FEDERAL UNIVERSITY OF SANTA CATARINA
TECHNOLOGY CENTER
AUTOMATION AND SYSTEMS DEPARTMENT

Matheus Bruhns Bastos

Hybrid Drone: Modelling and Simulations

Florianópolis
2020

Matheus Bruhns Bastos

Hybrid Drone: Modelling and Simulations

Final report of the subject DAS5511 (Course Final Project) as a Concluding Dissertation of the Undergraduate Course in Control and Automation Engineering at the Federal University of Santa Catarina in Florianópolis.
Advisor: Prof. Marcelo De Lellis Costa de Oliveira, Dr.
Supervisor: Jonathan Dumon, Eng.

Florianópolis
2020

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Bastos, Matheus Bruhns
Hybrid drone : Modelling and simulations / Matheus Bruhns Bastos ; orientador, Marcelo De Lellis Costa de Oliveira, 2020.
81 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia de Controle e Automação, Florianópolis, 2020.

Inclui referências.

1. Engenharia de Controle e Automação. 2. Drone híbrido. 3. Efeito Magnus. 4. Modelagem. 5. Simulação. I. Oliveira, Marcelo De Lellis Costa de. II. Universidade Federal de Santa Catarina. Graduação em Engenharia de Controle e Automação. III. Título.

Matheus Bruhns Bastos

Hybrid Drone: Modelling and Simulations

This dissertation was evaluated in the context of the subject DAS5511 (Course Final Project) and approved in its final form by the Undergraduate Course in Control and Automation Engineering.

Florianópolis, December 15, 2020.

Prof. Hector Bessa Silveira, Dr.
Course Coordinator

Examining Board:

Prof. Marcelo De Lellis Costa de Oliveira, Dr.
Advisor
UFSC/CTC/DAS

Prof. Eugênio de Bona Castelan Neto, Dr.
Evaluator
UFSC/CTC/DAS

Prof. Fabio Luis Baldissera, Dr.
Board President
UFSC/CTC/DAS

This work is dedicated to the entirety of humankind.

ACKNOWLEDGEMENTS

Firstly, I would like to acknowledge the main institutions that allowed such a work to be carried through: the Federal University of Santa Catarina (UFSC) and, specially, the Automation and Systems Department (DAS), for the excellent teaching; Grenoble Images Parole Signal Automatique (GIPSA-lab), that welcomed me for an innovative internship; the french National Centre for Scientific Research (CNRS), that financed the project and allowed for the internship to happen.

I would like to express my deepest gratitude to my parents, Katianne and Rossano, whose support was extremely helpful and needed, whose words of encouragement allowed me to persevere and without whom my stay in Grenoble would not have taken place. I must also thank my brother, Nicholas, for his confidence in me. I cannot begin to express my thanks to Marina, my long time partner, who provided continuous support, even at distance, valuable advice and who always had profound belief in my abilities.

I am also grateful to my advisors, Marcelo De Lellis and Alexandre Trofino, ever so understanding, who never wavered in their support, ever since we started working together. Many thanks to Jonathan, my supervisor, responsible for valuable contributions and guidance. It was a pleasure to work alongside Matthieu and Alexandre, both who played decisive roles in the development of the project and that welcomed me with such open arms and whose friendships I will carry onward.

I also gratefully acknowledge the help of Andrei, Firas, Makia and Omar – the time together was always much appreciated. Thanks should also go to all staff members, that aided me in various ways and, more specifically, to Martine from GIPSA-lab, who carried out a lot of work in order to allow my presence.

At last, I also thank all my family, friends and people that somehow helped me – directly or indirectly – get where I am.

“The saddest aspect of life right now
is that science gathers knowledge faster
than society gathers wisdom.”
(ASIMOV)

ABSTRACT

This work presents the problem of modelling and simulating a novel structure of hybrid drone, that uses rotating cylinders as wings, generating lift through the Magnus effect – also explored in this thesis. Then, it focuses on the development of such models with different tools in two distinct environments, Simulink and Gazebo simulator attached to ROS. It also proposes an approach to deal with the trajectory generation issue for autonomous aircrafts using their physical constraints as parameters. Next, it shows results of laboratory tests in a wind tunnel, and analyzes the data in order to provide models for important variables of the hybrid drone system. At last, the document is concluded with an overview of the contributions along with proposed future work in the field.

Keywords: Hybrid drone. Magnus effect. Modelling.

RESUMO

Este trabalho apresenta o problema de modelagem e simulação de uma estrutura nova de drone híbrido que utiliza cilindros rotativos como asas, gerando força de sustentação através do efeito Magnus – também explicado neste trabalho. A seguir, foca no desenvolvimento desses modelos com diferentes ferramentas e em dois ambientes diferentes, Simulink e o simulador Gazebo associado ao ROS. Além disso, também propõe uma abordagem para lidar com a questão de geração de trajetória para aeronaves autônomas utilizando suas restrições físicas como parâmetros. Então, apresenta os resultados de testes de laboratório em um túnel de vento, e analisa os dados a fim de fornecer modelos para variáveis importantes do sistema composto pelo drone híbrido. Ao final, conclui com uma revisão sobre as contribuições do trabalho, além de propor próximos passos para o desenvolvimento do trabalho no assunto.

Palavras-chave: Drone híbrido. Efeito Magnus. Modelagem.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

ANOVA	Analysis of Variance
AWE	Airborne Wind Energy
DOF	Degrees of Freedom
eRuby	Embedded Ruby
GIPSA-lab	Grenoble Images Parole Signal Automatique
GUI	Graphical User Interface
NWU	North, West, Up
ROS	Robot Operating System
SDF	Simulation Description Format
SI	International System of Units
UAV	Unmanned Aerial Vehicle
VTOL	Vertical Take-Off and Landing
XML	Extensible Markup Language

LIST OF SYMBOLS

C_L	Lift coefficient
C_D	Drag coefficient
χ	Spin ratio
$\mathbb{I}$	Inertial frame of reference
$\mathbb{B}$	Body frame of reference
R	Rotation matrix based on Tait-Bryan angles (Z-Y-X intrinsic)
F_L	Lift force generated by the wing
ω_M	Magnus wing angular speed
v_a	Apparent wind velocity vector
v_w	Wind velocity vector
V	Drone velocity vector
$v_{a_{XZ}}$	Apparent wind velocity vector projected on the **XZ-Plane**
F_{D_Y}	Drag force generated by the wing for **Y-Axis**
v_{a_Y}	Apparent wind velocity vector projected on the **Y-Axis**
ρ	Air density
S_M	Projected surface area of the wing (rectangular)
$F_{D_{XZ}}$	Drag force generated by the wing for **XZ-Plane** wind
r_M	Magnus wing radius
ℓ_M	Magnus wing length
$\mathbf{R^2}$	Coefficient of determination – *R-squared*

CONTENTS

1 INTRODUCTION AND MOTIVATION

During the last years, economy has seen a growing usage of Unmanned Aerial Vehicles (UAVs), commonly known as drones, for multipurpose activities which, accompanied by the fast adapting industry, led to a substantial growth in the sector (DRONEII, 2020). Specifically, quadcopter drones and other similar designs have been sighted performing various functions, even as part of inspection of industrial sites (SKYWORKS, 2020), aerial surveillance (AUTONOMOUS. . . , 2020), postal delivery (SWISS. . . , 2020), high quality image acquisition (DJI, 2020b), and more, in contexts as pest control in plantations (UAV-IQ, 2020) and recreational purposes (INTEL, 2020).

Most of these activities have their potential effectiveness reduced by known challenges faced by electric quadcopters, such as low limit for payload, high dependence on battery technology and complexity of control algorithms. Some concepts, however, are actually hindered by these limitations. The use of quadcopters for delivery over extended distances or with heavy payloads serves as example. Solutions that seek indoor usage of drones also need several new features, from positioning to obstacle avoidance and dangers of human interaction.

The payload capacity is limited by the overall size of the drone. Increasing the size of fixed propeller quadcopters diminishes their advantages, because it leads to an increase in the propellers' momentum and affects the controllability of the system, while making construction and maintenance more complex and expensive. Moreover, the low mass blades used in small UAVs have low kinetic energy, being reasonably safe for close interaction. The same is not valid for large rotors (and blades) in machines such as piloted helicopters.

The payload limitation affects the system by restricting the use of large and heavy batteries that could offer more charge for the flight. High-energy low-weight batteries are necessary to increase flight autonomy without compromising the structure of standard quadcopters, not to require a boost in size that would be accompanied by the aforementioned issues. One of the most advanced technologies in video recording commercial drones, for example, has an alleged autonomy of 34 minutes, measured at constant speed of 18 km/h, at sea level and without any wind (DJI, 2020a). Considering a round-trip, it would be able to reach places just past 5 km, in ideal conditions.

There are different approaches when dealing with flight autonomy of drones, from creating aircrafts with Vertical Take-Off and Landing (VTOL) capabilities using tilt rotors (CHEN; JIA, 2020), to innovative designs that can benefit from its own enhanced features to a more efficient flight (LYU et al., 2017). The project at GIPSA-lab suggests a classical solution with specific characteristics, aiming to design and control a hybrid quadcopter with not only propellers, but also wings that are responsible for the main lift force after take-off. This design allows for the propellers to be more efficiently used for

horizontal thrust, as further explained in the next sections.

The principle of this hybrid drone is that the wings are capable of providing extra lift when in horizontal flight, allowing the propellers to be used almost exclusively (depending on the parameters of the drone and the desired speed) for acceleration in the direction of the flight, instead of producing the major part of the lift force. This allows a reduction of power usage, improving autonomy in both flight time and reachable distance.

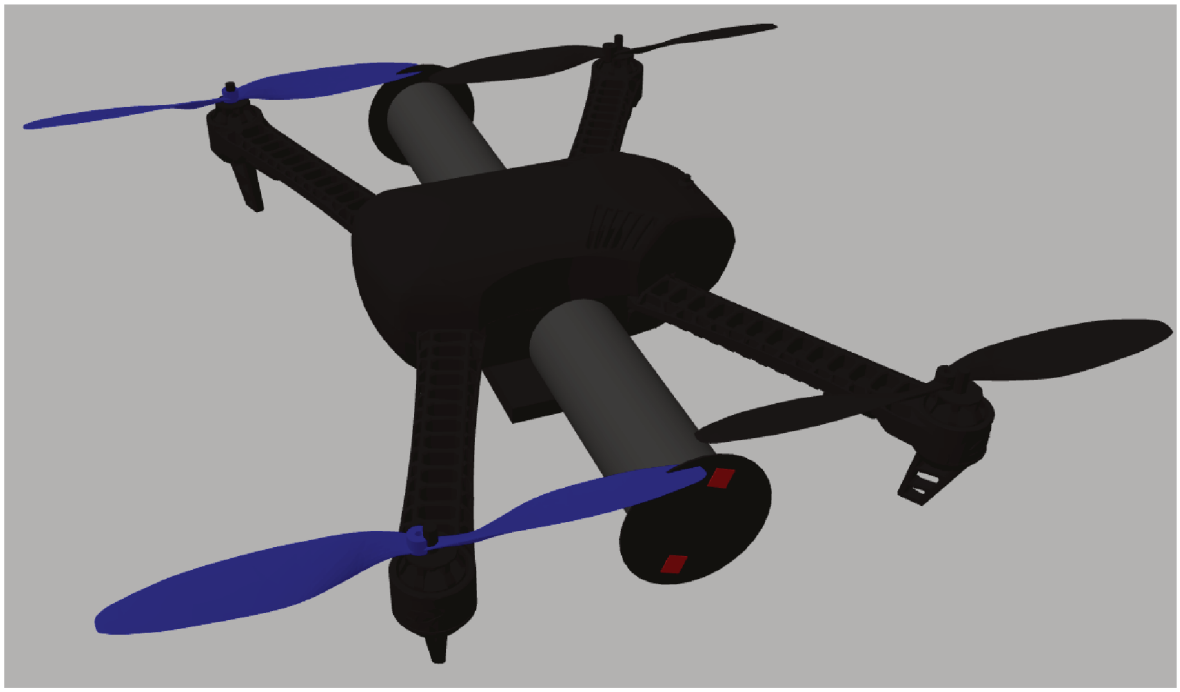
The idea in this thesis is to make use of Magnus effect-based systems to implement cylindrical rotating wings in a quadcopter. As indicated in (HABLY et al., 2018), one important advantage when compared to other wings is that the magnitude of lift and drag force vectors are unaffected by the *angle of attack* - the angle between fixed reference line on the wing and the apparent wind direction.

In Figure 1 an example of a quadcopter drone model is presented, with common architecture. In Figure 2 we have the hybrid counter part, with Magnus wings attached to it. More details on these models are presented in Chapter 3.

Figure 1 – Screenshot of the Iris drone model, the blue blades indicate the front of the drone. Conventional drone.

Source – original image, model from (PX4, 2020).

Figure 2 – Iris model with the Magnus wings model attached to it, composing the Hybrid Drone model.



Source – original.

1.1 OBJECTIVES

The goal of this thesis is to study the innovative UAV design based on a standard quadcopter and improved by a rotating cylinder-shaped wing able to generate lift through the Magnus effect.

The work then moves on to modelling the wings and the hybrid drone, simulating flights in various scenarios, designing tests for the wings and improving the simulated environments with real data, comparing with a regular drone of similar design and testing and analyzing a wing prototype.

Moreover, this thesis also presents a novel approach for the generation of feasible trajectories for drones based on constraints of speed, acceleration and/or *jerk*[1], using *spline*[2] interpolation. The method also ensures continuity of the trajectory.

1.2 KEY ASPECTS

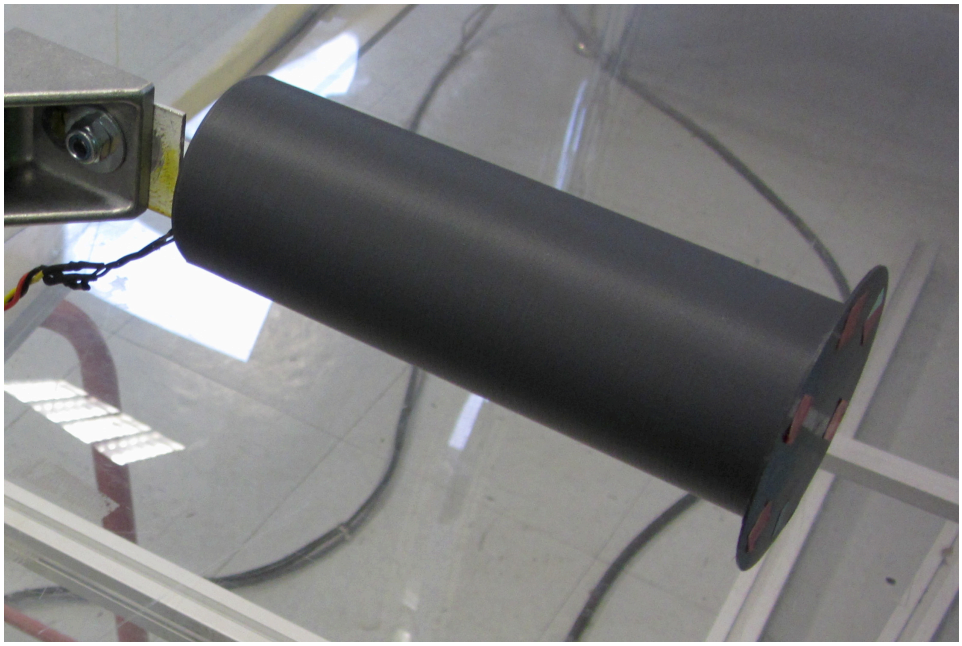
The proposed model has some specificities that can be stressed. Firstly, the *wings* mentioned are not standard wings, but rotating cylinders able to generate lift by

[1] Here on, 'jerk' is used to refer to the third derivative of the position in respect to time.

[2] Spline, in the mathematical sense, means a piecewise polynomial function, not limited to two dimensions.

the Magnus-effect. It is similar to standard wings in the sense that it also generates lift and drag forces whose magnitude depend on a specific system variable. This kind of structure was already thoroughly studied for use in Airborne Wind Energy (AWE) systems, as in (GUPTA et al., 2017). In Figure 3 there is an example of what the Magnus wing can look like.

Figure 3 – Magnus-effect wing prototype.



Source – GIPSA-lab archive (2020).

Given the symmetry of the wing along the axis of rotation, it is evident that the angle of attack, so important for common wings, is not relevant for the aerodynamical effects on the Magnus wing. Instead, both lift coefficient (C_L) and drag coefficient (C_D) are functions of another variable of interest, namely the *spin ratio* (X), that represents the ratio between the speed of the surface of the wing and the apparent wind in contact with it. The Magnus spin ratio is mathematically defined later on.

OUTLINE

The rest of this work is organized as follows.

Chapter 2 presents an overview of the base-model for the quadcopter, then introduces the adaptations for the hybrid drone model and the corresponding elements that need a more thorough explanation. The chapter finishes by presenting comparisons between the original and the hybrid drone, and also the hybrid drone with itself in different scenarios.

Chapter 3 brings the development of models in the ROS and Gazebo environment, introducing the architecture used for the high-fidelity simulation software while using real-life controllers based on ROS.

Chapter 4 presents the motivation for the development of new methods of trajectory generation, and the approach based on speed, acceleration and jerk constraints for a continuous trajectory, along with examples for 1 and 3 dimensions.

Chapter 5 presents the data obtained from lab tests with wing prototypes, and analyzes the results to draw conclusions over key aspects of the system. Among the results, models for important variables such as the coefficients of lift and drag are presented with detail.

Chapter 6, finally, presents conclusions and possible future steps on the development of the project.

2 SIMULINK MODEL

The *Simulink* model used in this work was built beforehand, based on a quadcopter model from a specific study within the laboratory, explained later on. Before we dive into the details of this simulator and the achieved results, we will first take a look into the base concepts that are necessary for a thorough understanding of it. That is the mathematical model with its considerations about reference frames and rotation, the definition of lift and drag forces as well as the apparent wind, and fundamentals of the Magnus effect.

2.1 QUADCOPTER MODEL

The mathematical model for the quadcopter used in this thesis is based on the detailed study of the dynamics of quadcopters, presented and explained in Chapter 2 of (MONTCEL, 2019), and was implemented in *MATLAB* - and also specifically *Simulink* - environment prior to the start of this work. This model allows for important customizations related to the drone structure, such as sizes and mass, maximum speed of the propellers and more.

The principle is to define the physical equations of a 6 Degrees of Freedom (DOF) rigid body, and then include the contributions that represent the difference between the rigid body and the quadcopter, as many as the desired precision of the model. Examples of those contributions are the effect of the motors and the propellers, deformation of components and air friction. Among the most important elements are, naturally, the thrust and the torque generated by the fast-rotating propellers.

