Università degli Studi di Padova

DEPARTMENT OF INFORMATION ENGINEERING
UNIVERSITY OF PADOVA

Department of Information Engineering
Control Systems Engineering

Force-Moment decoupled actuation property analysis for a Generically Tilted Multirotor platform

Candidate:
Marco Perin

Supervisor:
Prof. Angelo Cenedese
Co-supervisor:
Ph.D. Giulia Michieletto

Academic Year 2021/2022

Graduation date 20/10/2022

Abstract

Unmanned Aerial Vehicles are a rising topic in the control theory research, due to their emerging theoretical and practical challenges. This translates into the fact that recent studies focused on the structural properties of these aerial platforms actuated by more than four rotors, arbitrarily oriented, to allow them to perform better than a standard quadrotor, even in harsh conditions and in case of possible failures. Typically, to face these problems, the forces and moments generated these platforms turn out to be decoupled and can be treated independently.

In this thesis, the goal is to study the actuation and robustness properties of these multirotor platforms varying their physical construction parameters. The attention is first focused on the decoupling between the translational and rotational dynamics, that are intrinsically coupled in the original model, and then on the maximization of the force actuation capability that generates no angular momentum. Lastly all these studies are combined to plan a trajectory subject to all the given constraints, and that is not affected by propellers' failure.

Abstract (Italian)

Il controllo di veicoli aerei autonomi è un argomento di crescente interesse, dovuto all'affascinante teoria che ha come protagonisti questi dispositivi e che può condurre a impegnative ed entusiasmanti sfide sia in campo teorico che pratico. Questo si riflette nel fatto che studi recenti sono mirati a studiare le proprietà strutturali di questi multi rotori, permettendo performance migliori rispetto ai droni standard con quattro eliche, anche in situazioni anomale e nel caso di possibili guasti. Per affrontare il problema, i momenti e le forze sono stati trattati in modo indipendente le une dagli atri, rendendo possibile lo studio di come un numero maggiore di eliche possa portare a maggior robustezza e manovrabilità del drone stesso. Questi punti vengono trattati in primo luogo disaccoppiando le dinamiche di traslazione e rotazione, che nel modello dinamico sono intrinsecamente accoppiate, successivamente viene massimizzata la capacità di attuazione delle forze che non generano momento angolare. Tutto questo viene usato per generare una traiettoria che tenga in considerazione tutti i vincoli dati e che sia immune dai guasti alle eliche.

Contents

List of Figures

1 Introduction

1.1 State of the art

The aerial robotic modelization and control is a topic which is achieving resounding success in the research community, and lots of theoretical challenges still need to be addressed.

The current research topics span from single agent control problems, such as nonlinear control [1], to multiagent control, with for example consensus, flocking and formation control (using for example natural derived algorithms [2, 3] or rigidity theory [4]) on multi-UAV systems.

These topics are increasingly more relevant also because aerial platforms, with respect to ground ones, are extremely more versatile in environments where the terrain is uneven, for example in emergency situation where natural disasters occurs [5], or in other scenarios where they have a practical advantages for the task, like in mapping open environments [6].

In particular, great focus is being placed in the platforms with more than four, tilted propellers. Indeed, it has been proven that the standard quadrotor intrinsically lacks the full actuation property and are fully vulnerable to motor failures, even with tilted propellers [7].

1.2 Objectives

In this thesis, the first objective is to understand how the geometric design parameters of the multirotor impact the manoeuvrability of a generically tilted hexarotor. This condition is studied both with and without considering the constraint of generating zero moment from the platform.

This study is done because generating angular moment directly prevent the possibility

of attaining static hovering, that is when the forces and moments of the platform are in equilibrium, leading to a UAV being able to stand still in mid-air.

In order to address this, first what is required in order to maintain the platform in static hover, is defined, and then the impact of this problem in the analysis previously done.

The obtained results for the aforementioned topics will then be used in the last part of the thesis, to first analytically and numerically studying the problem of sideways static hovering, and then validating it with simulations. In the last part, with a nonlinear MPC controller, the problem of understanding the behaviour of the agent with the optimal parameters previously discussed will be addressed.

1.3 Structure of this elaborate

Mathematical model The first chapter presented in this thesis after this introduction, is a mathematical background chapter, used first to introduce basic concepts and notation, then for presenting the agent model used during the rest of the analysis. The problem of the decoupling of this model dynamic is then introduced, and a standard procedure to solve this is presented.

This model has firstly been studied in a symbolic framework, that is `SymPy`. This toolbox rapidly became unsuitable for the purpose, as it proved to have insufficient speed to manage the matrices and computations involved.

It became particularly clear when the symbolic representation of the (right) kernel of a 3 by 6 matrix had to be found, namely the matrix mapping the inputs of the agent to the angular momentum generated.

To cope with this, another toolbox has been adopted. By doing some researches, `SageMath` [8] turned out to be particularly well suited for this task, as the same computation was carried out with a significant speed-up, the order of eight to ten times with respect to `SymPy`.

Static analysis This modelization is then used in the next chapter to address the main focus of this thesis, that is to deal with the static analysis of the platform, meaning the description, study and discussion on the actuation capabilities of the platform.

Exploiting the symbolic calculus, some geometrical properties of the actuation capability geometric description. In particular, a closed formula in the case of non-constrained condition was found, thus leading to the possibility of exploring its value in a continuum of configurations, in contrapositive to previous studies [9], where only a numeric analysis was carried out. Comparing then the new results with the old one, a noticeable discrepancy was found, but it could be possibly attributed to the fact that the physical quantity considered is a volume, so the spatial discretization resolution used numerically can become significant for what concerns approximations, especially due to the shape of the volumes being formed

by all diagonal components, meaning that the discretized edge could suffer from cube voxel discretization.

After, the zero-moment constraint was addressed, but a closed formula was not found, so a numeric approach was still used. Nevertheless, a more accurate representation of the problem was found, and by exploiting some capabilities of `SageMath` it was possible to get accurate enough results.

Lastly, the study was expanded to consider a restriction of the problem, making interesting results emerge from this.

Dynamic analysis In the last chapter, the simulations that have been made to validate the results and to study the platform robustness to external forces noise are then presented. These simulations use a non-linear Model Predictive Control (MPC) controller to stabilize and assert control on the UAV.

In order to perform these simulations, a NMPC computational framework with the aid of MATMPC [10] was set up. Thanks to its modularity, it had been fairly simple to expand it to be able to run batches of simulations, with different modelizations and different parameters, such as various references and tasks to execute.

At this point the properties of this platform could be truly appreciated, with respect to a generic quadrotor, widely used nowadays in the industry, but with undoubtedly lower possibilities for complex manoeuvres. This was particularly clear when accounting also for its robustness, as its ability to counteract noise forces has proven to be considerable. This could be of particular interest for example when the application scenario requires tight standards of firmness during operation, like in delicate pick or place problems.

The last part of the thesis illustrates all these properties, starting from the study of the tilted static hovering problem, to the position reference tracking.

2 Mathematical Model

2.1 Mathematical Tools

2.1.1 Notation

The zero vector of size n is denoted by $\mathbf{0}_n$, and the vector constituted by 1 of size n is written as $\mathbf{1}_n$. $\mathbf{I}$ indicates the identity matrix, whereas $\boldsymbol{e}_i$ stands for its i-th columns.

Sine and cosine are sometimes denoted by their initial to prevent large equation size, meaning that $\cos(\alpha)$ is written as $c\alpha$ and $\sin(\alpha)$ becomes $s\alpha$.

Moreover, $\hat{\boldsymbol{x}}$ represents a versor in the same direction of $\boldsymbol{x}$, being defined as $\frac{\boldsymbol{x}}{|\boldsymbol{x}|}$.

Canonical Rotation Matrices

These matrices, belonging to $\mathbb{SO}(3)$, represent the standard rotations of α, β and γ around the axis x, y and z respectively.

$$\mathbf{R}_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \tag{2.1}$$

$$\mathbf{R}_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \tag{2.2}$$

$$\mathbf{R}_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

2.1.2 Pose Representation

Reference Frames

To represent the pose of an object, usually two reference frames are defined, being the inertial world frame $\mathcal{F}_\mathcal{W}$ and the body frame $\mathcal{F}_\mathcal{B}$, representing the object position, and fixed in its CoM.

We can then define some transformations between these two frames, in particular, in (2.4) we can see how to express a generic vector $\mathbf{v}_B \in \mathbb{R}^3$ in world frame, noted $\mathbf{v}_W$, with its inverse being simply (2.5).

$$\mathbf{v}_W = \mathbf{R}_{WB}\mathbf{v}_B \tag{2.4}$$

$$\mathbf{v}_B = \mathbf{R}_{WB}^T\mathbf{v}_W \tag{2.5}$$

If the translational component between the two frames is then considered, equation (2.4) becomes as in (2.6), with $\mathbf{T}_W$ being the position of the origin of the body frame with respect to the world frame. The inverse instead, becomes more elaborate, as the translational component must be taken into account into the inversion. This is shown in (2.7).

$$\mathbf{v}_W = \mathbf{R}_{WB}\mathbf{v}_B + \mathbf{T}_W \tag{2.6}$$

$$\mathbf{v}_B = \mathbf{R}_{WB}^T(\mathbf{v}_W - \mathbf{T}_W) \tag{2.7}$$

Skew Symmetric matrix

Considering a generic 3D vector $\boldsymbol{\omega} = [\omega_x\ \omega_y\ \omega_z]^T$, we define the skew operator $[\,\cdot\,]_\times$ applied to $\boldsymbol{\omega}$ as

$$[\boldsymbol{\omega}]_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_y & 0 & -\omega_x \\ -\omega_z & \omega_x & 0 \end{bmatrix} \tag{2.8}$$

Rotation Matrix derivative

It is useful to introduce also the derivative of a rotation matrix, in order to be able later to describe the UAV dynamics equations.

$$\mathbf{R}\mathbf{R}^T = \mathbf{I} \quad \Rightarrow \quad \dot{\mathbf{R}}\mathbf{R}^T + \mathbf{R}\dot{\mathbf{R}}^T = \mathbf{0} \tag{2.9}$$

This is done by starting from equation (2.9), deriving it, and then noticing that the term $\dot{\mathbf{R}}\mathbf{R}^T$ is a skew symmetric matrix $\boldsymbol{S}$. The passage to (2.10) is then trivial, but using this formula, it can be proven [11] that the matrix $\boldsymbol{S}$ is then equal to $[\boldsymbol{\omega}]_\times$, with $\boldsymbol{\omega}$ being the vector representing the angular velocity vector.

$$\dot{\mathbf{R}}\mathbf{R}^T = \boldsymbol{S} \Rightarrow \dot{\mathbf{R}} = \boldsymbol{S}\mathbf{R} \tag{2.10}$$

This means that equation (2.10) translates to (2.11), that becomes (2.12) if we want to express $\dot{\mathbf{R}}_{WB}$ with respect to $\boldsymbol{\omega}_B$.

$$\dot{\mathbf{R}}_{WB} = [\boldsymbol{\omega}_W]_\times \mathbf{R}_{WB} \tag{2.11}$$

$$\dot{\mathbf{R}}_{WB} = \mathbf{R}_{WB} [\boldsymbol{\omega}_B]_\times \tag{2.12}$$

2.1.3 System actuation Capability

Fully and Under-actuated systems

We consider a UAV to be a rigid body that follows a second order dynamic, of kind

$$\ddot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{u}, t) \tag{2.13}$$

Where $\boldsymbol{x}$ represents a generalized-coordinate state vector and $\boldsymbol{u}$ represents the input vector acting on the system.

We can also describe this relation by making explicit the part of the system where the input has effect, and the one where it does not.

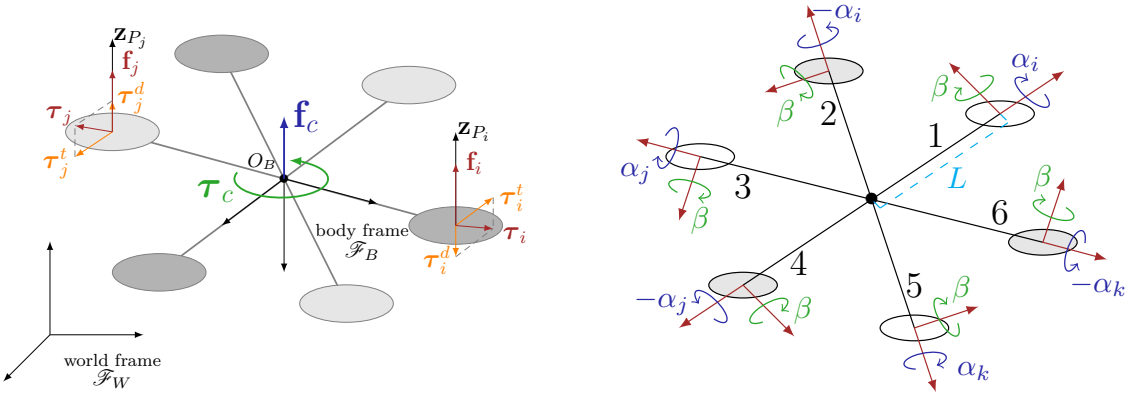
$$\ddot{\boldsymbol{x}} = \boldsymbol{f}_1(\boldsymbol{x}, \dot{\boldsymbol{x}}, t) + \boldsymbol{f}_2(\boldsymbol{x}, \dot{\boldsymbol{x}}, t)\boldsymbol{u} \tag{2.14}$$

A system that is capable of being commanded **any** instantaneous acceleration in $\boldsymbol{x}$, that translates to (2.15) is said to be **Fully-actuated**. If instead the system properties reflects in (2.16), the system is said to be **Under-actuated**

$$\dim [\boldsymbol{f}_2(\boldsymbol{x}, \dot{\boldsymbol{x}}, t)] = \dim [\boldsymbol{x}] \tag{2.15}$$

$$\dim [\boldsymbol{f}_2(\boldsymbol{x}, \dot{\boldsymbol{x}}, t)] < \dim [\boldsymbol{x}] \tag{2.16}$$

2.2 Standard n-rotor model



(a) Model overview (b) Rotor Arm geometric description

Figure 2.1: Hexa rotor agent model

A generic n-rotor UAV is made of a rigid body, composed of n arms, with motors attached at their end. For each motor shaft a propeller is attached, while rotating with sufficient speed. Each motor rotation axis is denoted by $\hat{\boldsymbol{u}}_{z_i}$ for the i-th motor.

These axes do not need to be parallel with one another, even if in many quadrotors seen nowadays, it is a common practice. That is, in fact, a special case that is often avoided in research as it is intrinsically under-actuated, and fully vulnerable to rotor failures [7]. Indeed, the way they are not parallel, determines the coefficients of the system matrices, and in particular the matrix that represents $\boldsymbol{f}_2$ in (2.14). Therefore, it is possible (and often done) to make the system fully actuated by changing orientation to the propellers.

To properly define the agent geometry, the reference frames have to be introduced, namely the inertial world frame $\mathcal{F}_W$ and the agent's body frame $\mathcal{F}_B$, with origin in O_B, that coincides with the Centre of Mass (CoM) of the platform.

The O_B position in $\mathcal{F}_B$ is denoted by the vector $\boldsymbol{p} \in \mathbb{R}^3$, while the orientation can be represented by a rotation matrix $\mathbf{R} \in \mathbb{SO}(3)$.

The structure of the multirotor, as depicted in Figure 2.1a, is a Generically Tilted MultiRotor (GTMR) model, with rotors positions and orientations as described by the parameters α, β and in general γ. The role of these construction variables, is described by (2.19)-(2.20) and shown in Figure 2.1b.

$$\gamma_i = (i-1)\frac{2\pi}{n} \tag{2.17}$$

$$\boldsymbol{p}_i = \mathbf{R}_z(\gamma_i)\ell\boldsymbol{e}_1 \tag{2.18}$$

$$\hat{\boldsymbol{u}}_{z_i} = \mathbf{R}_z(\gamma_i)\mathbf{R}_y(\beta)\mathbf{R}_x(\alpha_i) \tag{2.19}$$

$$= \mathbf{R}_(\gamma_i)\mathbf{R}_y(\beta)\mathbf{R}_x((-1)^i\alpha) \tag{2.20}$$

These equations, where ℓ is the length of each link, are used as the model of the

hexarotor considered is a star-shaped one, as its modelization is the one studied in this project.

2.2.1 Kinematics and dynamics

The agent pose can be represented by the pair $\mathcal{X} = (\boldsymbol{p}, \boldsymbol{R}) \in \mathbb{R}^3 \times \mathbb{SO}(3)$, describing the full pose of the vehicle in $\mathcal{F}_W$.

To indicate the structure twist instead, the pair $(\boldsymbol{v}, \boldsymbol{\omega})$ is presented where $\boldsymbol{v} = \dot{\boldsymbol{p}} \in \mathbb{R}^3$ symbolizes the linear velocity of O_B with respect to $\mathcal{F}_W$, and $\omega \in \mathbb{R}^3$ is the angular velocity of $\mathcal{F}_B$ with respect to $\mathcal{F}_W$, expressed in $\mathcal{F}_B$.

This lead to a kinematic formulation as presented in (2.21) and (2.22).

$$\dot{\boldsymbol{p}} = \boldsymbol{v} \tag{2.21}$$

$$\dot{\mathbf{R}} = \mathbf{R}\,[\boldsymbol{\omega}]_\times \tag{2.22}$$

The dynamic equations are then derived using the standard Newton-Euler approach, considering the forces and torques that are generated by each propeller. The i-th propeller, with $i = 1 \dots n$, rotates around its own spinning axis $\hat{\boldsymbol{u}}_{z_i}$, passing through its centre $\boldsymbol{p}_i$ with a controllable spinning rate that is $\omega_i \in \mathbb{R}$.

The model taken into account, namely a generically tilted star shaped multirotor, has each motor position defined as in (2.18).

Since each rotor spins in the Clockwise (CW) or Counterclockwise (CCW) direction with respect to its axis, the relative raw angular velocity is not used directly as the system input, instead we define $u_i = |\omega_i|\omega_i \in \mathbb{R}$ as the control input, representing the 'signed square' of the motor angular velocity.

This also have the benefit of having a linear map to the forces and moments of the agent, as in (2.23) and (2.24).

