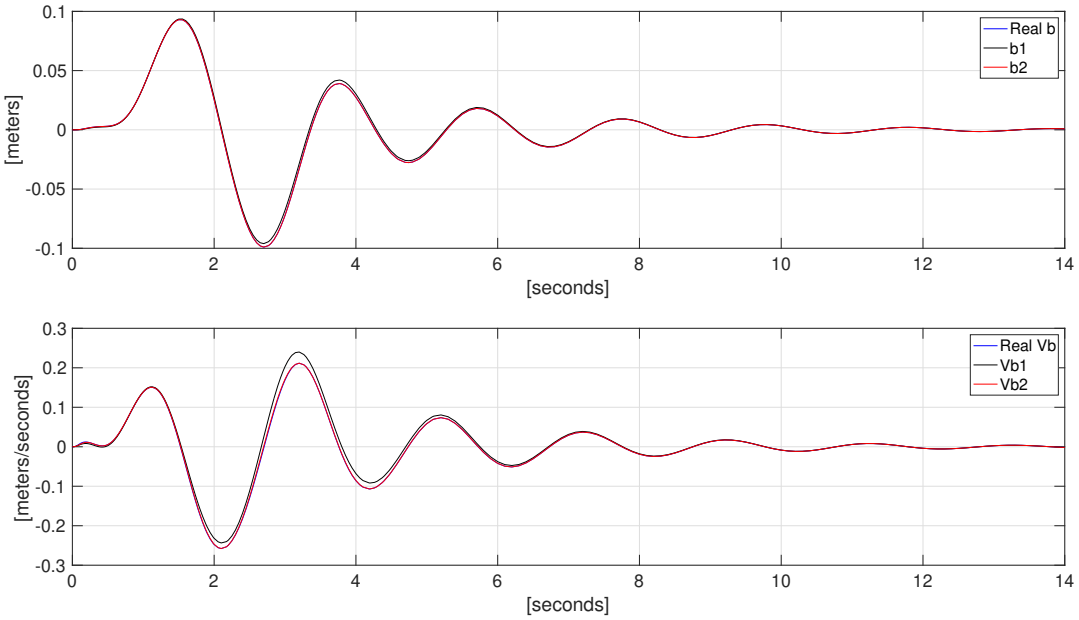


FIGURE 5.3: State b and V_b with no external disturbances



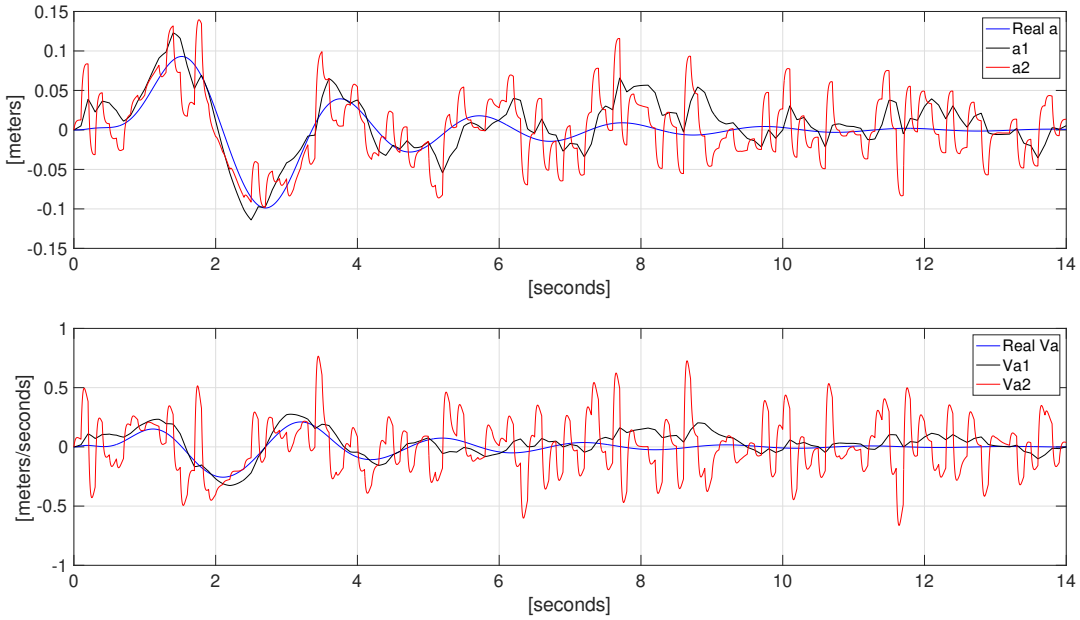
polynomial (*4.17*), that from now on will be called *observer 1*, and the elements with the number 2 (in red) are the estimates of the observer defined through the polynomial (*4.18*), that will be called *observer 2*. According to

what we have learned in the theory, the *observer 2* should have faster dynamics since it has poles further from the imaginary axis. Indeed, studying the graph we can barely distinguish the curve of the real states from those estimated by the *observer 2*, whereas in the first seconds of the plot we can still recognize the estimates of the *observer 1* that takes more time to converge.

