Universitat Politècnica de Catalunya

Master Thesis

Controlling a Quadrotor with a Robotic Arm using a Nonlinear Model Predictive Control

Author:
Sebastián Chávez-Ferrer Marcos

Supervisors:
Dr. Carlos Ocampo Martínez

Institut de Robòtica i Informàtica Industrial (CSIC-UPC)

January 2015

Acknowledgements

I would like to express my acknowledgements to the Master coordinator, Cecilio Angulo, who helps me to solve all my administrative problems and also who showed me the benefits of the master.

Also, I would like to thank to my supervisor Carlos Ocampo, who guided me through the goods and bads moments of the thesis. Without its help it would have been impossible to finish it and get these results.

On the other hand, I would like to express my appreciation to all the people from the Institut de Robòtica i Informàtica Industrial (IRI) where I was very well welcomed and where was helped me to solve most of my technical problems.

I would like to thank to all the people who suffered and enjoyed this Master with me, my class colleagues from the MAiR and my friends from ETSEIB.

Finally I would like to express my biggest acknowledgements to my parents, who supported me and encouraged me until the end of my studies.

Abstract

This thesis designs a method to control a quadrotor equipped with a robotic arm. The arm has been developed in Institut de Robòtica i Informàtica Industrial (CSIC-UPC).

During the project, an algorithm has been made as a first approximation to control a quadrotor which is working with the robotic arm.

In order to compensate the perturbations of the arm's dynamic, a nonlinear model predictive control algorithm has been chosen among other possible techniques. (PID, Model Predictive Control or LQR).

PID and model predictive control have been discarted because is not possible to control the nonlinearities of the system studied with a properly accuracy. Also there are no possibilites to restrict the system by using physical constraints.

Finally, five scenarios has been simulated and tested to verify the performances and robustness of the designed method. A takeoff maneuver, where the quadrotor reaches to a specific altitude. A hover mode where the system has to compensate the dynamics of the arm while it is static or it is in movement. Finally, the quadorotor has to move to a specific point in the space while the arm it is static or in movement.

The goal of the controller is to reject the perturbation due the arm and stabilize the system.

This thesis presents the results obtained after testing the controller designed with the scenarios created.

Contents

Acknowledgements iii

Abstract iv

List of Figures vii

List of Tables ix

Nomenclature xi

1 Introduction 1

1.1 Motivation . . . 1

1.2 Objectives . . . 2

1.3 Scope of Research . . . 2

1.4 Outline of the Thesis . . . 3

2 Background 5

2.1 State of the art . . . 5

2.2 Background . . . 7

2.2.1 Unmanned Aerial Vehicles . . . 8

2.2.2 Robotic arms . . . 10

2.2.3 Control Algorithms . . . 11

3 Implementation 15

3.1 Quadrotor Description . . . 15

3.1.1 Dynamic Equations of a X-Type Quadrotor . . . 15

3.1.2 Model of the Plant used by the NMPC . . . 22

3.1.3 Model to simulate the Real Plant . . . 25

3.1.4 Physical and Geometrical Parameters of the Quadrotor Model . . . 26

3.2 Robotic Arm Description . . . 27

3.2.1 Dynamic Equations of a Robotic Arm . . . 27

3.2.2 Model to simulate the Real Robotic Arm . . . 31

3.2.3 Physical and Geometrical Parameters of the Arm Model . . . 32

3.3 Nonlinear Model Predictive Control problem . . . 34

3.3.1 Dynamic model, state variables and control signals . . . 34

3.3.2 Horizon and Sampling Time . . . 35

3.3.3 Objective function . . . 36

3.3.4 Constraints . . . 37

4 Simulation Results **43**
4.1 Taking off 45
4.2 Static Control 51
4.2.1 Static arm 52
4.2.2 Dynamic arm 53
4.3 Dynamic Control 53
4.3.1 Static arm 55
4.3.2 Dynamic arm 56

Bibliography **57**

List of Figures

2.1 Examples of UAV 6
2.2 Top-Left: X-Type. Top-Right: V-Type. Bottom: Stingray-Type 9
2.3 Robotic arm designed by CSIC-UPC 10
2.4 Scheme of MPC algorithm 12
2.5 NMPC scheme 13

3.1 Blade system reference 16
3.2 Forward-Backward movement 18
3.3 Left-Right movement 19
3.4 Up-Down movement 19
3.5 Rotate around E_a^3 19
3.6 Parameters of a blade 20
3.7 Denavit hartenberg parameters 28
3.8 Matlab model of the robotic arm designed by CSIC-UPC. 33

4.1 Dynamic Reaction in the base of the arm due its no motion 45
4.2 Reference and Angle of each joint 46
4.3 Dynamic Reaction in the base of the arm due its motion 46
4.4 Cost function during the Take Off 48
4.5 Control signal for a Take Off 49
4.6 State variables for the Take off 50
4.7 State variables a static control with a static arm 52
4.8 Cost function during the static control with a static arm 53
4.9 Cost function during the dynamic control with a static arm 55
4.10 State variables during the dynamic control with a static arm 56

List of Tables

3.1 DH parameters of the robotic arm 33

4.1 Initial Conditions for Take Off 47
4.2 Summary of Take Off 50
4.3 Initial Conditions for Static Control 51
4.4 Summary of a static control with a static arm 53
4.5 Initial Conditions for dynamic Control 54
4.6 Summary of a dynamic control with a static arm 56

Nomenclature

A	Rotor disc area $[m^2]$
C_Q	Non-dimensional torque coefficient
C_T	Non-dimensional thrust coefficient
CoG	Center of gravity $[m]$
D_i	Rotor displacement from the flyer center of mass $[m]$
I	Rotational inertia $[kgm^2]$
ISE	Integral Squared Error
I_b	Rotor blade rotational inertia about the flapping hinge $[kgm^2]$
M_i	Momentum due the displacement of the thrust relative to the center of gravity $[Nm]$
Q_i	Torque from the i-th rotor
R	Rotation matrix
T_i	Thrust from the i-th rotor
ω_i	Angular velocity of rotor $[rad/s]$
ρ	Density of the air $[kg/m^3]$
σ	Non-dimensional rotor solidity
a	Non-dimensional blade lift slope gradient
$a1s_i$	First-harmonic longitudinal flapping coefficient $[rad]$
$b1s_i$	First-harmonic lateral flapping coefficient $[rad]$
c	Blade chord $[m]$

g Acceleration due the gravity $[m/s^2]$

m Mass of the Rotor $[kg]$

r Rotor radius $[m]$

$sk(\Omega)$ Skew-Symmetric matrix

UAV Unmanned Aerial Vehicles

Chapter 1

Introduction

1.1 Motivation

In the last years there has been an increasing importance of quadrotors in society and also of the tasks that they can do. There is a need for extending the capabilities of a quadrotor to satisfy the increasing demand of new services from the companies.

The motivation of this project is to provide a Nonlinear Model Predictive Control (NMPC) to a quadrotor's robotic arm attached to its body. Most of the literature explains methods about how to control an UAV applying a model predictive control to a linearized model, like in (Bouffard). Other kind of papers are focused on finding a properly mathematical model that explain most of the dynamics effects that occur during the maneuvers as it is well explained in (Pounds et al. [2006]). But there are also few papers like (Bangura and Mahony [2012]) that try to control the quadrotor by using a nonlinear model predictive control algorithm.

The quadrotor in which this research is based has been described in (Sanramaria and Andrade [2014]) and is used by Institut de Robòtica i Informàtica Industrial (CSIC-UPC) on its research in the Aerial Robotics Cooperative Assembly System ARCAS european project. This european project has the goal of design and develop cooperating flying robots for assembly operations.

This quadrotor will have to be able to cooperate with other robots in order to accomplish different goals like surveillance, assembly structures or track and detect objects.

With this thesis a new approach to control a quadrotor with a robotic arm is going to be presented.

1.2 Objectives

The main goal of this project is to use the NMPC method to control a quadrotor like in the literature existing. The new feature added is that the quadrotor has a robotic arm attached to its body and the NMPC has to compensate the perturbations generated by the motion of this arm.

The thesis presented here has the aim of developing a first approach of a controller able to manage these perturbations.

More specific objectives have been proposed as follows:

1. To design an algorithm to study the viability of the control.
2. To discretize and implement to Matlab environment the dynamic equations of the quadrotor and the robotic arm.
3. To couple the dynamic effects of the quadrotor and the arm.
4. To build a real parametrized model of a quadrotor and a robotic arm.
5. To design a NMPC to control the dynamics and carry the system to specific operational points.
6. To analyse the stability of the quadrotor while it is hovering and the arm is static or moving.
7. To analyse the stability of the quadrotor while it is moving and the arm it is static or moving.

1.3 Scope of Research

Some decisions have been taken by an heuristic method since were out of the scope of this research.

The sampling time and the control horizon have been chosen in order to properly run the simulations and verify the tests but in any case they have been optimized to run the NMPC algorithm at the fastest velocity.

The aim of this project is to design a NMPC for a simulation environment, so the code is not optimized to use it in real time.

Moreover, the identification of the models is out of the study proposed. During this thesis, parameters chosen to design the model of the quadrotor and the arm have been mixed from different researches. However, all the values selected have been verified to ensure that they were real and possible values.

Finally, the trajectory made by the arm is not controlled by the NMPC. Multiples PD have been designed to control each joint. Performance and stability of its trajectory are not taken into account during this study.

1.4 Outline of the Thesis

This study is organised as follows:

Chapter 2 - Background: The background contains all the theoretical information that is necessary to understand implementation and results. At the beginning a description of the state of the art has been presented. Below there is a detailed explanation about the UAV, the robotic arm and the control algorithm chosen for this study.

Chapter 3 - Implementation: In chapter 3 a description of the implementation has been presented. There is a detailed description of the equations that define the dynamics of the quadrotor and the robotic arm. There is also a explanation about the parameters that define the model of each device. Finally the NMPC designed is presented with all its features.

Chapter 4 - Results: In chapter 4 there are the results of the five scenarios that have been tested. Each section has a brief description about the initial conditions and a summary with the results obtained. Additionally in this chapter we have a description of the perturbations given by the arm.

Chapter 5 - Conclusions and Future Work: Conclusions derived from these results are presented in this chapter. It is commented the goals achieved and possible failures that happened. It is detailed possible improvements and fields with a interesting line of research.

Chapter 2

Background

2.1 State of the art

In the recent years, the relevance of the Unmanned Aerial Vehicles (UAV) has increased drastically. Nowadays there are a lot real examples where the UAV could be really helpful, and most of them are in the military field like in Sydney et al. [2013]. Moreover, according to Nonami [2006], the UAV will be completely integrated in our society in the next few years.

Although it is possible to delegate to an UAV a lot of tasks that for a human being would take much time or even risk (Bouabdallah et al. [2004]), the majority of those tasks still need the human intervention to achieve their goals. Some examples could be aerial photography, television, cinema shootings (Wai Weng [2006]), or even uses in the research field, which sometimes needs to perform aerial experiments.

However, there exists a huge number of potential applications to be developed where the UAV could be completely autonomous. Today, most of the big companies are working on implementing new services using autonomous UAVs. Surveillance, crowd control, mine detection or aerial delivery of payload like it is explained in (Oleg [2009]) are some of the examples.

In order to use autonomous UAVs it is necessary to develop new methods and algorithms to control these vehicles. The model predictive control methods used in Raemaekers [2007] or in Grancharova et al. [2012] is one clear example. However, right now almost

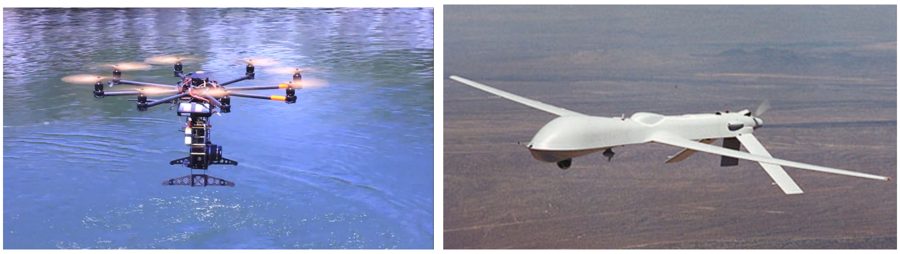

Figure 2.1: Examples of UAV

all the external perturbations are compensated by a human. In this way, the final target is to find a method able to understand the situation, predict what is going to happen and therefore correct the behaviour of the UAV in order to accomplish the necessary performance to reach the goal.

The are several types of UAVs (see figure 2.1) and each one reacts different in front the perturbations. At the moment, the most integrated and popular UAV in our society is the quadrotor. This mechanism was conceived in 1907 by Breguet and Richet as it is described in Leishman [2002]. The first model was a large and heavy model that could lifted only over a small height and for short duration. After a lot of efforts the quadrotor is having the people attention and each day its relevance is increasing for the companies.

To modelling the quadrotor physics, some works like Belkheiri et al. [2012] and Zhu and Huo [2010] approximate its dynamics by a linear system, for which standard linear controllers can be designed. Other authors use nonlinear control techniques, as for example Mellinger and V. [2011] that used feedback linearization, Vries and Subbarao [2010] employing backstepping techniques, or Benallegue et al. [2006] developing sliding mode control.

Relative to our control method, some authors like Bangura and Mahony [2012] have developed a nonlinear model predictive control (NMPC) to control the dynamics of a quadrotor. Other sophisticated projects have even designed a controller with a model of the wind perturbation like in Alexis et al. [2010].