One of the main objectives of the model is, at low-level, to transform the rotation speed of the 4 propellers into thrust and torque vectors, to be translated into resulting force and moment for the entire drone. At a higher level, the goal is to transform attitude reference into acceleration reference, or vice-versa depending on the situation, so that it eventually turns into torque and thrust, leading to the desired rotation speed for each propeller.

2.2 EARLY ADAPTATIONS

At first, the new hybrid drone model needs very basic updates to resemble the expected behaviour. The mass of the wings need to be added to the total mass of the drone, the lift and drag forces need to be considered when balancing the dynamic equations, and the estimation of the state of the batteries must take into account the extra power usage of the Magnus wings. This new setup - to be improved afterwards - gives us a starting point on the potential of this hybrid drone.

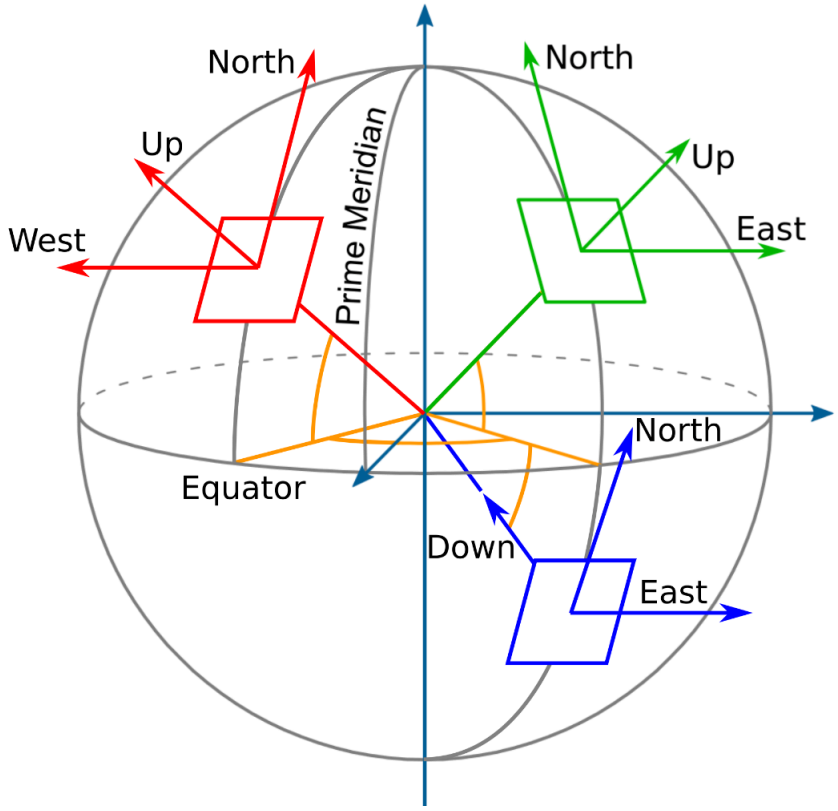
For the mass of the wings, the first prototypes were mounted and weighted, then

this value was added to the model accordingly. The lift and drag forces were added along the other forces when computing the resulting acceleration of the drone, and they are further detailed in the next sections.

2.2.1 Reference Frames

The first element to define is the reference frame. When modelling this system it is important to stress that there are two reference frames at stake. One is the inertial frame of the Earth, denoted by $\mathbb{I}$, using North, West, Up (NWU) coordinates such that the unit vectors i_1, i_2 and i_3 correspond respectively to North, West and Up directions. This frame is adequate to represent the trajectory of the drone in a "natural" way. In Figure 4 we have a simple diagram showing the construction of this (and others, unused) reference frame(s).

Figure 4 – Representation of different reference frames using local tangent planes.

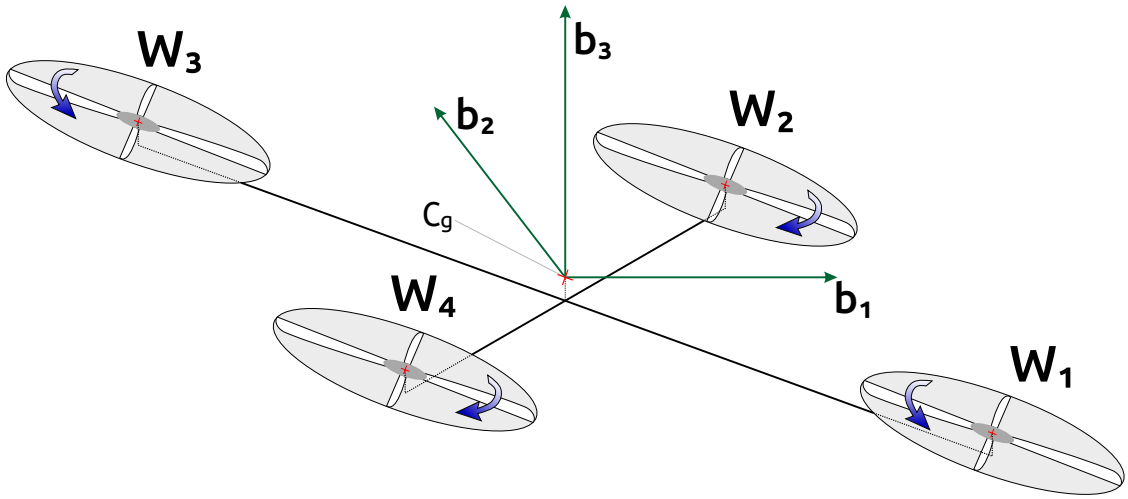


Source – (MONTCEL, 2019).

The other frame of interest is the body frame, denoted by $\mathbb{B}$, used to describe the dynamic effects on the drone. In Figure 5 we can see a representation of the body frame of reference, where W_1, W_2, W_3 and W_4 are the positions of each motor, C_g is the center of gravity of the quadcopter, and the unit vectors are defined as b_1, b_2 and

b_3, in a configuration referred as **X**, since the direction of *front* is halfway between two motors, as opposed to **+**, when the *front* direction is along a motor branch. The blue arrows represent the sense of rotation of each motor.

Figure 5 – Definition of body frame and relevant elements.



Source – Adapted from (MONTCEL, 2019).

2.2.2 Rotation Matrix

In simulation, world elements such as the wind and trajectory states (linear velocity, position and attitude) are defined in the inertial frame ($\mathbb{I}$), while aerodynamic effects such as the lift and drag forces are computed in the body frame ($\mathbb{B}$).

To transform from one frame to the other we use a rotation matrix, following a specific convention for Euler angles, namely the **Tait-Bryan** angles. This means that when in zero degrees of elevation, we have the drone in horizontal attitude. More specifically, we use the sequence identified as **Z-Y-X intrinsic**[1].

From $\mathbb{I}$, we apply a rotation ψ around the **Z-axis** (represented by i_3). This rotation is expressed through the matrix $R_Z(\psi)$. The resulting coordinates are rotated around the new **Y-axis** by an angle θ, through the matrix $R_{Y'}(\theta)$. Finally, we apply a rotation ϕ around the **X-axis** of the previous coordinates, through the matrix $R_{X'}(\phi)$. Given the conventions used for the Euler angles, the resulting rotation matrix from the inertial to the body frame is computed as:

$$R = R_Z(\psi) \cdot R_{Y'}(\theta) \cdot R_{X'}(\phi) \ , \tag{1}$$

[1] Intrinsic in this context means that each successive rotation is made in respect to the current rotating system of coordinates and not the original one, as opposed to extrinsic.

which yields:

$$R = \begin{pmatrix} c(\psi)c(\theta) & c(\psi)s(\theta)s(\phi) - s(\psi)c(\phi) & c(\psi)s(\theta)c(\phi) + s(\psi)s(\phi) \\ s(\psi)c(\theta) & s(\psi)s(\theta)s(\phi) + c(\psi)c(\phi) & s(\psi)s(\theta)c(\phi) - c(\psi)s(\phi) \\ -s(\theta) & c(\theta)s(\phi) & c(\theta)c(\phi) \end{pmatrix} , \quad (2)$$

where s and c represent sine and cosine, respectively (e.g., $s(n)$ represents the sine of n).

This matrix allows us to transform from any coordinate in the inertial frame $\mathbb{I}$ to the corresponding coordinate in the body frame $\mathbb{B}$. This can be done with the unit vectors for the 3 directions as $b_1 = R \cdot i_1$ and $b_2 = R \cdot i_2$ and $b_3 = R \cdot i_3$.

2.2.3 Magnus Effect

Before defining the important forces related to the wings, we first study the physical principle to generate lift on the wing: the Magnus effect. This effect presents itself with the movement of a rotating body through a fluid. A simple and well-known example of the effect is when a football *curves* from its path because it was shot with rotation, or the commonly known *topspin* used by tennis players. When the ball is spinning in the air, a force appears, one that changes its path in a manner that an otherwise non-rotating ball would not. The direction of the force depends directly on the axis and sense of rotation.

The Magnus effect is a direct consequence of Bernoulli's principle, for it is caused by the different speeds of the fluid around the body, since on one side the fluid is accelerated and on the other, decelerated, generating a difference in pressure and, therefore, a force.

This effect has been studied in the past decades with various use cases, one example being power contribution systems for large ships, as described in detail in (TRAUT et al., 2014).

The direction of the lift force that appears is simultaneously orthogonal to that of the apparent wind and the rotation of the body, and its sense can be described as:

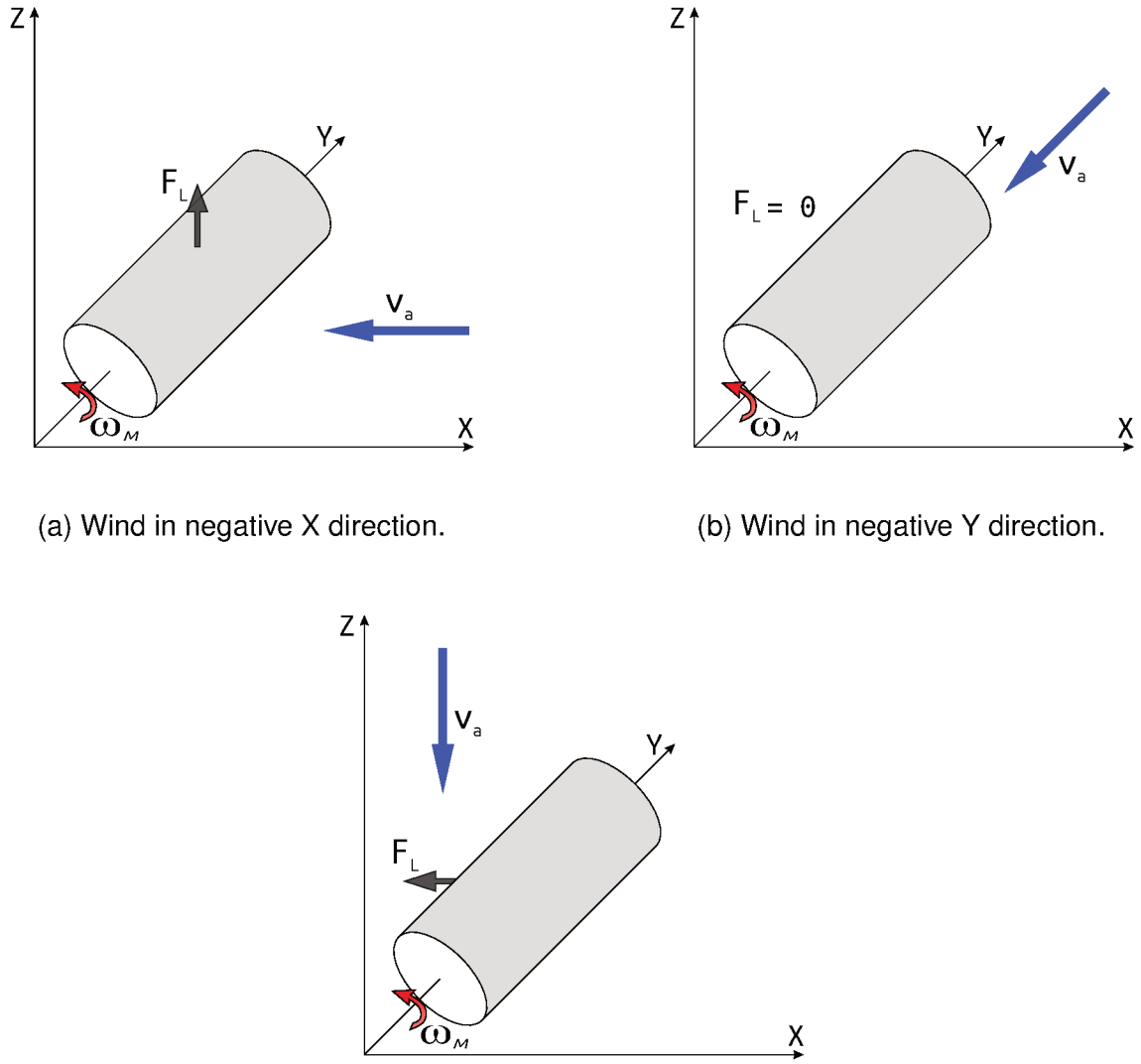
$$F_L = -(\omega_M \times v_a) , \quad (3)$$

where F_L indicates the direction of the lift force, ω_M indicates the axis of rotation of the body – in our case, the Magnus wing –, and v_a gives the direction of the apparent wind. The leading negative sign appears because the direction of the apparent wind vector v_a is opposed to the direction of the moving body. The proper definition of the apparent wind is the subject of the next section.

With that in mind, Figures 6a-6c present the cases in which the direction of the wind is along the X, Y and Z-axis, respectively. The red arrow indicates the sense of

rotation of the wing (ω_M), the blue arrow indicates the direction of the apparent wind (v_a), and the gray arrow indicates the direction of the resulting lift force (F_L).

Figure 6 – Diagram of Magnus wing in wind with different directions.



(a) Wind in negative X direction.

(b) Wind in negative Y direction.

(c) Wind in negative Z direction.

Source – original.

2.2.4 Apparent Wind

Having defined the rotation mechanism, we may define apparent wind velocity, used for the definition of spin ratio and lift and drag forces. The wind in the simulation world is defined as a three-dimensional vector in the inertial frame ($\mathbb{I}$), as $v_W \in \mathbb{R}^3$. The linear velocity of the drone is also a three-dimensional vector in the inertial frame, as

$V \in \mathbb{R}^3$. Equation 4 defines the apparent wind:

$$v_a = v_w - V \ ,$$
$$v_a = \begin{pmatrix} v_{a_x} \\ v_{a_y} \\ v_{a_z} \end{pmatrix} = \begin{pmatrix} v_{w_x} \\ v_{w_y} \\ v_{w_z} \end{pmatrix} - \begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix} \ , \tag{4}$$

and using the rotation matrix, in an analogous way as the transformation from i_1 to b_1, we can take the apparent wind vector to the body frame $\mathbb{B}$.

For the computation of lift and drag forces, there is yet another concern. Any substantial lift can only be generated by the interaction of the wind orthogonally to the rotating axis. Since the wings are to be mounted over the **Y-axis** of the drone (correspondent to b_2, illustrated in Figure 5), the portion of the wind that generates lift is in the **XZ-plane**. As seen in Figure 6b, it is clear that the wind in the direction of the **Y-axis** does not generate lift.

For that reason, the apparent wind vector needs to be decomposed, such that only its **XZ-plane** projection, represented by $v_{a_{XZ}}$, is used for computing spin ratio and lift and forces. However, it is not correct to assume that no substantial drag is generated by the wind in Y direction (or *lateral* wind) and, to compensate that, another component of the drag force – represented by F_{D_Y} – is computed using only the apparent lateral wind (v_{a_Y}).

For all simulations, a fixed coefficient for drag from lateral wind was used, based on experimental data, but no further study on the coefficient of drag for the lateral wind was developed during this thesis.

2.2.5 Lift and Drag Forces

To compute lift and drag forces, first we need to define these forces and their relation with other important variables. Equations 5 and 6 show how to compute these values:

$$F_L = \frac{1}{2} \rho \| v_{a_{XZ}} \|^2 S_M \ C_L \ , \tag{5}$$

$$F_{D_{XZ}} = \frac{1}{2} \rho \| v_{a_{XZ}} \|^2 S_M \ C_D \ , \tag{6}$$

where F_L is the lift force, $F_{D_{XZ}}$ is the drag force related to wind in **XZ-plane**, ρ is the air density, $\| v_{a_{XZ}} \|$ is the Euclidean norm of the apparent wind velocity vector projected on the **XZ-plane**, S_M is the projected surface area of the wing, computed as $S_M = 2 \ r_M \cdot \ell_M$, where r_M is the radius of the base of the Magnus wing, and ℓ_M is the length of the wing. Throughout all this work, assume units from the International System of Units (SI) if absent.

The values for C_L and C_D come from a general trend based on a study of research papers on the subject – rotating cylinders in high Reynolds number regime – as more detailed in (GUPTA et al., 2017). The proposed values are based on two polynomials, one of fourth degree for C_L, one of third degree for C_D, over the variable X, the Magnus wing spin ratio, defined as:

$$\mathrm{X} = \frac{\omega_M \cdot r_M}{\|v_{a_{XZ}}\|} \,, \tag{7}$$

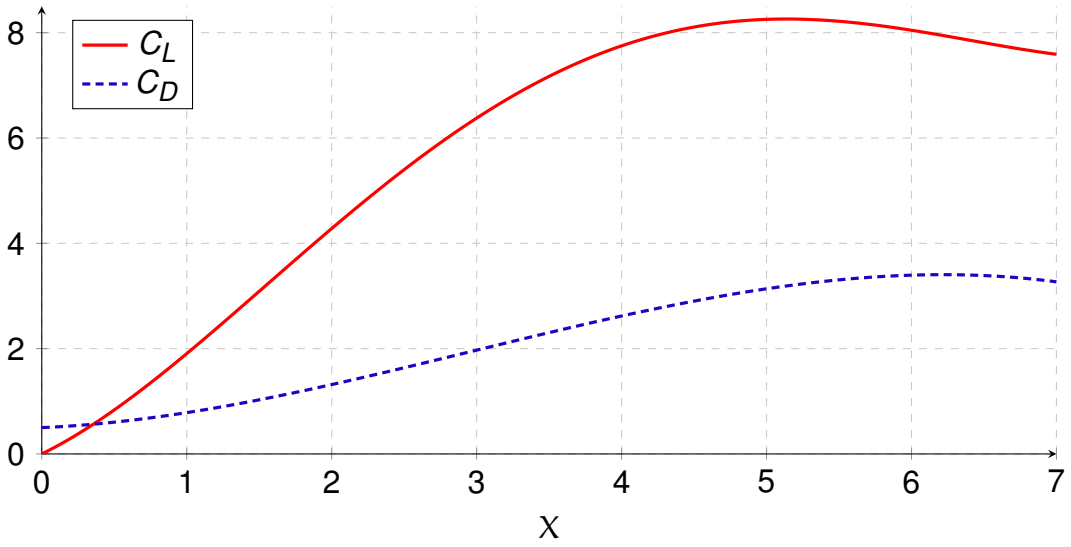
being ω_M the angular speed of the Magnus wing (expressed in radians per second). The polynomials are presented in Equations 8 and 9, respectively:

$$C_L(\mathrm{X}) = 0.0126\,\mathrm{X}^4 - 0.2004\,\mathrm{X}^3 + 0.7482\,\mathrm{X}^2 + 1.3447\,\mathrm{X}\,, \tag{8}$$

$$C_D(\mathrm{X}) = -\,0.0211\,\mathrm{X}^3 + 0.1873\,\mathrm{X}^2 + 0.1183\,\mathrm{X} + 0.5\,. \tag{9}$$

In Figure 7 we can see both of the polynomials. Here, a specific usage range is determined for the spin ratio as [0,6], as in the aforementioned study.