We then say, according to the most widely used model, that the i-th propeller applies at O_{P_i}:

- A thrust force $\boldsymbol{f}_i \in \mathbb{R}^3$ that is expressed by (2.23) in $\mathcal{F}_B$, where $c_{f_i} \in \mathbb{R}^+$ is the (constant) parameter that represents the force coefficient of the propeller.
- A drag moment $\boldsymbol{\tau}_i^d \in \mathbb{R}^3$ whose direction is opposite to the angular velocity of the propeller and whose expression in $\mathcal{F}_B$ is reported in (2.24).

  The parameter $c_{\tau_i} \in \mathbb{R}$ is the (constant) parameter representing the torque coefficient (positive if the propeller is CW and negative otherwise).

$$\boldsymbol{f}_i = c_{f_i} u_i \hat{\boldsymbol{u}}_{z_i} \tag{2.23}$$

$$\boldsymbol{\tau}_i^d = c_{\tau_i} u_i \hat{\boldsymbol{u}}_{z_i} \tag{2.24}$$

The entire structure of the UAV is now considered, so we have to take into account the thrust moment resulting by the force generated for each propeller, that is $\boldsymbol{\tau}_i^t \in \mathbb{R}^3$, defined in (2.26). Then the total control force $\boldsymbol{f}_c \in \mathbb{R}^3$ and the control moment $\boldsymbol{\tau}_c \in \mathbb{R}^3$, applied at O_B and expressed in $\mathcal{F}_B$, are reported respectively in (2.25)-(2.27)

$$\boldsymbol{f}_c = \sum_{i=1}^{n} \boldsymbol{f}_i = \sum_{i=1}^{n} c_f \hat{\boldsymbol{u}}_{zi} u_i \tag{2.25}$$

$$\boldsymbol{\tau}_i^t = \boldsymbol{p}_i \times \boldsymbol{f}_i \tag{2.26}$$

$$\boldsymbol{\tau}_c = \sum_{i=1}^{n} (\boldsymbol{\tau}_i^t + \boldsymbol{\tau}_i^d) = \sum_{i=1}^{n} (c_f \boldsymbol{p}_i \times \hat{\boldsymbol{u}}_{zi} + c_{\tau_i} \hat{\boldsymbol{u}}_{zi}) u_i \tag{2.27}$$

Considering now the control input vector $\boldsymbol{u} = [u_1 \dots u_n]^T \in \mathbb{R}^n$, the equations (2.25) and (2.27) can be rewritten shortly as:

$$\boldsymbol{f}_c = \mathbf{F}\boldsymbol{u} \tag{2.28}$$

$$\boldsymbol{\tau}_c = \mathbf{M}\boldsymbol{u} \tag{2.29}$$

That translates, into matrix form, in:

$$\begin{bmatrix} \boldsymbol{f}_c \\ \boldsymbol{\tau}_c \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ \mathbf{M} \end{bmatrix} \boldsymbol{u} \tag{2.30}$$

Where $\mathbf{F}$ represents the control force input matrix $\in \mathbb{R}^{3\times n}$, while $\mathbf{M} \in \mathbb{R}^{3\times n}$ is the control moment input matrix, and therefore these depend on the geometric and aerodynamic parameters of the agent.

The platform dynamic is then studied, and neglecting the second order effects, it is described by the following system of Newton-Euler equations:

$$m\ddot{\boldsymbol{p}} = -mg\boldsymbol{e}_3 + \mathbf{R}\boldsymbol{f}_c = -mg\boldsymbol{e}_3 + \boldsymbol{RFu} \tag{2.31}$$

$$\boldsymbol{J}\dot{\boldsymbol{\omega}} = -\boldsymbol{\omega} \times \boldsymbol{J}\boldsymbol{\omega} + \boldsymbol{\tau}_c = -\boldsymbol{\omega} \times \boldsymbol{J}\boldsymbol{\omega} + \boldsymbol{Mu} \tag{2.32}$$

In which $g > 0, m > 0$ and $\boldsymbol{J} \in \mathbb{R}^{3\times 3}$ are the gravitational acceleration, the total mass of the platform and its positive definite inertia matrix, respectively.

To summarize, the whole system equations are reported here.

$$\dot{\boldsymbol{p}} = \boldsymbol{v} \tag{2.33}$$

$$\dot{\mathbf{R}} = \mathbf{R}\,[\boldsymbol{\omega}]_\times \tag{2.34}$$

$$m\ddot{\boldsymbol{p}} = -mg\boldsymbol{e}_3 + \boldsymbol{RFu} \tag{2.35}$$

$$\boldsymbol{J}\dot{\boldsymbol{\omega}} = -\boldsymbol{\omega} \times \boldsymbol{J}\boldsymbol{\omega} + \boldsymbol{Mu} \tag{2.36}$$

2.2.2 On rotational velocity and Euler angles derivatives

At this point, it is important to study the relation between the rotational velocity in the body frame and the derivative of the roll, pitch and yaw euler angles, namely the relation between $\boldsymbol{\omega}_B$ and $\boldsymbol{\alpha}$, that has ϕ, θ and ψ angles as components.

Starting from equation (2.12), we can compare it to its formulation in (2.37).

$$\dot{\boldsymbol{R}}_{WB}(\phi,\theta,\psi) = \frac{\partial \dot{\boldsymbol{R}}_{WB}}{\partial \phi}\dot{\phi} + \frac{\partial \dot{\boldsymbol{R}}_{WB}}{\partial \theta}\dot{\theta} + \frac{\partial \dot{\boldsymbol{R}}_{WB}}{\partial \psi}\dot{\psi} \tag{2.37}$$

Indeed, we know that $[\boldsymbol{\omega}_B]_\times = \boldsymbol{R}_{WB}^T \dot{\boldsymbol{R}}_{WB} = f(\dot{\phi}, \dot{\theta}, \dot{\psi})$, and by performing the needed computations, (2.38) can be obtained.

$$\boldsymbol{\omega}_B = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \boldsymbol{W}\dot{\boldsymbol{\alpha}} = \begin{bmatrix} 1 & 0 & -s\theta \\ 0 & c\phi & c\theta s\phi \\ 0 & -s\phi & c\theta c\phi \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{2.38}$$

It can be remarked that the derivative of the pitch angle is modified by the rotation around x, the derivative of the roll angle is directly mapped to the component of the rotational velocity, and finally the derivative of the yaw angle is modified by the rotation around x and y. This can be seen clearly by decomposing the transformations to find $\boldsymbol{\omega}_B$, resulting in (2.39), where we see how each component is influenced by the relative transformations.

$$\boldsymbol{\omega}_B = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \boldsymbol{R}_x^T(\phi) \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \boldsymbol{R}_x^T(\phi)\boldsymbol{R}_y^T(\theta) \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \tag{2.39}$$

It is then noticeable that the aforementioned mapping is needed only in case of a precise modelization, and sometimes it can be relaxed. This is the case when, for example, is the almost hovering conditions. Indeed, in hovering we can say that $\theta \approx 0$ and $\phi \approx 0$, leading to an approximation of the rotational velocity as in (2.40)

$$\boldsymbol{\omega}_B = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \approx \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}. \tag{2.40}$$

2.3 Force-Moment decoupling

2.3.1 Problem description

In this section, the coupling between the forces and the moments generated by the propellers is studied. We can in fact emphasize how the translational dynamic (2.35) is influenced by the rotational one (2.36) through (2.34).

If we now recall the definition of fully (and under) actuated system in section 2.1.3, we may rewrite (2.14) as in (2.41).

$$\begin{bmatrix} \ddot{\boldsymbol{p}} \\ \dot{\boldsymbol{\omega}} \end{bmatrix} = \begin{bmatrix} -g\boldsymbol{e}_3 \\ -\boldsymbol{J}(\boldsymbol{\omega} \times \boldsymbol{J}\boldsymbol{\omega}) \end{bmatrix} + \begin{bmatrix} m^{-1}\mathbf{R}\mathbf{F} \\ \boldsymbol{J}^{-1}\mathbf{M} \end{bmatrix} \boldsymbol{u} \tag{2.41}$$

It is clear then that $\begin{bmatrix} m^{-1}\mathbf{R}\mathbf{F} \\ \boldsymbol{J}^{-1}\mathbf{M} \end{bmatrix} \in \mathbb{R}^{6\times n}$ needs to have full rank (6) in order to have a fully-actuated system:

$$\operatorname{rank} \begin{bmatrix} m^{-1}\mathbf{R}\mathbf{F} \\ \boldsymbol{J}^{-1}\mathbf{M} \end{bmatrix} = \operatorname{rank} \left(\begin{bmatrix} m^{-1}\mathbf{R} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{J}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{F} \\ \mathbf{M} \end{bmatrix} \right) \tag{2.42}$$

$$= \operatorname{rank} \begin{bmatrix} \mathbf{F} \\ \mathbf{M} \end{bmatrix} \tag{2.43}$$

It is immediate to see then, that a quadrotor will not be able to attain full actuation for any configuration of the propellers, as the matrices in (2.43) would have 4 columns.

2.3.2 Actuation

As we want to decouple the output force and momentum generated, we can firstly recall them, as

$$\mathbf{F} \in \mathbb{R}^{3\times n} \qquad \text{Control force } \boldsymbol{f}_c \text{ input matrix} \qquad 1 \leq \operatorname{rank}[\mathbf{F}] \leq 3 \tag{2.44}$$

$$\mathbf{M} \in \mathbb{R}^{3\times n} \qquad \text{Control moment } \boldsymbol{\tau}_c \text{ input matrix} \qquad 1 \leq \operatorname{rank}[\mathbf{M}] \leq 3 \tag{2.45}$$

It is then possible to define two new matrices, $\mathbf{A}$ and $\mathbf{B}$ such as in (2.46) and (2.47) respectively, to design a change of basis matrix to decouple the inputs.

$$\mathbf{A} \in \mathbb{R}^{n\times 3} \quad \text{s.t.} \quad Im[\mathbf{A}] = Im[\mathbf{M}^T] = (Ker[\mathbf{M}])^{\perp} \tag{2.46}$$

$$\mathbf{B} \in \mathbb{R}^{n\times(n-3)} \quad \text{s.t.} \quad Im[\mathbf{B}] = Ker[\mathbf{M}] \tag{2.47}$$

We can then define the change of basis matrix $\boldsymbol{T} \in \mathbb{R}^{n\times n}$ as $\boldsymbol{T} = [\mathbf{A}|\mathbf{B}]$, that is full rank as a consequence of the construction of the two matrices.

This can be used to create a new input vector $\tilde{\boldsymbol{u}}$ that becomes very convenient for the decoupling problem.

$$\boldsymbol{u} = \mathbf{T}\tilde{\boldsymbol{u}} \implies \boldsymbol{u} = [\mathbf{A}|\mathbf{B}] \begin{bmatrix} \tilde{\boldsymbol{u}}_A \\ \tilde{\boldsymbol{u}}_B \end{bmatrix} = \mathbf{A}\tilde{\boldsymbol{u}}_A + \mathbf{B}\tilde{\boldsymbol{u}}_B \tag{2.48}$$

Indeed, from (2.48) we can first expand (2.30) into (2.49).

$$\begin{bmatrix} \mathbf{F} \\ \mathbf{M} \end{bmatrix} \mathbf{T} = \begin{bmatrix} \mathbf{FA} & \mathbf{FB} \\ \mathbf{MA} & \mathbf{MB} \end{bmatrix} = \begin{bmatrix} \mathbf{FA} & \mathbf{FB} \\ \mathbf{MA} & \mathbf{0} \end{bmatrix} \tag{2.49}$$

Then we are finally able to see, thanks to (2.51), that the decomposition obtained is able to map the input into forces and moments component that are easy to decouple, in opposition to the original matrices in (2.30).

This is because now the moment can be directly allocated with $\tilde{\boldsymbol{u}}_A$, having then the ability to set the generated force entries by simply accounting for the term $\boldsymbol{f}_c^A$ generated by $\mathbf{FA}\tilde{\boldsymbol{u}}_A$.

$$\begin{bmatrix} \boldsymbol{f}_c \\ \boldsymbol{\tau}_c \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ \mathbf{M} \end{bmatrix} \boldsymbol{u} = \begin{bmatrix} \mathbf{F} \\ \mathbf{M} \end{bmatrix} \mathbf{T}\tilde{\boldsymbol{u}} = \begin{bmatrix} \mathbf{FA} & \mathbf{FB} \\ \mathbf{MA} & 0 \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{u}}_A \\ \tilde{\boldsymbol{u}}_B \end{bmatrix} \tag{2.50}$$

$$= \begin{bmatrix} \boldsymbol{f}_c^A + \boldsymbol{f}_c^B \\ \boldsymbol{\tau}_c^A \end{bmatrix} \tag{2.51}$$

3 Static Analysis

To find meaningful properties regarding the UAV manoeuvrability, this chapter dives into different aspects that can be of relevance for the problem.

First, a geometric procedure is defined, to calculate the volume of the agent's feasible force space. Using this formula, an analysis spanning the possible configuration space in $\alpha - \beta$ is done, to study which configuration can perform better, and what an angle variation actually changes in the total volume.

The problem is then restricted to the case of zero-moment generation, with a numerical but still refined approach, and the same analysis is repeated, finding now the differences with respect to the previous scenario.

At last, an approximate formulation for the volume of the constrained forces restricted to the forces that can at least keep the agent in static hover is being studied, as those are the forces that can effectively move the hexarotor in mid-air.

3.1 Force volume description

The UAV, as seen from the previous chapter, can benefit in multiple ways by having a geometry that improves its capability of generating forces in other directions than the vertical one.

At first, this provides the ability to counteract the lateral noise acting on the agent, and then it also adds possible instantaneous force vectors to be created, making possible more complex manoeuvres.

It has to be notice though, that even if the benefits of having the properties just mentioned are abundant, there are some disadvantages.

Having the rotor axis tilted in fact implies that to generate a purely vertical vector, the input forces, that are generally tilted, need to be combined to make each horizontal component cancel out. This becomes immediately a disadvantage when considering that

the consumption needed to keep the agent hovering increases. This is because to apply the same total vertical force having tilted propellers, some spurious lateral forces are created, requiring that opposed ones to be generated in order to counteract them.

From subsection 2.2.1 and in particular from (2.30), we can consider the columns of $\mathbf{F}$ as the set of vectors that are the generators of a cone representing the feasible forces that the agent is capable of generating, represented in Figure 3.4a.

Taking then $\boldsymbol{f}_i$ as the i-th force generator vector, all the forces that can be generated will then be the summation of all the k basis vectors, each one constrained to the input bounds. These are all the points belonging to the set $\boldsymbol{D}$ as in (3.1) for which the coefficients α_i exists in their respective domain.

$$\boldsymbol{D} = \sum_{i=1}^{k} \alpha_i \boldsymbol{f}_i \qquad \text{with} \quad \alpha_i \in [0, 1] \tag{3.1}$$

3.1.1 Volume Formula Study

To search for an analytic formula of the volume representing the space of feasible forces, at first the problem was reduced to a two-dimensional one. This has been done because the basic objective was to find a formula for the space spanned by any linear combination of vectors with coefficients in limited domain, as in (3.1).

It can be noticed that with this formulation, no information on the dimensionality of the problem is being used, as long as a metric of this space can be defined. It is clear though, that if the dimension of the space in which the vectors belong is 3, this metric will be the volume, whereas in the 2D case it reduces to an area.

Generally, the shape that we would get is that of a polyhedron, where various numeric method exists [12] to calculate its volume, but the underlying structure of the vectors generating it provides us with a possibility of finding a closed form solution.

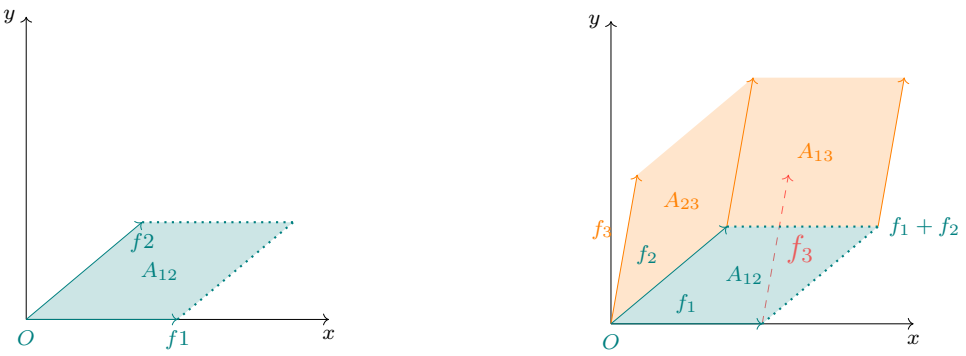
We begin from the basic case of two vectors, as in Figure 3.1a. Naming the two vectors $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$, we see that the area reachable from a combination of the two vectors can be calculated by the absolute value of determinant of the matrix formed by them, namely $\left\|\det \begin{bmatrix} \boldsymbol{f}_1 & \boldsymbol{f}_2 \end{bmatrix}\right\|$.

This polygon will be referred as P_{12}, whereas its area will be defined A_{12}, for shortness.

Now the existing polygon is extended with another vector $\boldsymbol{f}_3$, as in figure Figure 3.1b.

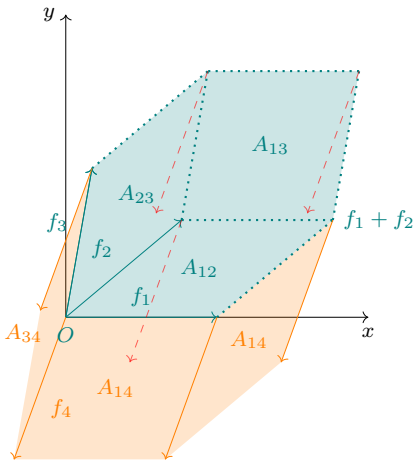
This is done by adding the new vector to each point of the convex hull, and considering only the ones that do not intersect with the existing polygon. It is possible to see that now the total area is the one spanned by the first two vectors, with the additional areas of the polygons composed by the combination of vector $\boldsymbol{f}_3$ with the first two, that is $A_{123}. = A_{12} + A_{13} + A_{23}$.

From this, we can extend up to four vectors, as depicted in Figure 3.2a, and it is possible to scale up to n vectors , leading to a total area of the resulting polytope equal to (3.2).