On the other hand, the use of robotics arms in the society is widespread in several areas. Some examples can be the use of robotic arms for cooperation tasks as did Hayati [1986], space as in Fukuda [1985] or even surgery as Velliste et al. [1995] did. Moreover, there are a lot of studies and works about how to control and minimize errors, like in Bicchi and Tonietti [2004], and their dynamics and behaviors are well known by studies like

Herrera et al. [2012]. In this way, it is a good choice to use robotic arms built on the quadrotors in order to add new features to this flying machines. By using both robotic structures, new applications could be developed like collaborative tasks to move objects by pincers, track object with a camera in the end-effector as it is described in (Allen et al. [1993]) or anchoring the quadrotors to recharge its batteries.

This thesis develops 6DOF model for a quadrotor that includes a model of the perturbation of the robotic arm. Also design a nonlinear model predictive control to compensate the perturbations due the arm.

2.2 Background

The three main elements used during this project are:

The first element is an UAV, more specifically a quadrotor. In these times, it is the most popular vehicle in the commercial field and so the most studied system, apart from being the cheapest model that can be found. This reasons are the ones why most of researcher laboratories are focusing their studies on this system.

The second element is a robotic arm. This mechanism increases the versatility of the quadrotor. At present there are lots of types of robotics arms. Considering that a quadrotor has a small payload, the arm has to be light but strong enough to carry the maximum weight that is possible. So, in that case, the robotic arm designed in the CSIC-UPC (section 2.2.2), described in (Sanramaria and Andrade [2014]), has been chosen to test the methods explained in this thesis.

The third element is the control algorithm. A nonlinear model predictive control has been selected to control the whole system. The main reason is because most of the problems that have been had to face could be represented as an optimization problem. This implies that it is possible to solve the problem subject to certain restrictions. It is also possible, because the dynamic systems of the quadrotor and the arm are well known, to use it as a model in the controller. On the other hand, due to nonlinearities, is important to use a robust algorithm to absorb all non controlled changes.

In the next sections more details of each part are presented.

2.2.1 Unmanned Aerial Vehicles

UAV is an acronym for Unmanned Aerial Vehicle, which is an aircraft with no pilot on board. Usually, UAVs are controlled remotely and can fly autonomously based on either pre-programmed flight or using more complex dynamics.

The UAV concept is becoming more popular each year. There are a lot of types, sizes and prices (see figure 2.1). The purpose of this thesis is to equip an arm to a quadrotor and control its dynamics by using a nonlinear model predictive control. The robotic arm generates a perturbation that the control algorithm has to manage in order to maintain the system in a fixed position or move the quadrotor to a desired position. So an UAV with the hover capability is needed in order to hold a fixed position while an arm is working in some task.

Keeping this in mind, the most used UAV are based on a system with vertical thrust. That kind of technology consists in a generation of a vertical force to compensate the gravity component of the whole dynamic system. It is also possible, by combining the different forces generated, to obtain other forces and torques in all axis to move the robot to any position with a desired motion as it is described in (Pounds et al. [2006]).

There are several types of rotor systems in function of the task that it has to do. Depending of the autonomy, the payload or maneuverability, the number of rotors can change between models for example this eight rotor UAV described in (Romero et al. [2009]). On the other hand, other features that could change drastically the behavior of the UAV are the propellers as it explained in (Pounds).

This study is based on a model of four rotors, henceforth named quadrotor.

These vehicles are used to be controlled with a electronic system able to stabilize the aircraft. These UAV can be flown indoors as well as outdoors because of the small size and agile maneuverability. However, if the quadrotor has any variable perturbation, its performance could drop until being unstable. This is the reason why it is important to design a control algorithm able to manage these perturbations.

Even in the quadrotor category, there are several types depending on the position of the rotors relative to its center of mass (X-type, V-type or Stingray-type, see figure 2.2).

During this thesis a X-Type has been chosen because their dynamic model is more easy to manage than other kind of structures. This geometry consists in situate four rotors in a cross position with two sets of identical rotorcrafts, two of them are spinning clockwise and two counter-clockwise in order to compensate the momentum generated as it is explained in (Martinez).

A mathematical model extracted from other researches has been used to emulate its behavior in a simulation environment. The majority of papers have used the same model introduced in the nineties by Pounds and Bouabdallah [2007]. In this papers, thrust and torque are modeled as a static function of the square of rotor speed. This model is based on static thrust characteristics of the rotor and holds for near hovering flights. But more complex model (Pounds et al. [2006]) includes the effect of the rotor blade flapping and the effects of translational lift.

Figure 2.2: Top-Left: X-Type. Top-Right: V-Type. Bottom: Stingray-Type

This research uses a generic model of quadrotor to apply a nonlinear control to accomplish certain targets. The model chosen it is described in Pounds et al. [2006] and during the next chapters, a more detailed explanation about the formulation is going to be presented.

2.2.2 Robotic arms

There are several kind of arms and a huge number of studies about their dynamics like in Featherstone and Orin [2000]. Implementing a robotic arm to a quadrotor would extend the capabilities and services that an UAV could offer.

During this thesis another research group in CSIC-UPC has been working on designing a robotic arm (see figure 2.3) able to be equipped to a quadrotor.

This arm is light and enough strong to carry several tools on its end effector. Actually, this group is working on using a camera to track a tag using the quadrotor (Sanramaria and Andrade [2014]).

In order to control both (quadrotor and arm) it is necessary to know precisely all the parameters of the robot to determine a mathematical model. Thanks to CSIC-UPC arm design, it is possible to know all this values and then use it in a simulation environment.

This arm is specially designed to be equipped in a quadrotor (lightness, strength, well balanced, center of mass aligned with its base and well known model) and because of this it has been selected as the arm to be modeled and to obtain the perturbations of the system.

Its dynamic has been found by using the Recursive Newton Euler algorithm as it is presented in Khosla and Kanade [1987].

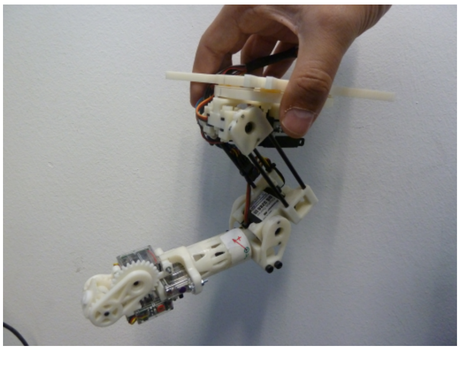

FIGURE 2.3: Robotic arm designed by CSIC-UPC

In chapter 3 a full description of the arm and its dynamic is going to be explained.

2.2.3 Control Algorithms

Finally, the last element necessary to control a quadrotor equipped with an arm, is it's control algorithm.

At the beginning, a black box model was selected as the dynamic model of the whole system, so the first try was study the problem as generic as possible.

After realizing that there are a strong coupled effects between the quadrotor and an arm, a different strategy was chosen. Those coupled effects prevented from finding a relation between the dynamics of the quadrotor and the dynamics of the robotic arm.

As the dynamic model of a quadrotor and an arm is well known, a black box model could be discarded in order to use parametrized models.Once the model type has been chosen, the whole problem could be solved as an optimization one.

Standard approaches to quadrotor control have been based on linear controller design. These methods include proportional integral derivative (PID) controllers (Noth et al. [2004]), linear quadratic regulator (LQR) (Shin et al. [2005]) or a robust H_∞.

Another method used, consists in linearize the model around a certain state of the quadrotor and then applying a predictive control model around this point (Bouffard), (Bresciani).

A MPC has the ability to anticipate future events and can take coordinated actions by using dynamic models of the process.

This algorithm is based on an iterative process which solves an optimization problem along a finite horizon. For each time t, the current plant state is sampled by using a linear model, and a minimizing of a cost function is computed for a short time horizon in the future $[t, t+T]$ (control horizon). Once the problem is solved a vector of control signals is found. This vector is the trajectory of the control signals that minimize the cost function along the control horizon. From all this vector, only the control signals corresponding to the time t are applied to the real plant in the evolution of the system. Then, a new value of the state variables of the plant is obtained and the algorithm is repeated again t.

In the figure (2.4) a brief scheme of the process is shown.

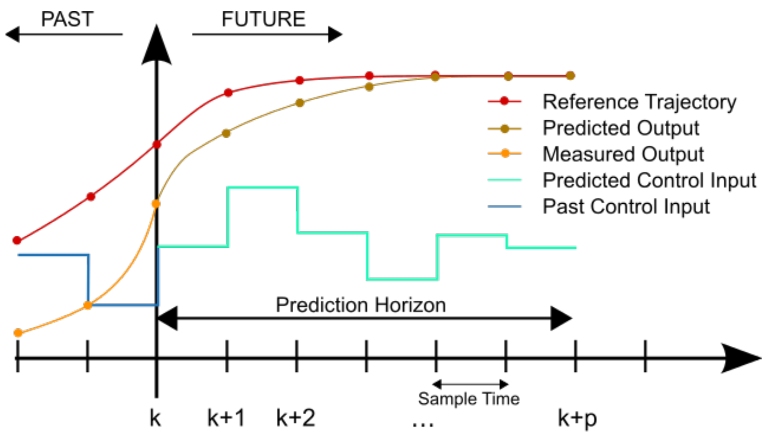

FIGURE 2.4: Scheme of MPC algorithm

The cost function to minimize involves all the state variables and the control signals as follows:

$$J = \sum_{i=1}^{N} \gamma_{x_i} (x_{desired} - x_i)^2 + \sum_{i=1}^{N} \gamma_{u_i} \Delta {u_i}^2$$

Where x_i is the state variable i-th, the $x_{desired}$ is the reference of the state variable i-th, the u_i is the control signal i-th, the γ_{x_i} is the reltive weight of the state variable i-th, γ_{u_i} is the relative weight of the control signal i-th and N is the control horizon length.

Also different types of cost function could be added to obtain the desired performance.

The minimization of the cost function is subjected to some certain constraints defined by the designer in order to determine a state-space where the problem should be solved. To obtain the state variables in a future prediction, is necessary a mathematical model of the plant.

The main restriction of this method is that the modeled plant used on the minimization of the cost function should be linear. This method ignores most of the dynamic effects of the system and also cannot represent accurately the real behavior of the quadrotor. I's also difficult to add a new model or perturbation with this strategy.

There is a variant of this algorithm for more complex systems called Nonlinear Model Predictive Control. Adding the nonlinear part improves the response of the system at the expense of increasing the necessary computational burden.

The numerical solution of the NMPC optimal control problems is typically based on direct optimal control methods using Newton-type optimization schemes. NMPC algorithms typically take advantage of the fact that consecutive optimal control problems are similar to each other.

This allows to initialize the Newton-type solution procedure efficiently by a suitable shifted guess from the previously computed optimal solution, saving considerable amounts of computation time.

A scheme of the global workflow is presented in the figure (2.5).

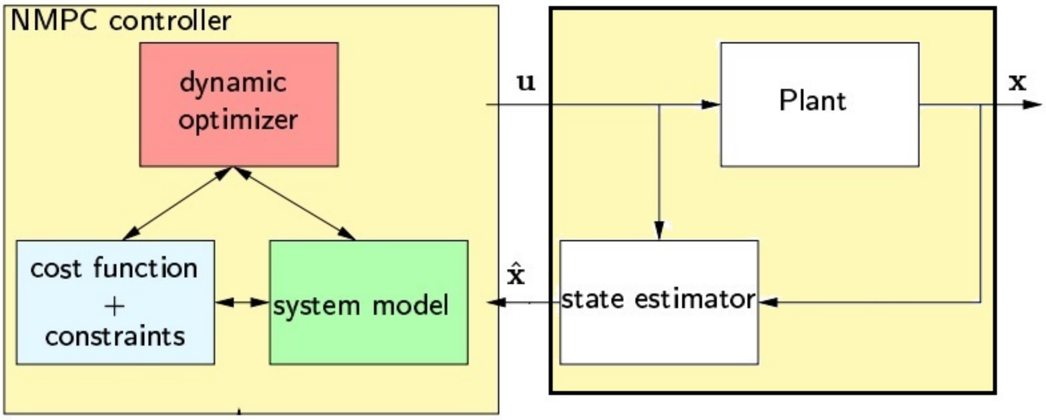

FIGURE 2.5: NMPC scheme

Plant: System to be controlled.

State Estimator: Make a estimation of the state variables.

Dynamic Optimazer: Solve the optimal problem.

System Model: Mathematical model of the behavior of the plant.

Cost Function: Function that should be minimized.

Constraints: Restrictions that should be respected to solve the optimal problem.

The Nonlinear Model Predictive Control algorithm has been chosen to control the quadrotor equipped with an arm. A more detailed explanation is presented in section (3.3).

On the other hand, the robotic arm has been treated as a dynamic perturbation of the quadrotor, so there is no need to control its joints. Nevertheless, is necessary to know the exact model of the arm to calculate the dynamic reactions applied to the quadrotor due to the motion of the arm during its trajectory.

The trajectory of the arm and their perturbations are calculated before applying the NMPC algorithm, so it turns out as a parameter of the system. This means that for each instant of time is possible to know the reaction of the arm into the quadrotor. As the aim of the Thesis is to compensate this perturbation, a predictive algorithm is used to minimize the error.

The main problem of the strategy in a real case is that a good estimation of the state variables of the quadrotor it is necessary. However, this project is developed in a simulation environment, so it's possible to obtain the state variables of the plant by using a model of the system.

During the next chapters, a more detailed specification of the problem will be presented.

Chapter 3

Implementation

In this section, a full description of the dynamic equations of the quadrotor and its model is explained. Also, the Newton-Euler solution for the particular case of the arm selected is detailed. Finally the description of NMPC designed it is presented at the end of this chapter.

3.1 Quadrotor Description

3.1.1 Dynamic Equations of a X-Type Quadrotor

The mathematical model of the quadrotor comes from the description done in (Pounds et al. [2006]).