Figure 7 – Polynomials for lift and drag coefficients C_L and C_D in respect to the spin ratio X.



Source – original.

2.3 SIMULATIONS

After the *Simulink* model was updated, simulations were made in different scenarios in order to gather data on the behaviour of the new model – with the Magnus wing – in comparison to the old one – without the wing –, but also between the new

model in different situations, changing parameters such as the rotation speed of the Magnus wing and more.

Since this was a preliminary study of the hybrid drone model, the trajectory chosen for the simulations was very simple, defined by a take-off phase to the desired height followed by a straight path flight along the X-axis (the *front*) at the desired cruise speed, ending in an idle phase (fixed position) as long as the take-off phase. There is no landing phase due to limitations of both models.

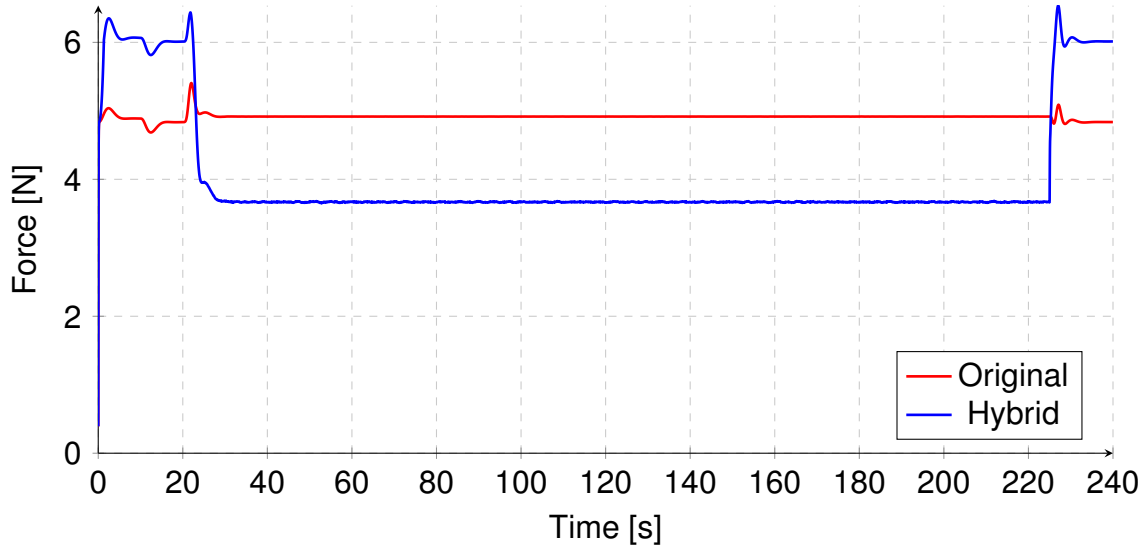
2.3.1 Comparison with and without wing

The first simulations aimed at comparing one version of the hybrid drone with the original drone model, and the metrics are the thrust generated by the propellers, the pitch angle of the drone and the overall power consumption of the system.

Thrust Force

Here, the thrust force is the resulting force in the Z-axis (*upwards*) that the drone generates through its propellers using the electric motors to which they are attached. This is an important aspect because the goal of the hybrid drone is to reduce the power usage and increase flight autonomy, by providing thrust that is not generated by these propellers, but by the Magnus wing.

Figure 8 – Thrust force on Z-axis on original and hybrid model. Simulation lasts for 4 minutes.



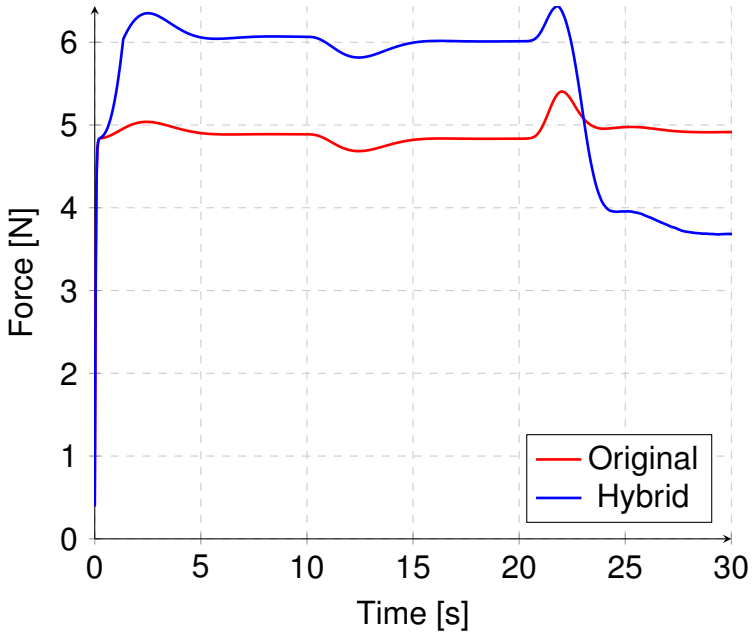
Source – original.

In Figure 8 both models ran for 4 minutes with desired height h_d = 10 m and cruise speed v_x = 10 m/s. The hybrid model had the rotation of the Magnus wings at

ω_M = 880 rad/s, only during horizontal flight. The take-off phase lasts from 0 to 20 seconds, then horizontal flight from 20 to 220 seconds, and then idle phase until 240 seconds.

It is clear from Figure 8 that the thrust generated by the propellers is significantly reduced on the hybrid model when compared to the original model, during horizontal flight. Also note the increase in thrust in the beginning and ending of the simulation on the hybrid model, more evident in the close-up of Figure 9. This is due to the increase in total mass of the drone on the hybrid model, since it has the two Magnus wings in addition to the main drone structure.

Figure 9 – Close-up of the first 30 seconds of simulation, when thrust has a higher component on the hybrid model.



Source – original.

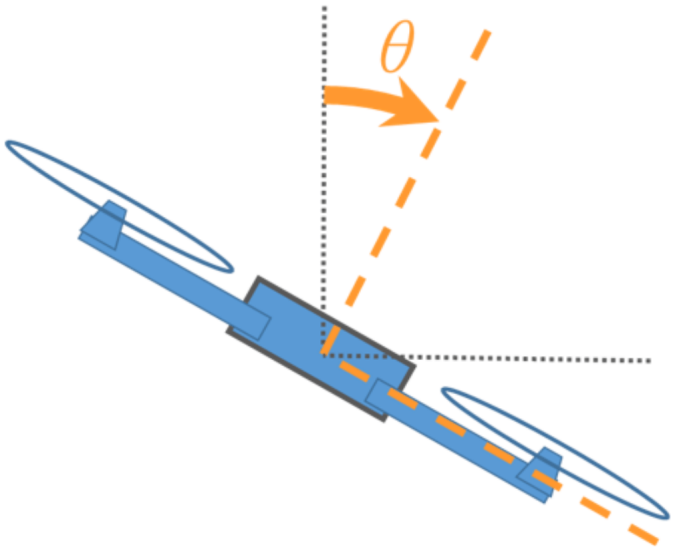
This increase in thrust is the expected behaviour, and it should occur whenever the Magnus wings are not producing upwards lift force.

Since the simulation is set such that the drone flies on a straight path along its X-axis (forwards) after take-off, the lift force generated by the Magnus wings is along the Z-axis (upwards), as we can understand given the previous explanation on the Magnus Effect. For that same reason, the potential lift during take-off would be along the X-axis, slightly forcing the drone backwards. Since this is not a desired effect, the rotation of the Magnus wing was only *turned on* after the take-off phase was over.

Pitch Angle

The pitch angle, from the Euler Angles convention, represents the rotation along the Y-axis of the drone, and is responsible for the balance between horizontal acceleration and compensation of weight. That happens because by rotating along the Y-axis, the drone can change the direction to which the generated thrust points, given the fixed position of the propellers. In Figure 10 there is an illustration of the pitch angle using a side-view, represented by the θ symbol.

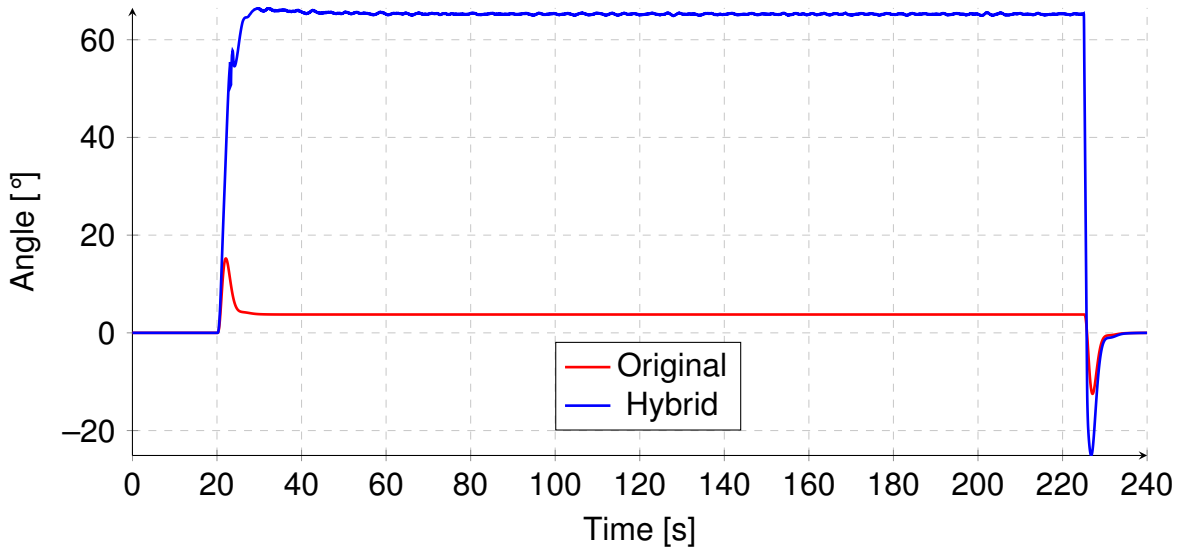
Figure 10 – Diagram illustrating the pitch angle on a simplified frame of a drone.



Source – (TYTLER, 2020).

Therefore, since the hybrid drone can use the propellers **and** the wings for the compensation of weight, it is expected an increase in the pitch angle, allowing the propellers to be more efficiently used for horizontal acceleration. In Figure 11 we can see the pitch angle on both the original and the hybrid model, using the same simulation as the previous comparison of thrust forces.

Figure 11 – Pitch angle on both models during 4 minute simulation.



Source – original.

While on the original model the pitch angle would remain around 5°, on the hybrid model, this value can rise up to around 60°. This means that after take-off phase, the hybrid drone can put its propellers to work much more on horizontal acceleration, and less on weight compensation, when compared to the original model.

One issue that arises with this scenario of a high pitch flight is the aerodynamical drag. When moving forward, the effective contact area of the drone with the air is directly related to the sine of θ. Therefore, its increase leads to an increase in the contact area, increasing the drag. Furthermore, the drag is also related to the square of the relative speed, and if we desire a higher speed, it would increase the pitch angle and then cause both factors to amplify the drag forces on the drone. With that in mind, a limitation on the pitch angle can be introduced in the system, in order to prevent these high drag flight situations.

Power Consumption

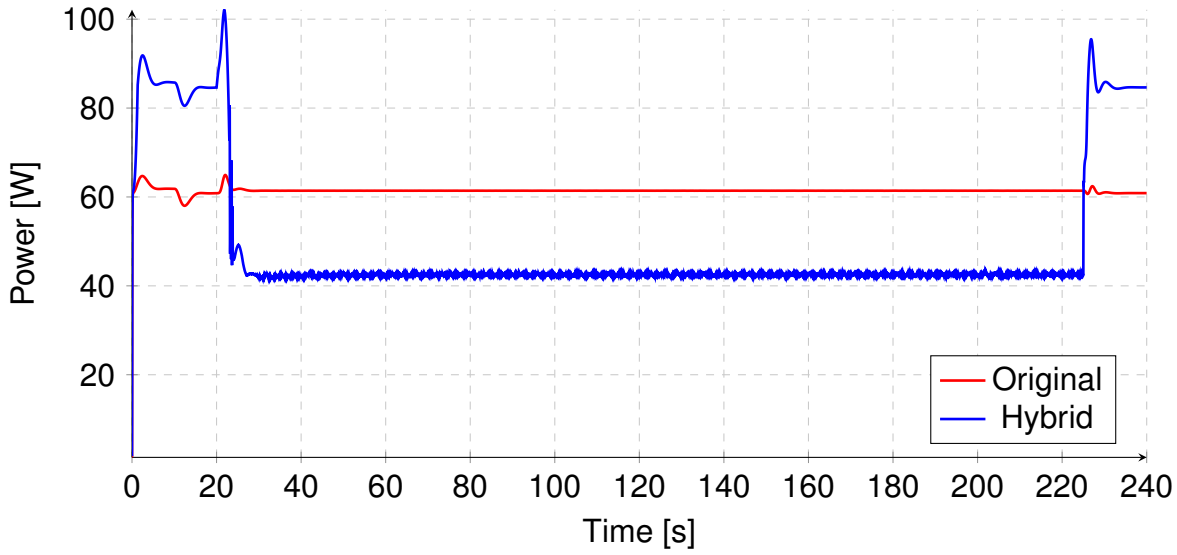
The power consumption is computed through simple estimators based on experimental values attained from batteries used for powering drones in the laboratory. The purpose of this analysis is to have an idea of the amount of power the hybrid drone needs to perform the same trajectory under similar conditions, also in comparison to the original model. The overall power usage of the original drone is computed based on the square of the rotation speed of the propellers, whilst the hybrid drone adds up the power of the Magnus wings, also related to the square of its rotation speed.

The generated lift depends on two main factors – the speed of the drone relative

to the air and the rotation speed of the wings. The increase in any of these factors should cause an increase in the lift force generated by the wings. However, it is important to note that the increase in relative speed implies an increase in rotation speed of the propellers, which means that both factors will affect significantly the power consumption of the drone.

Still, increasing the cruise speed will not affect the power consumption related to the Magnus wings if their rotation speed is left unchanged, but will yield an increase in lift force. The increase in cruise speed, though, will also result in extra drag on the wings, that might increase the power consumption of the propellers.

Figure 12 – Power consumption for the 4 minute simulation on both models.

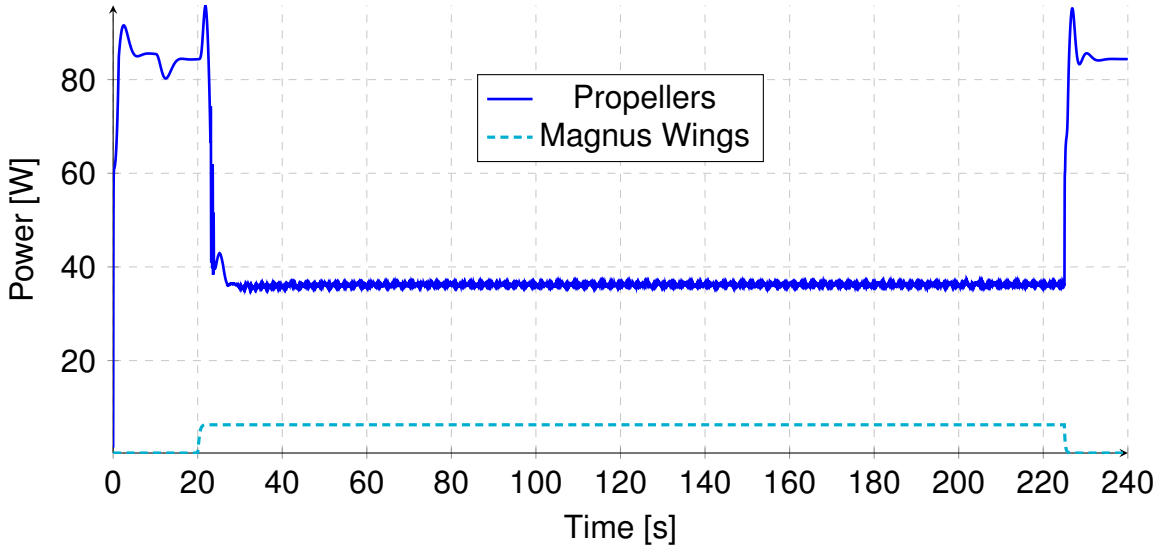


Source – original.

In Figure 12 we can see a comparison of the total power consumption of the drone – only the propellers for the original model and propellers plus wings for the hybrid model – in similar scenarios. The hybrid drone, between take-off and idle phases, uses up to 20 W less power (around a 33% decrease).

In Figure 13 we can see the two components of the hybrid model power consumption: the propellers and the Magnus wings.

Figure 13 – Components of power consumption of the hybrid model on the 4 minute simulation.