(a) 2 vectors polygon (b) 3 vectors polygon

Figure 3.1: Polytope growing with vector number - two to three vectors



(a) 4 vectors polygon

Figure 3.2: Polytope growing with vector number - four vectors

$$A_k = \sum_{i=1}^{\ell} A_{C_\ell} \qquad \text{with} \quad \begin{cases} \ell = \binom{n}{2} \\ C_\ell = \ell\text{-th combination with indexes } i,j \text{ of each base vector(Including } \mathbf{0}\text{), with } k \text{ basis vectors} \end{cases} \tag{3.2}$$

From this point, the problem in the third dimension was addressed, to match the dimensionality of the real problem. The difference, in this scenario, is that instead of reasoning in terms of area, the volume metric was finally stumbled upon.

It is well known that the volume of the space spanned by three vectors in 3D, like in Figure 3.3, is defined as the module of the determinant of the matrix composed by those vector stacked as columns. The area A_{ij} now becomes the volume V_{ijk}. In this way, (3.2) becomes (3.3), with the core concepts remaining the same .

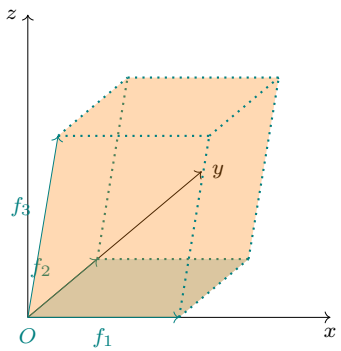


Figure 3.3: 3 vectors 3D polytope

$$V_k = \sum_{i=1}^{\ell} V_{C_\ell} \qquad \text{with} \quad \begin{cases} \ell = \binom{n}{3} \\ C_\ell = \ell\text{-th combination with indexes } i,j,k \text{ of each base vector (Including } \mathbf{0}\text{), with } k \text{ basis vectors} \end{cases} \tag{3.3}$$

$$= \sum_{i=1}^{\ell} \left\| \det \begin{bmatrix} \boldsymbol{f}_i & \boldsymbol{f}_j & \boldsymbol{f}_k \end{bmatrix} \right\| \tag{3.4}$$

3.1.2 Volume Physical meaning

Before moving to the value analysis of the volume, it is worthwhile to explain its underlying physical meaning.

Its basic definition is of a 3D space such that any point that lies inside of it, is reachable by a combination of some basis vectors. This can be seen in (3.1), with each vector coefficient α_i in the range $[0, 1]$, representing how much each vector is used.

This concept is now used with the basis vectors being the force vectors generated by each propeller, thus the coefficients stands for how much the corresponding propeller needs to spin.

Having a large volume of this polyhedron means that we can have a lot of freedom in allocating a desired control force vector, thus meaning more capability of controlling the agent.

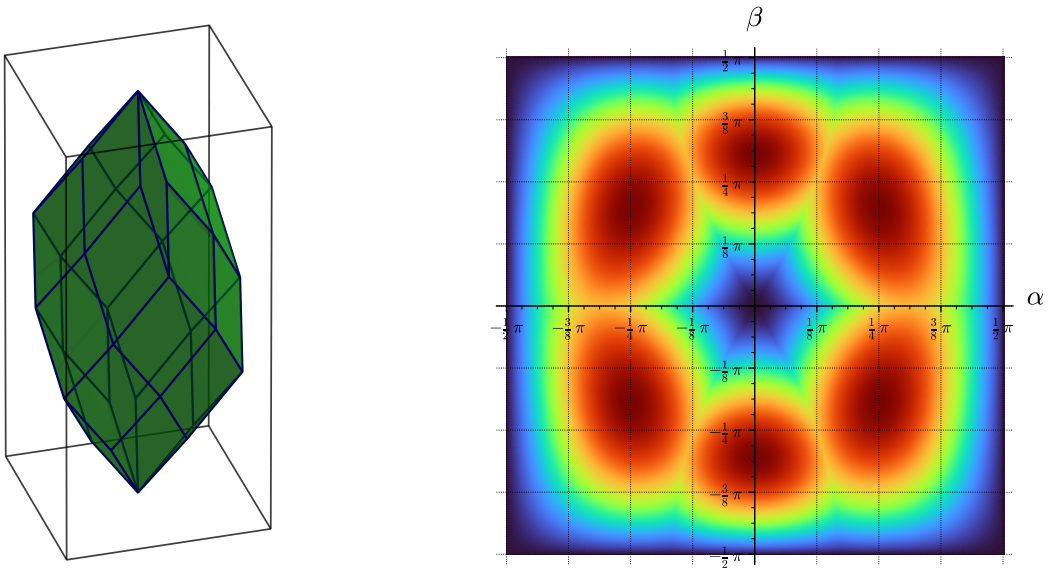
It has to be taken into account also that while the volume can be expanded by (properly) increasing the drone geometric angles, as it will be seen in the following section, this also generally decreases the z component of the total force that can be applied.

Using more aggressive angles, in fact, means that the propellers are oriented less perpendicular to the ground, meaning that the amount of force required by each one to keep the UAV hovering increases, leading to a higher power consumption just to achieve this common task.

On top of that, the decrease in the volume's vertical height limits the ability to generally create vertical forces, with consequences that can range from simply having less vertical

manoeuvrability, up to extreme cases in which also hovering is not feasible, as it will be seen in subsection 3.4.1.

3.2 Unconstrained Volume Analysis



(a) $V_6(\alpha, \beta)$ Polytope without constraints and ($\omega_{max} = 1, \alpha = 35°, \beta = 10°$)

(b) $V_6(\alpha, \beta)$ value with bothα and β in the interval $[-\pi/2, \pi/2]$

Figure 3.4: Hexarotor volume polytope shape for a fixed angle configuration and its volume value plot as a function of α and β

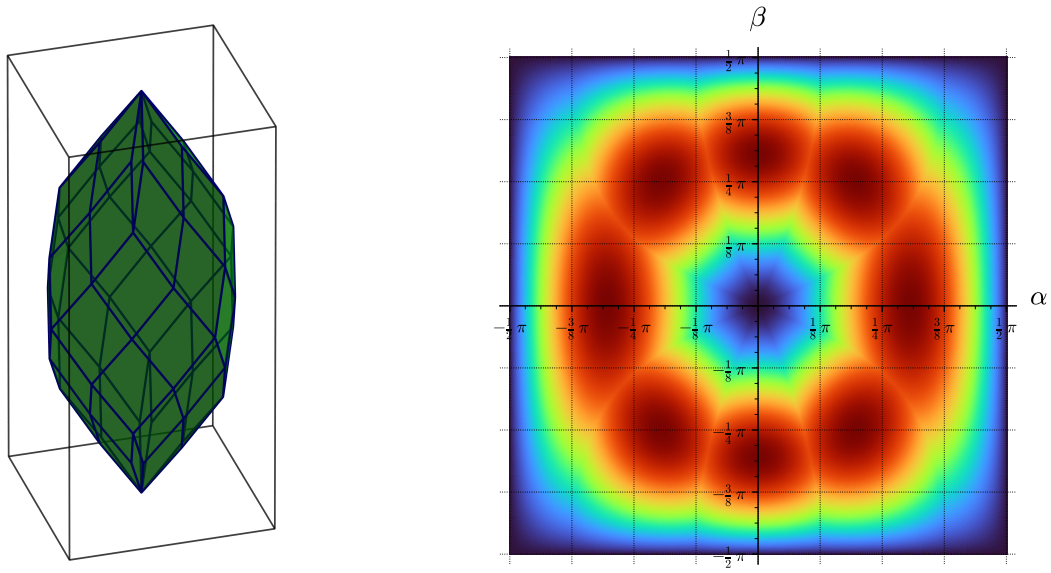
Having found the closed formula for the 3D volume, now its value is analysed varying the construction parameters of the UAV. Formally, the vectors taken as basis for constructing the force volume are shown in (3.5), that are simply the columns of the $\mathbf{F}$ matrix scaled by the maximum speed achievable by each propeller (squared, as from subsection 2.2.1).

$$\boldsymbol{f}_i' = \mathbf{F} u_{\max} = \mathbf{F} \omega_{\max}^2 \tag{3.5}$$

The shape that comes out by the procedure of taking the sum of all basis vectors, from all the combinations of size three, is depicted in Figure 3.4a, whereas the plot of the volume value is shown in Figure 3.4b as a function of the parameters α and β, each one in the range $[-\pi/2, \pi/2]$.

It is interesting to look at the symmetry in the plot. This still has to be further investigated, but the pattern strongly shows the same number of peaks as the number of rotors taken in exam (six), and the anti-peaks shows a radial pattern with the same number of axis too.

To investigate the fact that the number of peaks and valleys is the same as the number of propellers, the same analysis has been repeated for an octa rotor platform. The resulting volume plot is shown in Figure 3.5b: the overall pattern is very reminiscent of the hexa-rotor one, with of course the radial axis and the number of peaks being eight.



(a) $V_8(\alpha, \beta)$ Polytope without constraints ($\omega_{max} = 1, \alpha = 35°, \beta = 10°$)

(b) $V_8(\alpha, \beta)$ value in the interval $[-\pi/2, \pi/2]^2$

Figure 3.5: Octarotor volume polytope shape for a fixed angle configuration and its volume value plot as a function of α and β

In this case though, it is interesting to see that the stripes that start from the origin and expands towards the borders are not as straight as in the hexa-rotor case. They in fact tend to get warped as the α limit is approached.

The last observation regards the boundaries of the plot, but it is trivial to expect the volume value to collapse as soon as α or β drops to $\pm\pi/2$.

3.3 Zero-moment Constraint study

The next problem is to include the constraints in the formula. To do this, the whole analysis starts from using the results from the force-moment decoupling seen in section 2.3.

We then observe that the number of vectors in the kernel of $\mathbf{M}$ is three, and therefore the combinations of inputs that lead to zero moment is the same number.

This provides us with a basis made by $\mathbf{FB}$, from (2.50), that is at most of rank 3, as of (2.44) and the full rankness of $\mathbf{B}$ given by its construction.

3.3.1 B Matrix Feasibility

The matrix $\mathbf{B}$ calculated with `SageMath` by taking the right kernel of $\mathbf{M}$, even though trivially satisfying the zero-moment constraint takes values unfeasible for the input, like being negative or higher than the maximum value attainable by the motors.

Nevertheless, a non-trivial property of the $\mathbf{B}$ matrix is row-stochastic, meaning that its column sums up to one. This has various implications, with the first one being from a geometric point of view.

The fact that the matrix sums up to one, first implies that the vector $\mathbf{1}$ lies in the kernel of $\mathbf{M}$, but physically it translates to having all the motors spinning at the same rate. It is then trivial to think of its belonging to zero-moment space, as if we consider the geometrical symmetries of the UAV, it comes easy to deduce that the moments of the rotors cancel each other out if actuated evenly.

The other notable property of having the $\mathbf{1}$ vector in the kernel, is that it provides us the possibility to have it added to any column of $\mathbf{B}$ while having the resulting vector in the kernel. This provides us the ability to simply "shift" a column by its minimum component to make it greater than zero, rescaling it later to keep in the feasible domain of the motors.

Although this reasoning is easy to apply, at first a manual approach to condition the $\mathbf{B}$ was used, as it has been the first to be developed.

We can see in (3.6) how the matrix that `SageMath` produces, has its first rows as the identity ($\mathbf{I}_3$) matrix, possibly due to its procedure of calculating it, but with neither symmetries nor other noticeable properties. After some computations, the matrix is then transformed into the one in (3.7), that first satisfies all the constraints of the inputs, and on top of that a clear pattern in the elements is noticeable.

$$\mathbf{B}_{\text{sage}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0.01728 & -0.01728 \\ 0.01698 & 1 & -0.01728 \\ 0.01698 & -0.01698 & 1 \end{bmatrix} \tag{3.6}$$

$$\Downarrow$$

$$\mathbf{B}' = \begin{bmatrix} 1 & 0 & 0.01698 \\ 0 & 1 & 0.01698 \\ 0 & 0.01698 & 1 \\ 1 & 0.01698 & 0 \\ 0.01698 & 1 & 0 \\ 0.01698 & 0 & 1 \end{bmatrix} \tag{3.7}$$

Another way of finding a feasible matrix is to use the facts reported above, that is to possibly manipulate each column of the original matrix, and by first shifting it to have no negative terms, then scaling it to limit the maximum, resulting in a matrix with elements in the desired range.

$$\mathbf{B}'_{\text{auto}} = \begin{bmatrix} 1 & 0.01669 & 0.01698 \\ 0 & 0.9997 & 0.01698 \\ 0 & 0.01669 & 1 \\ 1 & 0.03368 & 0 \\ 0.01698 & 1 & 0 \\ 0.01698 & 0 & 1 \end{bmatrix} \tag{3.8}$$

An example of the result of this procedure is reported in (3.8). It can be seen though, that the first and last column are the same as the matrix found by applying the manual procedure, with the second column being the only one that differs. This matrix proved to generate a volume smaller than the one generated by the matrix found with the manual method, when applied to the **F** matrix. This is possibly due to the second column being less orthogonal to the other two, respect with the second column of (3.7).

3.3.2 Constrained polytopes calculation

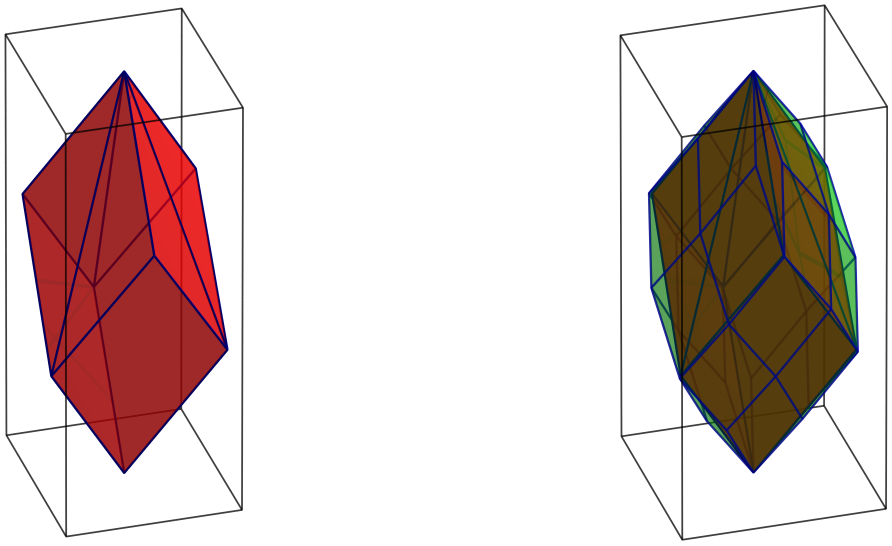
The shapes generated by the constrained basis vectors will now be discussed. Differently from the shapes in section 3.2, they are now simpler, as they are generated by three vectors. The result is a polyhedron similar to the one in Figure 3.3, as in both cases three basis vectors are used, but with the important difference that will make the use of the analytical formula found in the previous chapter to be of no use.

This difference lies in the fact that each vector $\boldsymbol{f}_i$ in the non-constrained formula is independent, making possible to sum them up by just making care of keeping the coefficient in the range $[0, 1]$, as previously said. In this scenario, however, it is not possible to simply sum up the basis vectors, as if we do, we exceed the input value bound for the overlapping terms.

An example of this can be seen from the previous section with the terms that if sum up, would clearly exceed one. The volume for each polytope in this section, is then calculated by firstly calculating all the sum of the combinations of the basis vectors, as in origin, then scaling each vector to have its maximum component as the maximum admittable value. The polyhedron is then build using the polyhedra class of `SageMath`, and then its volume is calculated using the included method `P.volume()` as it was of no use to apply other polytope volume computation methods. This is because `SageMath`'s one uses exact calculations, if needed[1], and more than that, it is already implemented.

3.3.3 Constrained polytopes analysis

The polytopes appear as in Figure 3.6a. It is clear that the polytope in discussion has eight vertices: the origin, the three vectors made from $\mathbf{B}'$, and their combinations.



(a) 3D polytope with zero-moment constraint

(b) 3D polytope confrontation with and without constraints(with: red, without: green)

Figure 3.6: Constrained polytope on the left and combined figure of it and the unconstrained one on the right
($\alpha = 34°$, $\beta = 10°$)

Comparing it with the unconstrained polytope, in Figure 3.6b, it can be seen that the constrained volume covers much of the latter, but since the volume scales as n^3, the volume in the figure is just 66% of the unconstrained one.

[1] polyhedra's base rings documentation

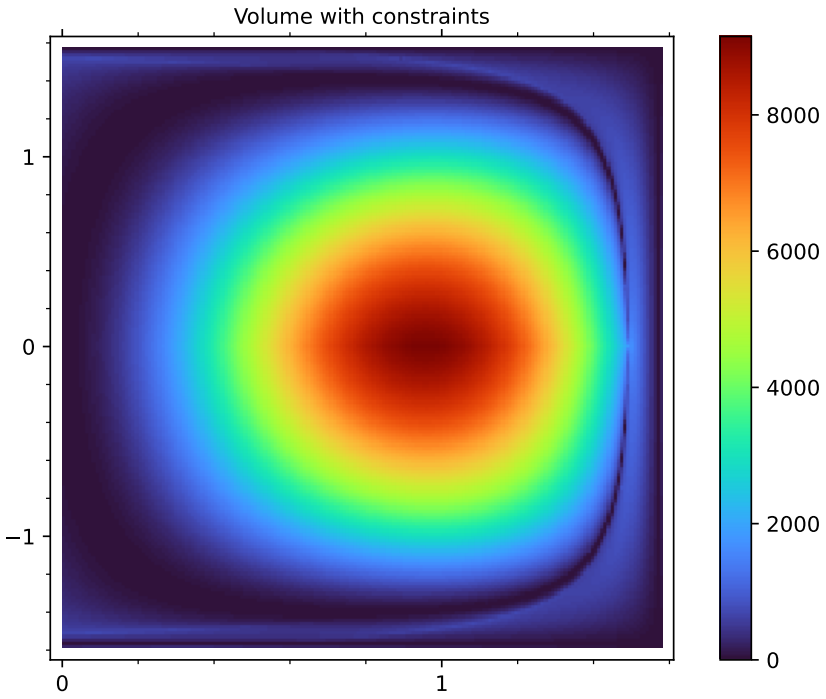


Figure 3.7: V_6 with zero-moment constraint plot
($\alpha \in [0, \pi/2], \beta \in [-pi/2, pi/2]$)

3.4 Static hovering analysis

3.4.1 Hovering feasibility

An important aspect to evaluate when dealing with the hovering problem, is how much the agent is able to move while standing still in the air. It is clear that a small angle configuration is not able to generate a lot of lateral forces whilst a UAV with a more aggressive geometry can be able to generate more downward force vectors.