Let's suppose two frames (see figure 3.1):

1. Inertial frame: $I = \{E_x, E_y, E_z\}$ where E_z is in the direction of gravity.
2. Body frame: A frame located in the body of the quadrotor. Its axis are denoted by $I = \{E_1^a, E_2^a, E_3^a\}$ and with the center in $\xi = \{E_x, E_y, E_z\}$.

Both frames are related by a rotation matrix $R : A \to I$

Let's say $V(t)$ and $\Omega(t)$ are the linear and angular velocity of the frame A in base A.

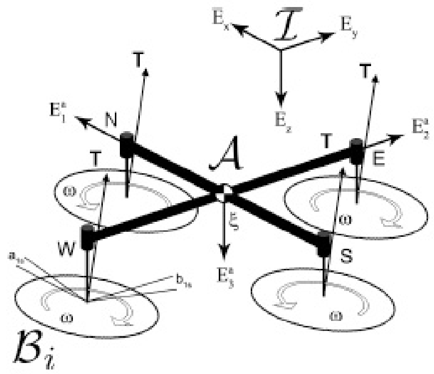

Figure 3.1: Blade system reference

The dynamic equations of the quadrotor are:

$$\dot{\xi}(t) = RV(t) \tag{3.1}$$

$$m\dot{V}(t) = -m\Omega(t) \times V(t) + mgR^T E_3^a + \sum_{i=1}^{4} T(t)_i \tag{3.2}$$

$$\dot{R}(t) = Rsk\left(\Omega(t)\right) \tag{3.3}$$

$$I\dot{\Omega}(t) = -\Omega(t) \times I\Omega(t) + \sum_{i=1}^{4} \left[Q(t)_i + M(t)_i\right] \tag{3.4}$$

$$T_i(t) = C_T \rho A r^2 \omega(t)_i^2 \begin{pmatrix} -cos(b1s_i)sin(a1s_i) \\ sin(b1s_i) \\ -cos(a1s_i)cos(b1s_i) \end{pmatrix} \tag{3.5}$$

$$Q(t)_i = C_Q \rho A r^3 \omega(t)_i |\omega(t)_i| E_3^a \tag{3.6}$$

$$M(t)_i = T(t)_i \times D_i \tag{3.7}$$

Where m is the mass of the rotor $[kg]$, I is the rotational inertia $[kgm^2]$, ρ is the density of the air $[kg/m^3]$, g is the acceleration due the gravity $[m/s^2]$, r is the rotor radius $[m]$, A is the rotor disc area $[m^2]$, ω_i is the angular velocity of the rotor i-th $[rad/s]$, $sk(\Omega)$ is the skew-Symmetric matrix, R is the rotation matrix which it is constructed by the yaw-pitch-roll = $(\varphi,\, \theta,\, \psi)$ euler angles.

D_i is the rotor displacement from the flyer center of gravity (CoG) $[m]$ ($D_1 = (0, d, h)$, $D_2 = (0, -d, h)$, $D_3 = (d, 0, h)$ and $D_4 = (-d, 0, h)$ with $d[m]$ and $h[m]$ as a length and height respectively above the CoG of the rotor).

T_i is the thrust from the i-th rotor $[N]$, Q_i is the torque from the i-th rotor $[Nm]$, M_i is the momentum due the displacement of the thrust relative to the CoG $[Nm]$, C_T is the non-dimensional thrust coefficient, which is an experimental parameter that could vary slightly. C_Q is the non-dimensional torque coefficient, which is another experimental parameter.

$a1s_i$ is the longitudinal flapping coefficient $[rad]$ and $b1s_i$ is the lateral flapping coefficient $[rad]$. Both parameters are related with the blade flapping effect. This values are going to be explained a few lines later.

By changing those parameters and using the equations (3.1), (3.2), (3.3), and (3.4) it is possible to determine the behavior of a quadrotor:

- Equation (3.1) relates the linear velocity of the CoG between inertial frame and the body frame.
- Equation (3.2) shows that the sum of the forces applied on the quadrotor have to be proportional to its linear acceleration. It is splitted in three different components. First, is the force due the coriolis effect. Second, is the force because the gravity. And finally, is the thrust generated by each rotor.
- Equation (3.3) allows to relate the euler angles rates with the angular velocity of the inertial frame.
- Equation (3.4) express that the sum of the torques applied have to be proportional to its angular acceleration. As before, the final torque it is explained in three elements. First, it is the torque created by an inertial system that is rotating around an axis. Q_i corresponds to the torque generated by each rotor due its

angular velocity. $M(t)_i$ is the momentum created by the difference between the thrust of one of the rotors and the thrust generated by the opposite rotor.

- Equation (3.5) is generated by the angular velocity of the rotor. It is dependant of the geometry of the propellers and the density of the air. It is multiplied by a vector that modify the final direction of the thrust. Instead of a vertical direction, appears a component on E_1^a and E_2^a due of the flapping blade effect (see equations (3.8), (3.9), (3.10)).

- Equation (3.6) is the torque generated by a mass (propeller) rotating around an axis (rotor).

- Equation (3.7) is generated because the thrust it is not applied in the CoG and then a torque appears in the frame of the quadrotor.

The quadrotor can move in the three axis by combining the thrust and the torque of each rotor:

Move forward-backward: To move forward or backward it is necessary to make a difference between the thrust generated by the front rotor and the back rotor as it is represented in the figure (3.2). By this process the $\sum_{i=1}^{4} M_i$ is not zero so the quadrotor has a slight torque that rotates the system until the forces and torques are balanced again. A pitch angle it is generated, which changes the direction of the thrust. A new component of force appears toward E_a^1 axis that allows an acceleration and finally a velocity.

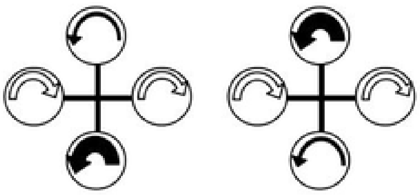

FIGURE 3.2: Forward-Backward movement

Move left-right: As it is shown in the figure (3.3), in order to accomplish these movements, it is necessary to create a difference between the thrust generated by the lateral rotors. With the same concept explained before, a new torque is generated and the quadrotor bends a roll angle. Finally the thrust is splitted in a vertical component and in a component in the E_a^2 axis.

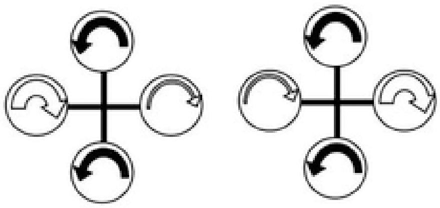

FIGURE 3.3: Left-Right movement

Move up-down: The figure (3.4) describes that to move up or down it is necessary that the sum of all vertical thrusts component is more or less than the force due of the gravity.

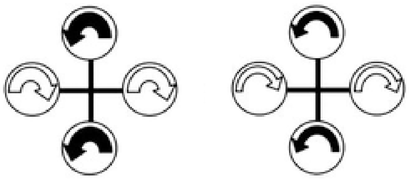

FIGURE 3.4: Up-Down movement

Rotate around E_a^3 (yaw motion): To rotate around the E_a^3 it should be a torque around this axis. There are two motors that are rotating clockwise and two motors that are rotating counterclockwise in order to compensate the torque created by the rotation of each rotors. (See figure 3.5). So the only way to change the yaw angles is unbalancing the torques generated by each rotor when they are spinning. As it is possible to check in the equation (3.6), the torque generated by the rotor it is controlled by its angular velocity. So changing the velocity of the rotors it is possible to reach to a determined yaw angle.

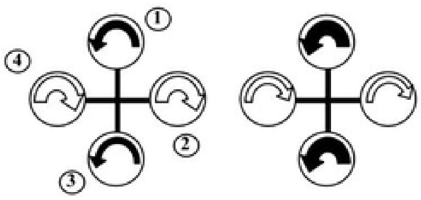

FIGURE 3.5: Rotate around E_a^3

By combining the fourth movements, it is possible to control its position and its yaw angle. Due the dynamic of the system, in order to fix a position, it is necessary to minimize the pitch and roll angles.

As it is detailed in the movement forward-backward, the pitch and roll angles generate a deviation of the thrust and finally become in a motion in E_a^1 and E_a^2.

On the other hand, the blade flapping effect also could change the vertical thrust and make new components in the E_a^1 and E_a^2.

Front side of the rotor disk is called the advancing side, and the back side is called the retreating side. Blade flapping occurs when the rotors translate horizontally. In this case, a different lift between the advancing and retreating blades appears causing the rotor tip path plane tilts. In order to modelize this effect it should solve the constant and sinusoidal components of the blade centrifugal aerodynamic weight and find the tilt angle of the plane as it is explained in (Hussein).

The flapping of the rotor it is found by calculating the magnitude and direction of the rotor's translation. It is necessary to define a new local reference frame located in the rotor and aligned in the direction of the rotor's movement (B_i) as it is described in (Pounds et al. [2006]).

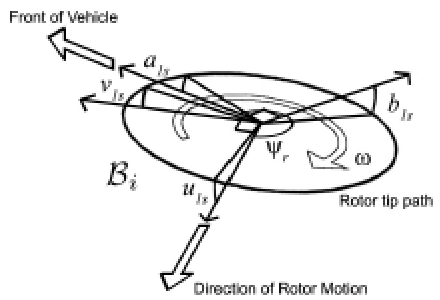

FIGURE 3.6: Parameters of a blade

In this frame, the longitudinal and lateral flapping angles (u_{1s_i}, v_{1s_i}) are calculated and then re-expressed them into the body frame of the quadrotor ($a1s_i$, $b1s_i$).

The expressions are the following:

$$v_{ri} = V(t) + \Omega(t) \times D_i \tag{3.8}$$

$$\mu(t)_{ri} = \frac{\|v(t)_{ri(1,2)}\|}{\omega(t)_i R} \tag{3.9}$$

$$\Psi(t)_{ri} = arctan(\frac{v(t)_{ri(2)}}{v(t)_{ri(1)}}) \tag{3.10}$$

Equation (3.8) denotes the velocity vector of the i-th rotor. It is the sum of the velocity of the quadrotor plus the velocity due the angular rotation of the system. Equation (3.9) is the ratio between the magnitude of the advance velocity and the linear velocity of the blades. It is called advance ratio. Equation (3.10) is the azimuthal direction of the motion. On the other hand, the longitudinal and lateral flapping angles in the local reference B_i are:

$$u(t)_{1s_i} = \frac{1}{1 - \frac{\mu(t)_{ri}^2}{2}} \mu(t)_{ri}^2 (4\theta_i - 2\lambda_i) \tag{3.11}$$

$$v(t)_{1s_i} = \frac{1}{1 + \frac{\mu(t)_{ri}^2}{2}} \frac{4}{3} (\frac{C_t}{\sigma} \frac{2}{3} \frac{\mu(t)_{ri}\gamma}{a} + \mu(t)_{ri}) \tag{3.12}$$

$$\lambda_i = \sqrt{\frac{C_T}{2}} \tag{3.13}$$

$$\gamma = \frac{\rho a c r_r^4}{I_b} \tag{3.14}$$

Where σ is the rotor solidity which is the ratio of the total blade area to the total disk area, a is the blade lift slope gradient, c is the blade chord $[m]$, r_r is the rotor radius $[m]$, I_b is the rotor blade rotational inertia about the flapping hinge $[kgm^2]$.

Equation (3.11) is the longitudinal flapping angle. Depends on parameters of the blade and the inflow of the rotor.

Equation (3.12) is the lateral flapping angle. Depends on the geometry of the blade, the inflow of the rotor and the Lock Number which is defined in Pounds et al. [2006].

Equation (3.13) is an approximation of the inflow of the rotor.

Equation (3.14) is the Lock Number that represents the ratio of the aerodynamic and inertial forces on the blade.

All these parameters are related with the geometry of the blades.

Once the longitudinal and lateral flapping angles are calculated in the local body frame of the rotor, they are transformed back into the quadrotor body frame using the frame mapping $J^A_{B_i}$.

$$J^A_{B_i} = \begin{pmatrix} cos(\Psi_{ri}) & -sin(\Psi_{ri}) \\ sin(\Psi_{ri}) & cos(\Psi_{ri}) \end{pmatrix} \tag{3.15}$$

Finally, the influence of the pitch and roll rates are added to the flapping angles.

$$\begin{pmatrix} a1s_i \\ b1s_i \end{pmatrix} = J^A_{B_i} \begin{pmatrix} u_1 s_i \\ v_1 s_i \end{pmatrix} + \begin{pmatrix} \frac{\frac{16}{\gamma} \cdot \frac{\Omega_x}{\omega_i} + \frac{\Omega_y}{\omega_i}}{1 - \frac{\mu_{ri}^2}{2}} \\ -\frac{\frac{16}{\gamma} \cdot \frac{\Omega_y}{\omega_i} + \frac{\Omega_x}{\omega_i}}{1 - \frac{\mu_{ri}^2}{2}} \end{pmatrix} \tag{3.16}$$

with Ω_x and Ω_y as a pitch and roll rates, ω_i angular rate of the rotor i-th.

These values represent the effect of the geometry and physics properties of the propellers. Depending on the model and the type of those blades, the behavior of the system varies.

Also, by modeling this effect, it is possible to correct the non controlled lateral movement when the quadrotor is hovering.

Still, the equations here presented are a mathematical approximation of the physics of a blade where some assumption have been done as it is eplained in Johnson [1994] and in Bangura and Mahony.

Using the equations described along this section, the behavior of a quadrotor is completely defined.

Those equations are going to be discretized in the next section.