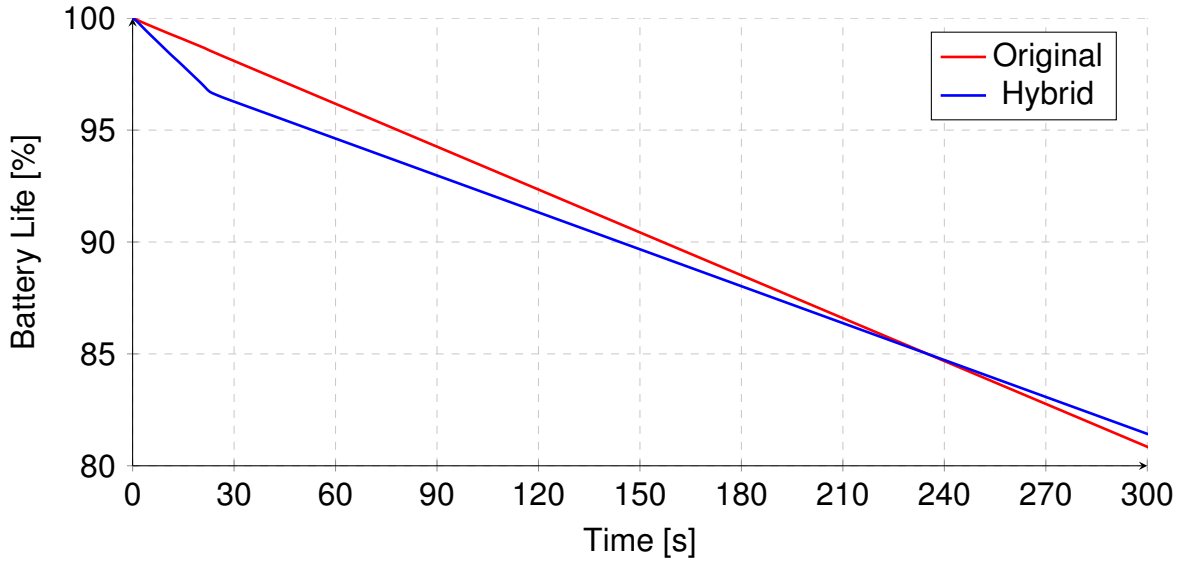


Source – original.

Another way of looking at power consumption is through the battery life of the drone. For safety reasons, the minimum battery level for any maneuver that is not part of landing phase is 20%. This rule keeps a safe margin for the drone to land, on most scenarios. Therefore, we can analyze the travelled distance, as well as the flight autonomy, for both models to have an idea of what the difference in power usage means during flight.

For the comparison, both models ran a simulation that was 21 minutes long, in order to catch the long-term effects of the power usage through the estimation of battery life. Since the hybrid model has larger mass, when the wings are not operating the battery is drained faster. However, as soon as the Magnus wings are turned on and start effectively generating lift for the drone, the decrease rate of the battery life is reduced. That reduction, after around 235 seconds (less than 4 minutes), puts the hybrid model in advantage over the original one. This can be seen in Figure 14, that shows the battery life estimation for the first 300 seconds of the simulation.

Figure 14 – Estimation of battery life of both models for the first 5 minutes of a 21 minute simulation.



Source – original.

Since the battery life decreases at constant rate for both models after the Magnus wings start operating, the long term effect can be – even intuitively – derived from this graph. This closer look at the initial 5 minutes shows that the Magnus wings make for a decrease in the absolute value of the slope of the battery life curve, only on the hybrid model, naturally.

2.3.2 Different scenarios with wing

Moving on, more simulations had the goal of understanding how different parameters – such as cruise speed and spin ratio – influence the hybrid drone model.

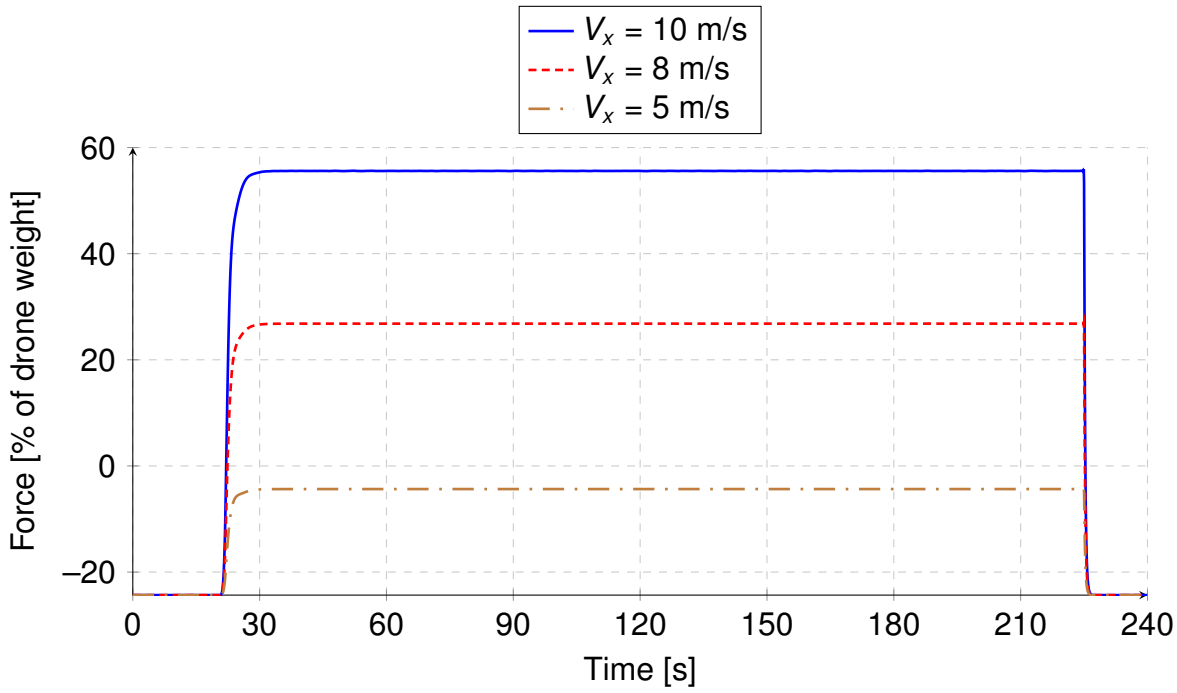
The reference scenario uses cruise speed as v_x = 10 m/s and rotation of the spin ratio X = 2.0 (which means that the Magnus wing rotation is ω_M = 880 rad/s). In all scenarios, height is set at h_d = 10 m. The simulations are 4 minutes long, and count with take-off, cruise flight and idle phases, as the previous ones.

Lift Force

In Figure 15 we can see the lift force on Z-axis, generated by the Magnus wings, in three different scenarios: the reference one, in solid blue, with cruise speed as v_x = 10 m/s; and two more, with smaller cruise speeds of 8 m/s and 5 m/s, in dashed red and dash-dotted brown, respectively. Here, the spin ratio is kept constant as X = 2.0, which means the rotation speed is set according to each cruise speed.

Instead of absolute values, the graph presents the lift generated by the Magnus wings in terms of the compensated weight of the drone. That is, the weight that the Magnus wings are lifting, **besides** their own weight. This *spare* lift represents the general lift gain of the hybrid model in respect to the original one.

Figure 15 – Lift force generated by Magnus wings on Z-axis, three different scenarios. Force is expressed in terms of percentage of the drone's weight, only considering spare lift.

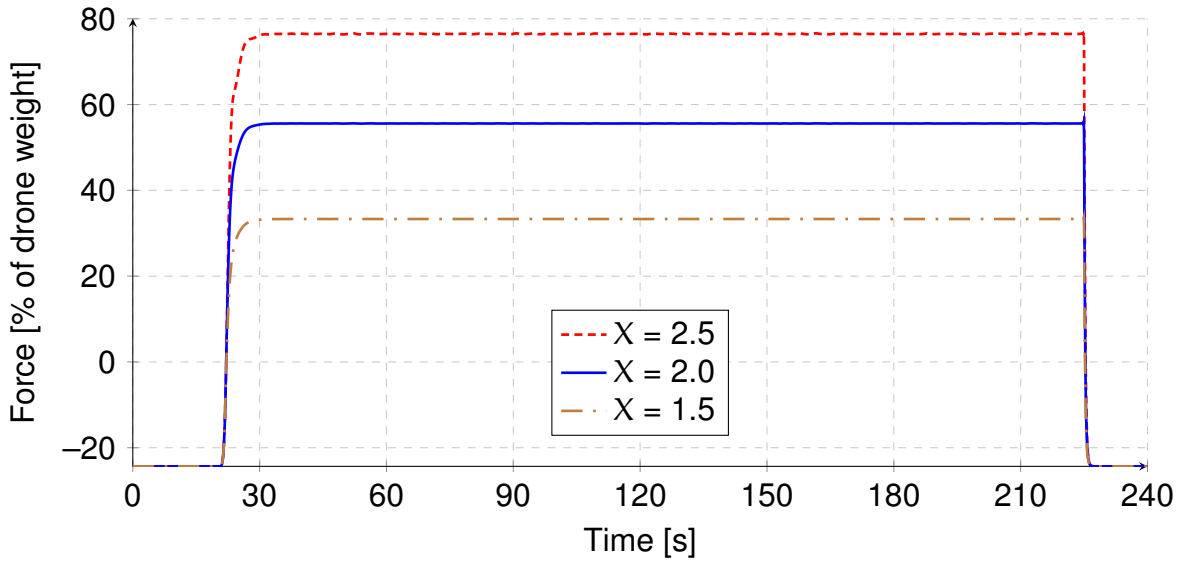


Source – original.

In the reference scenario, the lift in cruise flight phase settles around 3.86 N, which is translated to nearly 55% of the drone's weight, since the wings have a mass of 0.120 kg, and the rest of the drone of 0.493 kg. With cruise speed of 8 m/s, the weight compensation reaches 26.8% of the drone's weight, while with cruise speed of 5 m/s, the lift generated is not enough to compensate the weight of the wings, hence the negative values during cruise flight. In all cases, when the wings are turned off (take off and idle phases), the weight compensation marks negative 24.34%, indicating the ratio between the weight of the wings and the drone: $\frac{0.120}{0.493} \cdot 100 = 24.34\%$.

In Figure 16, the comparison brings the reference scenario – in solid blue –, along two other with spin ratio $X = 2.5$ and 1.5 – in dashed red and dash-dotted brown –, meaning the rotation speed of the Magnus wings is $\omega_M = 1000$ rad/s and 600 rad/s, respectively.

Figure 16 – Lift force generated by Magnus wings on Z-axis, three different scenarios. Reference scenario (solid blue) against scenario with spin ratio X = 2.5 (dashed red) and 1.5 (dash-dotted brown).



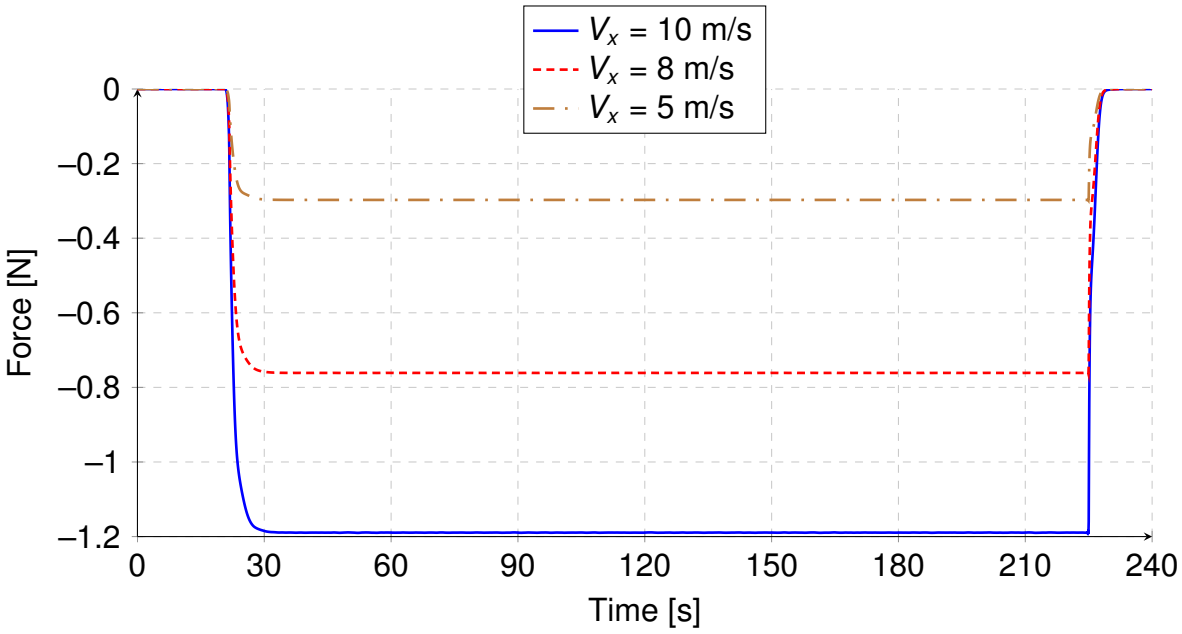
Source – original.

Naturally, the increase in spin from 2.0 to 2.5 leads to an increase in lift force, while the decrease from 2.0 to 1.5 leads to a decrease in lift force. That can be easily concluded from the polynomial that defines lift coefficient, around the interval [1.5, 2.5] for spin ratio (in Figure 7). For the increased spin ratio, the hybrid model compensates around 76.50% of the drone's weight, while for the decreased spin ratio, around 33.32%.

Drag Force

In Figure 17 we can see the drag force on X-axis, generated by the Magnus wings, again in the same three scenarios: the reference one, in solid blue, with cruise speed as v_X = 10 m/s; and two more, with smaller cruise speeds of 8 m/s and 5 m/s, in dashed red and dash-dotted brown, respectively. The drag is negative because the positive X-axis is forward, and the drag force opposes the movement of the drone.

Figure 17 – Drag force generated by Magnus wings on X-axis, three different scenarios. Reference scenario (solid blue) against scenario with cruise speed V_X = 8 m/s (dashed red) and 5 m/s (dash-dotted brown).

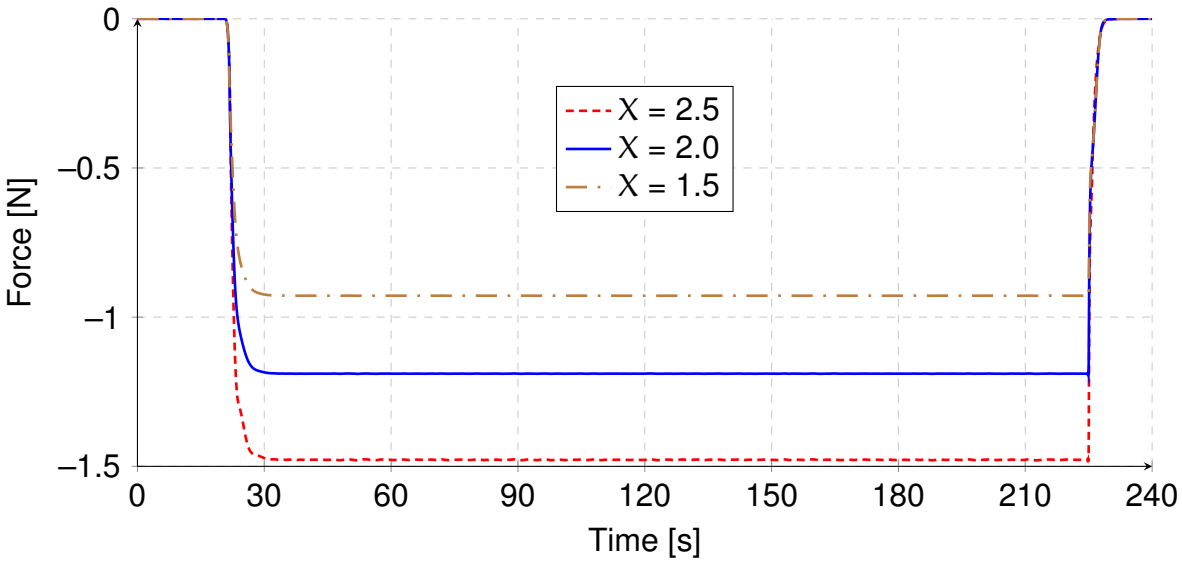


Source – original.

As expected, the absolute value of the drag force increases as the cruise speed is increased, since the relative speed is increased.

Next, in Figure 18, the increase in spin ration leads to an increase in drag force, and a decrease in spin ratio leads to a decrease in drag force, similar to the lift force behaviour, and also supported by the drag coefficient polynomial, presented in Figure 7.

Figure 18 – Drag force generated by Magnus wings on X-axis, three different scenarios. Reference scenario (solid blue) against scenario with spin ratio X = 2.5 (dashed red) and 1.5 (dash-dotted brown).



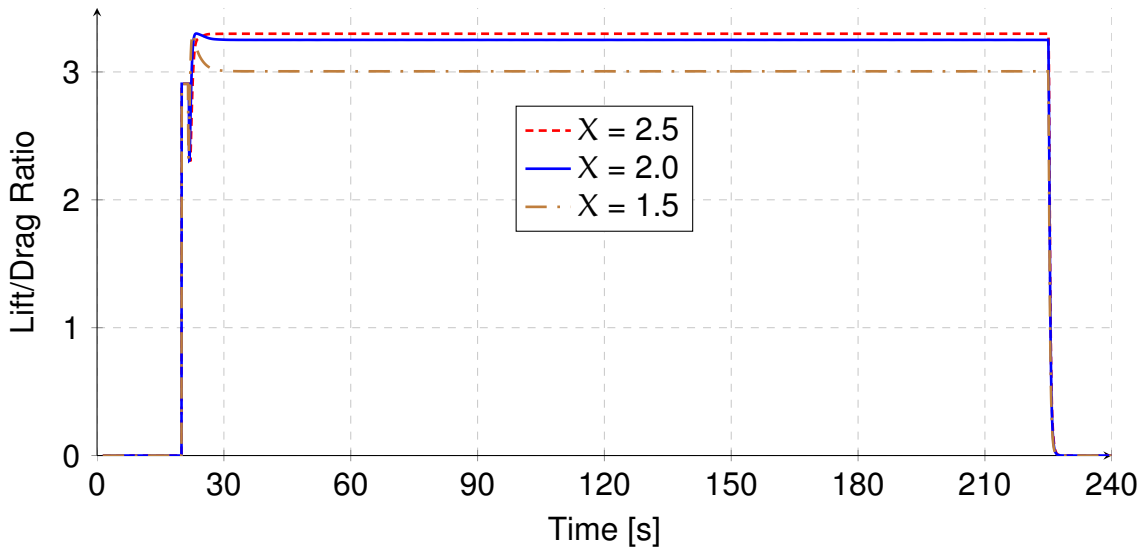
Source – original.

Lift/Drag Ratio

When analyzing the effects of a wing, one interesting parameter is the ratio between lift and drag forces generated by the wing. In this case, it depends on how we compute the coefficients for Lift and Drag, since all the other parameters are equal for both forces.

In the previous comparisons, when altering cruise speed, the spin ratio was kept constant. Therefore, there were no changes in the ratio between lift and drag. However, in comparisons between scenarios with different spin ratios, the lift/drag ratio changes. In Figure 19 we can see the lift/drag values for the three already presented scenarios, X = 2 (reference), X = 1.5 and X = 2.5.