5.1.2 Performances with an External Disturbance

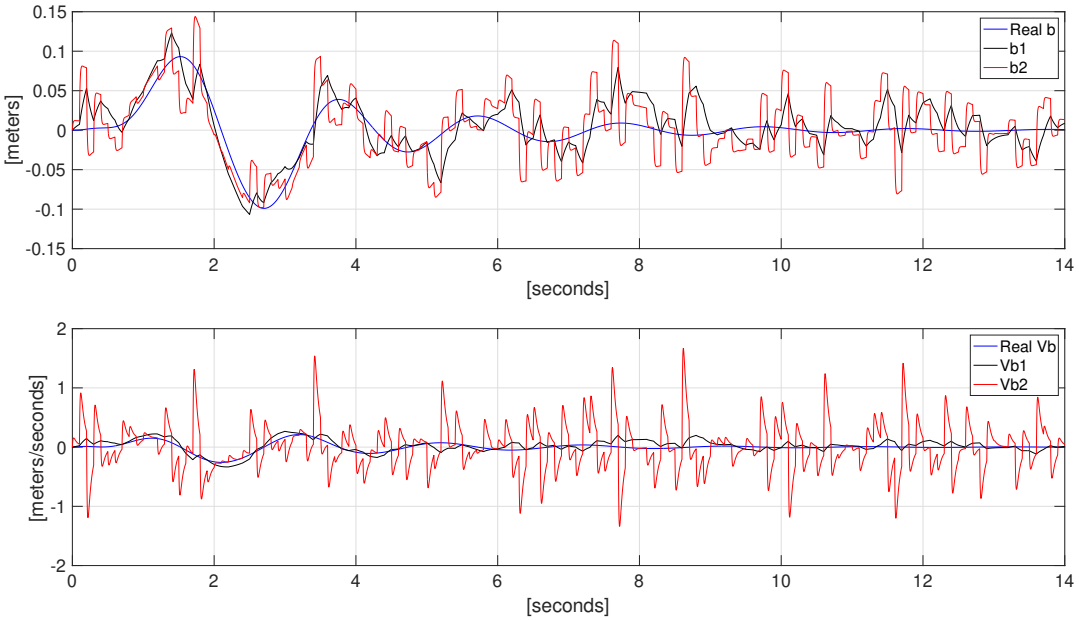
It is clear that in an ideal scenario the *observer 2* has better results, but now we have to understand if it is advisable to use it in every occasion, analyzing the Figure *5.4* and *5.5*.

FIGURE 5.4: State a and V_a with a disturbance of 0.0001 Hz



Especially from the graph of V_a and V_b we immediately notice that the estimates of the *observer 2* are deeply penalized from the external noise. For this reason, during the implementation of an observer we must keep into consideration this trade-off between accuracy and speed of convergence.

FIGURE 5.5: State b and V_b with a disturbance of 0.0001 Hz



5.2 FS-LQR Simulink Implementation and Performances

The choices and the structure of the controller we have decided to use are explained in Chapter *4*, now in this section, we will see how to implement our regulator in *Simulink* and its performances.

5.2.1 Simulink Scheme

First of all in Figure *5.6* it is possible to see the necessary scheme to implement an FS-LQR. The red block contains all the non-linear dynamics of the system, the orange blocks are the reference signals we would like to track, the blue one contains the filters on the states, in the green block there are the filters on the inputs, in the white rectangle the feed-forward action and finally in the red triangle we can see the constant gain applied on the feedback loop to obtain the desired control law.
In Figures *5.7*, *5.8*, *5.9* it is possible to have a look inside, respectively, of the red, the blue and the green block.