3.1.2 Model of the Plant used by the NMPC

To control the system through a NMPC it is necessary to have an accurate model of the plant. The equations presented in the section (3.1.1) have been discretized and transformed into an algorithm to predict the behavior of the quadrotor.

To define a model the state variables and the control signals have to be defined:

$(x_1, x_2, x_3) = \xi = (x, y, z)$: Position relative to the inertial frame.

$(x_4, x_5, x_6) = n = (\psi, \theta, \varphi)$: Euler angles (yaw, pitch, roll).

$(x_7, x_8, x_9) = V = (\dot{x_1}, \dot{x_2}, \dot{x_3}) = (V_x, V_y, V_z)$: Linear velocity relative to the inertial frame.

$(x_{10}, x_{11}, x_{12}) = \Omega = (\Omega_x, \Omega_y, \Omega_z)$: Angular velocity of the body frame.

The control variables are the signals able to control the whole plant. In this case, the controllable values are the angular rate of the rotors.

$(u_1, u_2, u_3, u_4) = u = (\omega_1, \omega_2, \omega_3, \omega_4)$: Angular rate of each rotor.

The equations (3.1), (3.2) and (3.4) provide the derivative of the position, orientation and velocities, so in order to compute the state variables in a simulation environment for each sampled time it is necessary to discretize them:

1. Position:

$$\frac{\xi(k+1) - \xi(k)}{T_s} = RV(k) \tag{3.17a}$$

$$\xi(k+1) = (x, y, z)^T(k+1) = (x, y, z)^T(k) + RV(k)T_s \tag{3.17b}$$

2. Linear Velocity:

$$m\frac{V(k+1) - V(k)}{T_s)} = -m\Omega(k) \times V(k) + mgR^T E_3^a + \sum_{i=1}^{4} T(k)_i \tag{3.18a}$$

$$linAcc(k) = -\Omega(k) \times V(k) + gR^T E_3^a + \frac{1}{m}\sum_{i=1}^{4} T(k)_i \tag{3.18b}$$

$$V(k+1) = (V_x, V_y, V_z)(k+1) = (V_x, V_y, V_z)^T(k) + linAcc(k)T_s \tag{3.18c}$$

$linAcc$: Linear acceleration.

3. Euler Angles:

$$\frac{n(k+1) - n(k)}{T_s} = W^{-1}\Omega(k) \tag{3.19a}$$

$$n(k+1) = (\psi, \theta, \varphi)^T(k+1) = (\psi, \theta, \varphi)^T(k) + W^{-1}\Omega(k)T_s \tag{3.19b}$$

W^{-1}: Is the inverse of the wronskian. This matrix transforms the euler angles rates to angles rates of the inertial frame (See figure 3.1).

4. Angular Velocity:

$$I\frac{\Omega(k+1)-\Omega(k)}{T_s} = -\Omega(k) \times I\Omega(k) + \sum_{i=1}^{4}[Q(k)_i + M(k)_i] \tag{3.20a}$$

$$angAcc(k) = -I^{-1}\Omega(k) \times I\Omega(k) + I^{-1}\sum_{i=1}^{4}[Q(k)_i + M(k)_i] \tag{3.20b}$$

$$\Omega(k+1) = (\Omega_x, \Omega_y, \Omega_z)^T(k+1) = (\Omega_x, \Omega_y, \Omega_z)^T(k) + angAcc(k)T_s \tag{3.20c}$$

$angAcc$: Angular acceleration.

5. Dynamic Equations:

$$T_i(k) = C_T \rho A r^2 \omega(k)_i^2 \begin{pmatrix} -cos(b1s_i)sin(a1s_i) \\ sin(b1s_i) \\ -cos(a1s_i)cos(b1s_i) \end{pmatrix} \tag{3.21a}$$

$$Q(k)_i = C_Q \rho A r^3 \omega(k)_i |\omega(k)_i| E_3^a \tag{3.21b}$$

$$M(k)_i = T(k)_i \times D_i \tag{3.21c}$$

These equations allow to compute the position of the quadrotor for each sampled time.

To couple the quadrotor and arm dynamics, the discretized equations (3.18) and (3.20) have to be modified:

$$linAcc(k) = -\Omega(k) \times V(k) + gR^T E_3^a + \frac{1}{m}\sum_{i=1}^{4}[T(k)_i] + F_{ar}(k) \tag{3.22a}$$

$$angAcc(k) = -I^{-1}\Omega(k) \times I\Omega(k) + I^{-1}\sum_{i=1}^{4}[Q(k)_i + M(k)_i] + \tau_{ar}(k) \tag{3.22b}$$

The dynamic effects of the robotic arm are coupled by adding their reaction forces F_{ar} and their reaction torques τ_{ar} into the equations that computes the accelerations.

In the section (3.2) it is presented a full detailed explanation about how to obtain the reaction of the forces and torques of the arm.

The model is defined by the state variables, obtained by the equations, and the control signals, defined as the angular rate of each rotor. By changing the rotors speeds it is possible to move the quadrotor to a desired position.

3.1.3 Model to simulate the Real Plant

Once the NMPC solves the problem, the control signals that minimize the cost function are found. The framework of this study is limited to simulation scenarios so it is necessary to have a model to emulate the real behavior for each sampled time and to obtain the states variables for the next optimization.

I.t is important that the model used for the controller is different from the model used to simulate the real plant. In this way it is possible to tune the NMPC to absorb the possible errors and increase its robustness.

So the model to emulate the behavior of the quadrotor with the arm it is a simplification of the real model described in the section (3.1.2). In this case, all the flapping effects will be removed from the dynamic equations.

$$a1s_i = 0$$

$$b1s_i = 0$$

Which means that the equation (3.21) has been modified as follows:

$$T_i(k) = C_T \rho A r^2 \omega(k)_i^2 \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} \tag{3.23}$$

Usually, in a real environment, these values are calculated using an estimator with sensor fusion algorithms like it is described in Kis and Lantos [2011]. These techniques give an estimation of the state variables of the real system which are used to set the initial conditions of the NMPC algorithm for the next iteration.

3.1.4 Physical and Geometrical Parameters of the Quadrotor Model

The equations defined in the section (3.1.1) are a parametrization of the behavior of the system.

To obtain a realistic behavior of the quadrotor is important to adjust the physical and geometrical parameters to real values. Focused on this aim, most of them have been taken from the quadrotor used by the researcher group that has motivated this study.

This team has been working with a modification of the Pelican UAV from Ascending Technologies and its parameters could be found in its datasheet.

Other parameters that are used during this thesis come from the studies Bresciani and Gruene and Pannek [2011].

Some of them are experimental and identify them were out of the scope of this research, so they could not be verified. However, these parameters are taken from other studies that have checked its authenticity.

Those parameters are split in several categories:

1. Physical: Gravity, viscosity and density of the air are defined to compute all the physical equations.

2. Airframe: Its mass and its inertial matrix have been defined in this section. Also the horizontal and vertical displacement of the rotors relative to the CoG.

3. Rotor: All the parameters about the blade geometry and its physics are described in this category. Here is where all the flapping blade effects are determined in function of the type of blade used by the quadrotor. Also its mass and its inertial matrix are detailed.

4. Constants: Some constants are precalculated to optimize the code during the control loop.

By changing this parameters and using the dynamic equations shown in the section (3.1.1), it is possible to simulate the behavior of any type of X-type quadrotor.

The versatility of this method has allowed us to create an algorithm that is independent of the quadrotor used during the control process.

3.2 Robotic Arm Description

In this section the parameters and the physics that define the behavior of a robotic arm will be detailed.

3.2.1 Dynamic Equations of a Robotic Arm

The aim of this section is to explain the dynamic equations of the robotic arm. First, it is necessary to explain the parametrization of the arm and then how to calculate the dynamic reaction on its base.

It is possible to modelize a robotic arm with serial joints by using the Denavit-Hartenberg parameters.

These variables (also called DH parameters) are four values that define a particular convention of how to attach the reference frame of the links for a chain of joints.

This convention consists in assign a coordinate frame to each link. For each joint there is a matrix transformation that allows to change from one frame to the next. Concatenating these transformation matrix it is possible to relate the end-effector frame with the base frame.

Depending of the type of the joint (hinge or sliding), one of the four parameters is the variable value and the other are constants.

The algorithm to select the right frames S_i that fulfil the DH parameters is:

1. The z-axis is in the direction of the joint axis. I.e. the rotation axis or the displacement axis.
2. The x-axis is the parallel to the common normal $x_n = z_{n-1} \times z_n$. The direction of x_n axis is from z_{n-1} to z_n.
3. The y-axis is selected to get the frame coordinate that accomplish the right-handed rule.

The four parameters are:

d_i: Offset along previous z to the common normal. $Distance_{z_{i-1}}(S_{i-1}, z_{i-1} \cap x_i)$.

θ_i: Angle about previous z from old x to new x. $Angle_{z_{i-1}}(x_{i-1}, x_i)$.

a_i: Length of the common normal. For a revolute joint, is the radius about previous z. $Distance_{x_i}(z_{i-1} \cap x_i, S_i)$.

α_i: Angle about common normal from the old z to new z. $Angle_{x_i}(z_{i-1}, z_i)$.

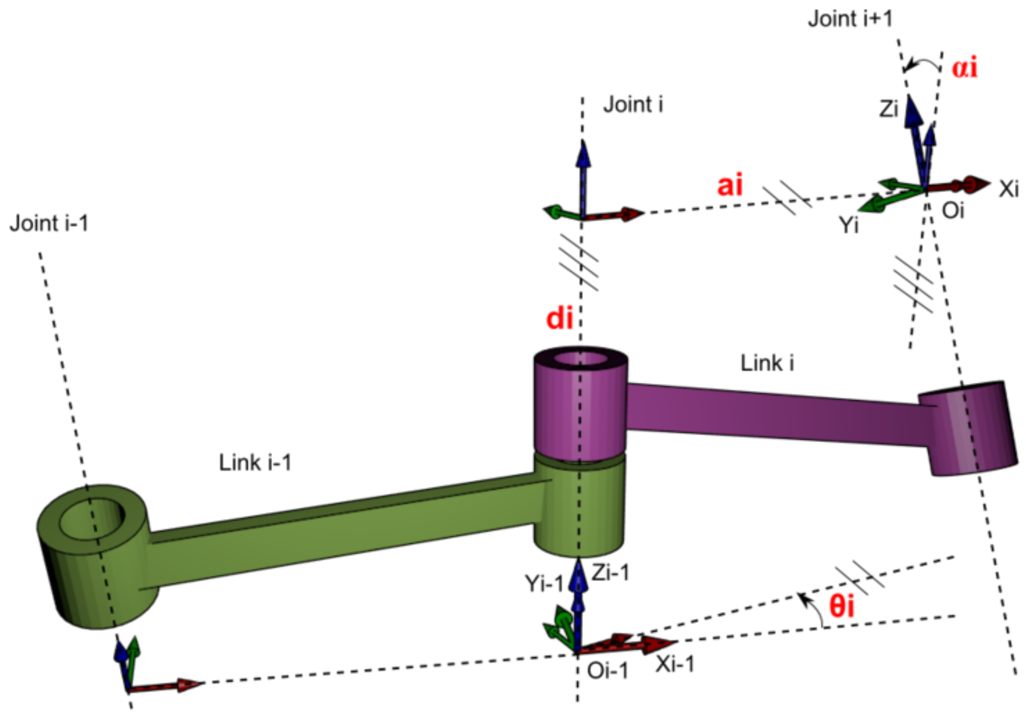

Figure 3.7: Denavit hartenberg parameters

The DH algorithm determines the configuration of the robotic arm and gives to us an unambiguous description of the system. Also, knowing the angles of each joint it is possible to exactly calculate the position and orientation of the end-effector relative to its base.

This information will be important to estimate the dynamic reactions into the base due the motion of the end-effector. The final target is to obtain these forces and torques in order to couple them to the quadrotor as a perturbation as it is defined in section (3.1.2).

The method selected to get the dynamic equations is the Recursive Newton-Euler (RNE) algorithm applied into a robotic arm as it is explained in Featherstone and Orin [2000].

The RNE algorithm has two steps. The first step allows to calculate the linear and angular velocity of each link by going from the base to the end-effector. The second step uses those velocities to solve the dynamic equations to obtain the force and torque of each joint from end-effector to base. So finally it is possible to get the reaction in the base due the motion of the system.

The formulation for the case of revolution joints is:

1. Forward step:

   - Angular Velocity. The angular velocity of the frame k is:

     $$\omega(t)_k = \omega(t)_{k-1} + \dot{q}(t)_k \; Z_{k-1} \tag{3.24}$$

     where ω_{k-1} is the vector of the angular velocity of the previous frame, $\dot{q}(k)$ is the joint velocity and Z_{k-1} is the revolution axis.

   - Angular Acceleration. The angular acceleration of the frame k is:

     $$\dot{\omega}(t)_k = \dot{\omega}(t)_{k-1} + \dot{q}(t)_k^2 \; Z_{k-1} + \omega(t)_{k-1} \times (\dot{q}(t)_k \; Z_{k-1}) \tag{3.25}$$

     where $\dot{\omega}(t)_k$ is the vector of the angular acceleration of the frame k and $\dot{q}(t)_k^2$ is the joint acceleration.

   - Linear Velocity. The linear velocity of the frame k is:

     $$v(t)_k = v(t)_{k-1} + (\omega(t)_k \times \triangle s_k) \tag{3.26}$$

     where $v(t)_{k-1}$ is the linear velocity of the of the frame $k-1$. The $\triangle s_k = d_k - d_{k-1}$ is the vector difference between the position of the frame $k-1$ and the current frame k. Note that to calculate linear velocity it is necessary to have the angular velocity of the frame, so the order is mandatory.