We first start by defining $\boldsymbol{f}_z$ as the force necessary to be produced by the UAV to stay in hovering. This is simply equal to $\boldsymbol{f}_z = mg\hat{\boldsymbol{z}}_W$, and from the UAV model, it is also equal to $\boldsymbol{e}_3^T \mathbf{R}\mathbf{F}\boldsymbol{u}$.

Displaying this necessary force as a horizontal plane, parallel to the $x - y$ one and at a distance of $f_{z,h}$ from it, is done in Figure 3.8, where both the constrained and the unconstrained plots are used.

It is to notice that it can be easy to underestimate how, increasing α and β to increase the volume, the maximum force appliable to the vertical component diminishes, leading to the possibility of being unable to keep the agent hovering.

An example of this can be seen in Figure 3.9b, where the high α value, combined with a small β one (60° and 10°, respectively) lead the feasible forces being all below the required hovering force.

This brought up the analysis of which angles can and cannot be used for the required task.

To deal with this, the condition to satisfy is reported in (3.9), and by defining

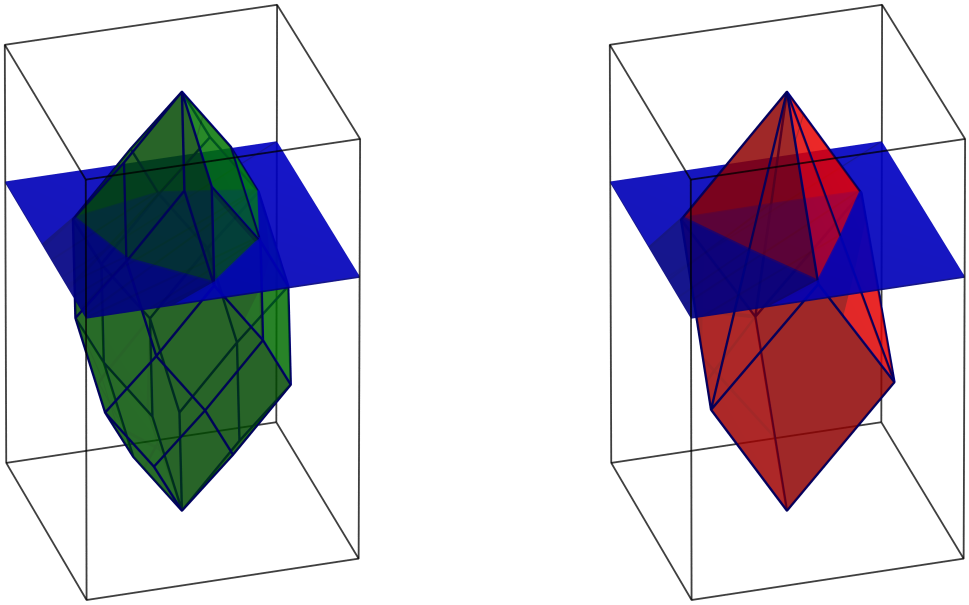
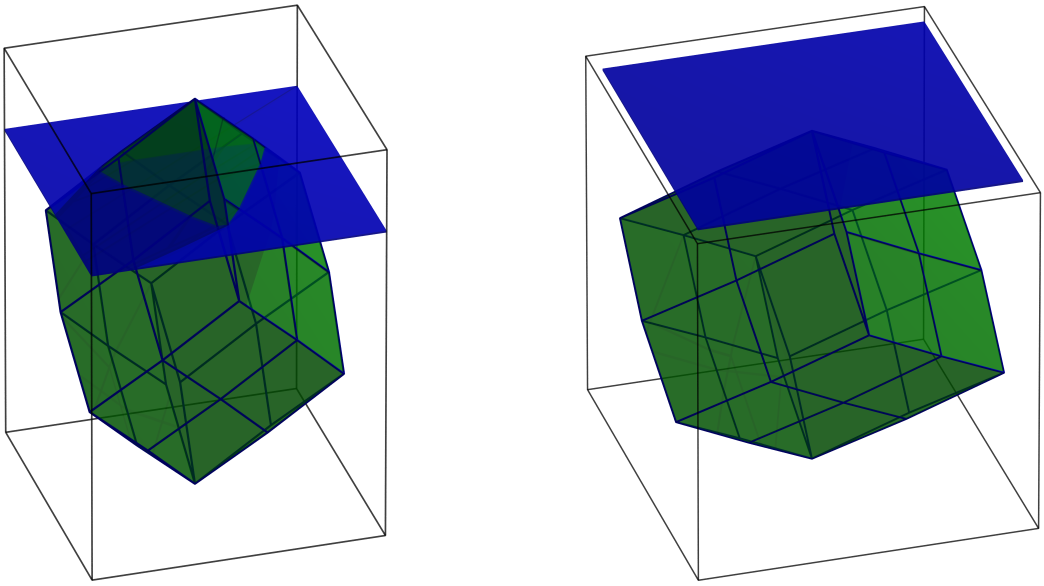


Figure 3.8: Constrained polytopes with $f_{z,h}$ spherical cap



(a) Hovering possible

(b) Hovering not possible

Figure 3.9: Constrained polytopes with $f_{z,h}$ spherical cap

$h_h = \boldsymbol{f}_{c,\max_z}$ as the hoverable height, we can rewrite the inequality as in (3.10).

$$\boldsymbol{f}_{c,\max_z}(\alpha,\beta) = e_3^T \mathbf{F}\mathbf{1}_6 \geqslant f_{z,h} \tag{3.9}$$

$$= h_h \geqslant 0 \tag{3.10}$$

Plotting the condition boundary, as in Figure 3.10 and Figure 3.11, over the graphs of the constrained and unconstrained volumes, makes clear how the results obtained in the previous sections need to be reconsidered. This is because what seemed to be an optimal configuration for the constrained volume in subsection 3.3.3, is clearly not an optimal for the hovering problem, as it is extremely close to being unable to hover.

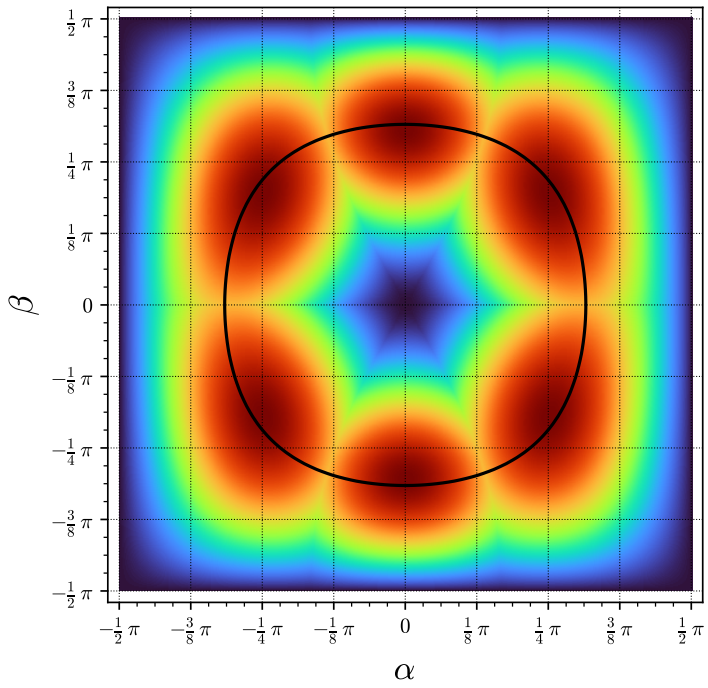


Figure 3.10: Unconstrained volume with hoverable boundary

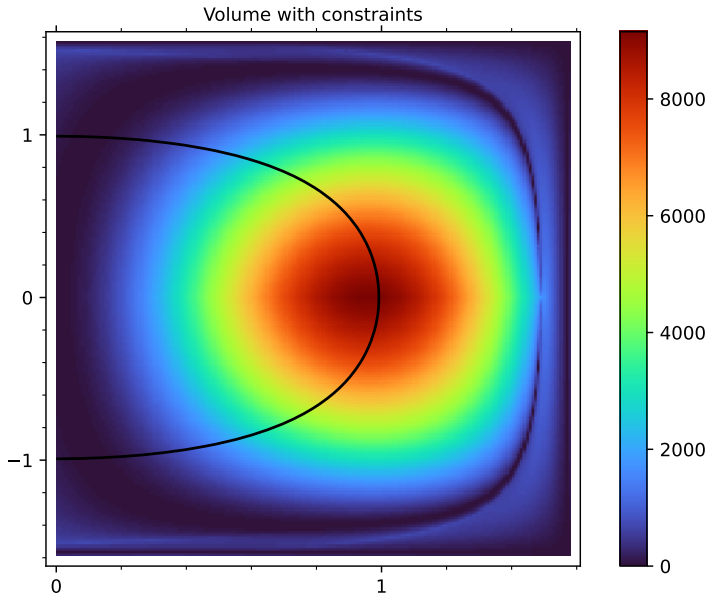


Figure 3.11: Constrained volume with hoverable boundary

In particular, in Figure 3.10, we see that the optimal point for the constrained volume would be around one radian, but the boundary imposed by the hoverability constraint shows that most of the angles values that represent are around the peak are not feasible.

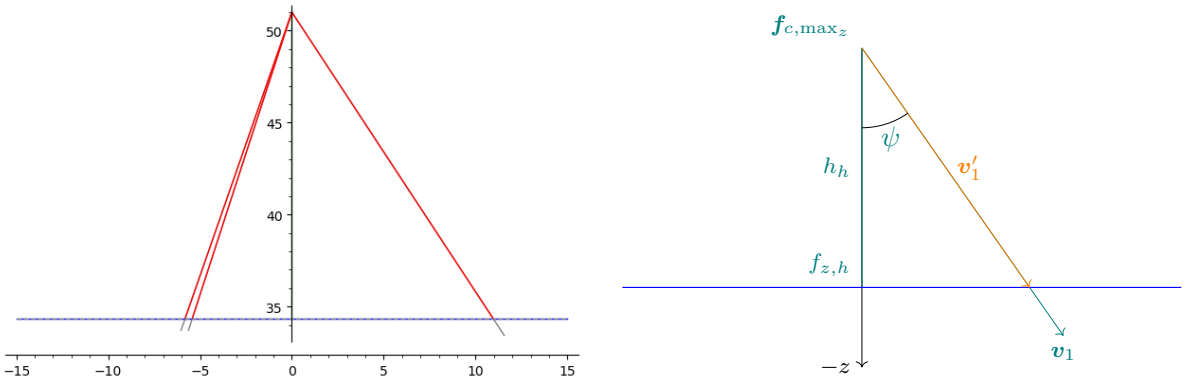
We then address the problem of finding the volume of the region over the required vertical hover force, considering only the case of the zero-moment forces.

The first step is to observe that the shape that comes from the polyhedron being cut by the $f_{z,h}$ plane, is a pyramid, which volume is reported in (3.11), with A_p being the basis of the pyramid, and h being its height.

$$V_p = \frac{1}{3} A_p h \tag{3.11}$$

The height now simply becomes the maximum total force which can be developed by the UAV , removed of $f_{z,h}$. For what concerns the base area, the intersection between the hover plane and the three top vectors is calculated.

To explain this process, a 2D projection of the scene to a plane passing through the z axis and one of the three vectors (rotated around z by one additional degree for clarity) is shown in figure Figure 3.12a.



(a) 2D projection of the constrained polytope and the hover plane

(b) 2D projection parameters

Figure 3.12: 2D projection overview and detail

First, the three top vectors are found by mirroring the three base vectors upwards, as this would give different vectors from the ones we are looking for, but with just a rotational difference around the z axis.

In particular, considering as $\boldsymbol{v}_i$ the new defined vectors, and $\boldsymbol{v}_{i,z}$ as the z component of the i-th base vector, their new z value is considered as $\boldsymbol{f}_{c,\max_z} - \boldsymbol{v}_{i,z}$, basically mirroring it vertically with respect to the vertical midplane of the polytope, as previously said. Each vector's origin is then considered to be the top vertex of the polyhedra, that is $\boldsymbol{f}_{c,\max}$, to have then a new triplet of vectors to use.

Having these new vectors, we then find their intersection with the hovering force plane by firstly finding the angle ψ between them and the vertical axes, by inverting the dot product formula in (3.12), resulting in (3.14).

$$\boldsymbol{a} \cdot \boldsymbol{b} = |\boldsymbol{a}||\boldsymbol{b}| cos(\psi) \tag{3.12}$$

$$\Downarrow$$

$$cos(\psi) = \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{|\boldsymbol{a}||\boldsymbol{b}|} \tag{3.13}$$

$$\Downarrow$$

$$\psi = \cos^{-1}\left(\frac{\boldsymbol{a} \cdot \boldsymbol{b}}{|\boldsymbol{a}||\boldsymbol{b}|}\right) \tag{3.14}$$

Then noticing that, by (3.15), where v_1 is the first top vector, h_h is the height of the hoverable volume pyramid, and we can resort to (3.17) to calculate the length of the vector going from the top vertex to the intersection.

$$h_h = |\boldsymbol{v}_1'| \cos(\psi) \tag{3.15}$$

$$\Downarrow$$

$$|\boldsymbol{v}_1'| = \frac{h_h}{\cos(\psi)} \tag{3.16}$$

$$\Downarrow$$

$$\boldsymbol{v}_1' = |\boldsymbol{v}_1'|\hat{\boldsymbol{v}}_1 = \frac{h_h}{\cos(\psi)}\hat{\boldsymbol{v}}_1 \tag{3.17}$$

A representation of these parameters is displayed in Figure 3.12b.

Now a situation that can happen and that would lead to a major volume overestimate will be addressed. This is when condition (3.18) is violated, resulting in an arrangement as in Figure 3.13.

$$h_\Delta = h_h - v_{i,z} \leqslant 0 \tag{3.18}$$

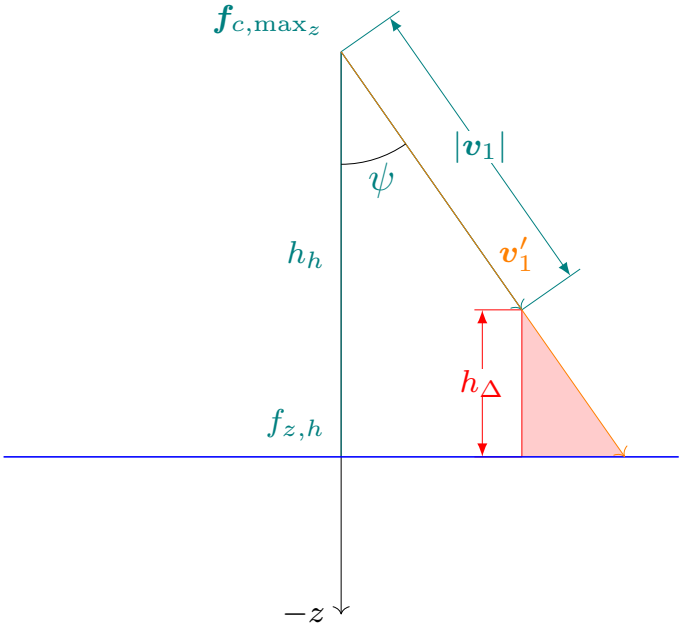


Figure 3.13: 2D projection parameters, in the case of excessive elongation of the vector $\boldsymbol{v}_1$

Using the same volume formula as before, for this new case would include the extension of the vectors over their maximum length, resulting in an obvious unfeasibility of the resulting vector. To cope with this, the approach that have been used is still an approximated one, but with a lower error.

The chosen approach was to basically check if the condition in (3.18) has been violated, and in that case, to clamp the length of the vector, increasing then the missing volume portion by just using $V_p' = A_b h_\Delta$.