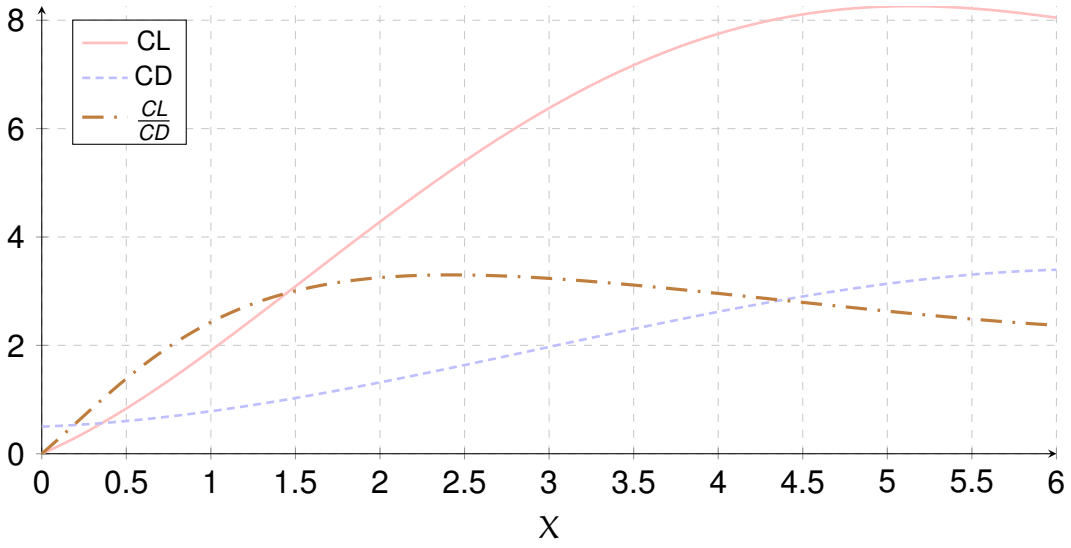
Figure 19 – Lift over drag ratio related to the Magnus wings, reference scenario (solid blue) against scenarios with spin ratio X = 1.5 (dash-dotted brown) and X = 2.5 (dashed red).



Source – original.

It is clear that the decrease in spin ratio causes an obvious decrease in the lift/drag ratio, which makes sense given the computations of lift and drag coefficients. If we divide the polynomials, we can see that the slope of the curve from $1.5 \leq X \leq 2.0$ is greater than the slope from $2.0 \leq X \leq 2.5$, which corroborates the data in Figure 19. We can also note from Figure 20 that just before X= 2.5 we have a global maximum on the function.

Figure 20 – Ratio between C_L and C_D (dash-dotted brown) along with C_L and C_D polynomials (solid light red and dashed light blue, respectively).



Source – original.

From that, we reinforce that the model for the coefficients for lift and drag forces are a key aspect of the hybrid model, and that the apparent wind (cruise speed in the simulations, since there was no actual wind) and the spin ratio are important in-flight aspects to track the efficiency of the Magnus wings.

3 GAZEBO/ROS

As we saw in the previous chapter, the original drone model used for the analysis was implemented using *Simulink* in Matlab environment, as was the first implementation of the hybrid model. Although *Simulink* is a very powerful tool, it lacks some key aspects, such as portability, for exporting models into a controlled open-world simulator as, for example, Gazebo offers. Therefore, the next step was to create a model for the hybrid drone on a more versatile platform. The choice naturally led to the ROS/Gazebo platform, already used in the laboratory, and very well established in the community in respect to robot simulation.

The goal is to simulate the drone in a *Software in the loop* strategy, using the tools used in the laboratory to control the real drone on the simulated hybrid one. The Gazebo structure, basically, allows us to create a model file that contains physical properties, defining links and joints and its relations, and also adding the so called *plugins* to the model. Plugins are chunks of code that execute specific tasks on the Gazebo world or model. They are usually implemented in `C++` language and can effectively use the ROS packages to communicate and control aspects of the simulation.

3.1 HYBRID DRONE MODEL

Firstly, the wings of the hybrid model were added to the original model as a permanent addition, but soon it became clear that a better approach would be to have an attachable module of the wings, allowing the developers to add it to any other aircraft models, as desired. That also required an easily customizable model, and Gazebo does not offer a very straight-forward solution to that type of demand.

The currently used format to describe models on Gazebo is called SDF, and it is *"an XML format that describes objects and environments for robot simulators, visualization, and control"* as stated in (OSRF, 2020). As usual in Extensible Markup Language (XML) formats, there is no built-in feature to parameterize arguments and create functions or variables, therefore making the description of the model very repetitive and hard to maintain or update. One of the consequences of this is that any numeric value representing an important parameter, for example the mass of the wing, needs to be copied as a number to each point in the XML description. Should this value be updated, the developer would need to carefully follow all the model description, understanding *the meaning* of each tag, to update all its instances.

In order to avoid this extensive unpractical work, the solution found and implemented was to embed Ruby code to the XML description, allowing to escape the limitations of raw SDF by adding specific signatures that tell the compiler where it should look for the compilation of a Ruby script. This feature is implemented using the `ERB` command, from the ERB library, which is an implementation of Embedded Ruby

(eRuby), described in detail in (SEKI, 2020).

With that approach, a model file with eRuby is created, and all the parameters can then be referenced as variables in the code structure. Computations of values such as inertia of the wings can also be made, making it easier to change the few parameters that define the wing and its position on the model. In Figure 21 we can see a short sample of code, describing only a few aspects of a wing, how it should be written without the use of eRuby, using only raw SDF.

Figure 21 – Sample of SDF model file without the use of embedded Ruby.

```
<sdf version='1.4'>
    <model name='wings'>
        <link name="left_wing">
            <pose>0 0.14 0.035 1.570796 0 0</pose>
            <inertial>
                <mass>0.06</mass>
                <inertia>
                    <ixx>0.00013125</ixx>
                    <ixy>0</ixy>
                    <ixz>0</ixz>
                    <iyy>3.75e-05</iyy>
                    <iyz>0</iyz>
                    <izz>0.00013125</izz>
                </inertia>
            </inertial>
        </link>
    </model>
</sdf>
```

Source – original.

By defining variables on an “eRuby header” on the top of the file, as in Figure 22, we can use those throughout all of the model file, as seen in the much simpler version of the sample code in Figure 23.

Figure 22 – Example of usable header written in Ruby for the use of eRuby on model files.

```
<%
    # Geometry
    wing_mass   = 0.06
    wing_height = 0.025
    wing_width = wing_height
    wing_depth = 0.15

    # Position
    wing_y = 0.14
    wing_z = (wing_height/2.0).round(6)
    wing_roll = (3.14159265359/2).round(6)

    # Inertia
    wing_ixx = ((1.0/12)*wing_mass*(wing_height**2 + wing_width**2)).round(10)
    wing_iyy = ((1.0/12)*wing_mass*(wing_height**2 + wing_depth**2)).round(10)
    wing_izz = ((1.0/12)*wing_mass*(wing_depth**2 + wing_width**2)).round(10)
%>
```

Source – original.

Figure 23 – Sample of SDF model file with the use of embedded Ruby (header not shown).

```
<sdf version='1.4'>
    <model name='wings'>
        <link name="left_wing">
            <pose>0 <%=wing_y%> <%=wing_z%> <%=wing_roll%> 0 0</pose>
            <inertial>
                <mass><%=wing_mass%></mass>
                <inertia>
                    <ixx><%=wing_ixx%></ixx>
                    <ixy>0</ixy>
                    <ixz>0</ixz>
                    <iyy><%=wing_iyy%></iyy>
                    <iyz>0</iyz>
                    <izz><%=wing_izz%></izz>
                </inertia>
            </inertial>
        </link>
    </model>
</sdf>
```

Source – original.

With this adaptation, we can go on to the construction of the hybrid drone model as a whole. One thing that should be noted is that only one model of a wing would be needed, since both wings (for left and right sides) are the same, interchangeable. This

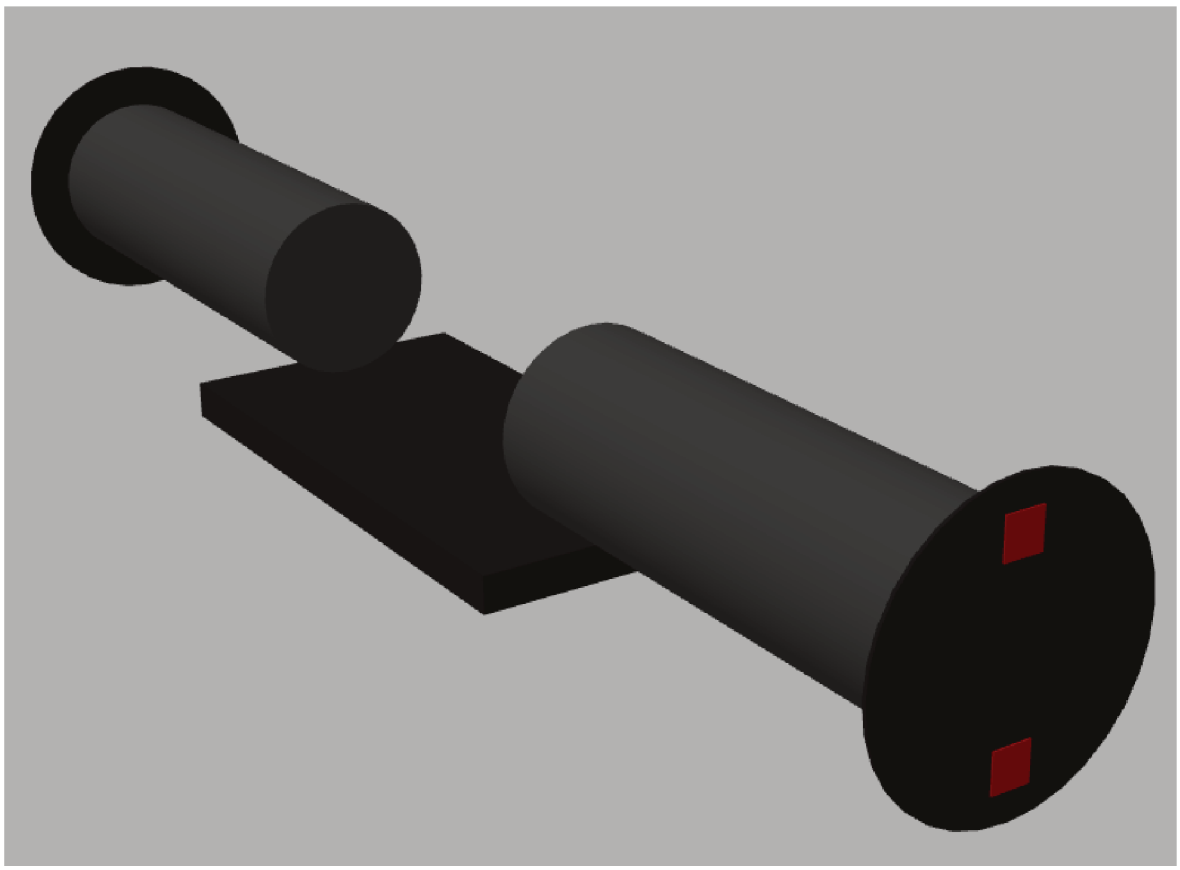
same model could be attached twice, with different position and orientation. However, the structure for the fixation of the Magnus wings on the real hybrid drone prototype is such that both wings are first attached to a common frame, which is then attached to the drone.

In order to follow closely the structure of the prototype, for the hybrid model file, a single model containing both wings and the common frame was created, referenced as `wings` model. Further modularization could be achieved by constructing a single wing and using it twice for the `wings` model, yet it was considered unnecessary.

We can now take a look at the main drone model, to which the wings will be attached. The model is present in PX4 Drone Autopilot's repository `sitl_gazebo`, available at (PX4, 2020), and is named "Iris". This repository uses another repository resulted from (FURRER et al., 2016) to model the rotors of the drones. The Iris model was already used for simulations in the laboratory, related to drone modelling and control. Hence the choice to keep it for the hybrid drone simulations. In Figure 1, the simple visuals of the Iris model were presented.

In order to compose the hybrid structure, then, we can see a simplified model of the Magnus wings (and the common frame for fixation), in Figure 24. The model is composed by a `base link` – which is the support structure –, two cylinders that represent the Magnus wings – each one with its own Thom disc at the outer edge –, and four red markers – two in each disc, for visual effects. It is worth noting that the joints, that connect each wing to the base link, don't need visual elements, therefore they don't appear in the image. Still, the model has two revolute joints located at the center of the wings, inner edge.

Figure 24 – Example of Magnus wings model visuals on Gazebo GUI.



Source – original.

By attaching the Magnus wings model to the Iris model, through a fixed joint, we get the full hybrid drone model, and its visuals were presented in Figure 2.

The new hybrid model was validated using the project's trajectory generator, before moving on to the development and addition of plugins. Using it, we can easily set up a desired trajectory for the drone, while the intercommunication with the low-level controls is handled by PX4's autopilot (flight controller). This is to make sure that the model file works as expected, since without the plugin there are no computations of lift, drag or any other forces and moments, and there is no difference for the drone controller besides the added mass of the new structure, which is very small, following the real values.

3.2 PLUGIN

After the validation of the hybrid drone model, the plugins could be added, in order to model the behaviour of the forces and moments involved in the new configuration of the drone. The wing plugin was developed in `C++`, and it can access most of the variables related to the simulation, the links and joints, and the model itself. The

plugin describes the behaviour of the wings when interacting with movement around the simulated world, with wind and with the attachment to the drone. Only one plugin is needed, and it is attached twice to the model, once in each wing.

Since the plugin needs to communicate with elements external to the simulation – such as reference command for the rotation speed of the wings –, a communication platform based on ROS topics was implemented and tested, and a ROS control node also had its implementation started, in order to control the new variables from outside of the plugin and Gazebo simulation, taking into account the new structure of the system. That is an important step because the goal of the plugin is to model the effects of the physical entities of the drone model, not to properly control any variable of interest.

The main focus of the external controller is to control the lift and drag forces generated by the wings. We saw in previous chapters that these forces depend on a specific parameter of the wing – the spin ratio (X). Thus, the controller uses the angular speed of the wings – commanded by DC motors – to change the spin ratio, directly affecting lift and drag generated by the wings.

It is important to note that, in simulation, obtaining the real velocities is a fairly trivial task, allowing for an easy computation of related variables, such as spin ratio that depends on the apparent wind. That is not the case in a real world scenario, and the implementation of control and estimation algorithms, as well as instrumentation related to the measured variables, is a critical concern when moving on from the simulation environment.

Continuing the plugin description, one important aspect is that it is responsible for computing the forces of lift and drag, on each wing, and applying it to the correct points on the model, taking into account the different frames of reference. The different frames arise from different ways to represent data in an adequate way since, as already discussed, some variables can be naturally understood from a specific frame of reference, while others make no sense in that same frame.

As an example about this difference in coordinate frames: the link's velocity vector (velocity vector of a wing) is defined in a global reference frame, while the application of forces is given in the local/body frame, therefore a rotation is needed. The data available to the plugin is in the form of a 3-dimensional `Pose`, that provides a quaternion indicating the rotation of the body relative to the global world frame. In the scope of the plugin, there are methods that transform quaternions to rotation matrices and rotate them, along with many other functionalities, making the change between reference frames a trivial task inside the plugin.

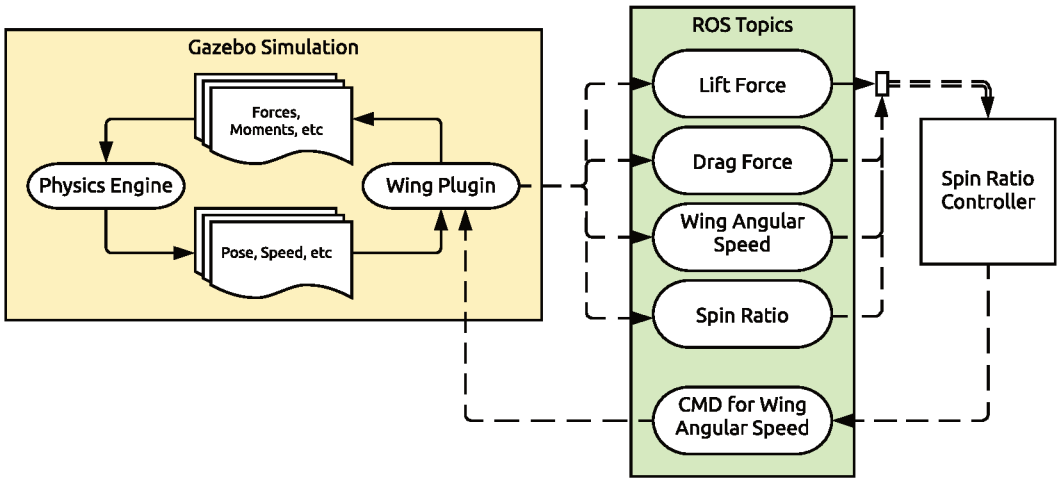
Any other forces or moments caused by the Magnus wings are also computed by the plugin and applied making the above considerations. Specifically, the dynamics of the angular speed of the wings are also described by the plugin, and therefore the real value of this speed can differ from the value published by the controller on the designed

ROS topic – the main channel between external controller and simulation plugin.

The plugin also subscribes to another ROS topic, to which the spin ratio controller publishes, giving the command data for the plugin to compute the real rotation speed of the wing. Besides that, the plugin publishes to other topics the values of the real angular speed (control variable) and the computed lift and drag forces and spin ratio – each on their own topic. Any other data the plugin needs comes from a direct interface between Gazebo Physics engine and the Gazebo plugin.

In Figure 25 we can see a diagram indicating the simplified data flow through the system. The solid lines represent internal communication, that is given between instances of different elements inside the Gazebo simulator, while the dashed lines represent the act of publishing or subscribing. The arrows point from publisher to topic to subscriber.

Figure 25 – Simplified chart showing data flow between the simulation elements on Gazebo, the ROS Topics and the spin ratio controller.



Source – original.

The plugin was built as to also run checks for some possible inconsistencies in the model file, such as missing tags that lead to undefined model parameters. In some cases, in the absence, the plugin sets a default value, in other cases – more critical ones – it raises an error and does not load, enforcing the explicit declaration of those parameters.

The validation of the plugin used the same trajectory generator used in the validation of the hybrid model, placing the hybrid drone in known positions and setting its speed, checking the profile of the forces and moments resulting from the plugin computations and evaluating the simulation output.