   - Linear Acceleration. The linear acceleration of the frame k is:

     $$\dot{v}(t)_k = \dot{v}(t)_{k-1} + \dot{\omega}(t)_k \times \triangle s_k + \omega(t)_k \times (\omega(t)_k \times \triangle s_k) \tag{3.27}$$

     where $\omega(t)_k \times (\omega(t)_k \times \triangle s_k)$ is the centrifugal acceleration. Again note that to calculate linear acceleration it is necessary to have the angular acceleration.

Finally, to start the iteration, initial conditions have to be set:

$\omega(0)_0$: Initial angular velocity of the base.

$\dot{\omega}(0)_0$: Initial angular acceleration of the base.

$v(0)_0$: Initial linear velocity of the base.

$\dot{v}(0)_0$: Initial linear acceleration.

2. Backward step:

   - Forces: Let's define a vector from the end of a link to its CoG $\triangle r_k = \overline{c_k} - d_k$ where $\overline{c_k}$ is the location of the center of mass of link k.

     The resulting force applied in the CoG is:

$$f(t)_k = f(t)_{k+1} + m_k[\dot{v}(t)_k + \dot{\omega}(t)_k \times \triangle r_k + \omega(t)_k \times (\omega(t)_k \times \triangle r_k)] \quad (3.28)$$

   - Torques: First it should be calculated the moment of the frame k.

$$n(t)_k = n(t)_{k-1} + (\triangle s_k + \triangle r_k) \times f(t)_k - \triangle r_k \times f(t)_{k+1} + D_k\ \dot{\omega}(t)_k + \omega(t)_k \times (D_k\ \omega(t)_k) \quad (3.29)$$

     where D_k is the inertial tensor of link k in base space.

     Once the moment it is solved, the torque it is computed as:

$$\tau(t)_k = n(t)_k^T\ Z_{k-1} + b_k\ \dot{q}(t)_k \quad (3.30)$$

     where Z_{k-1} is the axis of the revolution and b_k is the viscous friction coefficient.

Once the backward step is finished the forces and torques of each joint are obtained and the RNE algorithm is finished.

The reaction force and reaction torque have been defined as follows:

Reaction force of the arm: The reaction force in the base is the force computed in the joint 1. $f_{ar} = f(t)_1$.

Reaction torque of the arm: The reaction torque in the base is the torque computed in the joint 1. $\tau_{ar} = \tau(t)_1$.

These reactions has been coupled in the section (3.1.2).

3.2.2 Model to simulate the Real Robotic Arm

In order to implement the RNE algorithm using the DH parameters the Robotics toolbox from Corke [2011] has been used. This tool allow to use some methods to calculate the RNE algorithm or the acceleration of each joint given the current angles of the joints, their velocities and the torques desired. Also, it is possible to calculate the torque necessary to compensate the gravity effect.

To calculate the dynamic reactions of the base for each sampled time, it is necessary to have the angles, the velocities and the accelerations of each joint.

For the simulation model, a discretized method has been implemented. The method applied in this thesis to obtain the dynamic reactions is:

1. To calculate the Coriolis velocity using the current angles and velocities of the joints.
2. To estimate the necessary torque to be applied to each joint to compensate the effect of the gravity for the current position of the joints.
3. To compute the necessary torque to compensate the centrifugal force due the Coriolis effect.
4. To solve the direct kinematics using the current angles, velocities and the torques desired for each joint. This torques contemplate the enough torque to compensate the gravity and Coriolis effect, and also to achieve the motion planned. After solve it, the accelerations for each joint are obtained.
5. To update the joint velocity using the acceleration and the sampling time.
6. To update the joint angle using the velocity and the sampling time.
7. Finally, using the updated angles, velocities and accelerations, the RNE algorithm is applied. After this step, the base forces and torques reactions are obtained.

This algorithm is repeated for each sampled time. Due the complexity of the operations, it is an expensive method that spends a lot of CPU time.

Once it is finished, the result is a vector with three forces and three moments (one per axis) represented in its base frame.

This information will be used during the control loop as a perturbation of the system.

3.2.3 Physical and Geometrical Parameters of the Arm Model

There are some geometrical and physical parameters that define the dynamic model of the system.

The robotic arm used is designed by CSIC-UPC on collaboration with european project Arcas (Sanramaria and Andrade [2014]).

This arm is a prototype and its final design will be different from the model presented in this section.

However, the algorithm developed during this thesis are general and can be applied to any robotic arm. So, even if the arm changes, only by changing the physical and geometrical parameters the method explained in this study will work properly.

The system designed by CSIC-UPC has 6 joints set in different ways in order to reach to any orientation as it is represented in the figure (3.8).

The reference frame follows the same convention as the body frame of the quadrotor:

1. x $axis$: Aligned with the forward of the quadrotor.
2. y $axis$: Pointing to the floor (same direction as the gravity).
3. z $axis$: Completes the reference frame with the right-hand rule.

The first joint is to rotate the arm along $z - axis$ and move all the other articulations to any location of the xy plane . The next two joints move the rest of the chain in the xz plane. Finally, the last three joints are to orientate the end-effector independently of the other articulations.

During the simulation, the motion of each joint it is managed with a PID controller. The end-effector has a load of 0.05 kg to simulate that it is carrying an object.

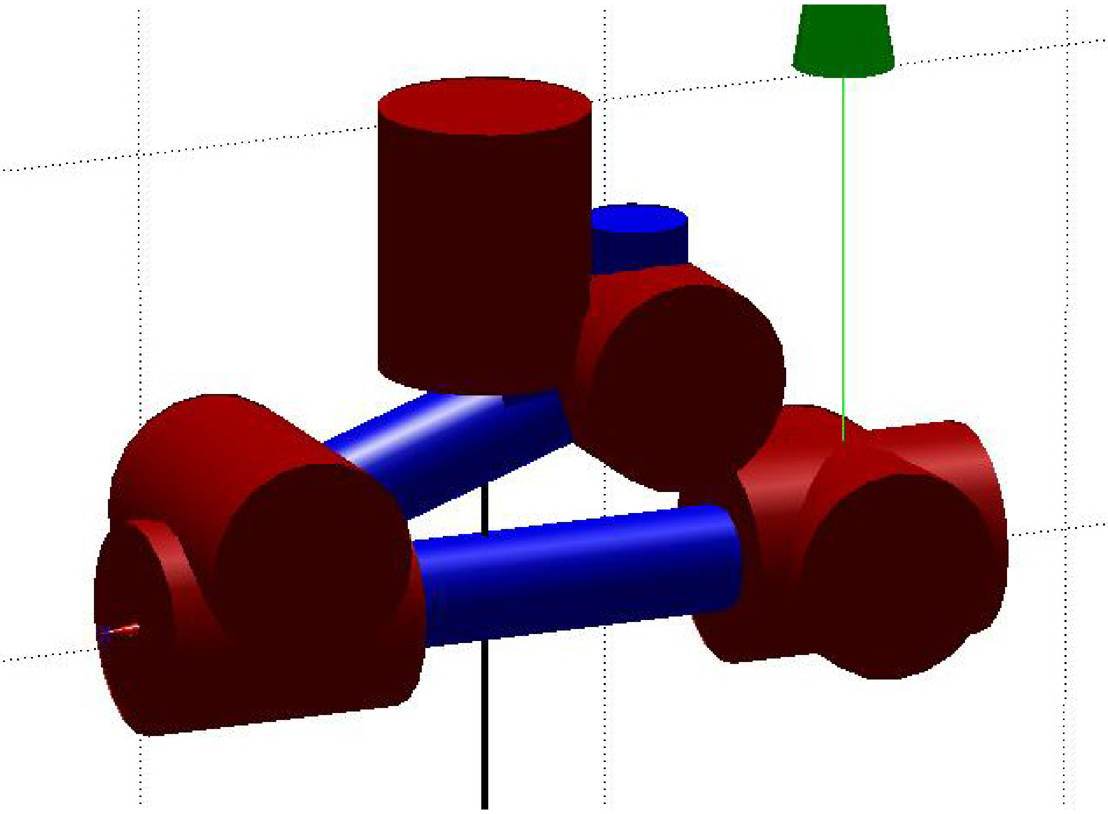

FIGURE 3.8: Matlab model of the robotic arm designed by CSIC-UPC.

To compute the Recursive Newton Euler algorithm it is necessary to know Denavit–Hartenberg parameters and the physical magnitudes of the system. All these values are taken from the original design of the arm done by CSIC-UPC.

The next table (3.1) has all the DH parameters of the robotic arm used during the simulations.

TABLE 3.1: DH parameters of the robotic arm

Link i	a_i	d_i	α_i	θ_i
1	0.030	-0.026	$\pi/2$	q_1
2	0.079	0	0	$q_2 + \pi + 0.325$
3	0.0158	0	$\pi/2$	$q_3 + \pi - 0.325$
4	0	-0.118	π	q_4
5	0	0	$-\pi/2$	q_5
6	0	0	0	$q_6 + \pi/2$

3.3 Nonlinear Model Predictive Control problem

3.3.1 Dynamic model, state variables and control signals

The dynamic model of the whole system includes the quadrotor and the effect of the motion of the robotic arm.

This thesis develops a method to couple the effect of the arm with the quadrotor dynamics. First of all it is important to specify the state variables chosen for this study.

The sign of these variables determine if the propellers are rotating clockwise or counter-clockwise. This sign is important to compute the torque generated by its rotation.

On the other hand, the arm is considered as a perturbation of the system, so the NMPC has not the task of control its trajectory. To add this perturbation it is necessary to calculate the RNE algorithm for each sampled time before to apply the NMPC.

In order to simulate the trajectory of the arm, it has been necessary to build a PID for each joint. These controller is enough to move a joint from an angle to another one in a smoothly way.

Once all the reactions are calculated, it is possible to simulate the behavior of the whole system.

To couple both dynamics, the discretized equations of the model of the quadrotor has to be modified:

$$m \cdot \dot{V} = -m \cdot \Omega \times V + m \cdot g \cdot R^T \cdot E_3^a + \sum_1^4 T_i - F_{ar} \tag{3.31}$$

$$I \cdot \dot{\Omega} = -\Omega \times I \cdot \Omega + \sum_1^4 [Q_i + M_i] - \tau_{ar} \tag{3.32}$$

with:

F_{ar}: Reaction forces of the arm.

τ_{ar}: Reaction torques of the arm.

Note the sign of these reactions. The negative sign is because the Peter Corke Toolbox gave us the reaction of the virtual anchor of the arm base, but the for the dynamics, it is necessary the opposite.

As it is described above, adding just a new parameter, it is possible to couple the perturbation of the arm in the dynamic behavior of the quadrotor. For each sampled time, the value of the F_{ar} and a_r may change in function of the precalculated arm motion.

Once the reactions are calculated and the equations are modified, it is possible to start the NMPC loop to optimize the control of the system.

3.3.2 Horizon and Sampling Time

Choose a properly control horizon and sampling time is important to control the plant.

The horizon of the control is the number of steps that the NMPC evolves the system toward the future to find the optimal control taking into account the future predictions. As the horizon increases, the complexity to solve the problem augments, and the CPU time required increases dramatically. So a constraint to limit the number of the steps is the power of CPU. On the other hand, a not enough control horizon, the dynamic of the system can not be predicted and then the control could be impossible.

The sampling time is the amount of time that the time advances for each step of the control horizon. If is too large, the dynamics of the system will occur faster than the NMPC could control. On the other hand, if the sampling time is too short, the future predicted is near of the current time and the inertias of the motion are not possible to be predicted.

So it is important to balance both parameters in order to have a far horizon of control but with a short sampling time to detect the dynamics of the plant and also taking into account the power of the CPU required.

After some experimentation and evaluating several parameters of the horizon control and sampling time, a good balance has been found.

Depending of the control mode, the parameters could vary. These variations are because the dynamics effects could affect more or less according of the motion of the quadrotor and then it is necessary a larger or shorter horizon control to stabilize the plant.

In the Chapter 3, more detailed explanation has been presented.

3.3.3 Objective function

The objective function, also called value function, is the expression that the NMPC has to minimize by looking the best control signals possibles.

This function is the global target of the quadrotor and describes how far is from its goal. The goal could be dynamic or a trajectory of the state variables. On the other hand, this equation does not show the performance to accomplish the minimization.

During this thesis, two different functions has been developed depending if the quadrotor has to stand still or if it has to move to some position.

The value function has subgoals for each subtarget that it has to accomplish. Each sub function has its weight in order to prioritize which ones are more important others. But to use a weight system, first is important to normalize them. To do it, this project has used the range of the constraint for each variable involved in each subfunction:

$$J = \sum_{1}^{p} \gamma_i \cdot J_i \tag{3.33}$$

with:

J: Total cost function.

γ_i: Weight of the subfunction $i - th$

J_i: Subfunction $i - th$

In the particular case of this thesis, there are up to 4 subfunctions:

$$J = \sum_{1}^{4} \gamma_i \cdot \frac{(x_i - x_{id})^2}{(x_{iMax} - x_{iMin})} \tag{3.34}$$

$x_i d$: Desired value of x_i.

x_iMax: Maximum possible value of the state variable x_i.

x_iMin: Minimum possible value of the state variable x_i.

The state variables involved in the value function are $(x_1, x_2, x_3, x_4) = (x, y, z, \psi)$ corresponding of quadrotor position and yaw.

On the other hand, pitch and roll are not included because if the system is forced to have a determined value of these states, directly implies that the quadrotor will move along its x or y axis.

As it was explained before, a tilt on its x or y axis generates a change of the direction of the thrust vector, and a component of force could appear in the horizontal plane. So it is important to let free the pitch and roll in order to allow the proper movement of the quadrotor.