FIGURE 5.6: Full *Simulink* scheme

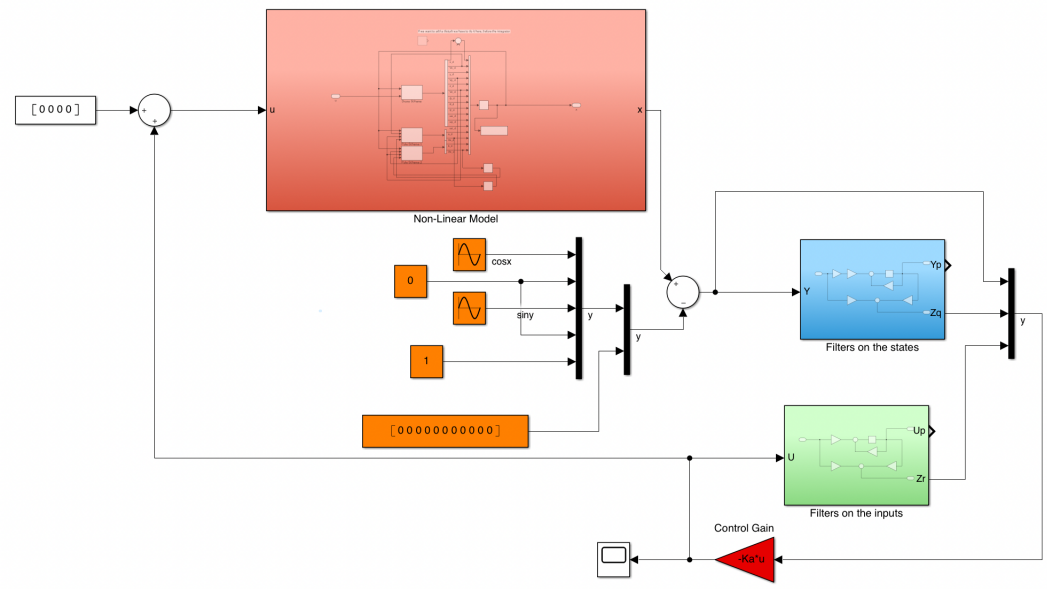


FIGURE 5.7: Non-linear system scheme

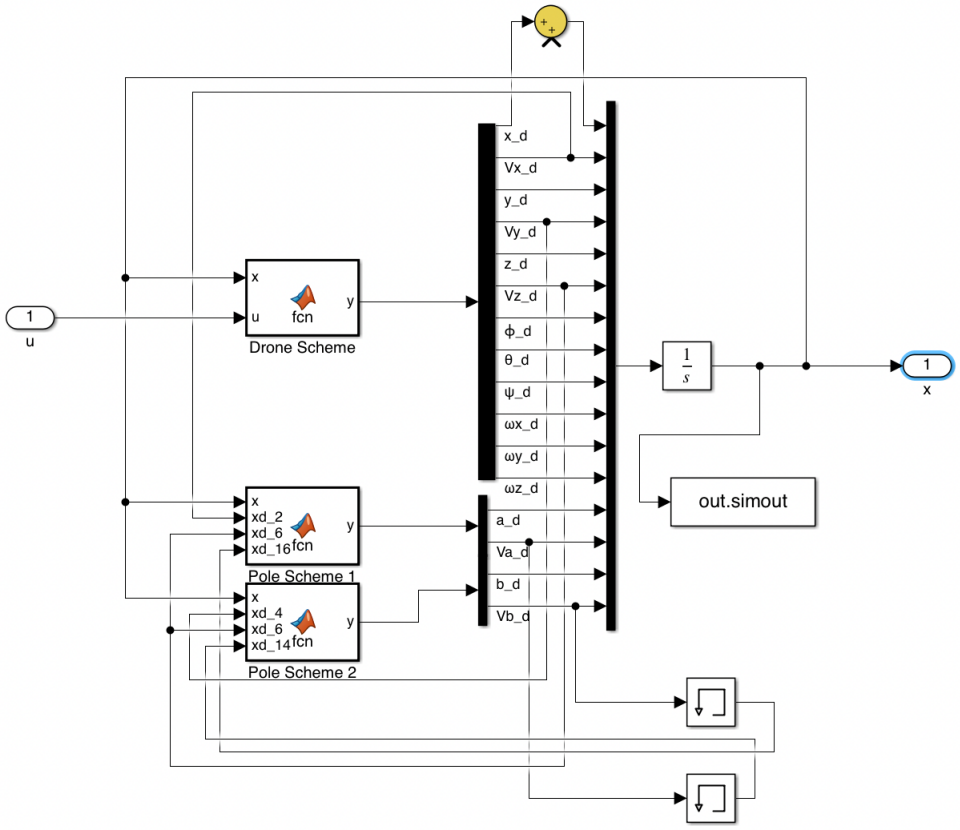


FIGURE 5.8: States filters scheme

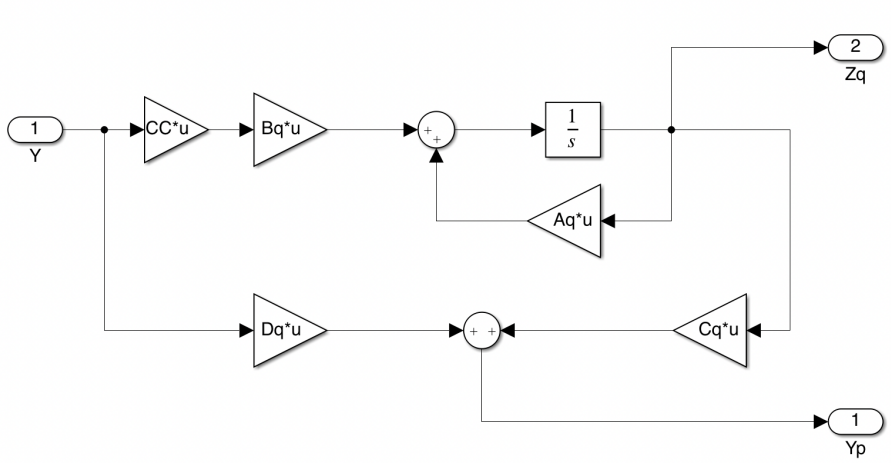
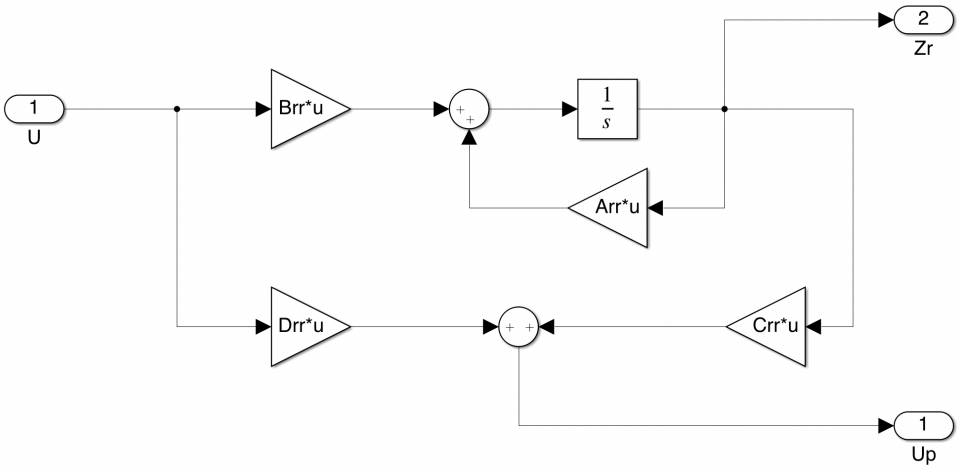