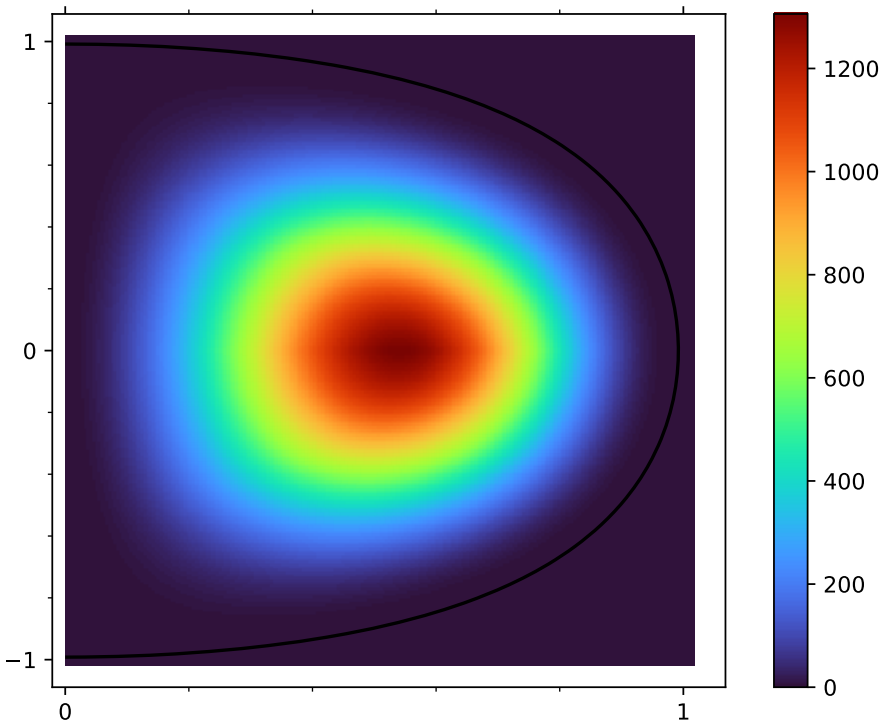
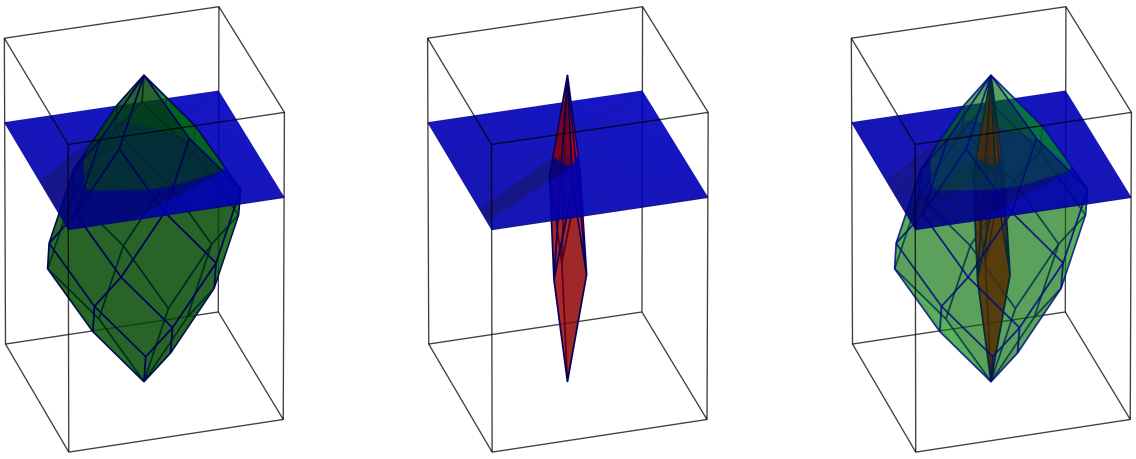


Figure 3.14: Hoverable volume with limit indicated by the black contour

This lead to a volume like in Figure 3.14. It is clear from the values in the colorbar placed right of the plot, that its value is lesser than the unconstrained one.

It is also interesting to investigate the resulting constrained with a low α value. This creates unconstrained polytopes as in Figure 3.15a, very similar to the configuration with a bigger α, but the constrained polytope, as in Figure 3.15b is very narrow.



(a) Unconstrained polytopes (b) Constrained polytopes (c) Combined polytope s

Figure 3.15: Polytopes with $\alpha = 10°$ and $\beta = 40°$

Comparing them in Figure 3.15c, shows how the constrained volume is very small with respect to all the unconstrained volume. All of these considerations lead to the conclusion that the β angle has not a lot of use for the constrained volume, but this parameter can

still useful in the unconstrained volume maximization problem, though having it small may compromise the platform robustness, as already said.

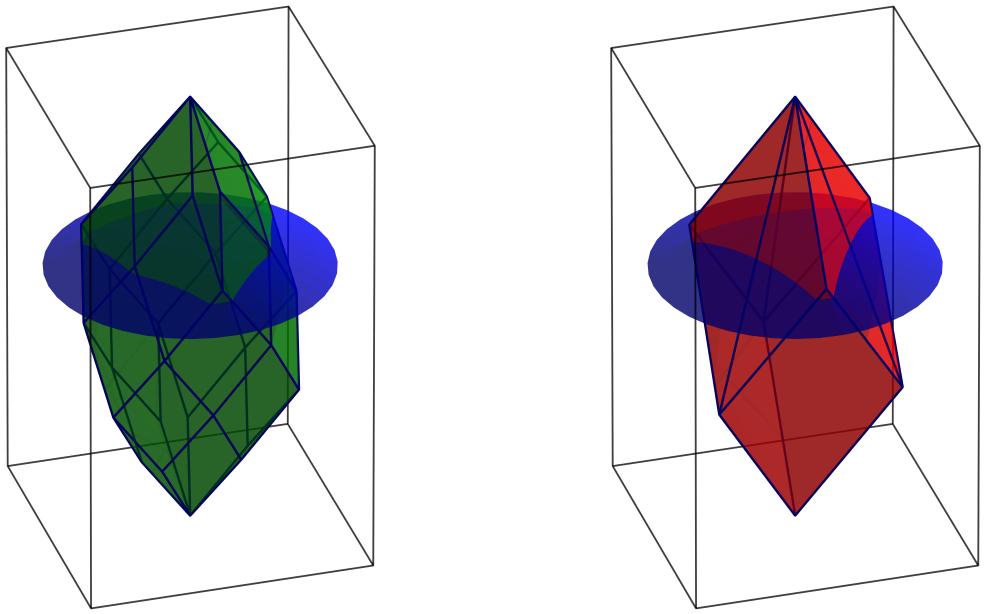
3.4.2 Maximum angle attainable

The last property of the UAV that has been studied in this thesis, for what concerns the static analysis, is the ability to keep hovering while not being parallel to the ground, but instead being tilted.

This could be of interest of for other purposes, as it would empower the agent be able to, for example, touch a vertical surface with the aid of a bar attached to the bottom, or to be able to land and take off from a non-horizontal surface, by modifying the force along $\boldsymbol{z}_W$.

To address this, in this section the problem of finding the maximum inclination angle in hovering is studied.

First of all, we notice that the study done in the previous section has to be modified, as we are not interested any more in a plane, but instead we now have a spherical cap as in Figure 3.16, centred at $\mathbf{0}$ and with radius $f_{z,h}$. This is because to keep hovering while tilted, the required vector will have different direction than $\boldsymbol{z}_B$, but its magnitude remains the same, by just being perpendicular to the ground.

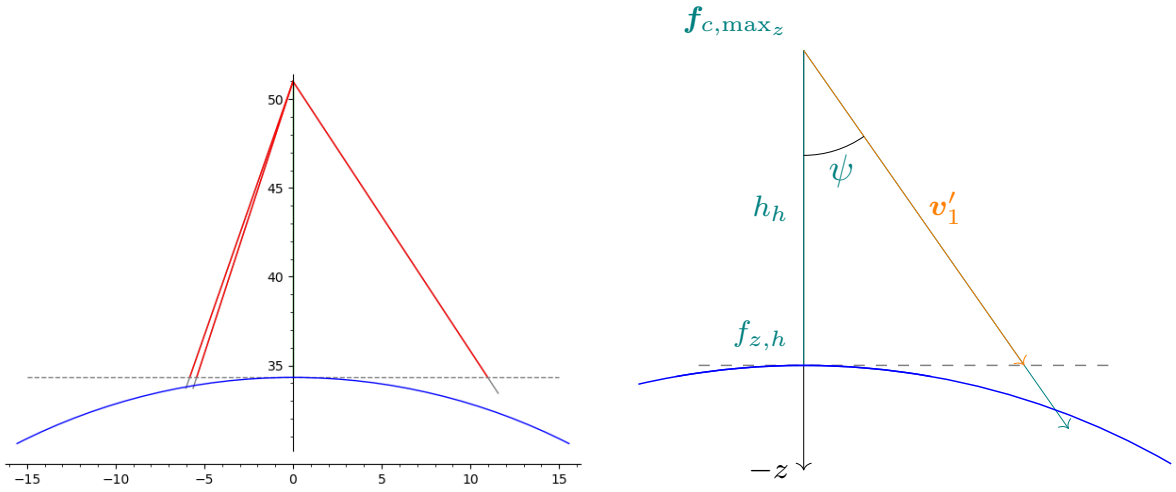


(a) Unconstrained polytope (b) Constrained polytope

Figure 3.16: Polytopes with $f_{z,h}$ spherical cap

We than have that the projection in Figure 3.12a becomes the one in Figure 3.17a, and we see that the previous formula would underestimate the new volume, but this issue has not being addressed, as it was out of scope for the current analysis.

We can see then, that the force vectors generating no moment that can be applied as a control force, lie in one of the three intersection between the top vectors and the circular cap, as they are the point both furthest from the central axis and with the smallest vertical component of the whole volume. This fact let us simply modify Figure 3.12 into Figure 3.17, letting us notice that now the volume would not include the space between the dome and the previously defined plane.



(a) 2D projection of the constrained polytope and the hover plane

(b) 2D projection parameters

Figure 3.17: 2D projection overview and detail, with circular force boundary

Having adapted the previous framework to the current problem, we define the new intersection between vector $\boldsymbol{v}_1$ and the arc as $\boldsymbol{v}_1''$. Then the angle that we are interested in, that will be called δ, is the angle between that point and the vertical axis. All these parameters are shown graphically in Figure 3.17b.

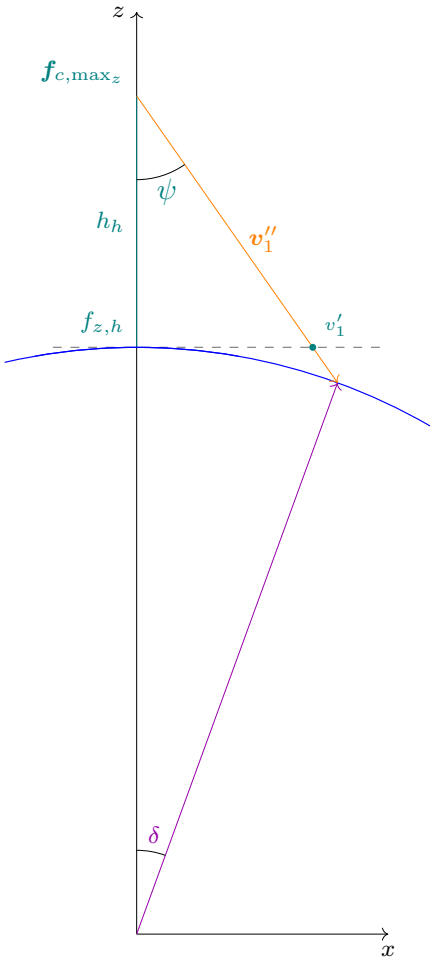


Figure 3.18: 2D projection parameters

To find the value that we are interested in, even though possible, we do not use the exact computation to find the true vector $\boldsymbol{v}_1''$, but we use the previous one.

This has a small impact on the values found, but it is used nonetheless as the resulting angle will not be overestimated, so in the worst case scenario, the UAV will have some control that will be able to apply around that point to adjust to small variations.

We can then see that the hoverable volume plot, seen in the previous section (Figure 3.14), seems strongly correlated with the maximum inclination angle, even though this is expected. In particular, the only noticeable differences comparing the new plot in

Figure 3.19, are now discussed. First we can see that the hoverable volume peak has a smooth shape, whereas the maximum angle plot has a neat path where the angle has a local maximum. We can also see that the angle plot around the contour representing the hoverable limit, has still a meaningful value until the very proximity to it, in opposition to the volume plot, that collapses more rapidly. The last prominent difference is that the peak is shifted in the α axis towards 40°, whilst previously was located around 28°.

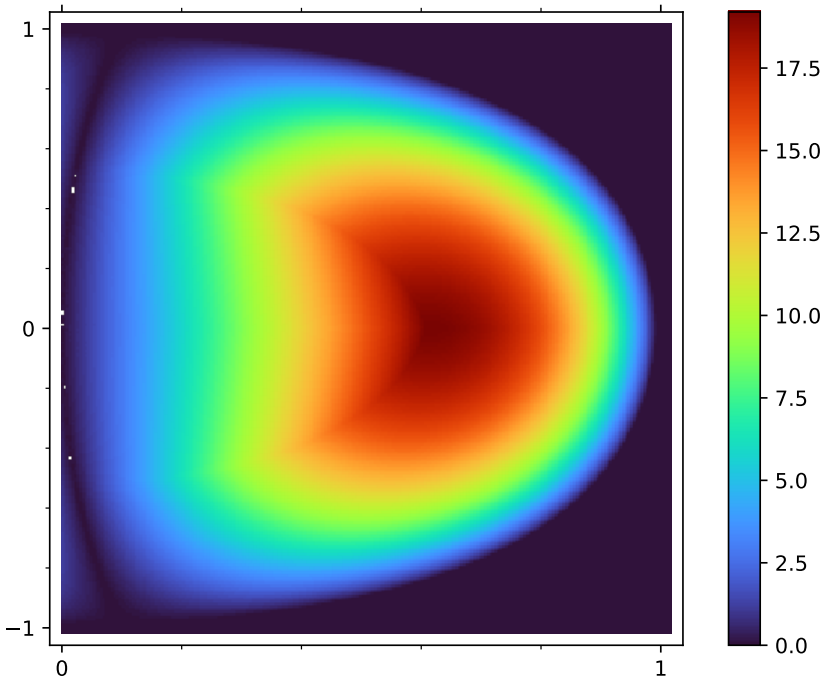


Figure 3.19: Maximum angle of inclination attainable varying α and β

In light of these facts, the best angles with respect both to the hovering problem and in general the control under zero-moment constraints, appears to be then with β sufficiently small to stay close to the peaks, while still being greater than 0 to keep the robustness to faults. Moving on to the α angle instead, the optimal values possibly are between from 0.5 rad to 0.7 rad, that is from 28.6° to 40°, resulting in angle to balance between stability and manoeuvrability of the UAV.

4 Dynamic Analysis

In this Chapter we investigate the dynamic behaviour of the UAV. There exists controllers already exploiting the zero moment components [1, 13], but since this thesis focus is studying the fundamental properties of a nonlinear MPC has been applied, since it was ready to be used.

4.1 MPC

4.1.1 MPC Description

The MPC is chosen because it can predict the system behaviour, exploiting the plant model inserted in the controller.

On the basis of the system prediction, the MPC evaluates a control input, which is optimal with respect to a set function.

The MPC is characterized by three main elements, which are:

- an internal system model, that is exploited to predict the system variables evolution;
- a cost function that has to be minimized (e.g. distance of a variable from the given reference);
- a set of constraints to be satisfied that can be of different type (e.g., linear, quadratic, etc.)

The modelization of the system to control plays a fundamental part in order to obtain a satisfying control law.

To make prediction close to the real system evolution, a deep analysis on the process has to be carried out, being cautious about the fact that a model that is excessively complex can lead to an expensive control law in terms of computational effort.

Therefore, the model inserted in MPC has to be a compromise between complexity and efficiency.

The principal advantage of MPC is its robustness and its ability to handle constraints. For example, some bounds can be set on the system variables in order to describe a real problem, like $u_i \geqslant 0$.: All of these constraints can be hard or soft, depending on how they are defined. In general, the hard constraints describe physical limitations related to the input actuators, whereas the soft ones are used to describe desired system behaviours. Therefore, hard constraints are imposed on input variables, while the output and states are characterized by the soft ones.

The Model Predictive Control implements the Receding Horizon strategy. This means that at each instant, the control follows three steps.

1. It predicts the system evolution, in terms of states and outputs variables;
2. it evaluates a control sequence that minimizes an objective function, and then
3. it applies only the first element of the control sequence.

The procedure just depicted is schematized in Figure 4.1.

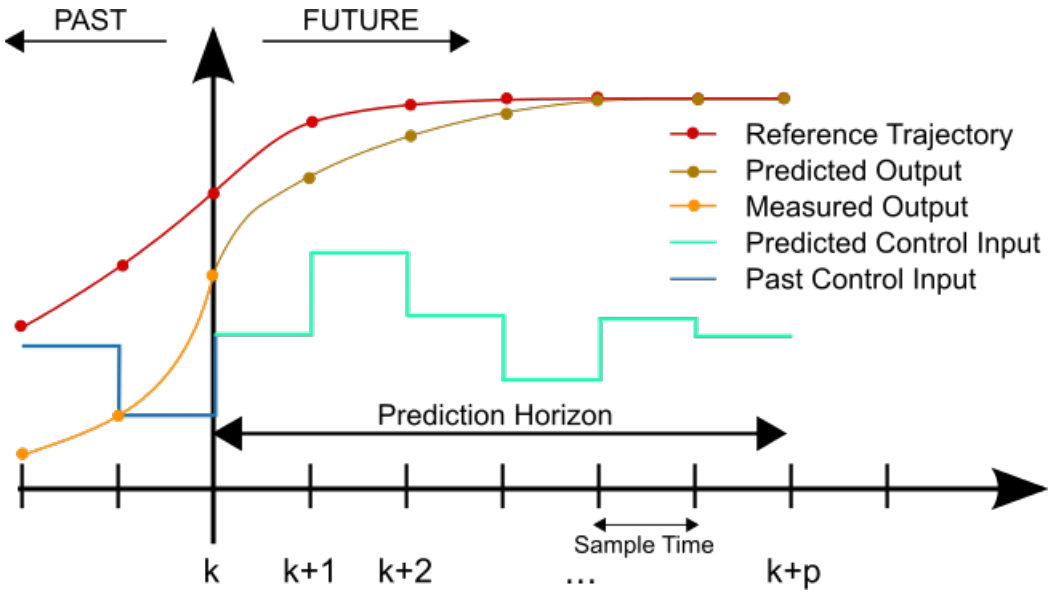


Figure 4.1: MPC working procedure (Source: Wikipedia)

At each sample time the procedure is repeated, but neglecting the other control input values obtained before and not used to control the system.

The input control sequence computed by the controller is the result of the MPC capability to predict the future system evolution. The ability of looking the plant future behaviour is defined by a parameter called prediction horizon, in Figure 4.1 is noted with letter p. It is a parameter set in the design control phase, and it is chosen as compromise between the computational limits and the improvement of the optimal control input obtained from the controller.

In this Figure, it can be noticed how MPC solves the optimization problem using all the entire control sequence computed at time k to reach the desired trajectory; but then only the first value of the control sequence is applied to the system.

Summarizing, at each samples, MPC has to solve the following optimization problem:

$$\min_{\boldsymbol{u}(\cdot)} \boldsymbol{M}(\boldsymbol{x}(t_f)) + \int_{t_0}^{t_f} \boldsymbol{L}(\boldsymbol{x}(t), \boldsymbol{u}(t))dt \tag{4.1}$$

$$\text{such that} \quad \begin{cases} \boldsymbol{x}(t_0) = \hat{\boldsymbol{x}}_0 \\ \dot{\boldsymbol{x}}(t) = F((\boldsymbol{x}(t), \boldsymbol{u}(t)), t \in [t_0, t_f] \\ H((\boldsymbol{x}(t), \boldsymbol{u}(t)) \leq 0, t \in [t_0, t_f] \end{cases} \tag{4.2}$$

Where $\boldsymbol{L}$ represents the cost function, t_0 is the initial time (the actual time instant), while t_f is the final sample of the prediction horizon. The final stage has a separated treatment, indeed $\boldsymbol{M}$ is the cost function for the final instant. In (4.2) the initial condition, the state evolution and the nonlinear limitations are noted.