When the system moves along an horizontal axis, it is mandatory to increase the difference of the rotor speed between two opposite rotors. But doing this, the total moment of the system is not compensated and starts to rotate in the z axis.

So it is necessary to fix the desired yaw angle to ensure that the plant orientation is constant. If this restriction is not imposed, it is possible to move from one position to another but losing the orientation due the change of rotation of the rotor when the quadrotor is trying to move.

On the other hand, nor linear nor angular velocity have been included due that it is impossible to hover a quadrotor if the system is not allowed to change its velocity conveniently.

The NMPC uses this equation in order to find the optima control signals to minimize the function value, which means accomplish all the goals.

3.3.4 Constraints

The constraints define the state space of the problem. It is a polyhedric space where the NMPC has to find the optimal solution.

As bigger is this space, larger it is the search. So it is important to limit each variable with an upper and lower value. Also, if there are any linear or nonlinear relation between variables it should be implemented in the problem.

The constraints help to find the optimal and also limit all possibles solutions. Also, those restrictions allow to define the performance desired by the system.

While the cost function to minimize express the goal to reach, the constraints tell to NMPC how to accomplish its target.

During this thesis, the constraints have been categorized in two types:

1. State Variables Constraints: All the constraints related with the state variables are defined in this category. The upper and lower values have to be defined for each of the 12 state variables of the plant.

   - Position Bounds: Defines the volume allowed to move the quadrotor.

     $$(x_{1Upper}, x_{2Upper}, x_{3Upper}) = (x_{upper}, y_{upper}, z_{upper})$$

     $$(x_{1Lower}, x_{2Lower}, x_{3Lower}) = (x_{lower}, y_{lower}, z_{lower})$$

     The trajectory of the quadrotor have to stay inside this volume in order to find a solution of the problem. Also, this volume has to be at least enough big to contain the possible error obtained during the hover maneuver.

   - Euler Angles Bounds: Defines the maximum and minimum euler angles allowed.

     $$(x_{4Upper}, x_{5Upper}, x_{6Upper}) = (\psi_{upper}, \theta_{upper}, \varphi_{upper})$$

     $$(x_{4Lower}, x_{5Lower}, x_{6Lower}) = (\psi_{lower}, \theta_{lower}, \varphi_{lower})$$

     Taking into account that the $x_4 = \psi$ is the yaw angle, the range of this variable have to be enough wide to contain all the possible orientations in the xy plane necessary to accomplish the targets of the quadrotor.

     On the other hand, the range of pitch ($x_5 = 0$) and roll ($x_6 = 0$) angles have to be enough small to permit a slight tilt that allows to move the quadrotor but without losing the control of the plant. For large values of pitch or roll, the system can rotate completely until the thrust points to the same direction as the gravity, which would be critical for the plant.

To avoid this, a small range has been chosen.

- Linear Velocity Bounds: Defines the maximum and minimum linear velocity allowed. These are referenced to the inertial frame.

$$(x_{7Upper}, x_{8Upper}, x_{9Upper}) = (\dot{x}_{upper}, \dot{y}_{upper}, \dot{z}_{upper})$$

$$(x_{7Lower}, x_{8Lower}, x_{9Lower}) = (\dot{x}_{lower}, \dot{y}_{lower}, \dot{z}_{lower})$$

Those constraints allow to smooth the behaviour of the quadrotor. Due the limit of its speed, the effect of its dynamics are less aggressive than for high speeds.

- Angular Velocity Bounds: Defines the range of the angular velocity of the quadrotor.

$$(x_{10Upper}, x_{11Upper}, x_{12Upper}) = (\Omega_{xUpper}, \Omega yUpper, \Omega zUpper)$$

$$(x_{10Lower}, x_{11Lower}, x_{12Lower}) = (\Omega_{xLower}, \Omega_{yLower}, \Omega_{zLower})$$

As before, it is important to define a narrow range of values in order to do not lose the control of the motion and avoid the possible problems with the dynamics effects.

2. Control signal Constraints: The restrictions of the control signal usually corresponds to the physics capabilities of the actuators. In this thesis the control signals are the angular rate of the rotors.

   So depending of the kind of motor used, its limits and its behavior could change.

   The basics aspects that it should be limited are:

   - Angular Rates of the Rotors: Defines the range of the angular velocity of the each rotor. The upper bounds correspond the maximum possible angular rate that the motor is able to turn.

These constraints take into account the sign of the value because the velocity could be positive or negative depending if it turns clockwise or counter-clockwise.

$$u_{1Upper}(k) = \omega_{1Upper} > 0$$

$$u_{2Upper}(k) = \omega_{2Upper} < 0$$

$$u_{3Upper}(k) = \omega_{3Upper} > 0$$

$$u_{4Upper}(k) = \omega_{4Upper} < 0$$

On the other hand, the lower bounds are the minimum values experimentally found to avoid lose the control of the system. Under these limits the quadrotor may fall down. So actually is not the minimum angular speed of the rotors.

$$u_{1Lower}(k) = \omega_{1Lower} > 0$$

$$u_{2Lower}(k) = \omega_{2Lower} < 0$$

$$u_{3Lower}(k) = \omega_{3Lower} > 0$$

$$u_{4Lower}(k) = \omega_{4Lower} < 0$$

So:

$$\omega_{1Lower} < u_1(k) < \omega_{1Upper}$$

$$\omega_{2Lower} < u_2(k) < \omega_{2Upper}$$

$$\omega_{3Lower} < u_3(k) < \omega_{3Upper}$$

$$\omega_{4Lower} < u_4(k) < \omega_{4Upper}$$

- Angular Accelerations of the Rotors: Defines the range of the angular acceleration for each rotor. It is important to define the maximum possible acceleration because this constraint limits how fast could change the angular speed of the rotors and then the dynamics of the quadrotor.

  For wide ranges the maneuvers are faster but also more aggressive so the system could become uncontrolled.

On the other hand, for a narrow ranges, change the state variables of the plant could be too slow and then it may be difficult to accomplish the goals.

The maximum width of the ranges are determined by the physical properties of the rotor.

To implement this restriction it is necessary to know the last angular rate of the rotor and compute the current increment:

$$\triangle u_i(k) = u_i(k) - u_i(k-1) < max\triangle \rightarrow u_i(k) < max\triangle + u_i(k-1)$$

$$-\triangle u_i(k) = -u_i(k) + u_i(k-1) < max\triangle \rightarrow -u_i(k) < max\triangle - u_i(k-1)$$

for i $\in$ [1,4].

Once all those constraints are applied, the states space of the system is defined and the optimizer can find the solution in the control space.

Chapter 4

Simulation Results

During this section, the results obtained are going to be presented.

The problem to solve consists to use an NMPC to control the dynamics of a quadrotor that is perturbed by external forces and torques.

Those perturbations are represented by the movement of a robotic arm.

In order to proceed to the simulations, some assumptions have been done:

- Enough computational resources: Let's suppose that the code and the processor are enough to compute all this calculations in real time.
- Accurate estimator of the state variables: Let's assume that the are a good filter that return to us a good estimation of the 12 state variables. During these tests, this filter is emulated by using a mathematical model of the quadrotor.
- Point of application: The force/torque point of application is in the origin of the body frame.
- Control the angular rate of the rotors: Let's suppose that is possible to control the speed of the rotors and their dynamics are instant.
- There are no wind: Let's assume a controlled environment.

- The sampling time is constant: The clock of the computer is constant and the algorithm is solved without any delay.
- Robotic arm motion: The angular joint position are going to have small increments.
- Empty end-effector. No tools attached to the end-effector.

To test the algorithm developed during the thesis, five different scenarios have been prepared:

1. Taking off while carrying a robotic arm: The quadrotor take off and flies until reach to a specific altitude. The arm stays in a safe position.
2. The quadrotor in hover mode and the robotic arm in a safe position: The aim of this test is to try if the system is able to keep a fixed position while its center of mass has changed.
3. The quadrotor in hover mode and the robotic arm moving toward a target position: This scenario is a demonstration about how the NMPC compensate the dynamics of the arm due its motion.
4. The quadrotor moving toward location and the robotic arm in a safe position: The target is show how the quadrotor can move to a fixed position while is carrying an object that changes its center of mass.
5. The quadrotor moving toward a location and the robotic arm moving toward a target position: This environment has been prepared to test the MPC against the dynamics of the quadrotor perturbed by the robotic arm.

During the next sections, a comparison between those modes are going to be presented. In each part, some statistics have been compared by changing some of the parameters of the NMPC to show how they affect to the system. To obtain the perturbation is necessary to simulate a motion of the arm. For these tests, the same arm motion are going to be applied in order to understand better the different behavior of the whole system . The motion of the arm could be or stay in a safe position, or move to a target pose. Both cases has different reactions. This plot shows the reaction when the arm robotic stays in a safe position. The forces and torques are constant during all the simulation. Actually, the only remarcable force that is acting to the system is the force in the z axis which corresponds to the force of the gravity. Also, there is a slight moment in the y axis.

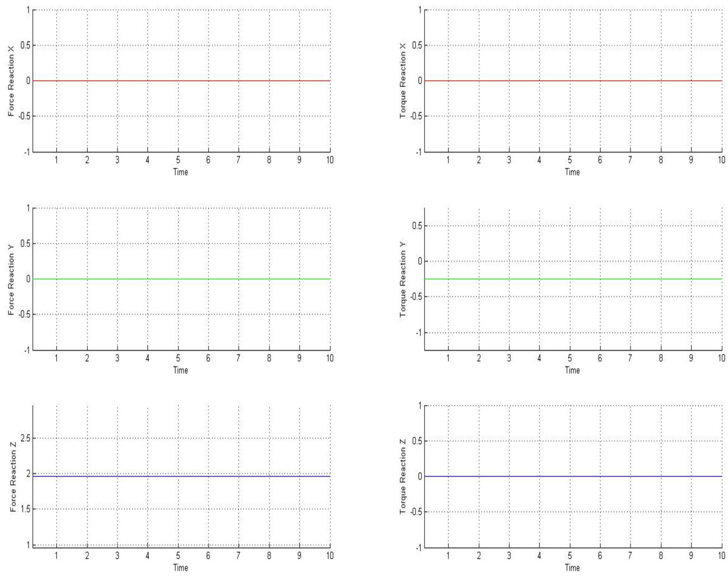

FIGURE 4.1: Dynamic Reaction in the base of the arm due its no motion

On the other hand, the motion chosen have been small step for each of the joints.

The graphic shows the forces that the quadrotor will receive when the robotic arm is moving. Those dynamics have to be compensate by the control algorithm in order to stabilize the system.

Depending on the scenario, the perturbations will change. It is possible to observe how when the angle reference change, appear forces and torques reactions until the position is reached. Once the arm it is stable, the dynamic reactions disappear.

4.1 Taking off

The test consists on take off the quadrotor until reach a safe position in the air. The robotic arm stays in a safe position during all this maneuver.

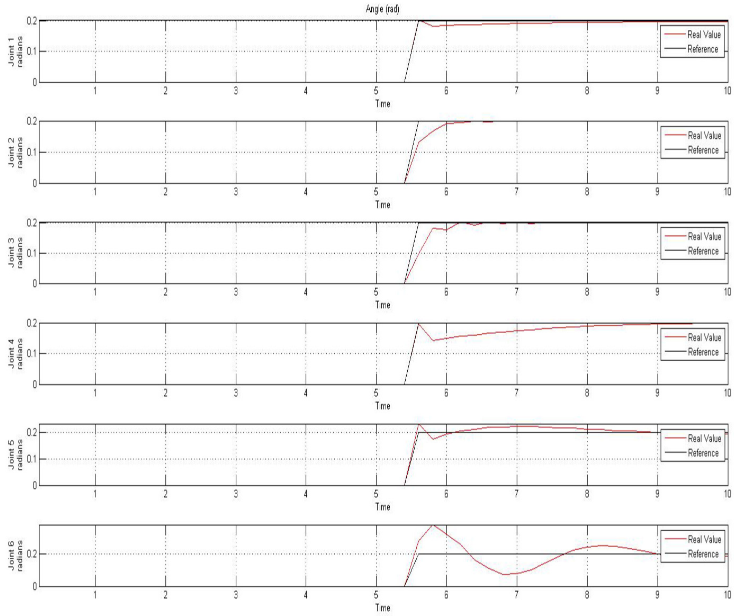

FIGURE 4.2: Reference and Angle of each joint

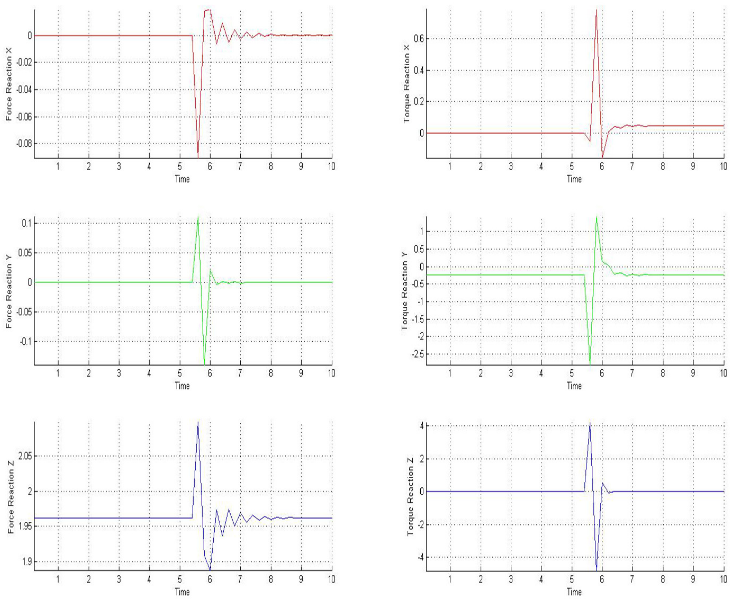

FIGURE 4.3: Dynamic Reaction in the base of the arm due its motion

The whole system starts into the ground. The NMPC has to find the right combination of the inputs to take off.