4 TRAJECTORY GENERATION

One of the challenges regarding control of a quadcopter drone is trajectory generation. The position controller is limited due to the physical limitations of the hardware on the drone. Thus, if there is no concern about continuity or physical constraints on the given references, it might lead to inadequate – and even dangerous – in-flight situations. The drone has limitations on speed and acceleration, that are at times neglected at the trajectory generation phase. Seeking to produce a feasible trajectory for the drone during this phase, a script to adapt trajectories (defined as points in 3D space) to speed, acceleration and jerk constraints was developed, by use of spline interpolation.

4.1 ALGORITHM

The basic structure of the process starts with a set of points (there are variations for 1D, 2D and 3D points) and the constraints on speed, acceleration and jerk. The first step is to create a time reference, which is how long the drone would take to visit all points by flying at the maximum allowed speed in a straight line between each two points. This value is used as the `minimum travel time`, though it is, in most cases, unachievable. Then, the degree of the polynomials used to create the splines, to interpolate the points, can be chosen. Values can range from 3 to around 20, but experiments indicated that a degree around 6 generates good results. To ensure continuity on speed and acceleration, at least 3rd degree is necessary for the position curve (to be generated). However, that usually does not comply with jerk limitations. A 4th order spline position curve ensures a linear jerk. It was observed that, with 2nd or 3rd degree jerk profiles, the constraints were more easily respected.

The next step is to create a curve using spline interpolation. The first dimension of the function is always time, and subsequent dimensions are added as desired. This curve will portray a continuous flight path passing by every point from the initial reference, and finishing at the so-called `current travel time` (which will be compared to the `minimum travel time`).

Then the derivatives, until the 3rd order, are computed and checked against each respective constraint. Since the function is piecewise polynomial, it is trivial to compute its derivatives. If there is any violation of the constraints, the current travel time is increased, as to allow the function to *stretch* along the time vector, decreasing the derivatives and making it smoother – possibly landing the derivatives all inside their boundaries. This stretching is done by multiplying the `current travel time` by a factor (usually very close, but always above 1). From this point on, the algorithm repeats the steps of computing the derivatives, checking them against the constraints and stretching the time vector – if necessary. Given this dynamic of “stretching” the time vector, the algorithm is referred to, originally, as **Time-Stretching Trajectory Generation**.

It is interesting to note that an execution using a factor very close to 1 will usually take more iterations to complete, while a greater factor should require less iterations. However, the closer the factor is to 1, the closer to the `real minimum travel time` the output gets. This `real minimum travel time` is the lowest travel time that generates a trajectory adequately bounded by the derivative constraints for that specific spline function. Given the continuity of the function, if the initial `minimum travel time` violates the constraints of the drone, the following is guaranteed:

$$t_{ini} \leq t_{real} \leq t_{cur} \, ,$$

with t_{ini} representing the `minimum travel time`, t_{real} the `real minimum travel time`, and t_{cur} the `current travel time`.

Another point is that the use of spline interpolation ensures that the complexity of the curve doesn't scale too much, since the interpolation uses `(n-1)` points at a time for a `n`th degree spline, connecting each segment in a convenient way. That – besides allowing for curves with many segments, making them larger than an approximation purely on high-order polynomials could achieve – also helps avoiding *Runge's phenomenon*.

One of the highlights of the algorithm is that it allows for choosing the degree of the polynomial used for the spline generation. It is also possible to define values for the derivatives at any number of specific points (such as start and end), making it easy to integrate different trajectories (generated through this method or not). For soft start and end, for example, it can be defined that speed and acceleration must be zero at the first and last points. In Algorithm 1, there is a simple pseudocode of the script:

Algorithm 1: Time-Stretching Trajectory Generation

Input : A set of points in space $p \in P$, a desired polynomial degree n, a sample frequency f_s, stretching factor k_t, constraints v_{max}, a_{max}, j_{max}.

Output : A set of points of the spline curve $s \in S_p$ representing the trajectory, a time vector T.

```
1  t_ini ← 0;
2  for i ← 1 to (length(P) - 1) do
3  |   t_ini ← t_ini + |p_{i+1} − p_i| / v_max;
4  end
5  t_cur ← linspace(0, t_ini, f_s · t_ini);
6  p_traj ← spline(n, t_cur, P);
7  v_traj ← diff(p_traj, t_cur);
8  a_traj ← diff(p_traj, t_cur);
9  j_traj ← diff(p_traj, t_cur);
10 while (failsConstraints(v_traj, v_max)
11         OR failsConstraints(a_traj, a_max)
12         OR failsConstraints(j_traj, j_max)) do
13 |   t_cur ← t_cur · k_t;
14 |   p_traj ← spline(n, t_cur, P);
15 |   v_traj ← diff(p_traj, t_cur);
16 |   a_traj ← diff(v_traj, t_cur);
17 |   j_traj ← diff(a_traj, t_cur);
18 end
19 S ← p_traj;
20 T ← t_cur;
```

On the pseudocode, the description of some functions is absent in order to simplify the structure. In Table 1 those descriptions are presented more thoroughly. The main development of the algorithm took place under *MATLAB*, though the names of the functions do not necessarily correspond to the known definition under *MATLAB*, since adaptations were made for the sake of presentation of the pseudocode.

Table 1 – Description of functions used in Algorithm 1.

`length(X)`	returns length of the largest dimension in `X`.
`linspace(a,b,n)`	returns array with `n` points equally spaced by $\frac{b-a}{n-1}$.
`spline(n,T,X)`	returns a spline curve `f` of `nth` degree such that `f(T(i))=X(i)`, for all `i`.
`diff(X,T)`	returns array of approximate derivatives of `X` in respect to `T`.
`failsConstraints(X,c)`	returns `True` if the absolute value of any point in array `X` is greater than `c`. Else, returns `False`.

Source – original.

In this simplified code there is also no reference to the application of boundary conditions on the function, though the actual script developed does take that into account and enables one to set boundary conditions for any of the derivatives of the function.

4.2 TESTING

In order to test the algorithm, the values chosen for the constraints were taken from real constraints of the drone, either from its hardware or from specific controllers that limit certain outputs. These values are presented in Table 2.

Table 2 – Constraints on the maximum magnitude of each variable used for the trajectory generation algorithm.

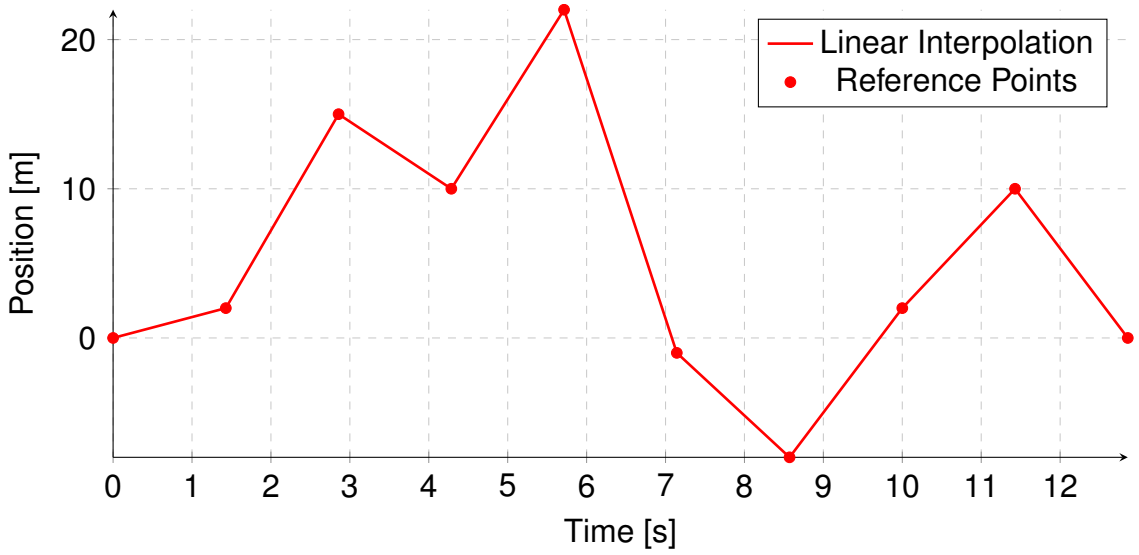
Variable	Value
Speed [m/s]	7
Acceleration [m/s^2]	10
Jerk [m/s^3]	20

4.2.1 1D Test

The results vary with the choice of the reference points used as input, naturally. For the first test, a 1D set o reference points was defined. In order to run the algorithm, another important parameter is the sample frequency, related to the overall update rate of the system. For this test, the value of the sample frequency was set as f_s = 100 Hz.

Since the `minimum travel time` considers a linear movement between each two points in the position reference, the original reference points are interpolated linearly, producing the profile in Figure 26.

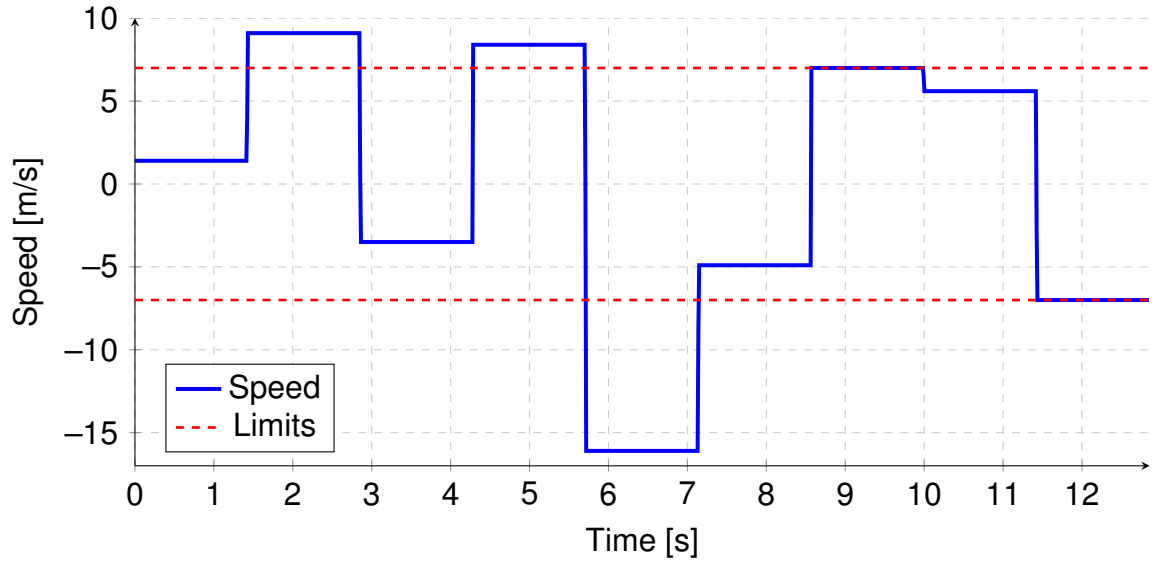
Figure 26 – Linearly interpolated function, in solid red, of the reference points, round markers in red.



Source – original.

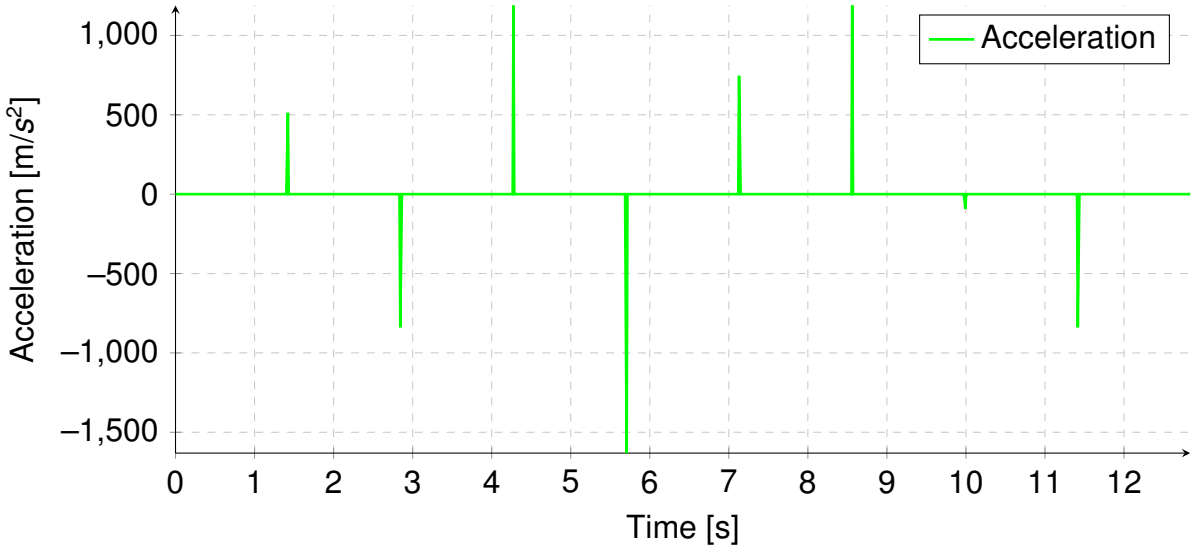
This interpolation leads to specific speed and acceleration profiles, that we can see in Figures 27 and 28, respectively. For the speed profile, the limits set by the constraints are present, while for the acceleration, given the magnitude of the data, they are absent.

Figure 27 – Speed derived from the piece-wise linear position function, in solid blue, dashed red shows the limit set by the constraint.



Source – original.

Figure 28 – Acceleration derived from the piece-wise linear position function, in solid green.



Source – original.

In this scenario, the original duration of the trajectory, the `minimum travel time`, is $t_{ini} \approx 12.86$ seconds. It is clear from the speed and acceleration profiles that the drone will not follow this trajectory as it is. In order to help the drone follow a given

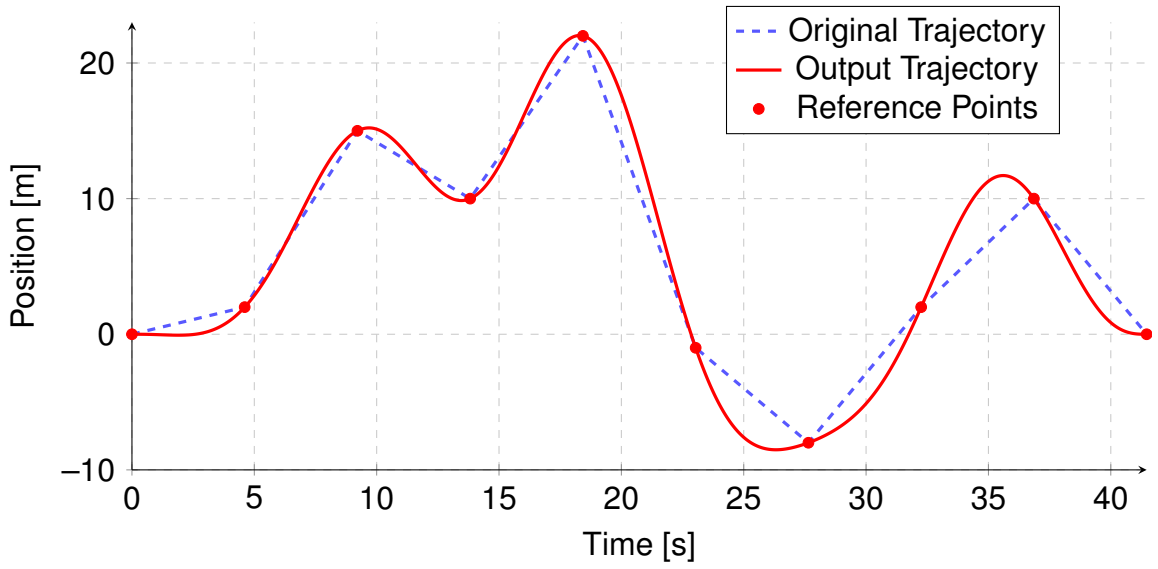
trajectory and avoid unexpected behaviour, we can use the developed algorithm to adapt this trajectory.

Running this trajectory through the algorithm, it iterates a total of 24 times, with the stretching factor set to $k_t = 1.05$. The degree for the spline interpolation was chosen as $\mathtt{n} = 6$. The total stretching in the time vector amounts to $(k_t)^{24} = (1.05)^{24} \approx 3.22$. This means that the time to complete the trajectory went from said 12.86 to $t_{cur} \approx 41.46$ seconds, getting closer to what a drone with those constraints is actually able to perform.

In Figure 29 we can see the new position reference generated by the algorithm, along with the original trajectory and the reference points indicating the desired position. Note that the original trajectory was *stretched* until it reached the duration of the new trajectory, **only** for visual aspects in the graph.

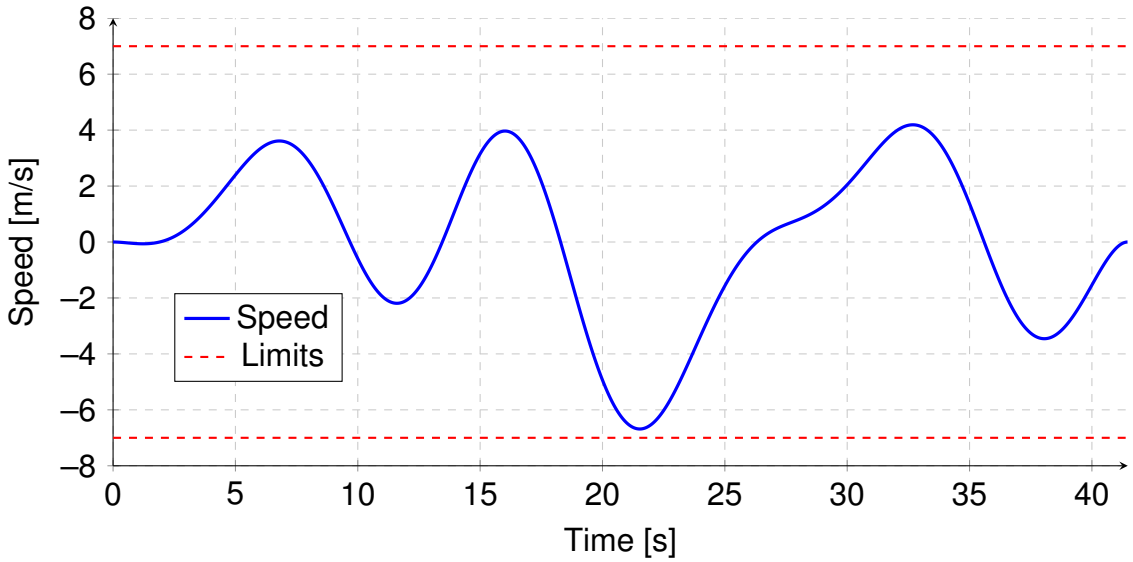
In Figure 30 we have the derived position profile, along with the visual representation of the speed constraints. Note that both the initial and final speed are zero, as this was specified as boundary condition for the algorithm.

Figure 29 – Position output, after using the "Time-Stretching Trajectory Generation" algorithm, in solid red, original trajectory (stretched for visual purposes) in dashed light blue, and red dots as the reference points.