4.1.2 NMPC

In case of a nonlinear model dynamics, the MPC can be adapted, thus requiring a non-linear optimization problem to be solved. This is addressed by using various solvers, ranging from Active Set solvers to Interior Point Method solvers. The non-linearity of the problem now poses also the issue of the local and global minimums. Indeed, under the non-linearity condition, the critical point is not generally guaranteed to be also a global one.

4.1.3 Model implementation

The mathematical model is written in continuous time, and it is then discretized by integration through the explicit Runge-Kutta discretization method. In particular, the two model used, are shown in (4.5) and (4.7), both sharing the system state that is as in (4.3).

$$\boldsymbol{x} = \left[\boldsymbol{p}, \boldsymbol{\alpha}, \boldsymbol{v}, \boldsymbol{\omega}\right]^T \tag{4.3}$$

With $\boldsymbol{v}$ as the spatial velocity, being then $\dot{\boldsymbol{p}}$.

$$\dot{\boldsymbol{x}} = \boldsymbol{f}_1(\boldsymbol{x}) + \boldsymbol{g}_1(\boldsymbol{x}, \boldsymbol{u}) \tag{4.4}$$

$$\begin{bmatrix} \dot{\boldsymbol{p}} \\ \dot{\boldsymbol{\alpha}} \\ \ddot{\boldsymbol{p}} \\ \dot{\boldsymbol{\omega}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{W}^{-1}\boldsymbol{\omega} \\ -g\boldsymbol{e}_3 \\ -\boldsymbol{J}(\boldsymbol{\omega} \times \boldsymbol{J}\boldsymbol{\omega}) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ m^{-1}\mathbf{RF} \\ \boldsymbol{J}^{-1}\mathbf{M} \end{bmatrix} \boldsymbol{u} \tag{4.5}$$

$$\dot{\boldsymbol{x}} = \boldsymbol{f}_2(\boldsymbol{x}) + \boldsymbol{g}_2(\boldsymbol{x}, \boldsymbol{u}) \tag{4.6}$$

$$\begin{bmatrix} \dot{\boldsymbol{p}} \\ \dot{\boldsymbol{\alpha}} \\ \ddot{\boldsymbol{p}} \\ \dot{\boldsymbol{\omega}} \end{bmatrix} = \boldsymbol{f}_1(\boldsymbol{x}) + \begin{bmatrix} 0 \\ 0 \\ m^{-1}\mathbf{RF} \\ \boldsymbol{J}^{-1}\mathbf{M} \end{bmatrix} \mathbf{T} \begin{bmatrix} \tilde{\boldsymbol{u}_A} \\ \tilde{\boldsymbol{u}_B} \end{bmatrix} \tag{4.7}$$

The model just described contains the inverse of the matrix $\boldsymbol{W}$, that comes from subsection 2.2.2, to correctly integrate the Euler angles. Aside from this, the first model directly derives from (2.33)-(2.36), whereas the second one uses the input $\tilde{\boldsymbol{u}}$.

These models require different constraint sets, as the first requires a soft constraint to force the zero-moment direction, while the second one instead embeds that constraint directly on the input. To limit the input instead, the first model can take the constraint as a direct hard constraint on the input, bounding it in the feasible range of the motor squared speed. These constraints are shown in detail in Table 4.1

Model	Input constraints	Soft constraints
Model 1	$0 \leqslant \boldsymbol{u}_i \leqslant u_{\max}$	$\lvert\mathbf{B}\boldsymbol{u}\rvert \leqslant \varepsilon$
Model 2	$\tilde{\boldsymbol{u}_A} = \mathbf{0}$	$-\mathbf{T}\tilde{\boldsymbol{u}} \leqslant 0$ $\mathbf{T}\tilde{\boldsymbol{u}} - u_{\max} \leqslant 0$

Table 4.1: MPC Constraints for each model

Both models contain a cost function like the one in (4.10), where the parameters that are involved in are the $\boldsymbol{h}$ function, defined as in (4.8) and in the matrix $\boldsymbol{Q}$ as in (4.9), with each matrix used inside that represents the weight matrix for the respective term in (4.8).

$$\boldsymbol{h}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{p} - \boldsymbol{p}_{\text{ref}} \\ \boldsymbol{\alpha} - \boldsymbol{\alpha}_{\text{ref}} \\ \dot{\boldsymbol{p}} \\ \dot{\boldsymbol{\alpha}} \\ \boldsymbol{u} - \boldsymbol{u}_{\text{hover}} \end{bmatrix} \qquad \boldsymbol{h}_f(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{p} - \boldsymbol{p}_{\text{ref}} \\ \boldsymbol{\alpha} - \boldsymbol{\alpha}_{\text{ref}} \\ \dot{\boldsymbol{p}} \\ \dot{\boldsymbol{\alpha}} \end{bmatrix} \tag{4.8}$$

$$\boldsymbol{Q} = \text{diag}\left[\boldsymbol{Q}_p, \boldsymbol{Q}_\alpha, \boldsymbol{Q}_v, \boldsymbol{Q}_{\dot{alpha}}, \boldsymbol{Q}_u\right] \qquad \boldsymbol{Q}_f = \text{diag}\left[\boldsymbol{Q}_p, \boldsymbol{Q}_\alpha, \boldsymbol{Q}_v, \boldsymbol{Q}_{\dot{alpha}}\right] \tag{4.9}$$

$$\boldsymbol{L} = \frac{1}{2}\boldsymbol{h}^T\boldsymbol{Q}\boldsymbol{h} \qquad \boldsymbol{L}_f = \boldsymbol{M} = \frac{1}{2}\boldsymbol{h}_f^T\boldsymbol{Q}_F\boldsymbol{h}_f \tag{4.10}$$

4.2 Simulations

4.2.1 Tilted Hovering Simulations

After all the analysis done, some simulations have been carried out to test the effective ability of the UAV to maintain the tilted hovering under the optimum configurations.

These simulations have been done using an MPC controller to bring the agent from the horizontal hovering position to the final point of the desired static hovering test.

Tilted reference generation

To test our hypothesis, we firstly have to define the correct references to pass to the MPC controller. We first start by analysing the projection on the $x-y$ plane of the three vertices belonging to the top vectors of the constrained polyhedron, namely the one used in subsection 3.4.1, in particular in Figure 3.17.

Noticing than that one of them lies along the y axis, and in particular in the negative part of it. This has then also been verified also symbolically, empowering us to just assign the ϕ angle with a negative value, to place the reference $\boldsymbol{z}_B$ vector in the same direction.

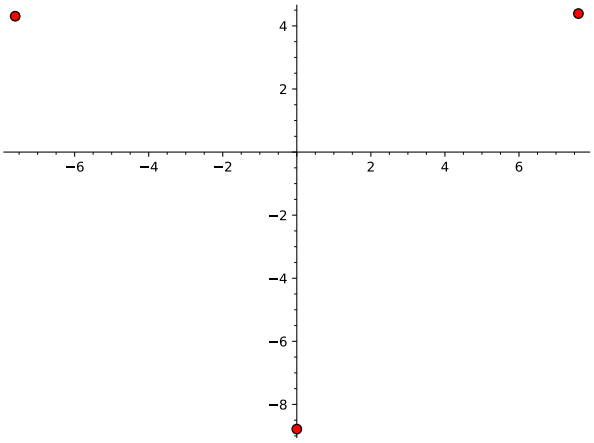


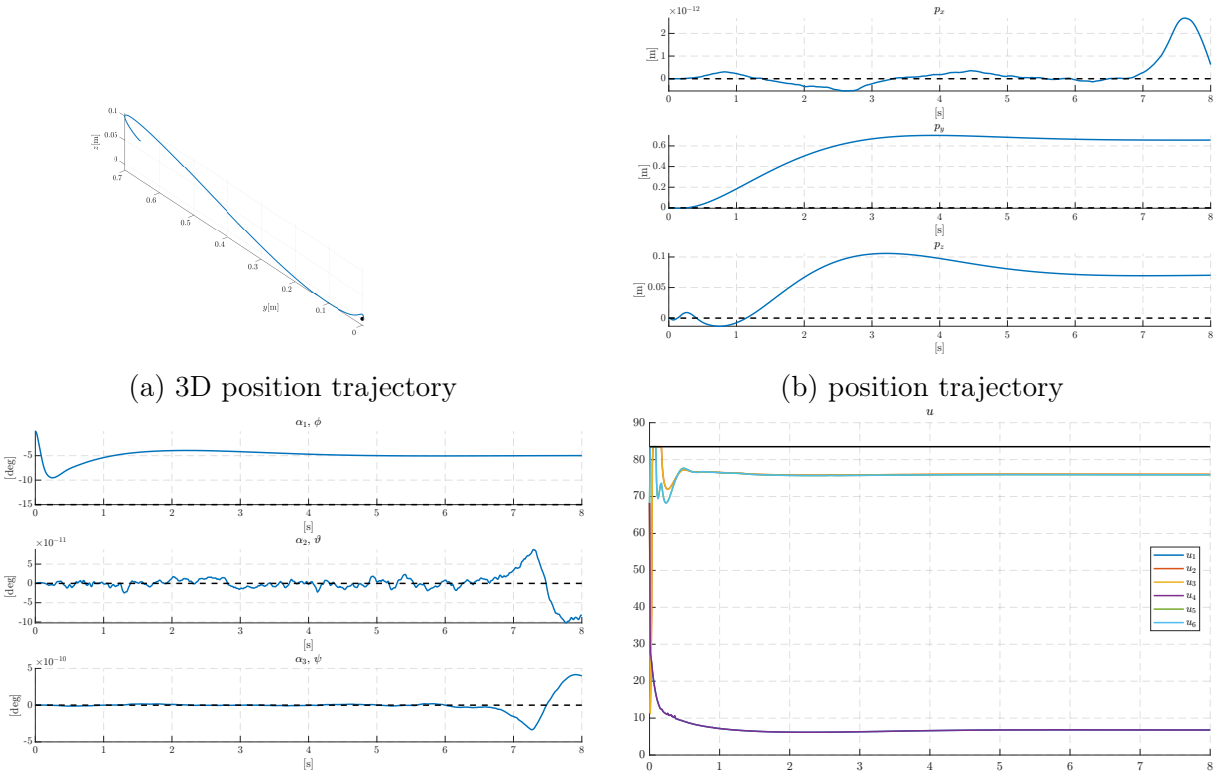
Figure 4.2: Top vertices constrained polyhedron $x-y$ displacement

Noiseless simulations

Firstly some simulation in ideal conditions have been carried out to test the effectiveness of the MPC controller implementation, to validate the hypothesis done in the angle attainable only for the theoretical work done.

This started by setting the reference position to be zero, as we are only interested in the angular behaviour.

We can see that the agent is capable of correctly stabilize itself with 15 degrees of rotation around the x axis. The inclination has been introduced by setting $\boldsymbol{\alpha}_{\text{ref}}$ to have the resulting desired $\boldsymbol{z}_B$ vector in the same direction as one of the three vertices of the hoverable volume.



(a) 3D position trajectory

(b) position trajectory

(c) angles trajectory

(d) inputs trajectory

Figure 4.3: simulation with $\phi_{ref} : -15°, \alpha : 10°, \beta : 2°$

This is done by projecting in 2D these points, in the $x - y$ plane, as explained in section 4.2.1.

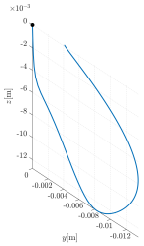
The simulation with this parameter is done in Figure 4.4 and Figure 4.3, with the inclination angle set at 15 degrees, and only the construction parameters changing between the two. By observing in particular the positions in Figure 4.3a and Figure 4.4a, and the rotations in Figure 4.3c and Figure 4.4c, it is evident that they get stabilized.

Furthermore, the input assumes values that let us have a significant range of control over it (except for the initial peaks), making possible to, for example, counteract noise and model uncertainties.

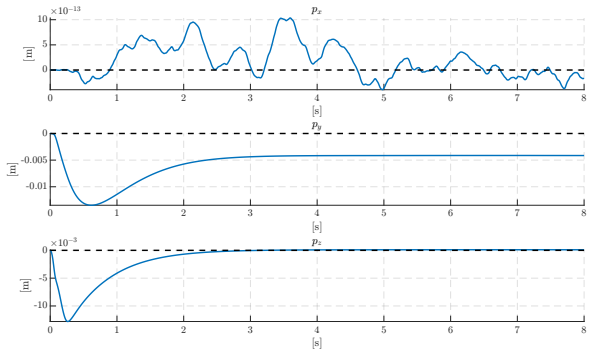
On top of that, it is interesting to see how the six inputs group to form three main pairs, namely (u_1, u_4), (u_2, u_3), (u_5, u_6). This behaviour is observed as soon as the tolerance applied to the soft constraint in Table 4.1 rises. This also means that even if this behaviour does not imply the input belonging to the $\mathbf{M}$ kernel, as clearly in the initial simulation instants the multirotor apply torque to rotate itself, it is clearly correlated.

Assigning the same inclination but with the opposite sign, as in Figure 4.5, shows how the input pairs now change roles, making the $u_1 - u_4$ pair to become dominant.

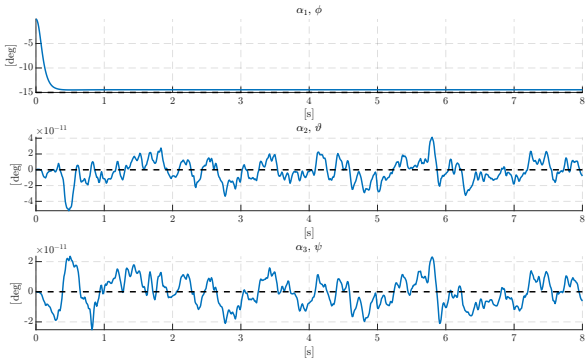
Nonetheless, the force vector is not feasible any more. This can be seen directly by observing Figure 3.16b: if the desired vector is allocated along one of the three top vertices, inverting its angle means rotating the force vector around the z axis by pi radians. It is then easy to see that by applying that rotation to a desired force vector close to one of those vertices easily trespass the polyhedron boundary, thus becoming unfeasible. The



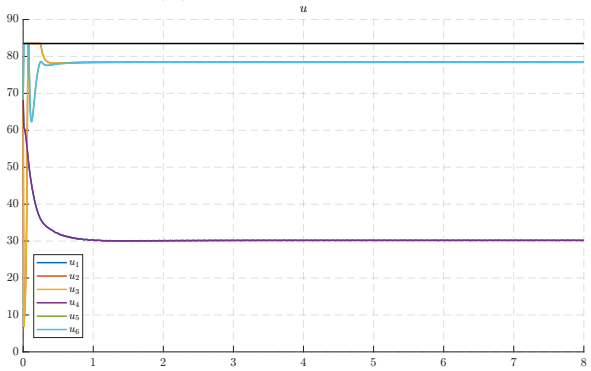
(a) 3D position trajectory



(b) position trajectory



(c) angles trajectory

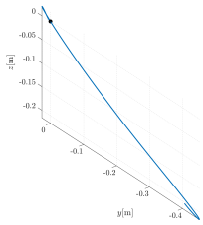


(d) inputs trajectory

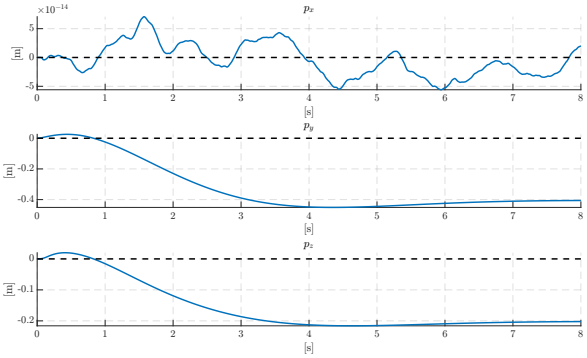
Figure 4.4: simulation with $\phi_{ref} : -15°, \alpha : 33°, \beta : 2°$

other reasoning that can lead to the unfeasibility conclusion, is that the vector in the first simulation is generated mainly by the input pairs (u_2, u_3), (u_5, u_6), where the contribution of the rotors one and four is not entirely used. This though, is expected, as these propellers lies along the x axis, therefore being the least effective in creating forces along it, as they are tilted. Following the same reasoning, it is possible to notice in Figure 4.5 that the inputs roles are inverted. Indeed, now the pair that previously was greatly unused, is subjected to an excessive load request, leading to its saturation in the first instants of the simulation. This means that the input required to stabilize the reference vector is not feasible with the current angle configuration.

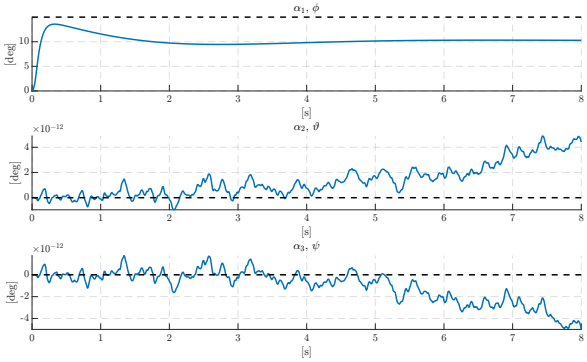
It is noticeable that the parameters in Figure 4.3a, that are $\phi_{ref} = -15°, \alpha = 10°$ and $\beta = 10°$ should not be able to maintain a stable configuration, based on Figure 3.19. This in fact could explain why the controller is not able to get to more than 5 degrees of tilt, as it can be seen in Figure 4.3c.