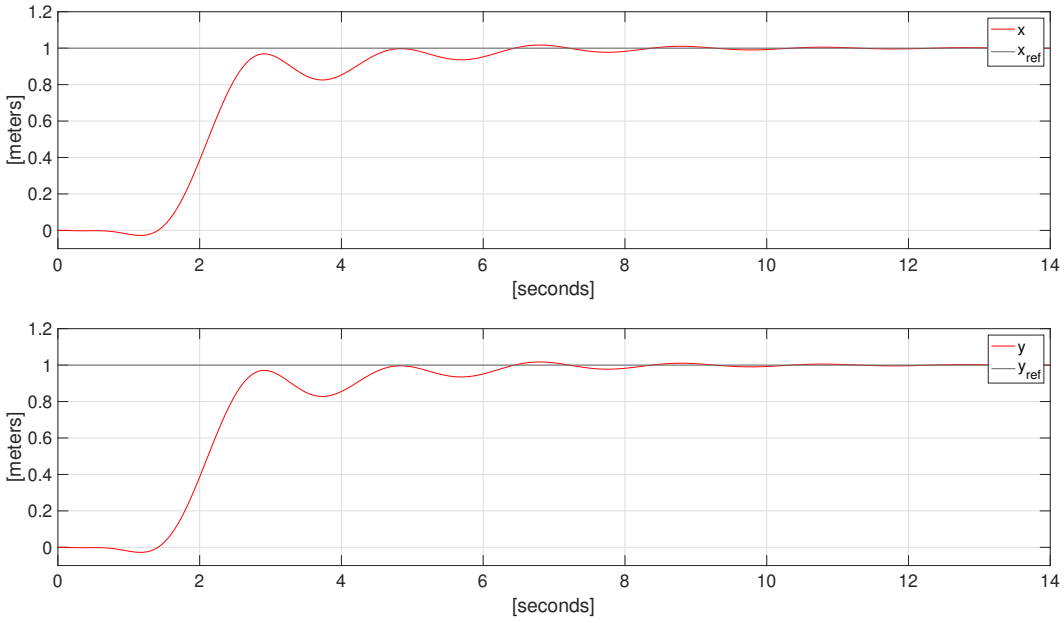
FIGURE 5.9: Inputs filters scheme



5.2.2 FS-LQR Performances

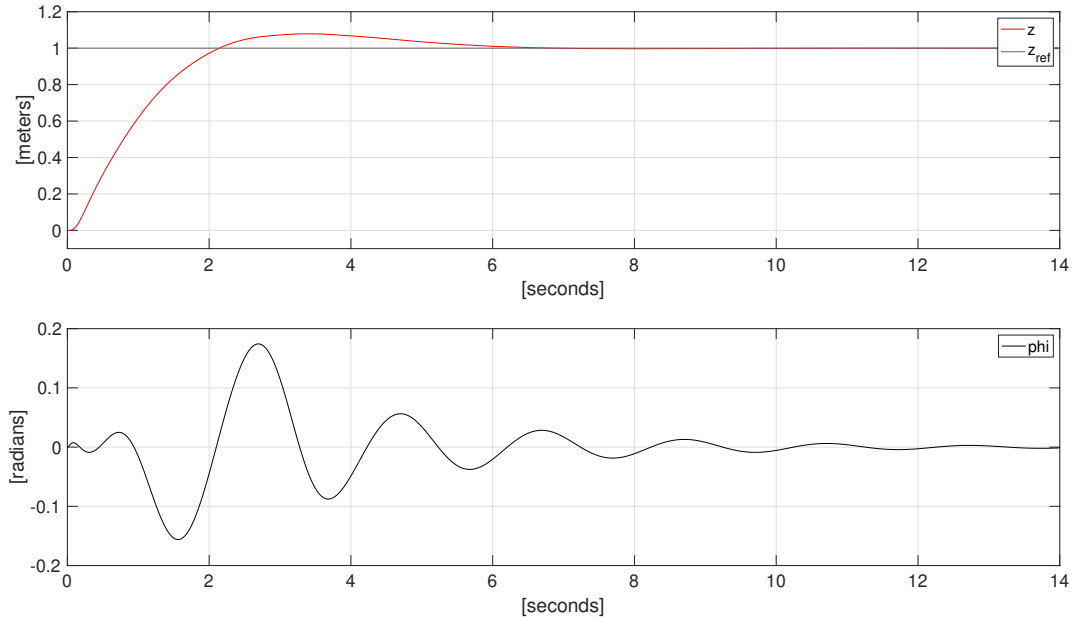
Set-point tracking: For the analysis of the performances we start first from a *set-point tracking* scenario, where the reference signal is set to have as final spatial position for the drone $(x, y, z) = (1, 1, 1)$. The results are summarized in Figures *5.10*, *5.11*, *5.12* and *5.13*.