In this case, the cost function is:

$if x_3 >= 0$

$$J = 10 \cdot \frac{(x_1 - 0)^2}{100} + 10 \cdot \frac{(x_2 - 0)^2}{100} + 10 \cdot \frac{(x_4 - 0)^2}{4} + 10 \cdot \frac{(x_9 + 1)^2}{100}$$

$else$

$$J = 10 \cdot \frac{(x_1 - 0)^2}{100} + 10 \cdot \frac{(x_2 - 0)^2}{100} + 2.5 \cdot \frac{(x_3 + 4)^2}{100} + 10 \cdot \frac{(x_4 - 0)^2}{4}$$

end with: $(x_1, x_2, x_3, x_4, x_9) = (x, y, z, \varphi, \dot{z})$

By using this strategy, the NMPC is forced to try push the quadrotor until obtain a velocity in the z axis.

Once the quadrotor starts to fly $(x_3 < 0)$ will mean that has enough thrust to reach to the altitude desired, so the cost function changes to find the right location in the space.

The prioritization of the cost function is first of all, maintain the yaw orientation then fix the position in the plane xy and finally move to a z altitude or move along the z axis.

Parameters	Value
Initial State Variables	$[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$
Desired State Variables	$[0, 0, -4, 0, 0, 0, 0, 0, 0, 0, 0, 0]$
Initial Control Signal	$[200, -200, 200, -200]$
Horizon Control	10
Sampling Time	$0.2s$
Simulation Time	$10s$

TABLE 4.1: Initial Conditions for Take Off

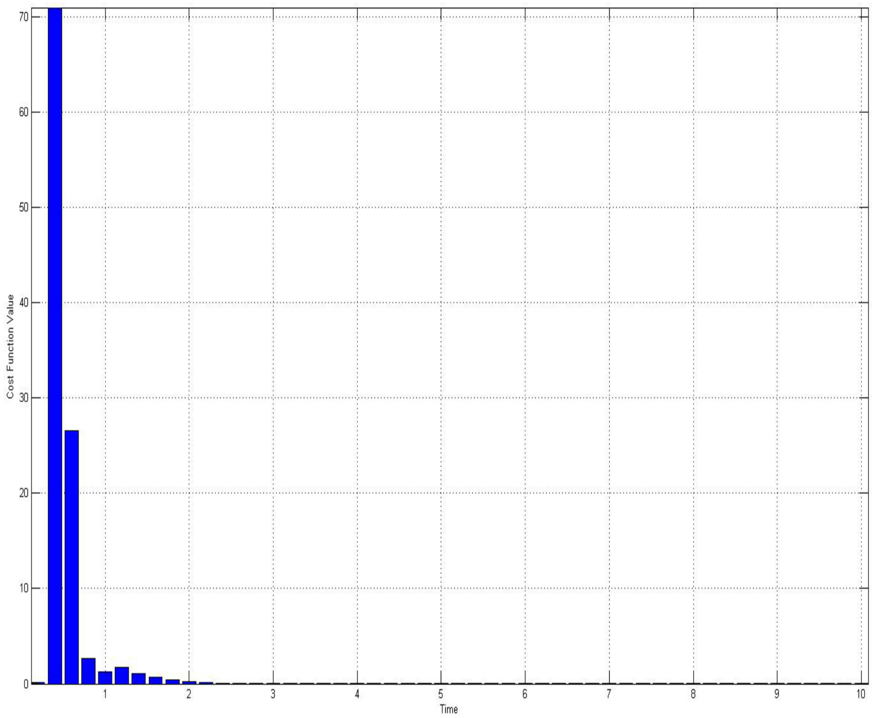

FIGURE 4.4: Cost function during the Take Off

As it is possible to see above, there is a peak. Before this points, is the step where the controller tries to push the quadrotor to move at $1m/s$ to up. After this point corresponds when the cost function has changed its branch, and the NMPC tried to reach to 4 m from the ground.

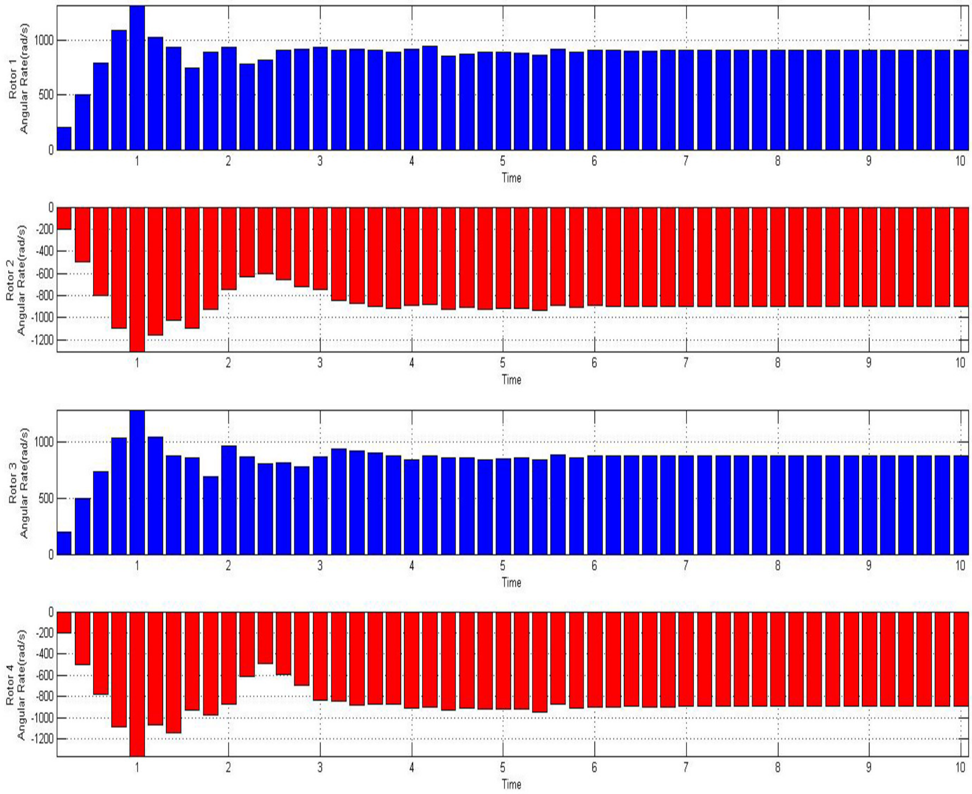

FIGURE 4.5: Control signal for a Take Off

At the beginning, the angular rate of the rotors increase its value in order to accelerate the quadrotor. Once it is flying, the control signal is looking for the way to stabilize the system. On the other hand, it is possible to notice the change of the behavior in the control signal when the quadrotor reach to the desired position.

Finally, the performance of the motion could be observed in the following graph.

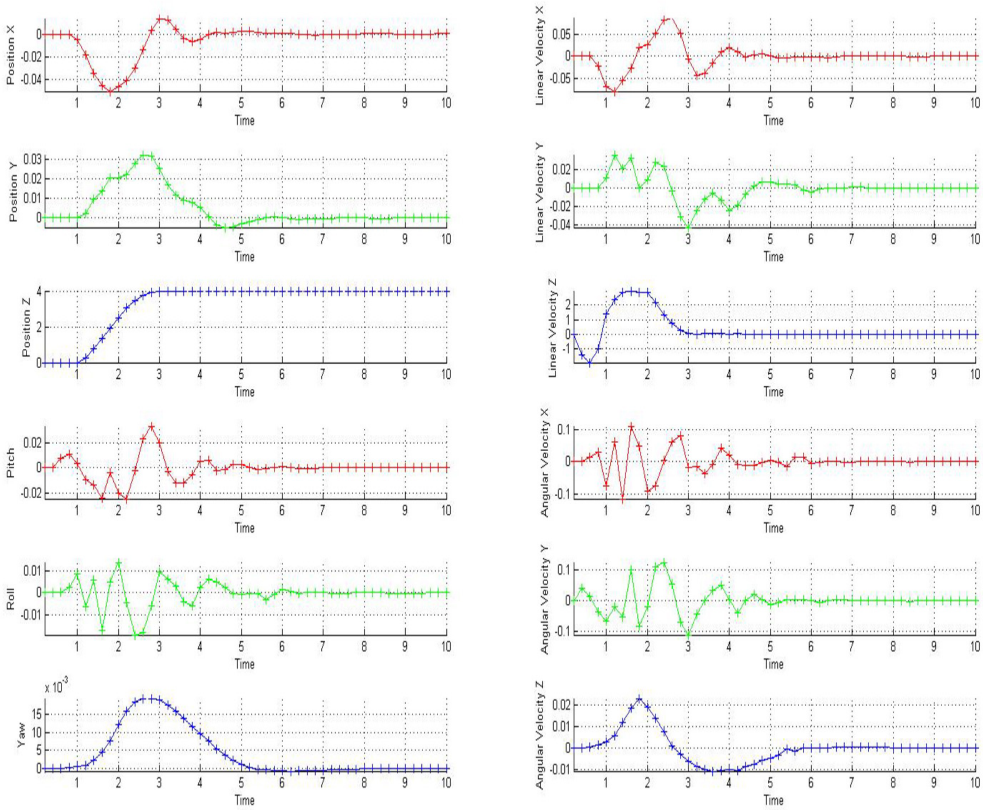

Figure 4.6: State variables for the Take off

The system, after reach to the enough velocity to take off, starts increasing its altitude until the $4m$. At the same time is compensating the movement in the xy plane and the tentative to change its yaw.

Parameters	Value
ISE	6.2
Maximum error	2.5
Time To Get Stable	$3.2s$

Table 4.2: Summary of Take Off

4.2 Static Control

The aim of this section is to try to stabilize the quadrotor in a fixed position in the space.

$$J = 10 \cdot \frac{(x_1 - 0)^2}{100} + 10 \cdot \frac{(x_2 - 0)^2}{100} + 10 \cdot \frac{(x_3 + 4)^2}{100} + 10 \cdot \frac{(x_4 - 0)^2}{4} \tag{4.1}$$

This cost function assure that the quadrotor will stay near the position $(0, 0, 4)$ and with a fixed orientation.

At the beginning, the rotors start with an initial angular speed $(900, -900, 900, -900)$. Those rates are the necessary to avoid the fall of the system.

Parameters	Value
Initial State Variables	$[0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0]$
Desired State Variables	$[0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0]$
Initial Control Signal	$[900, -900, 900, -900]$
Horizon Control	10
Sampling Time	$0.2s$
Simulation Time	$10s$

Table 4.3: Initial Conditions for Static Control

4.2.1 Static arm

This is the simplest test where the robotic arm stays in a safe position and the quadrotor has to compensate its perturbations.

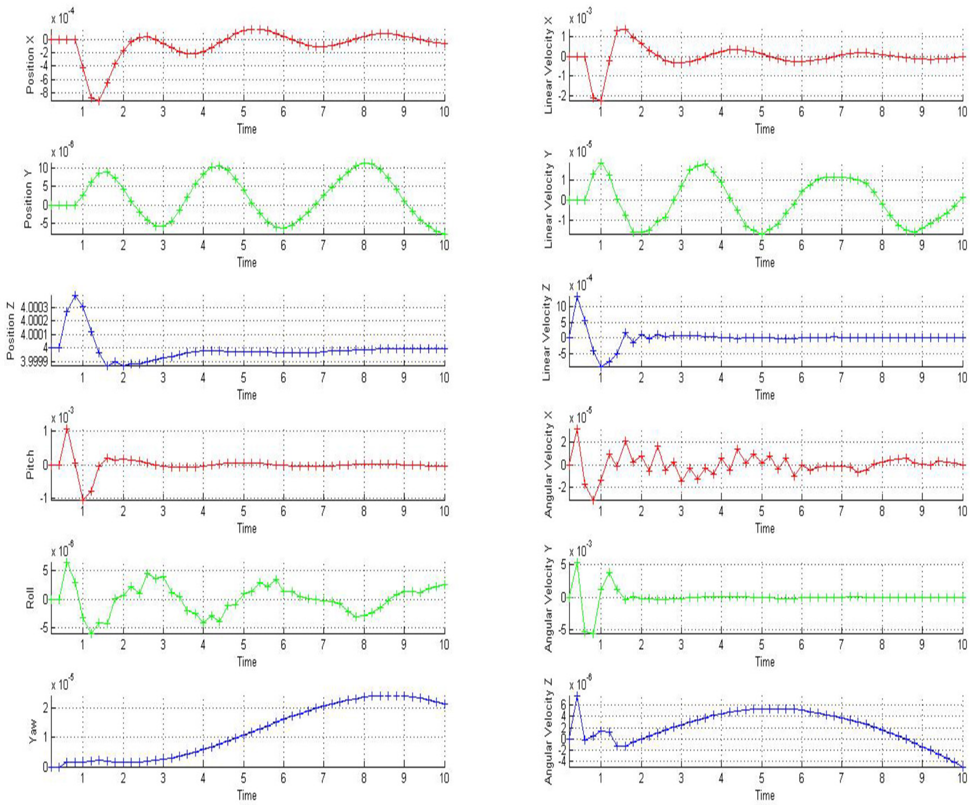

FIGURE 4.7: State variables a static control with a static arm

As it is possible to see, the NMPC could stabilize the quadrotor and bring it to the desired state variables.