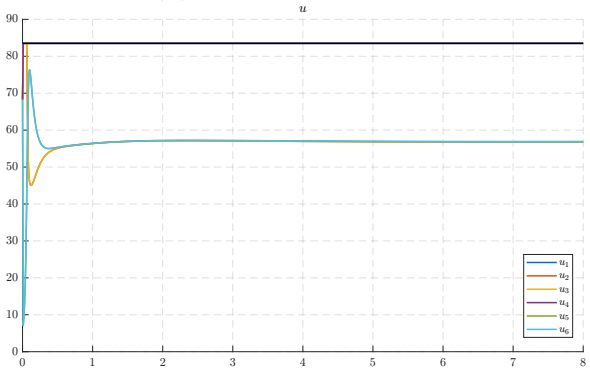
(a) 3D position trajectory



(b) position trajectory



(c) angles trajectory



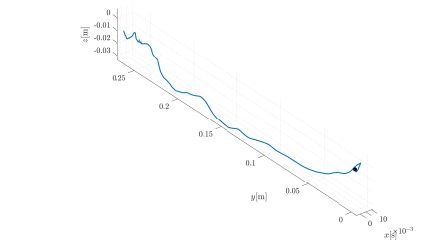
(d) inputs trajectory

Figure 4.5: simulation with $\phi_{ref} : 15°, \alpha : 33°, \beta : 2°$

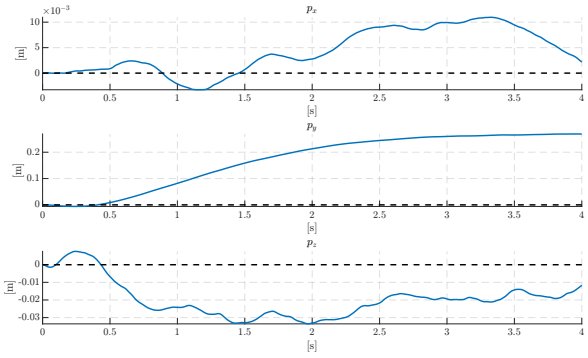
4.2.2 Realistic simulations

To improve the realistic features of the simulations, some random noise has been added to the UAV in the form of a force applied to the agent model at each simulation instant.

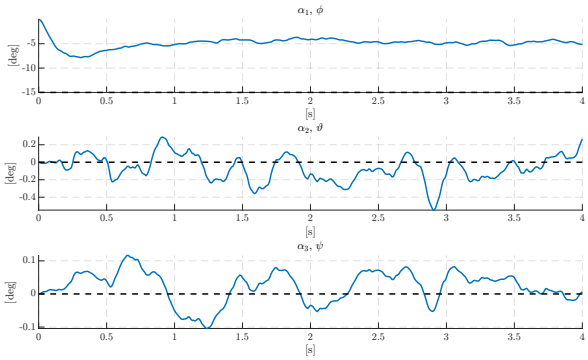
This proved the robustness of the controller, being able to counteract the noise and keeping the agent under control.



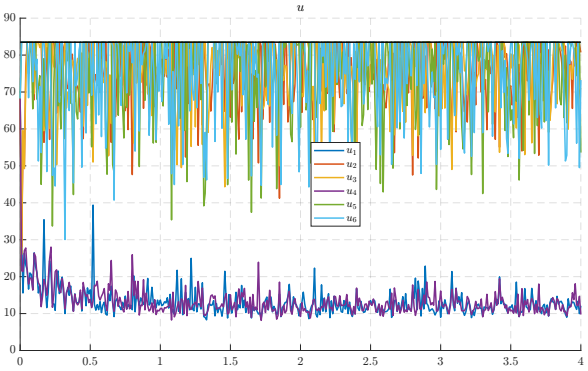
(a) 3D position trajectory



(b) position trajectory



(c) angles trajectory

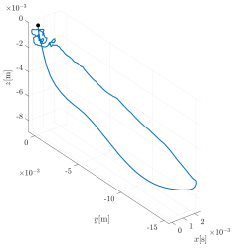


(d) inputs trajectory

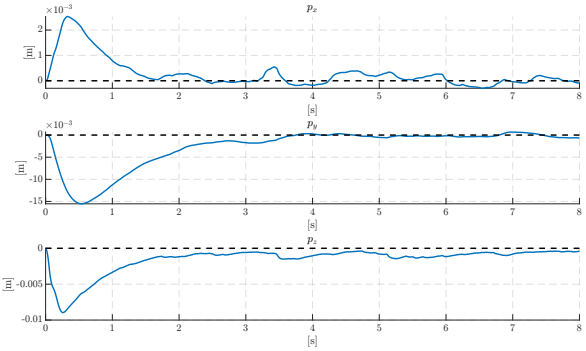
Figure 4.6: simulation with $\phi_{ref} : -15°, \alpha : 10°, \beta : 2_n1°$

These simulations proved to be comparable to the ideal ones, even considered the large amount of noise. Nevertheless, the impact of this disturbance is very noticeable in the inputs, and in Figure 4.7, in particular in Figure 4.7c and Figure 4.7a, how the order of magnitude of the signals change. The other aspect to consider is that in this plot, the input values tend to saturate a lot, due to the disturbance. It seems though, that the controller is able to overcome these saturations. In simulations shown in Figure 4.6 and Figure 4.8, this phenomenon is even more evident, leading to even more noise in the control inputs, as in these configurations it is more difficult to control the UAV due to the possibly unfeasible reference.

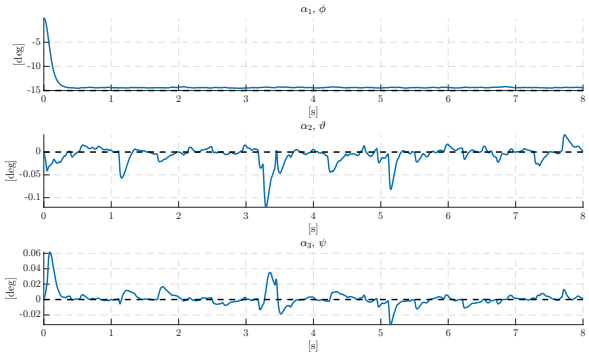
The last observation regards the fact that while in the last two mentioned simulations the noise is noticeable, the configuration with an α value of 33° in Figure 4.8, is still able to perform comparably to its noiseless counterpart in Figure 4.5, proving that indeed it is still more robust with respect to the configuration with $\alpha = 10°$.



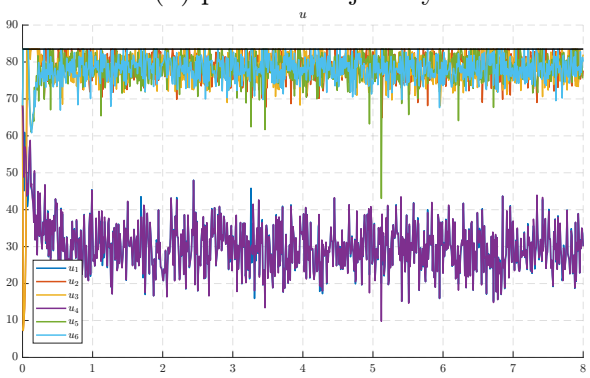
(a) 3D position trajectory



(b) position trajectory

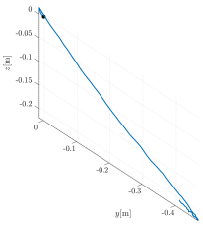


(c) angles trajectory

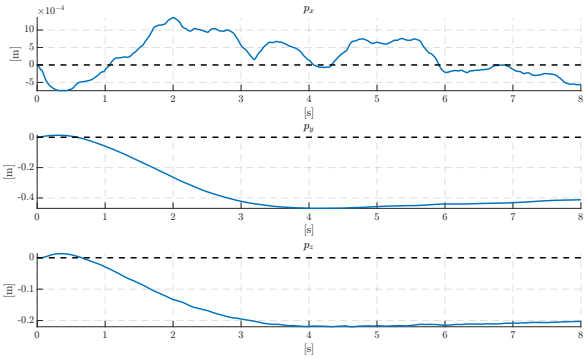


(d) inputs trajectory

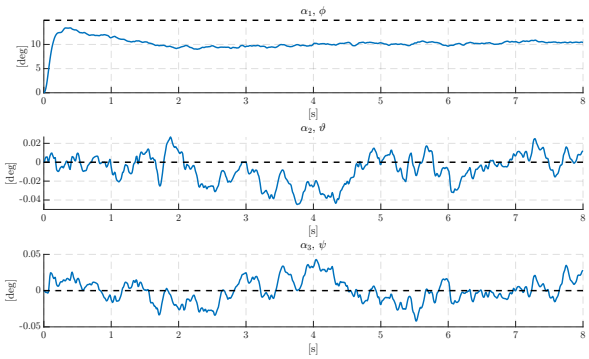
Figure 4.7: simulation with $\phi_{ref} : -15°, \alpha : 33°, \beta : 2_n1°$



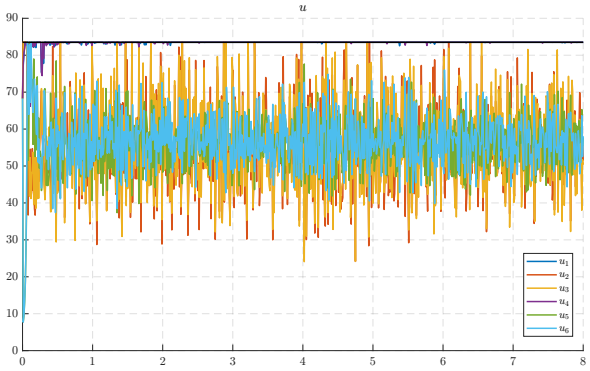
(a) 3D position trajectory



(b) position trajectory



(c) angles trajectory



(d) inputs trajectory

Figure 4.8: simulation with $\phi_{ref} : 15°, \alpha : 33°, \beta : 2_n1°$

4.2.3 Movement Simulations

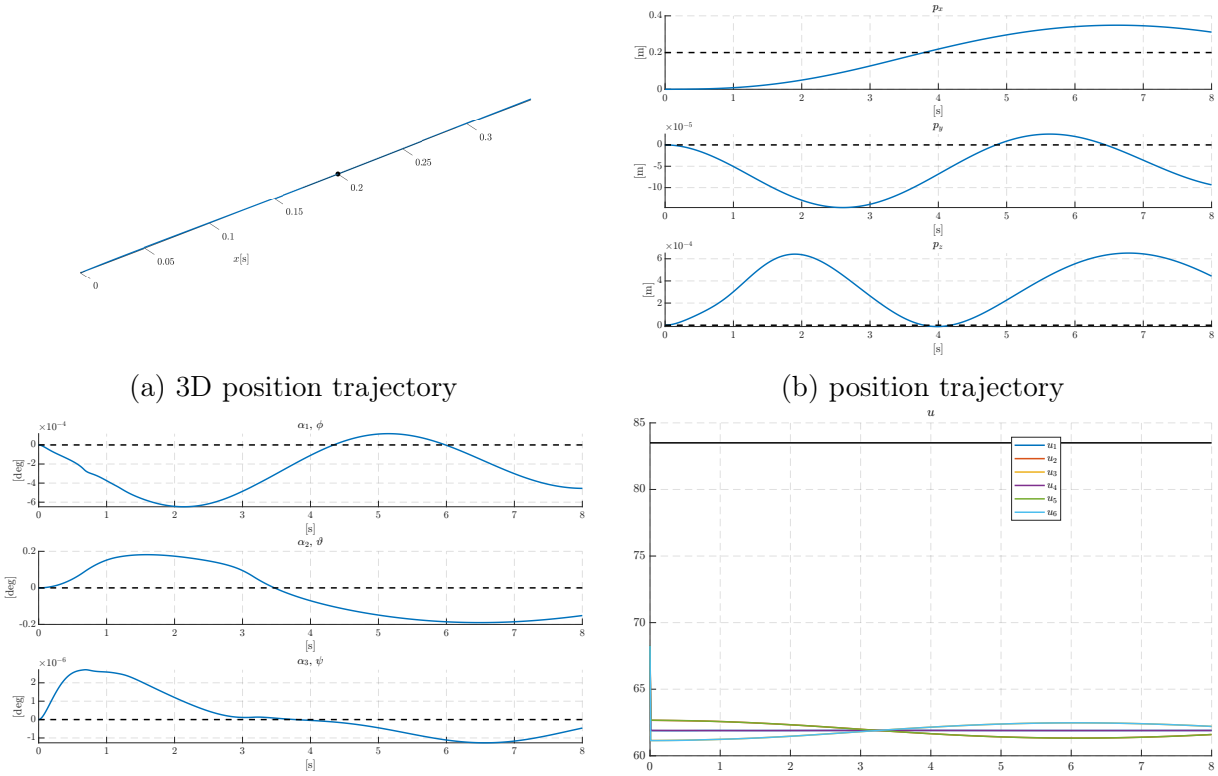
In this section, simulation with various parameters are shown, to validate the results discussed in the rest of the thesis.

In particular, three simulations are made at first, setting as reference zero in all three components of the α vector, and by setting a constant position reference in one of the components of the position. These simulations have been made to study the behaviour of the agent in the three canonical directions, verifying its robustness and behaviour depending on the geometric parameters.

4.2.4 Three axis simulations

It is of interest to perform the simulations with the reference position being along one of the Cartesian axis, as this can aid in the process of better understanding the decoupled dynamic of the agent.

x-axis



(a) 3D position trajectory

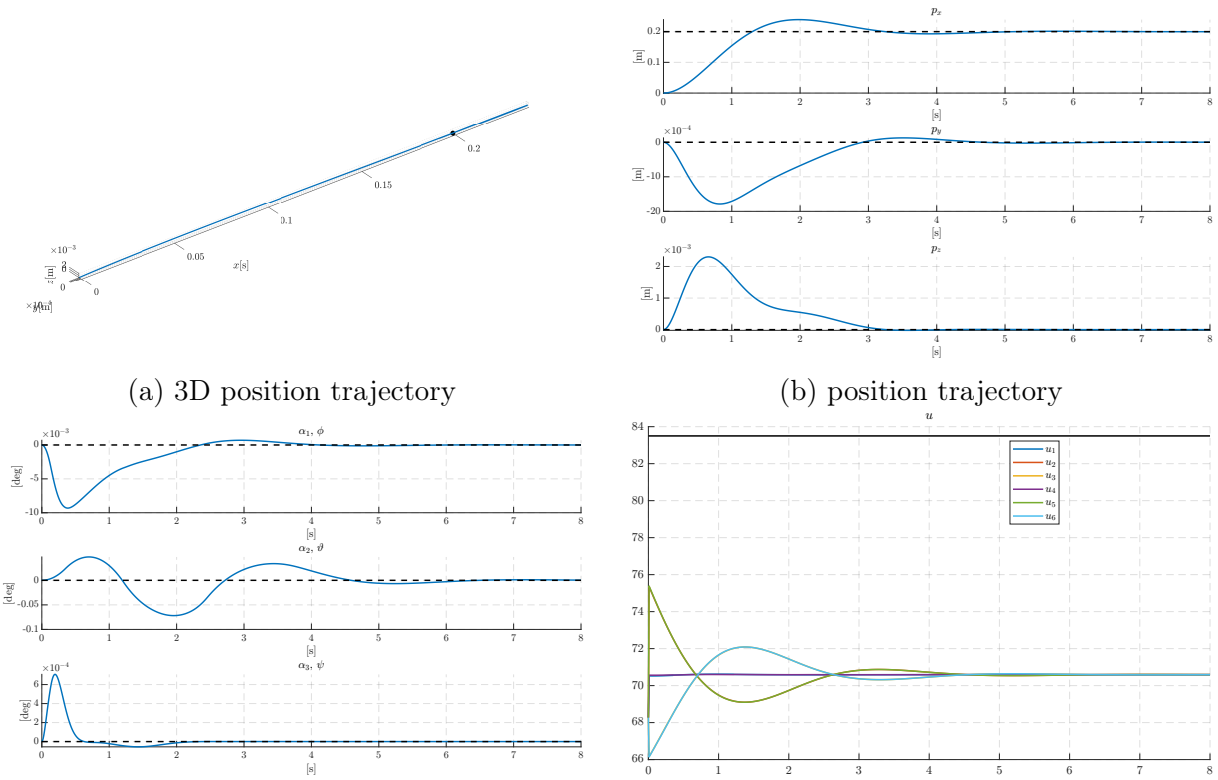
(b) position trajectory

(c) angles trajectory

(d) inputs trajectory

Figure 4.9: simulation along x axis, $\alpha : 5°, \beta : 2°$

Starting from the first axis, we can see how, in the 3D plot in Figure 4.9a the agent is capable of performing a quasi-straight manoeuvre. In particular, the y and z takes very small values compared to the x one. It can be observed that also the angles assumes small values, indicating that the zero moment constraint is working up to a certain tolerance.



(a) 3D position trajectory

(b) position trajectory

(c) angles trajectory

(d) inputs trajectory

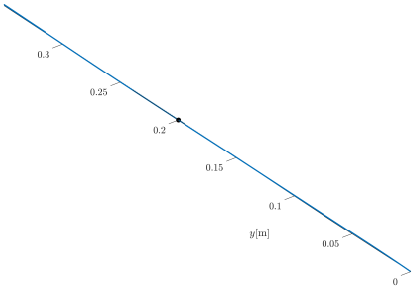
Figure 4.10: simulation along x axis, $\alpha : 40°, \beta : 2°$

It is important to point out though that it is not able to stabilize to the reference value in the simulation time. This is probably due to the value of α being too small, leading to small forces components that can be allocated horizontally. This problem is solved by increasing the α angle, as in Figure 4.10a. There, it has been set to 40 degrees, and we can see that the stability is attained, moreover in a short time window.

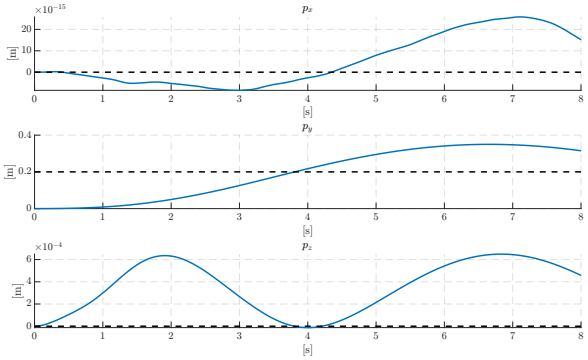
y-axis

Also in this case we have a good straightness of the path, seen in Figure 4.11a except with respect to the previous case, a peculiar wave in the z axis is observed in Figure 4.11b reminiscing of a sinusoidal wave, nonetheless, it is still in the order of 10^{-4}, so it is negligible in the problem.