FIGURE 5.10: Set-point tracking: response of x and y



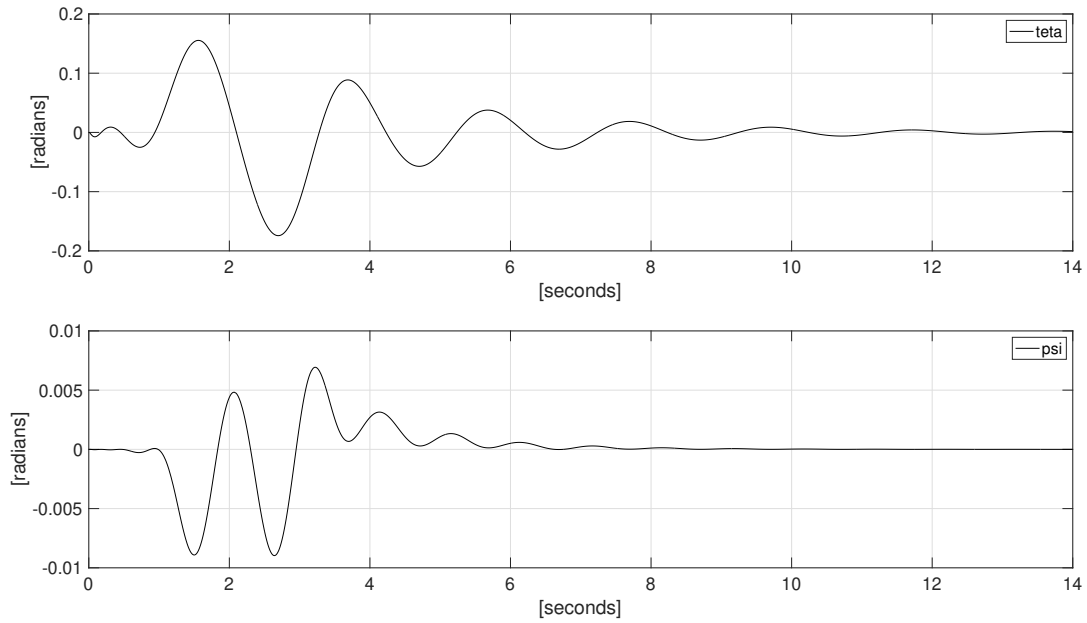
Studying Figure *5.10* we can notice that, even if on x and y we do not have an integrator, we still have a convergence to the reference signal. This happens because the poles used to generate the notch filter, even if they are imaginary, lay on the imaginary axis and therefore they still give to the system the property of converging to reference signal.

FIGURE 5.11: Set-point tracking: response of z and ϕ



The behavior of z, in Figure *5.11*, is exactly what we were expecting from an integrator.

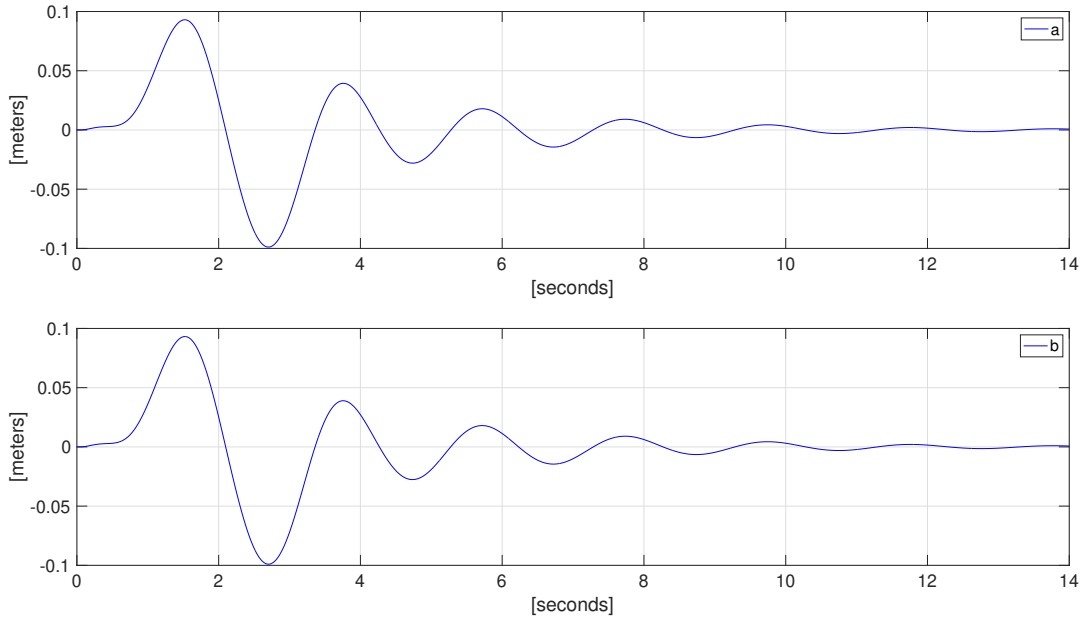
FIGURE 5.12: Set-point tracking: response of θ and ψ



Analysing RPY from Figure *5.11* and *5.12*, we see that the controller is able to keep the angles as closer as possible to the *hovering* configuration. Indeed

the maximum inclination that the drone assumes is lower than 11^o.