In order to compensate the moment generated by the arm, the four rotors has different speeds $(911.2652, -894.7493, 877.9722, -894.7513)$. Notice that the forward and backward rotors are unbalanced while the left and right are rotating at the same rate.

All the values are enough small to consider them as noise. It is clear to see how the velocity in the y axis is trying to correct the deviation of the system in this axis.

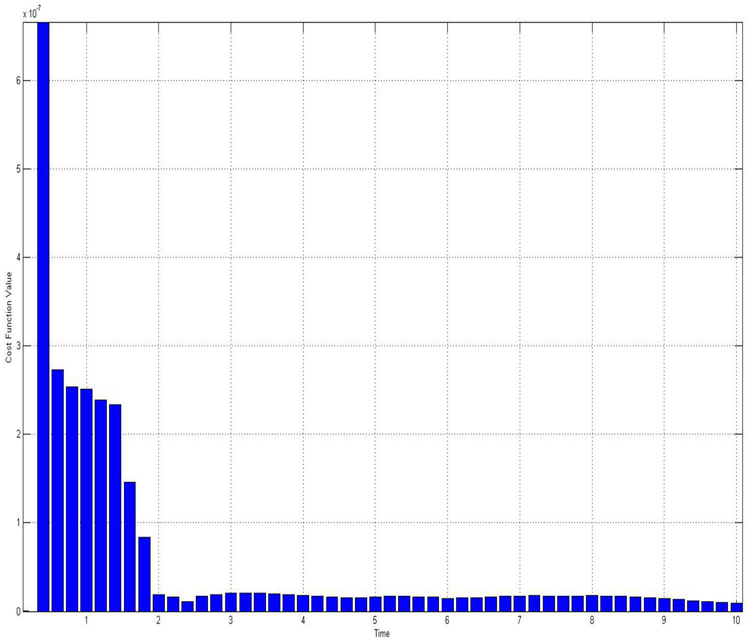

FIGURE 4.8: Cost function during the static control with a static arm

Approximately at 2 seconds, the system finds the correct combination of the rotor to reach at its stable position.The first peak is because the forces are unbalanced and the quadrotor moves slightly to up. Once the quadrotor is completely stable, it remains in a fixed position until the end of the simulation.

Parameters	Value
ISE	$2.75e^{-6}$
Maximum error	$6.75e^{-7}$
Time To Get Stable	$2.0s$

TABLE 4.4: Summary of a static control with a static arm

4.2.2 Dynamic arm

During this test, the perturbation of the arm robotic due its motion it has been applied to show its impact in the whole system.

4.3 Dynamic Control

The aim of this section is to try to carry the whole system to a position in the space.

$$J = 10 \cdot \frac{(x_1 - 0.75)^2}{100} + 10 \cdot \frac{(x_2 - 0.75)^2}{100} + 2.5 \cdot \frac{(x_3 + 4)^2}{100} + 10 \cdot \frac{(x_4 - 0)^2}{4} \tag{4.2}$$

Again the prioritization of the cost function is control yaw orientation then xy position and finally move to a z altitude.

The cost function has the target of move the quadrotor from the $(0, 0, 4)$ to $(1, 1, 4)$ and stay stable in this position.

Again the rotors start with an initial angular speed $(900, -900, 900, -900)$.

Parameters	Value
Initial State Variables	$[0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0]$
Desired State Variables	$[0.75, 0.75, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0]$
Initial Control Signal	$[900, -900, 900, -900]$
Horizon Control	10
Sampling Time	$0.2s$
Simulation Time	$10s$

TABLE 4.5: Initial Conditions for dynamic Control

4.3.1 Static arm

In this scenario, the quadrotors move to a position while the robotic arm stays immobile.

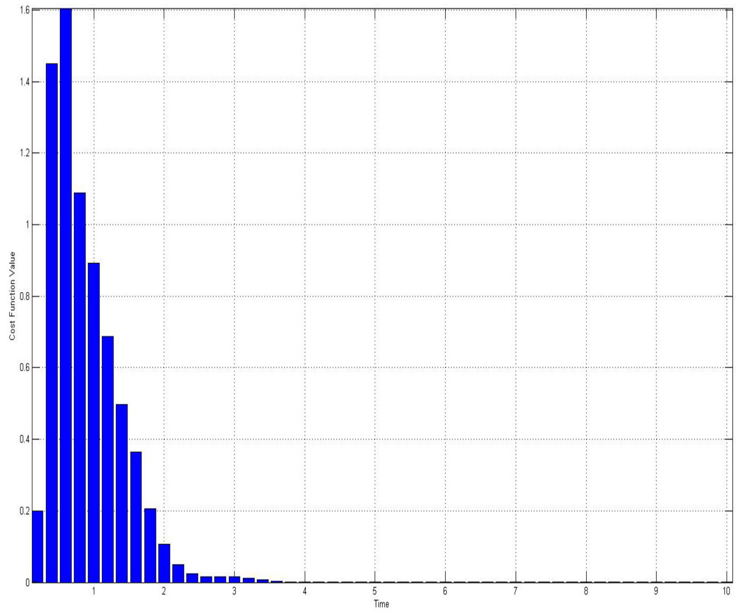

FIGURE 4.9: Cost function during the dynamic control with a static arm

The peak shown above corresponds the maximum error due that the system change its Z position, even if has the lowest weight in the function cost. The change of its altitude is because two different factors:

1. The system does not start from a stable position. The start values of the rotors speeds are not the values that maintain the quadrotor in a fixed position. So the NMPC search the right approach to stabilize the system around the altitude desired.

2. The system has as a priority maintain the yaw orientation all the time.

Then the NMPC tries to keep its orientation and move toward a position but using the prioritization explained before, which means that the worst performance will be in the z altitude control.

The quadrotor is approaching to its target at each instant of time so the cost of the function is decreasing.

The following figure shows the performance of the system. The target position x and y is reached with a slight peak. Also, it is possible to see how the effect of the movement implies a modification of the z position as explained above.

On the other hand, is possible to observe that to move in the xy plane, a tilt in roll and pitch is necessary.

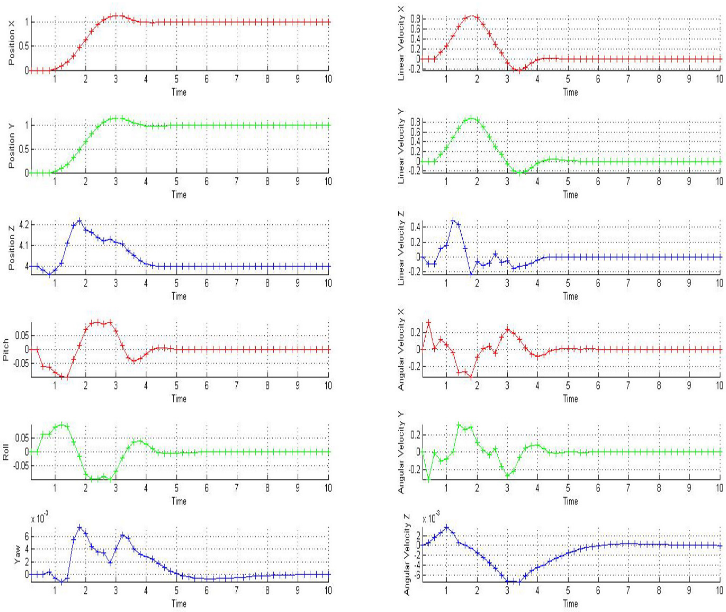

FIGURE 4.10: State variables during the dynamic control with a static arm

Parameters	Value
ISE	7.25
Maximum error	1.6
Time To Get Stable	$3.75s$

TABLE 4.6: Summary of a dynamic control with a static arm

4.3.2 Dynamic arm

Finally, the last case is to analyse how the perturbations from the robotic arm affects to the quadrotor while it is moving.

Bibliography

Alexis, K., Nikolakopoulos, G., and Tzes, A. (2010). Experimental model predictive attitude tracking control of a quadrotor helicopter subject to wind-gusts. *18th Mediterranean Conf. on Control Automation (MED)*, page 1461–1466.

Allen, P. K., Timcenko, A., and Yoshimi, B.and Michelman, P. (1993). Automated tracking and grasping of a moving object with a robotic hand-eye system. *Robotics and Automation, IEEE Transactions*, 9:152–165.

Bangura, M. and Mahony, R. Nonlinear dynamic modeling for high performance control of a quadrotor.

Bangura, M. and Mahony, R. (2012). Nonlinear dynamic modeling for high performance control of a quadrotor. *Australasian Conference on Robotics and Automation*, pages 1–10.

Belkheiri, M., Rabhi, A. Hajjaji, A. E., and Pegard, C. (2012). Different linearization control techniques for a quadrotor system. *In 2nd Int. Conf. on Communications, Computing and Control Applications (CCCA)*, pages 1–6.

Benallegue, A., Mokhtari, A., and Fridman, L. (2006). Feedback linearization and high order sliding mode observer for a quadrotor uav. *Workshop on Variable Structure Systems*, pages 365–372.

Bicchi, A. and Tonietti, G. (2004). Fast and" soft-arm" tactics [robot arm design]. *Robotics and Automation Magazine*, 11:22–33.

Bouabdallah, S. (2007). Design and control of quadrotors with application to autonomous flying. Master's thesis.

Bouabdallah, S., Pierpaolo, M., and Siegwart, R. (2004). Design and control of an indoor micro quadrotor. *IEEE International Conference*, 5:4398.

Bouffard, P. On-board model predictive control of a quadrotor helicopter: Design, implementation, and experiments. Master's thesis.

Bresciani, T. Modelling, identification and control of a quadrotor helicopter. Master's thesis.

Corke, P. (2011). *Robotics, vision and control: fundamental algorithms in MATLAB*. Springer.

Featherstone, R. and Orin, D. (2000). Robot dynamics: Equations and algorithms. In *ICRA*, pages 826–834.

Fukuda, T. (1985). Flexibility control of elastic robotic arms. *Journal of Robotic Systems (ISSN 0741-2223)*, 2:73–88.

Grancharova, A., Grotli, E., and Johansen, T. (2012). Distributed mpc-based path planning for uavs under radio communication path loss constraints. Master's thesis.

Gruene, L. and Pannek, J. (2011). *Nonlinear Model Predictive Control. Theory and Algorithms*. Springer.

Hayati, S. (1986). Hybrid position/force control of multi-arm cooperating robots. *Robotics and Automation. Proceedings. 1986 IEEE International Conference*, 3:82–89.

Herrera, R. T., Alcantara, S., Meda, J., and Velazquez, A. (2012). *Kinematic and Dynamic Modelling of Serial Robotic Manipulators Using Dual Number Algebra*.

Hussein, W. M. Quadrotor design, simulation and implementation. Master's thesis.

Johnson, W. (1994). *Helicopter Theory*. Dover.

Khosla, P. K. and Kanade, T. (1987). An algorithm to estimate manipulator dynamics parameters.

Kis, L. and Lantos, B. (2011). Sensor fusion and actuator system of a quadrotor helicopter. *Electrical Engineering*, 53(3-4):139–150.

Leishman, J. G. (2002). The breguet-richet quad-rotor helicopter of 1907. *Vertiflite*, 47:58–60.

Martinez, V. Modeling of flight dynamics of a quadrotor helicopter. Master's thesis.

Mellinger, D. and V., K. (2011). Minimum snap trajectory generation and control for quadrotors. *Int. Conf. on Robotics and Automation (ICRA)*, pages 2520–2525.

Nonami, K. (2006). Prospect and recent research and development for civil use autonomous unmanned aircraft as uav and mav. *Journal of system Design and Dynamics*, 1:120–128.

Noth, A., Bouabdallah, S., and Siegwart, R. (2004). Pid vs lq control techniques applied to an indoor micro quadrotor. *International Conf. on Intelligent Robots and Systems*, pages 2451–2456.

Oleg, Y. (2009). Development and testing of the miniature aerial delivery system snowflake. *20TH AIAA Aerodynamic deceleration systems technology conference and seminar.*

Pounds, P. Design, construction and control of a large quadrotor micro air vehicle. Master's thesis.

Pounds, P., Mahony, R., and Corke, P. (2006). Modelling and control of a quad-rotor robot. *Proceedings Australasian Conference on Robotics and Automation.*

Raemaekers, A. (2007). Design of a model predictive controller to control uavs. Master's thesis.

Romero, H., Salazar, S., and Lozano, R. (2009). Real-time stabilization of an eight-rotor uav using optical flow. *Robotics, IEEE Transactions*, 25:809–817.

Sanramaria, A. and Andrade, J. (2014). Hierarchical task control for aerial inspection. *euRathlon-ARCAS Workshop and Summer School on Field Robotics.*

Shin, J., Fujiwara, D., Nonami, K., and K., H. (2005). Model-based optimal attitude and positioning control of small-scale unmanned helicopter. *Robotica*, 23:51–63.

Sydney, N., Smyth, B., and Paley, D. A. (2013). Dynamic control of autonomous quadrotor flight in an estimated wind field.

Velliste, M., Perel, S., Spalding, M. C., Whitford, A. S., and Schwartz, A. B. (1995). Cortical control of a prosthetic arm for self-feeding. *Nature*, 453:1098–1101.

Vries, E. and Subbarao, K. (2010). Backstepping based nested multi-loop control laws for a quadrotor. *Int. Conf. on Control Automation Robotics Vision (ICARCV)*, pages 1911–1916.

Wai Weng, K. (2006). Design and control of an indoor micro quadrotor. *Research and Development, 2006. SCOReD 2006*, pages 173 – 177.

Zhu, B. and Huo, W. (2010). Trajectory linearization control for a quadrotor helicopter. *IEEE Int. Conf. on Control and Automation (ICCA)*, pages 34–39.