Source – original.

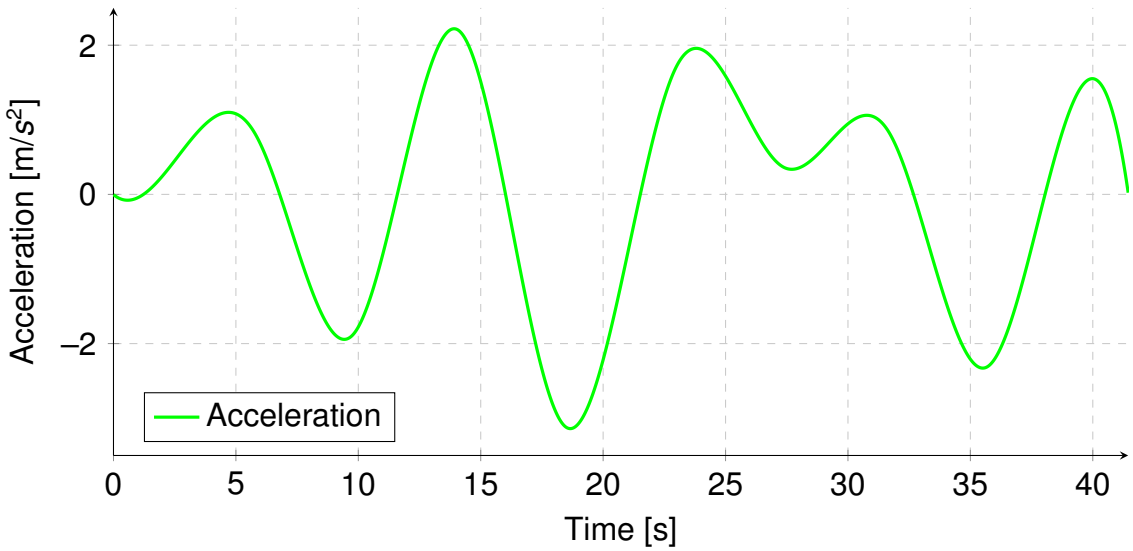
Figure 30 – Speed output, after using the “Time-Stretching Trajectory Generation” algorithm, in solid blue, and dashed red indicates the speed constraint.



Source – original.

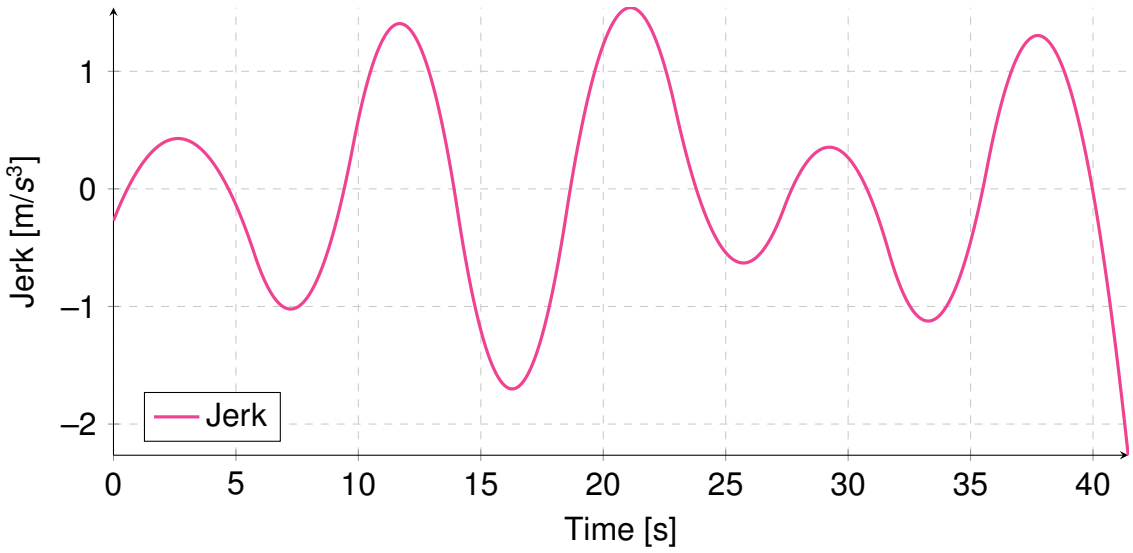
In Figure 31, there is the acceleration profile, also with boundary conditions that lead both start and end to zero. In Figure 32 we have the computed jerk for the same output. Note that it appears that these constraints are such that the speed limit is the actual bottleneck, since both acceleration and jerk profiles progress while keeping a reasonable margin from its limits. Also, given the difference in amplitude of the acceleration and, even more, the jerk curves in respect to their constraints, the graphs are shown without the limit lines, emphasizing the profile itself. The jerk, since the order of the spline chosen was 6, is a piece wise cubic spline, while the acceleration is a 4 degree spline.

Figure 31 – Acceleration output in solid green.



Source – original.

Figure 32 – Jerk output in solid magenta.



Source – original.

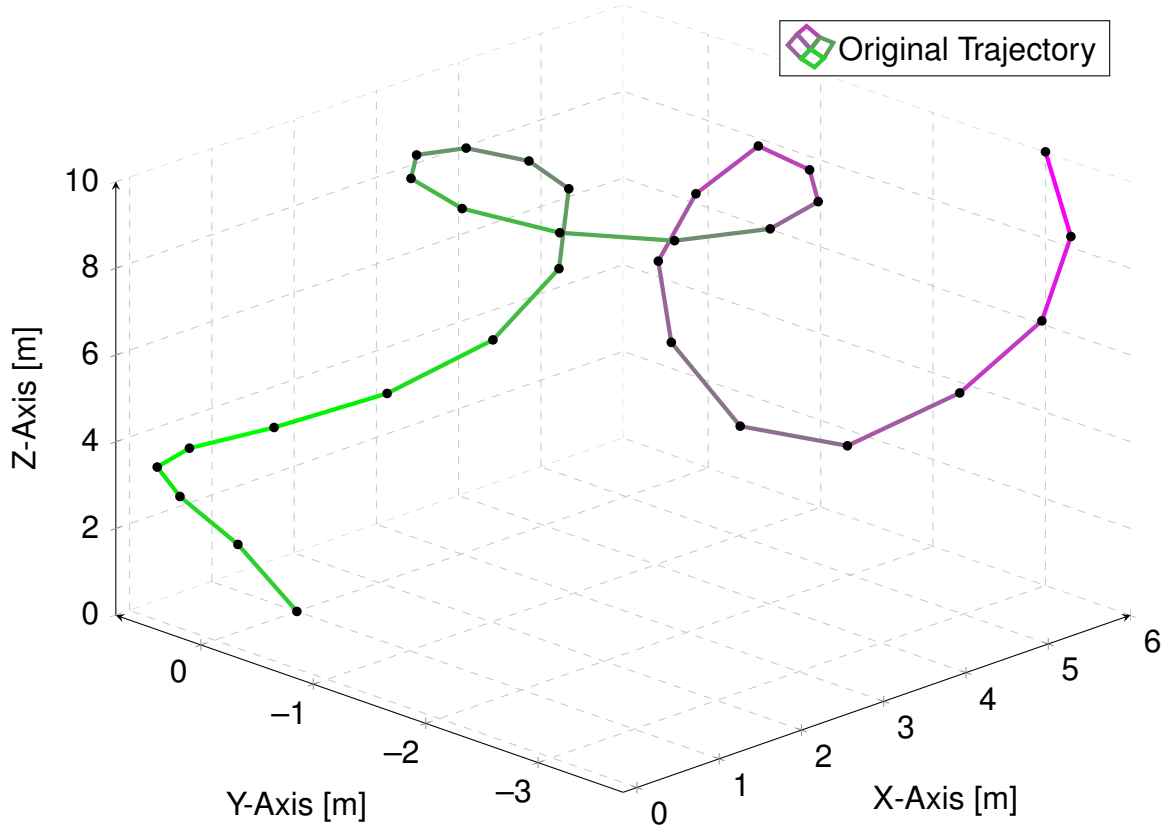
4.2.2 3D Test

In order to show another perspective on the use cases of this algorithm, a similar simulation is presented, but using 3D points as initial reference, instead of the 1D example seen in the previous section. The expansion for multiple dimensions can be done in more than one way. For this example, the approach was to separate the

components of the trajectory – namely, X, Y and Z directions – and create a specific spline curve for each one.

The trajectory chosen for this case was more complex, but not particularly uncommon in respect to obstacle avoidance or even stunt performance. In Figure 33 we can see the original reference points and the corresponding curve after linearly interpolating those points.

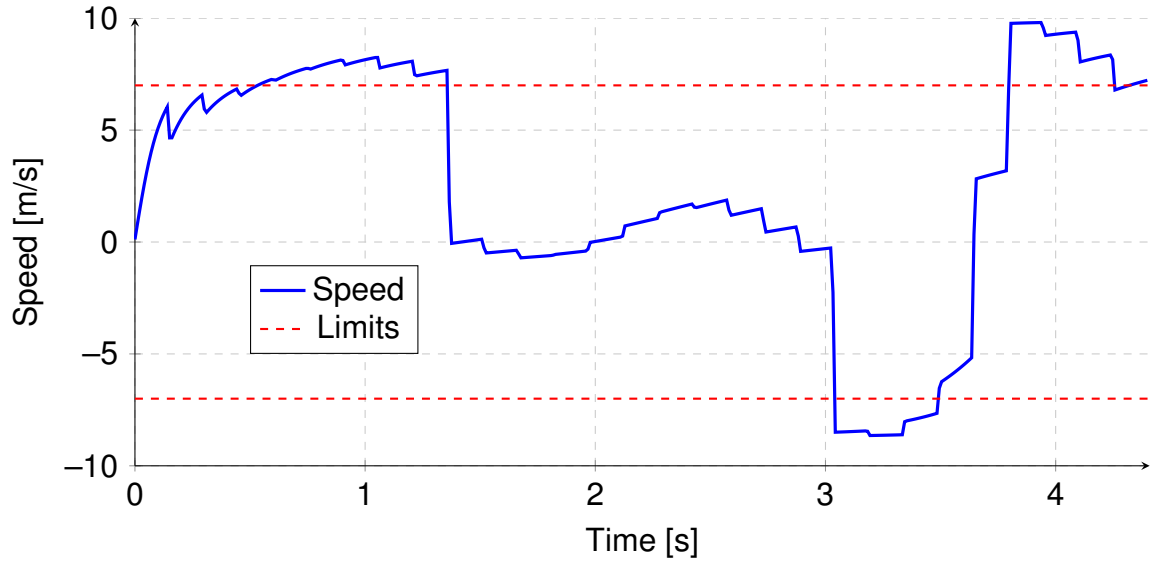
Figure 33 – Original trajectory build with linear segments, solid line from green to magenta. Reference points in black.



Source – original.

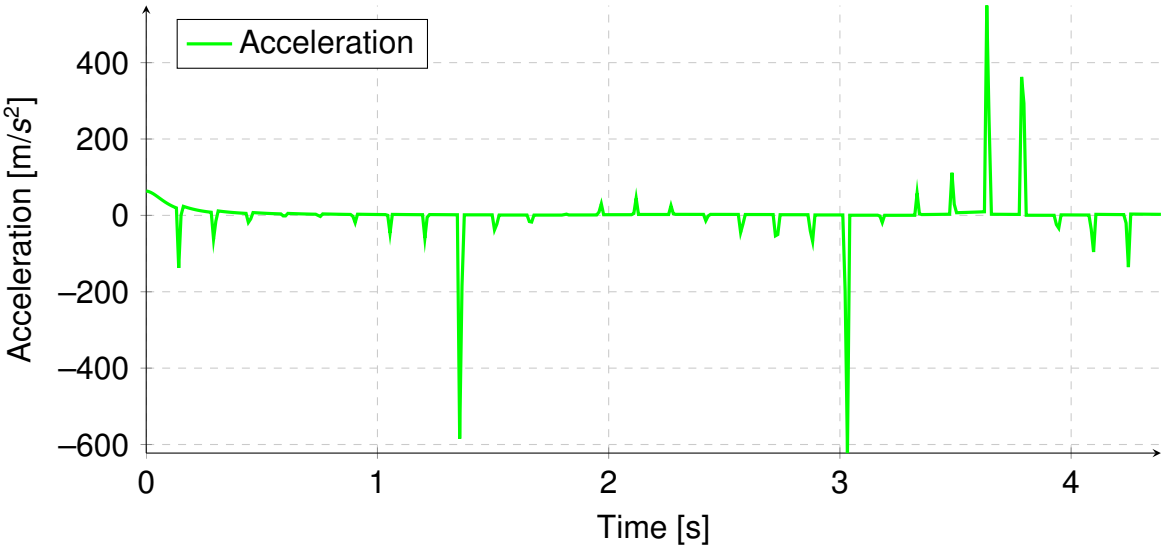
The profiles of linear speed and acceleration that result from this trajectory are in Figures 34 and 35. Here, the norms of speed and acceleration vectors are used, since the constraints apply on the linear parameters, not on specific axis. A *per axis* type of constraint could be easily implemented into the algorithm, if desired.

Figure 34 – Original linear speed profile, in solid blue, along with speed constraint in dashed red.



Source – original.

Figure 35 – Original linear acceleration profile, limits are absent given the scale of the ordinate axis.



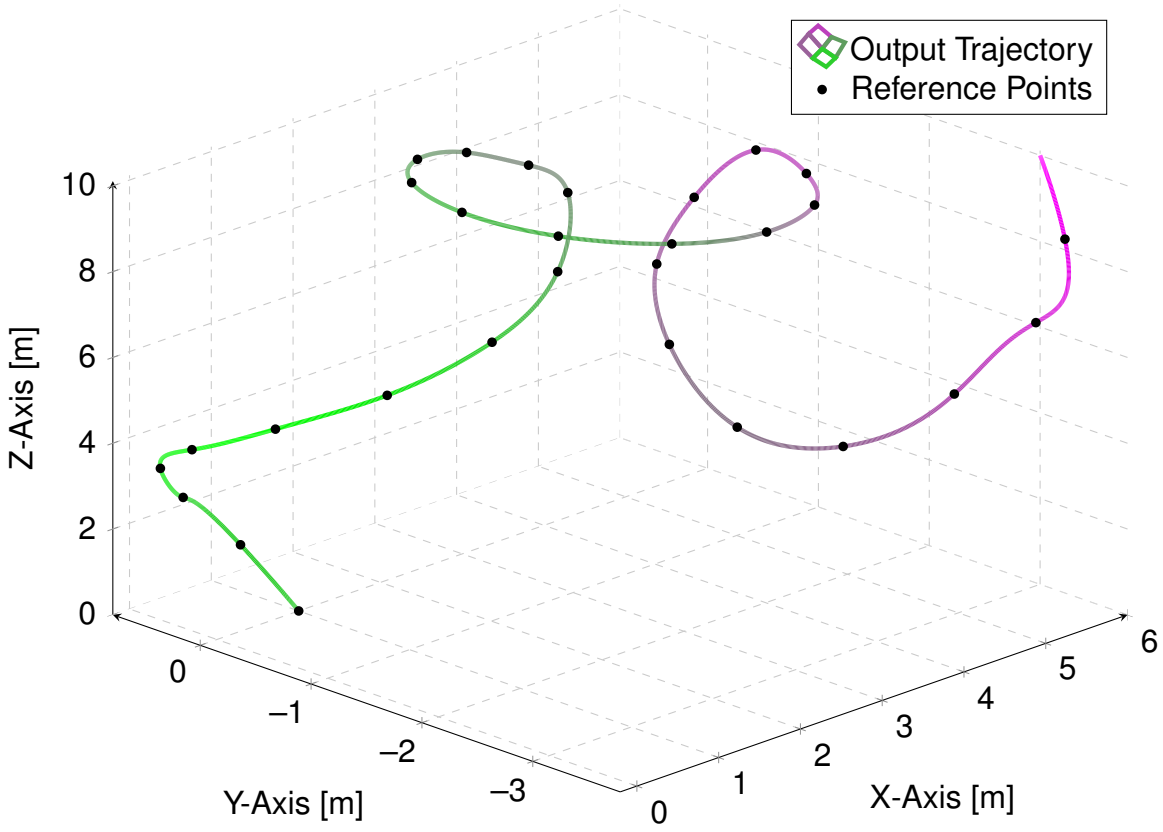
Source – original.

Although the speed profile is not that far out of the defined limits, the acceleration profile reinforces that this trajectory is not feasible. The jerk profile is omitted since it would be redundant, in a way, given the acceleration profile.

In this scenario, the original duration of the trajectory, the `minimum travel time`, is $t_{ini} \approx 4.41$ seconds. Running this trajectory through the algorithm, it iterated a total of 38 times, with the same stretching factor as the previous test, $k_t = 1.05$. The degree for the spline interpolation was also kept at `n` = 6. The total stretching in the time vector amounts to $(k_t)^{38} = (1.05)^{38} \approx 6.39$. This means that the time to complete the trajectory went from said 4.41 to $t_{cur} \approx 28.15$ seconds, getting closer to what a drone with those constraints is actually able to perform.

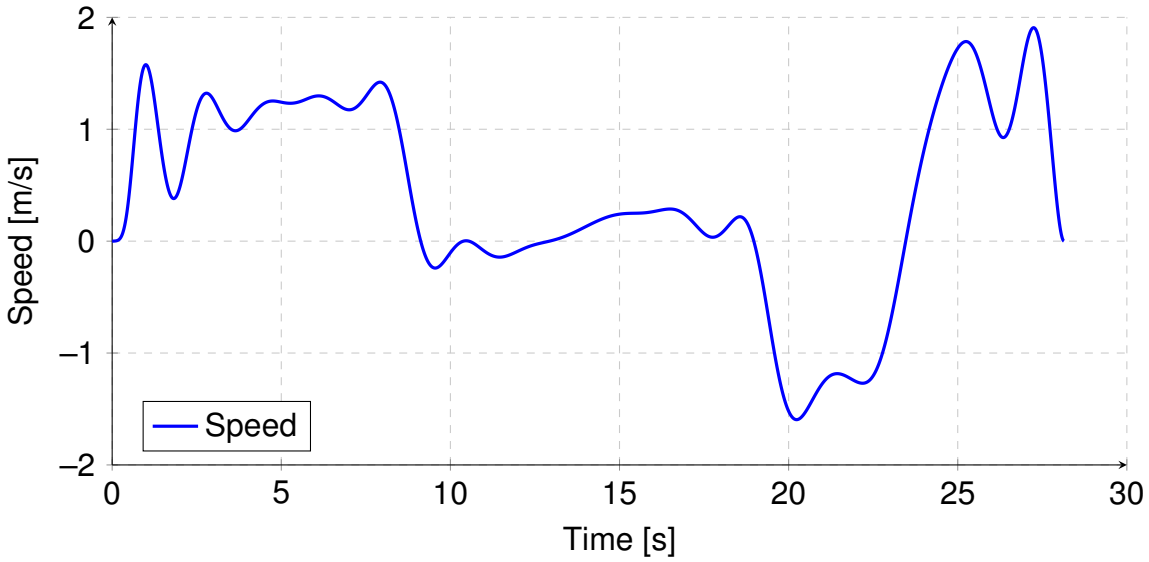
The new 3D trajectory is shown in Figure 36, along with the original reference points. In the following Figures 37, 38 and 39 we have the new linear speed, acceleration and jerk profiles, all adequately bounded by their constraints. The limits were hidden for the speed and acceleration given the scale of the plot, whilst they were kept for the jerk, for the same reason.