FIGURE 5.13: Set-point tracking: response of a and b



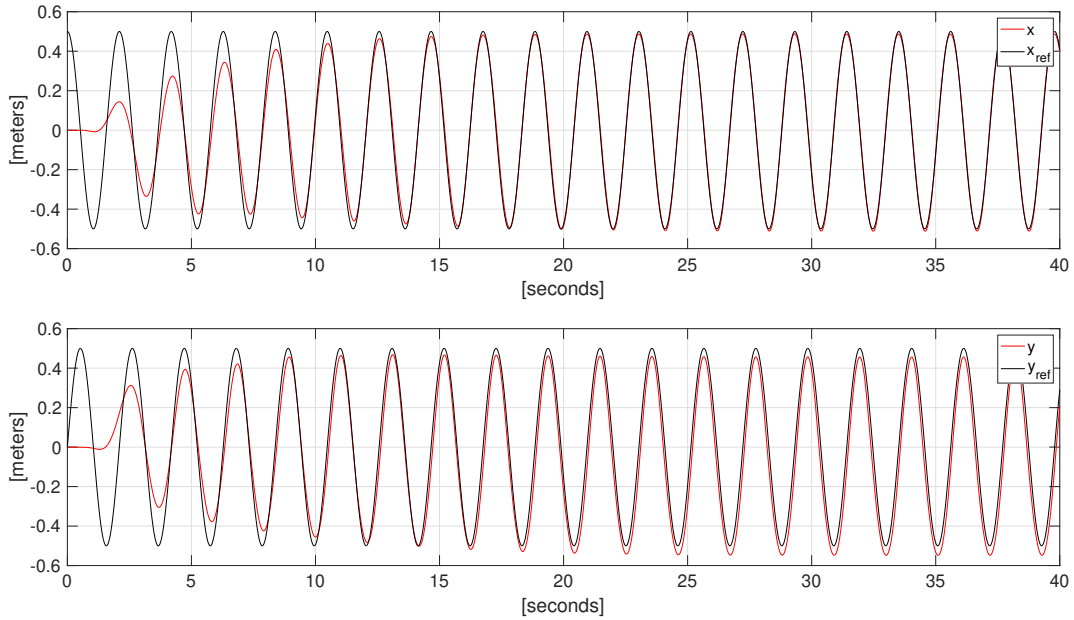
Also the inclination of the pole is kept under control, indeed the maximum angle is lower than 6^o. It can be calculated through basic trigonometry formulas keeping in mind the Figure *3.3*:

$$\alpha = sin^{-1}\left(\frac{b}{L}\right)$$

where α is the inclination of the rod with respect to the *z-axis* of the inertial frame and L is the total length of the pole.

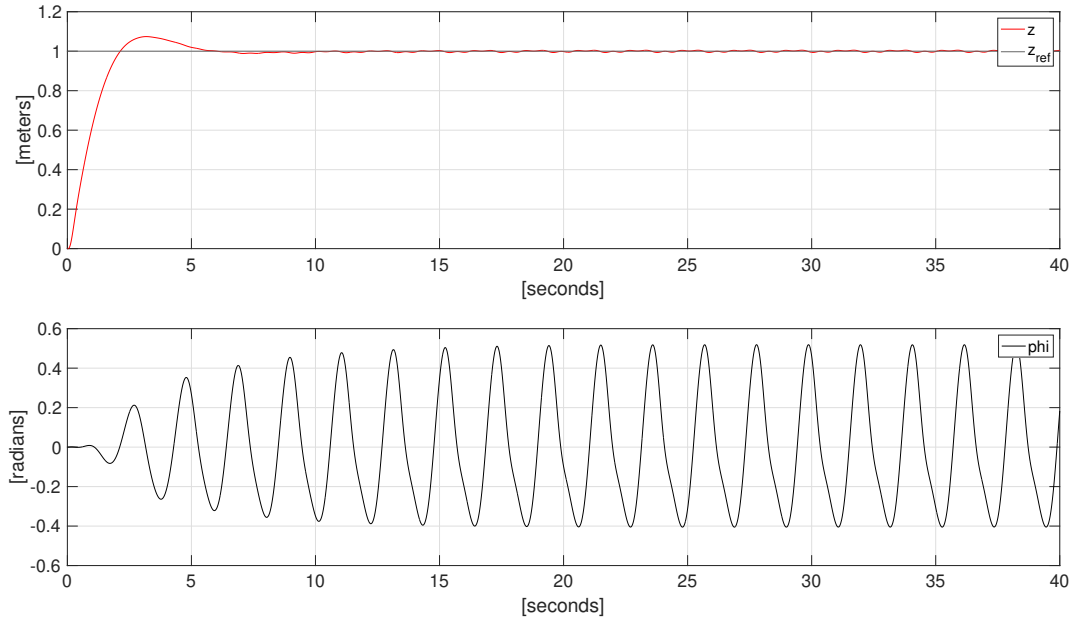
Circumference at constant height: If we what to make a *circumference at constant height* we need to change the reference signals for x and y. Now, indeed, they become a sine and a cosine function with an amplitude of $0,5\,m$. All the important results are shown in Figures *5.14*, *5.15*, *5.16* and *5.17*.

FIGURE 5.14: Circumference: response of x and y



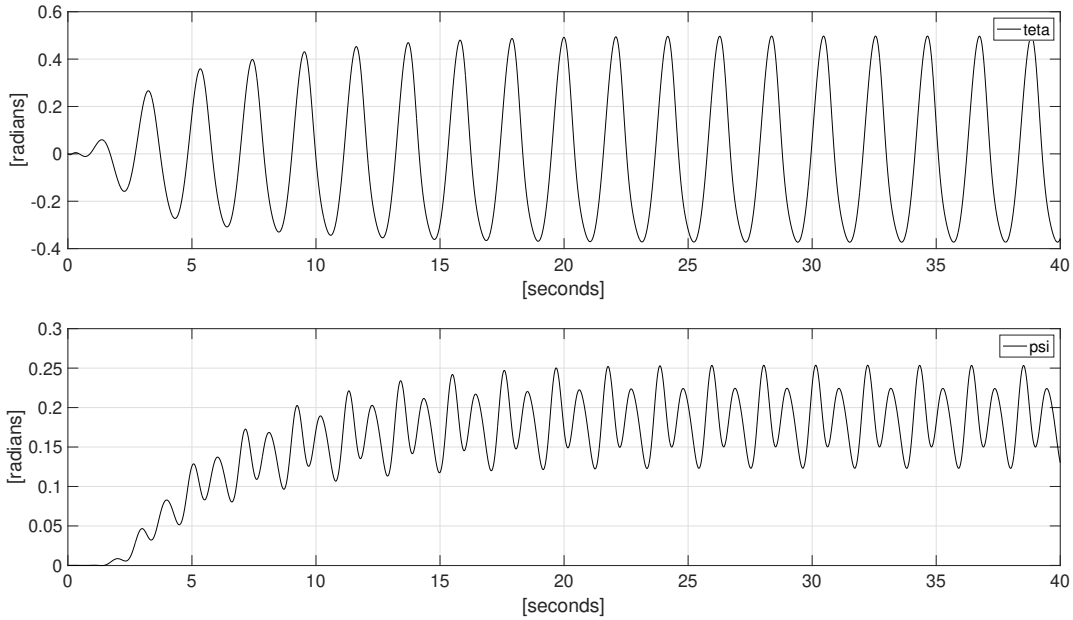
From Figure *5.14* we can see that the drone tracks the sinusoidals pretty well. There is just a little constant error that it is around the 5% of the total amplitude of the signals.