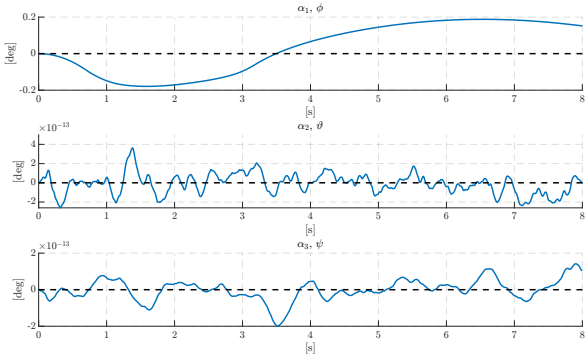
The simulation with α angle of 40 degrees in Figure 4.12 instead attains a much more stable final state.



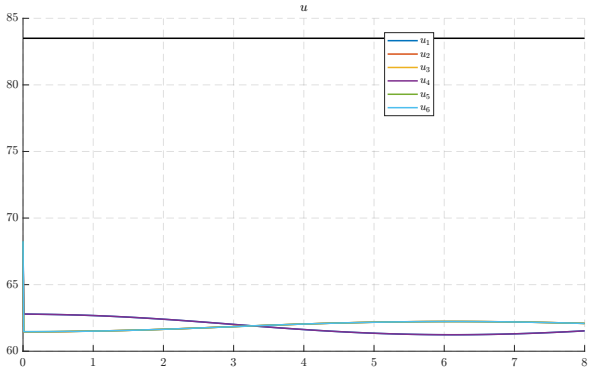
(a) 3D position trajectory



(b) position trajectory

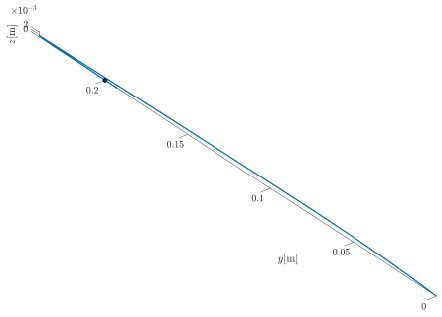


(c) angles trajectory

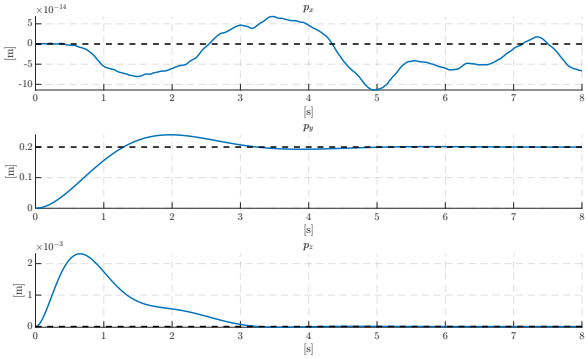


(d) inputs trajectory

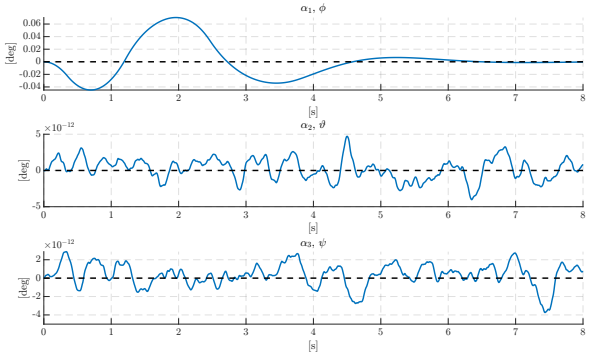
Figure 4.11: simulation along y axis, $\alpha : 5°, \beta : 2°$



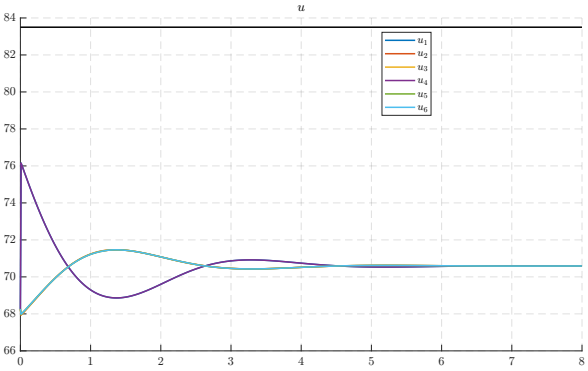
(a) 3D position trajectory



(b) position trajectory



(c) angles trajectory

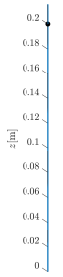


(d) inputs trajectory

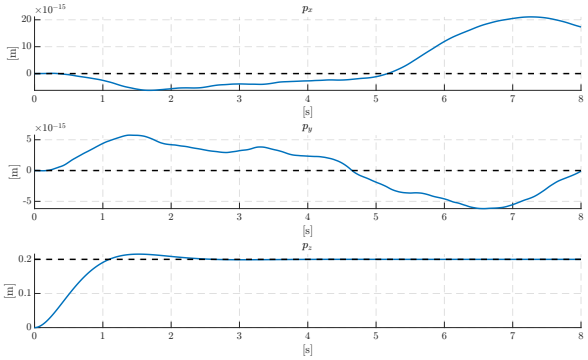
Figure 4.12: simulation along y axis, $\alpha : 40°, \beta : 2°$

z-axis

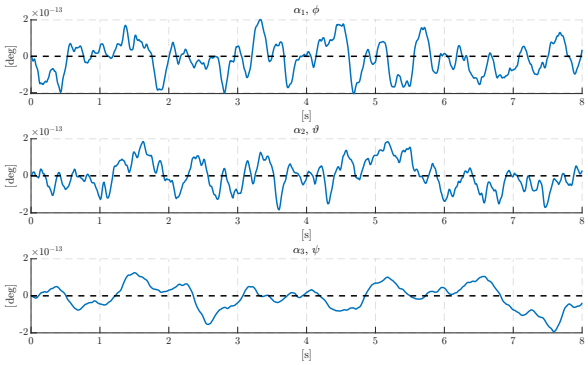
In the last axis, we can see the simplest case that we can think of. This is because, as mentioned in the **B** analysis in subsection 3.3.1, the one vector belongs to the kernel. Implying a that a direct control with all the propellers with the same spinning rates lead to a perfect vertical movement. In fact, this is confirmed from both Figure 4.13 and Figure 4.14, where all the inputs almost takes the same value along all the duration of the simulation.



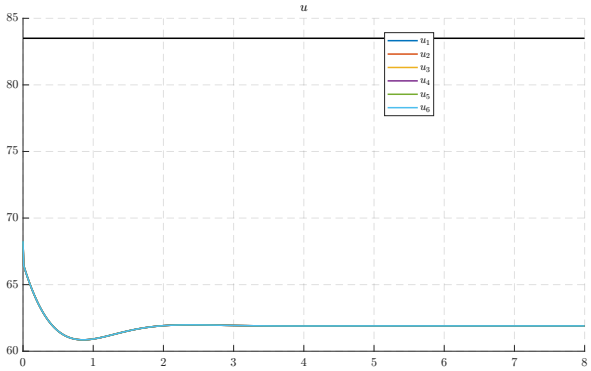
(a) 3D position trajectory



(b) position trajectory

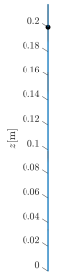


(c) angles trajectory

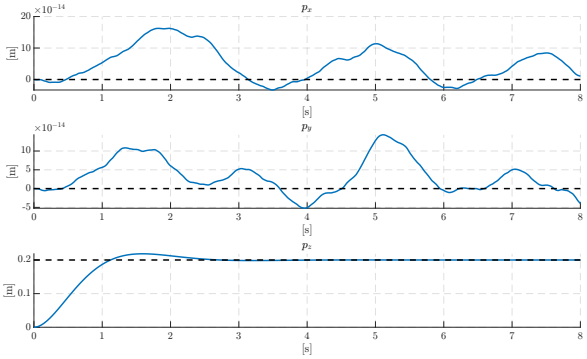


(d) inputs trajectory

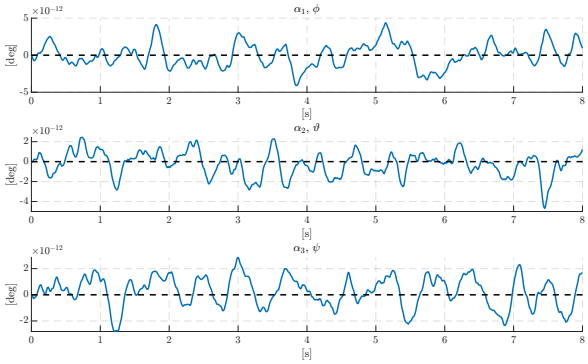
Figure 4.13: simulation along z axis, $\alpha : 5°, \beta : 2°$



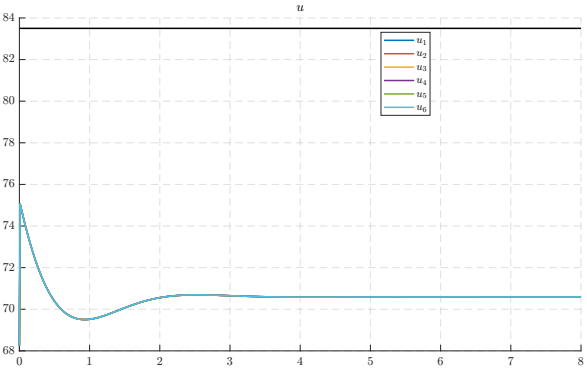
(a) 3D position trajectory



(b) position trajectory



(c) angles trajectory



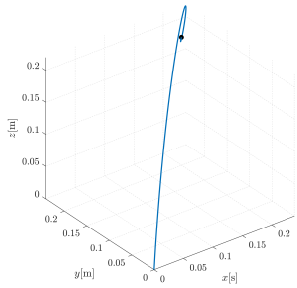
(d) inputs trajectory

Figure 4.14: simulation along z axis, $\alpha : 40°, \beta : 2°$

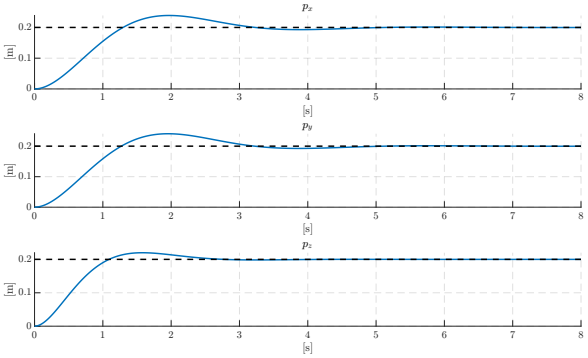
4.2.5 Generic point

To combine all the simulation together, and to see how the agent would react to the tracking of a generic position reference, here in the final simulation a reference of 0.2**1** is given.

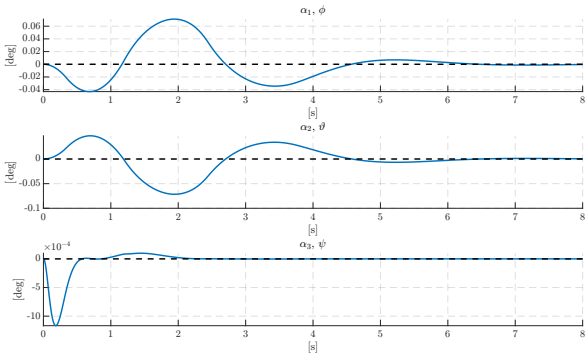
It can be seen that the agent achieve perfect tracking, with the zero moment constraint being sufficiently satisfied, as the angle variations suggests that a small amount of it has been generated, but the angles components assume fairly small values again.



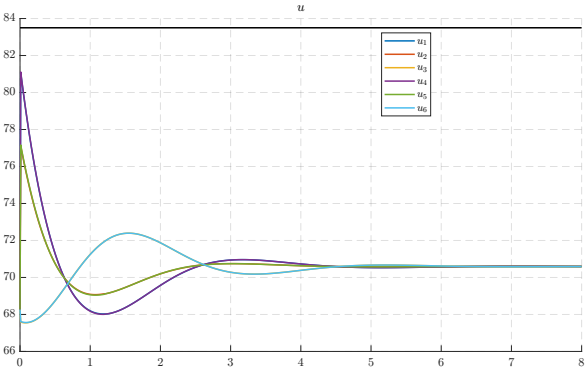
(a) 3D position trajectory



(b) position trajectory



(c) angles trajectory



(d) inputs trajectory

Figure 4.15: simulation along all axis, $\alpha : 40°, \beta : 2°$

5 Conclusion

All the studies carried out in this thesis, have proven that a generically cited multirotor possesses a large manoeuvrability and robustness if the geometry of the platform is chosen accordingly.

The first important result is the closed formula that has been found for the volume representing the space of feasible forces. This made possible to study its value in a continuum of the platform configurations, and not only restricted to the hexa-rotor case, but with a generically tilted multirotor platform.

In particular, it has been studied how the rotor sideways inclination angle α, has a more critical role in the actuation capability, in contraposition to the β angle, responsible for the inward or outwards inclination of the propellers with respect to its Centre of mass.

New studies brought attention to the problem of considering the volume of forces that can effectively maintain in mid-air the UAV, even if only by an approximated formula, but with results that have received an affirmation in the simulation, first without noise, than with a more realistic simulation that included lateral and vertical forces noise.

5.1 Future works

To continue the work that have been done until now, what is missing is experimental tests. This would validate all the results of this thesis, starting from the tilted hovering problem, to the general robust trajectory tracking problem.

Further generalization of the codes could lead to be able to repeat the analysis also for the case of possible rotor losses, and the majority of the results achieved in this thesis could help to make a path planning algorithm capable of fault tolerance and fail safeness.

A

Computational Tools

A.1 SageMath

`SageMath` [8] ‘ is a free, open-source mathematics software system licensed under the GPL’. It is built on top of many open-source python packages, amongst which we find `NumPy`, `Sympy` and `Maxima`.

It provides a convenient Python interface to symbolic computation, that combines the power of its underlying frameworks, under a common interface.

This means for example that, for symbolic computation, the `Sympy` engine mixed with `Maxima` are used. In this way, it is possible to exploit the higher speed of `Maxima`, while being able to use all the feature of `Sympy`. These include for example code compiling and export, whereas for plotting, the base engine used is `matplotlib`. These are only some basic examples of `SageMath`’s use of the many libraries it is build upon.

A.2 MATLAB

MATLAB®

MATLAB is both a programming language, and an IDE specifically designed for it. It combines data analysis capabilities with easiness of expressing dense and sparse matrices and array.

Another important feature is its ability to easily scale the application running, starting from multicore processing and GPU parallelization, to be able then to make the same code run on a computing cluster.

A.2.1 Simulink

Simulink is a whole environment built to work in synopsis with MATLAB, that provides tools to build and simulate models, with various ODE solvers.

A.3 MATMPC

MATMPC [10] is a toolbox developed in the University of Padova that provides the user an easy-to-use simulation environment for NMPC computation and simulation.

It is developed to work with MATLAB, and the majority of its code is written in it, whereas the most time critical code sections are written in C code, embedded in Matlab through the Matlab C API.

Bibliography

[1] Giulia Michieletto et al. 'Nonlinear Control of Multi-Rotor Aerial Vehicles Based on the Zero-Moment Direction'. In: *IFAC-PapersOnLine* 50.1 (2017). 20th IFAC World Congress, pp. 13144–13149. ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2017.08.2168.

[2] J. J. Liang et al. 'Performance Evaluation of Multiagent Genetic Algorithm'. In: *Natural Computing* 5.1 (Mar. 2006), pp. 83–96. DOI: 10.1007/s11047-005-1625-y.

[3] Pedro Trueba et al. 'Self-organization and Specialization in Multiagent Systems through Open-Ended Natural Evolution'. In: *Applications of Evolutionary Computation*. Springer Berlin Heidelberg, 2012, pp. 93–102. DOI: 10.1007/978-3-642-29178-4_10.

[4] Beniamino Pozzan et al. 'Heterogeneous Formation Control: a Bearing Rigidity Approach'. In: (2021), pp. 6451–6456. DOI: 10.1109/CDC45484.2021.9683374.

[5] Nan Zhao et al. 'UAV-Assisted Emergency Networks in Disasters'. In: *IEEE Wireless Communications* 26.1 (Feb. 2019), pp. 45–51. DOI: 10.1109/mwc.2018.1800160.

[6] Francesco Nex and Fabio Remondino. 'UAV for 3D mapping applications: a review'. In: *Applied Geomatics* 6.1 (Nov. 2013), pp. 1–15. DOI: 10.1007/s12518-013-0120-x.

[7] Giulia Michieletto, Angelo Cenedese and Antonio Franchi. 'Force-Moment Decoupling and Rotor-Failure Robustness for Star-Shaped Generically-Tilted Multi-Rotors'. In: (Dec. 2019), pp. 2132–2137. DOI: 10.1109/CDC40024.2019.9030008.

[8] W. A. Stein et al. *Sage Mathematics Software*. Version 9.7. The Sage Development Team. 2022. URL: http://www.sagemath.org.

[9] Marco Mantovanelli. *HR01 PROJECT Progettazione di un esarotore & analisi delle proprietà di attuazione*. 2021.

[10] Yutao Chen et al. 'MATMPC - A MATLAB Based Toolbox for Real-time Nonlinear Model Predictive Control'. In: *CoRR* abs/1811.08761 (2018). URL: http://arxiv.org/abs/1811.08761.

[11] Bruno Siciliano et al. *Robotics*. Springer London, 2009, pp. 106–107. DOI: 10.1007/978-1-84628-642-1.

[12] Ioannis Z. Emiris and Vissarion Fisikopoulos. 'Practical Polytope Volume Approximation'. In: *ACM Trans. Math. Softw.* 44.4 (June 2018). ISSN: 0098-3500. DOI: 10.1145/3194656.

[13] Giulia Michieletto et al. 'Hierarchical nonlinear control for multi-rotor asymptotic stabilization based on zero-moment direction'. In: *Automatica* 117 (2020), p. 108991. ISSN: 0005-1098. DOI: 10.1016/j.automatica.2020.108991.