FIGURE 5.15: Circumference: response of z and ϕ



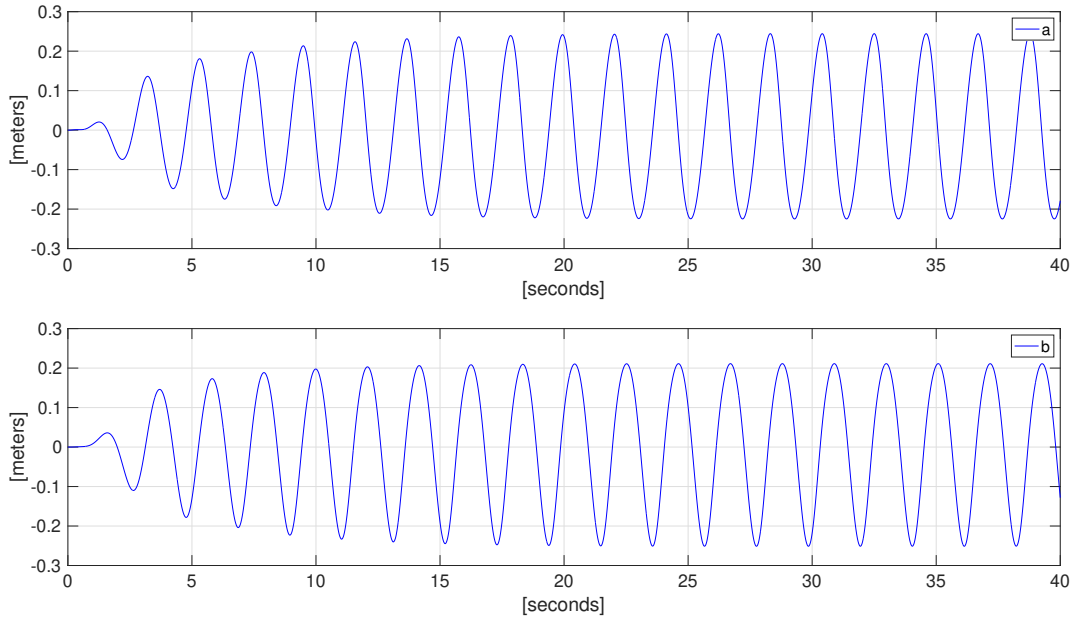
In Figures *5.15* and *5.16* we can see the evolution of RPY. If compared to the case of the *set-point tracking* here the angles are greater, but this is due to

FIGURE 5.16: Circumference: response of θ and ψ



the complex task that the drone has to achieve, anyway the maximum angle reached is still below 30^o.

FIGURE 5.17: Circumference: response of a and b

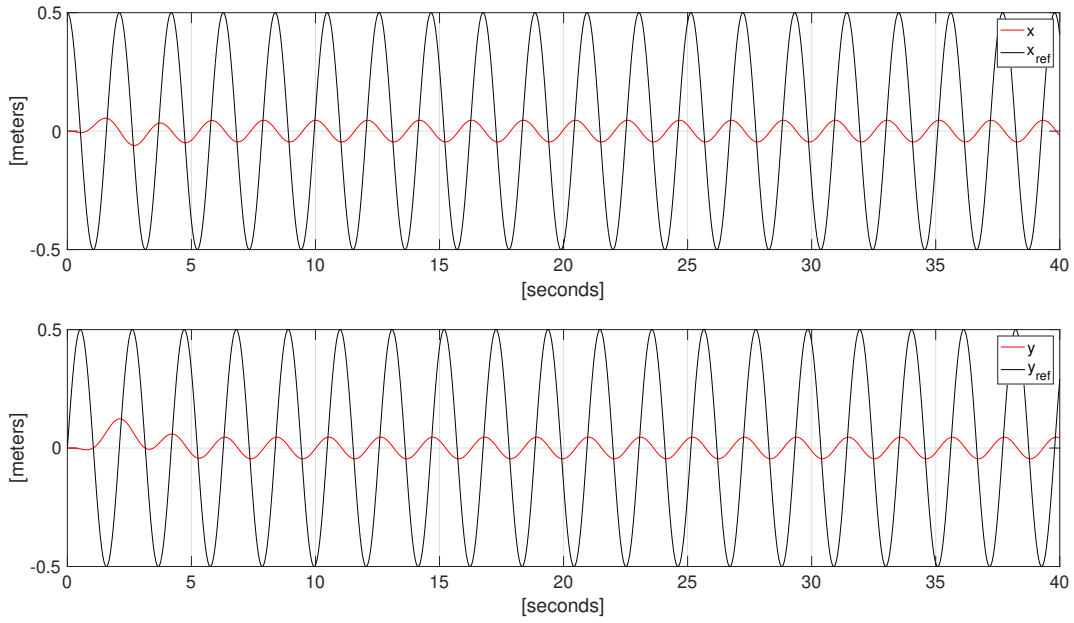


Also in the case of a and b, shown in Figure *5.17*, now we have greater values, but the inclination of the pole is still fine as it is below 15^o.

5.3 LQR Performances

To understand why an LQR is not enough for our goal, we will report in Figure *5.18* the behavior of the states x and y. It is clear that such a controller is not capable of tracking a sinusoidal. Indeed, the constant error is huge and the response of x and y is not in phase with the reference signal.

FIGURE 5.18: LQR: response of x and y



The above performances are obtained with the following weights in the matrices Q and R:

$$Q = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 15 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 15 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 \end{pmatrix},$$

$$R = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}.$$

Chapter 6

Conclusions and Future Developments

As pointed out earlier, the purpose of the paper is to show, through simulations, that, using a control system based on concepts like *full state feedback* and *optimal control*, it is possible to obtain a quadcopter, with a rod balanced on top of it, able to perform both a *set-point tracking* and circumference at a constant height.

To achieve this goal, we started from the nonlinear model of the drone, which allows us to accurately describe the dynamics of the system under investigation, and then, we linearized it around an equilibrium position, i.e., the hovering position of the drone. The linearization step is fundamental because all the control theory behind the regulator implemented is valid exclusively for linear systems. With a linearized system available, we can apply the frequency shaped linear quadratic regulator, which is a nice tool to track a sinusoidal signal, like the one needed to perform a circumference. The other main task of the controller is to keep the drone as close as possible to the equilibrium point, otherwise the linear model would lose its validity. To ensure this result is crucial the design of the filters on states and inputs, since through these tools we are able to describe the desired behavior for the quadcopter. The power of all possible types of optimal control relies upon a single cost function, which must be designed with the desired outcome in mind. The solution to the optimal control is obtained by finding a point of minimum for the objective function.
Last but not least, to develop a project ready to be implemented in a real environment, such as the CASY laboratory, it is necessary also to develop an observer for the states that cannot be measured by specific sensors.

Then, even if the pole balancing drone does not have a real application, we have proved that, nowadays, thanks to the control techniques known and to the technologies available on the market, we can realize autonomous drones able to perform very complicated tasks with incomparable accuracy.
The importance of this conclusion lies not only in the possibility of speeding up processes such as the transport of goods or the reconnaissance of agricultural environments, but also in the fact of being able to carry out operations autonomously and with greater accuracy. As a practical example, in agriculture, the immediate advantage is greater control of the environmental

impact thanks to the possibility of a more accurate dosage of fertilizers and pesticides, which is particularly important to achieve higher and higher standards of sustainable agriculture. On the other hand, these technologies can also be used in inhospitable scenarios that are difficult to reach by land.

The goal reached with this thesis can be seen also as a starting point for future developments. The results obtained are ready to be used for the final step: the implementation. Indeed, considering the tools available in the CASY laboratory, with a little bit of extra work it is possible to check if the project works properly once it has to face the reality. On the regulator side, an advisable improvement can be related to the suppression of the constant error on the states x and y when they track the sinusoidals. This, probably, can be achieved by implementing a double filter on x and y: the first filter to select the frequency of the sinusoidal and the second that has to work as an integrator.

Bibliography

[1] H. Alemi Ardakani and T. J. Bridges. "Review of the 3-2-1 Euler Angles: a yaw–pitch–roll sequence". en. In: (Apr. 2010), pp. 1–9.

[2] Paolo Bolzern, Riccardo Scattolini, and Nicola Schiavoni. *Fondamenti di controlli automatici*. 4° edizione. Milano: McGraw-Hill Education, 2015.

[3] Olivier Chocron. *Euler ZYX Convention*. shorturl.at/egDPY. Oct. 2000.

[4] N. K. GUPTA. "Frequency-shaped cost functionals - Extension of linear-quadratic-Gaussian design methods". In: *Journal of Guidance and Control* 3 (1980). Publisher: American Institute of Aeronautics and Astronautics, pp. 529–535.

[5] Fumio Hamano. "Derivative of Rotation Matrix Direct Matrix Derivation of Well Known Formula". In: *arXiv:1311.6010 [cs]* (Nov. 2013). URL: http://arxiv.org/abs/1311.6010.

[6] Markus Hehn and Raffaello D'Andrea. "A flying inverted pendulum". In: *2011 IEEE International Conference on Robotics and Automation*. May 2011, pp. 763–770.

[7] Faryar Jabbari. *EXTENSIONS TO LQR*. http://mae2.eng.uci.edu/~fjabbari//me270b/chap5.pdf. Apr. 2003.

[8] Claudio Melchiorri. *Introduzione al Controllo Ottimo*. shorturl.at/dtHM6. 2019.

[9] Claudio Melchiorri. *Ridig Body Motion – Homogeneous Transformations*. shorturl.at/lwIKQ. 2019.

[10] Bruno Siciliano et al. *Robotics: Modelling, Planning and Control*. en. London: Springer-Verlag, 2009.

[11] Shiyu Zhao. "Time Derivative of Rotation Matrices: A Tutorial". In: (Sept. 2016).