Figure 36 – New trajectory, solid line from green to magenta, with reference points in black.



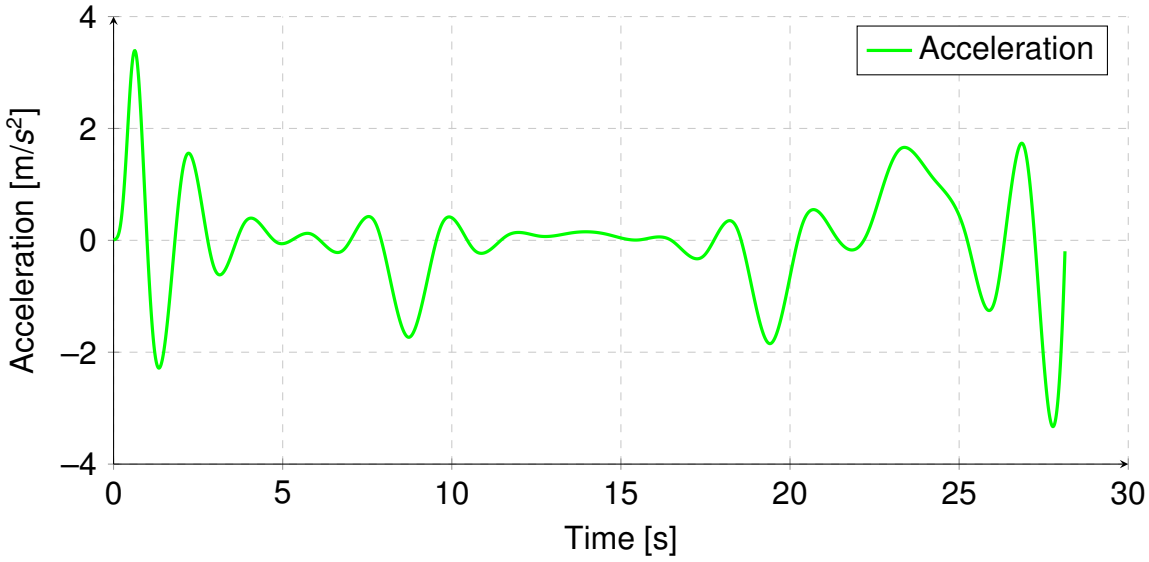
Source – original.

Figure 37 – New linear speed profile, solid blue.



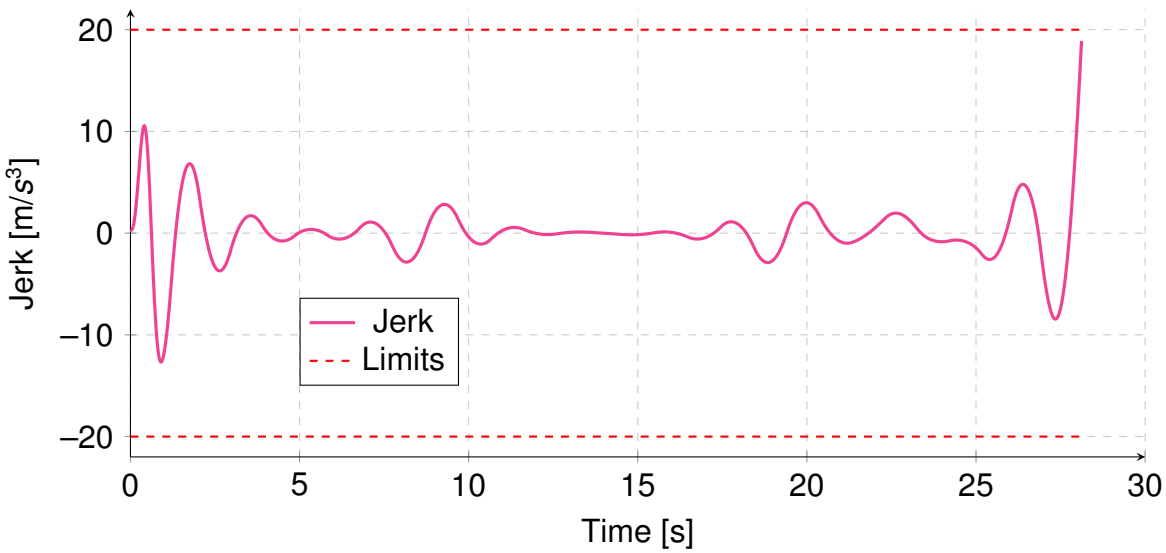
Source – original.

Figure 38 – New acceleration profile, solid green.



Source – original.

Figure 39 – New jerk profile, solid magenta, along with its constraints, dashed red.



Source – original.

Again, the algorithm succeeded in creating a continuous reference trajectory with the desired boundary conditions, taking into account the physical constraints of a drone in respect to linear speed, acceleration and jerk.

5 TESTS ON THE PROTOTYPE

In order to create a dynamic model that corresponds to the behavior of the hybrid drone, a couple of parameters regarding its functionalities are necessary. To achieve that, we have measured variables of interest during the lab tests related to power consumption, spin ratio and aerodynamical forces.

The main goal was to come up with a model for the power consumption of the wings and for the coefficients of lift and drag forces. Those elements would enable a fair enhancement to the hybrid drone model.

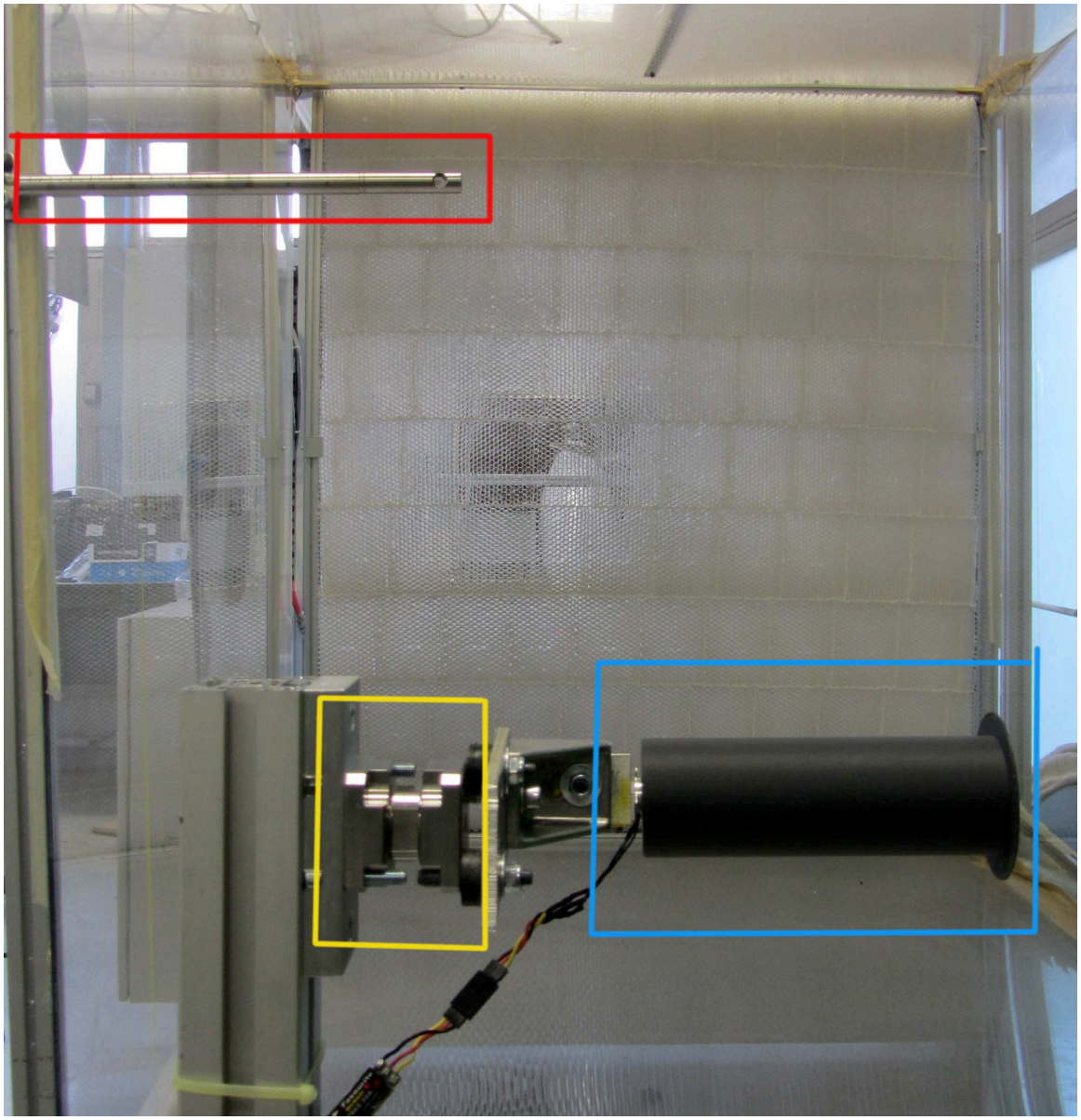
5.1 SETUP

The tests were carried out using a wind tunnel, consisting of nine 800 W brushless motors distributed over a surface of 1.2 m^2. The maximum wind speed achievable at the time of the tests was approximately 7.0 m/s. The tests were conducted on a single wing prototype, the cylinder being fixed orthogonally to the expected wind flow. The length of the wing was 0.15 m, with a radius of 0.025 m. For all the tests, the wing had Thom (THOM, 1934) discs at the edge of the cylinder.

In order to attain a good estimation of the parameters, the same test was conducted multiple times with different wind speeds. In each iteration, the electric current used by the wing motor was measured and, since the voltage was constant at 12 V, the power usage was easily derived. The rotation speed of the wing was measured with an optical tachometer, by attaching a small piece of reflective tape on the surface of the Thom disc. The measurement of wind speed was done with a hot wire anemometer, and displayed on an oscilloscope. Lift and drag forces were extracted from the 3-axis force sensor, considering upward lift and forward drag (in respect to the wind speed).

In Figure 40, a picture where part of the setup is shown, with highlighted key elements.

Figure 40 – Picture of part of the setup, including the hot wire anemometer (highlighted in red), the cylinder (in blue) and the 3-axis force sensor (in yellow).



Source – adapted from GIPSA-lab archive (2020).

5.2 POWER MODEL

When analyzing the scatter plot of the power data, it became evident that the power usage of the wing does not depend on the wind speed and, therefore, the Reynolds number. This conclusion is in line with the expectation, since at a constant voltage, the motor current would depend specifically on the motor speed, not on external factors such as the wind speed.

For that reason, initially all the data points were used to set up a model for power

usage in respect to the rotation speed of the wings. By analyzing the scatter plot of the data, a group of outliers appeared: four measurements that indicated a higher power usage than the overall expected trend of the data. Table 3 shows those values.

Table 3 – Power outliers.

Wings Speed [rad/s]	Power [W]
691,2	4,20
703,7	4,20
722,6	3,84
722,6	3,96

During the tests it was noted that the structure in which the wing and force sensor was mounted was very prone to undesired vibrations, depending on the rotation speed of the wing. Those vibrations strongly affected the force measurements, compromising part of the data, besides the fact it could damage the structure and the components. It was noted that between 670 rad/s and 730 rad/s these oscillations were very strong, given some level of resonance with the structure. It is speculated that this could cause an increase on the power consumption. Given that, the outliers were removed from the data set in order to adjust a model.

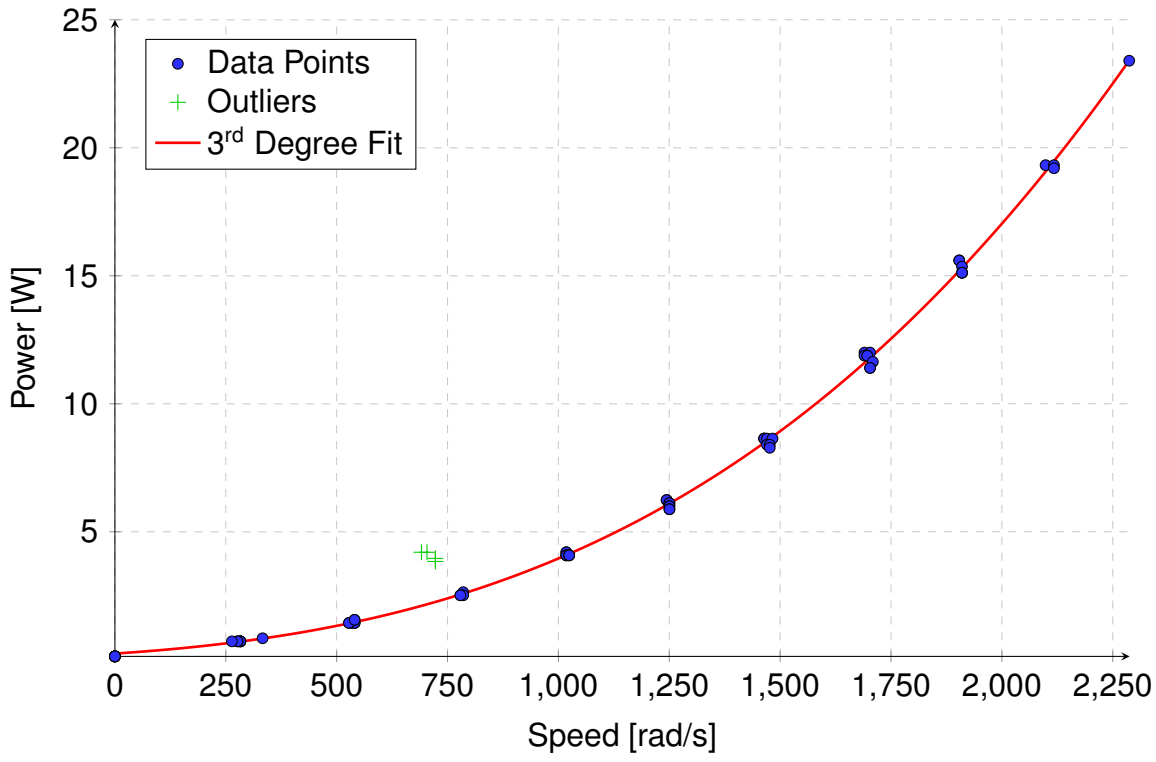
After different approaches on the fitting strategy, it was decided to use a 3rd degree polynomial to fit the data, a non linear ever increasing trend for power usage over wing speed. The model also comprehends an intercept value greater than zero, as expected, meaning the power usage is positive when the motor is idle – in accordance with the expected behaviour. This is desirable because real electric motors, along with their drivers, consume power while idle, given the imperfect efficiency of the circuits, unwanted impedances and other non-ideal elements. The 3rd degree polynomial can be described as:

$$P(\omega_M) = 1.057 \cdot 10^{-9} \cdot {\omega_M}^3 \ + \ 1.510 \cdot 10^{-6} \cdot {\omega_M}^2 \ + \ 1.148 \cdot \omega_M \ + \ 2.420 \cdot 10^{-1} \ , \quad (10)$$

with ω_M as the rotation speed in radians per second and $P(\omega_M)$ the power usage.

In Figure 41 we have the scatter plot of the data along with the polynomial fit, also indicating the ignored outliers.

Figure 41 – Scatter plot of the power data (blue dots), outliers (green crosses) and 3rd degree polynomial fit (solid red line).



Source – original.

The computation of the model and the necessary statistics for its evaluation were made using **R** – a "free software environment for statistical computing and graphics" (documentation of R, 2020). After the definition of the model, it is interesting to test whether or not the model is good or even useful and, for that, different tests can be used. At first, we can evaluate the coefficient of determination, known as $\mathbf{R^2}$. This coefficient ranges from 0 to 1. The closer it is to 1, the greater the portion of the total mean variability of the response variable is explained by the independent variable. The coefficient of determination is defined as:

$$\mathbf{R^2} = \frac{TSS - RSS}{TSS}, \tag{11}$$

where TSS stands for *total sum of squares* and RSS for *residual sum of squares*, defined respectively as:

$$TSS = \sum_{i=1}^{n} (Y_i - \bar{Y})^2 \ ,$$
$$RSS = \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2 \ ,$$

with n being the number of data points, Y_i the i^{th} sample, $\bar{Y}$ the mean of the response variable and $\hat{Y}_i$ the value predicted by the model at i.

For the polynomial model, then, the computed value of the coefficient of determination is 0.9991, very close to 1, which indicates that the model is adequate to portray the data.

Another tool to test the utility of the model, we can use Analysis of Variance (ANOVA), through an F-test. The hypothesis tested with the F-test in ANOVA are:

$$\begin{cases} H_0 \text{(null hypothesis)} : & \text{speed does not affect power,} \\ H_1 \text{(alternative hypothesis)} : & \text{speed affects power.} \end{cases}$$

Using the `anova` command in **R**, we get the details of the analysis and the values of interest. To draw a conclusion, there are two options, either comparing the F-value of the model with the reference value of the F-distribution (the F-value should be greater than or equal to the reference value in order to reject the null hypothesis), or comparing the *p-value* and the significance level (the p-value should be less than or equal to the significance level to reject the null hypothesis).

The so called p-value is very useful in null hypothesis significance testing, and it is defined as the probability of attaining results as extreme as the observed values, considering the null hypothesis to be correct. This is why it is compared with the significance level. In this specific case, considering a significance level of 5%, as it is commonly done, the null hypothesis can be rejected given the p-value of $2.2 \cdot 10^{-16} < 0.05$.

By rejecting the null hypothesis, at the significance level of 5%, we can conclude that the model is significant, there is enough evidence to say that the variation of the rotation speed influences the variation of the power usage.

Next, we test the normality of the residuals. One key assumption related to this model is that the response variable should have normal distribution. That is, its mean value is correlated to the value of the explanatory variable, and the dispersion around it follows a constant variance, normally distributed. For this analysis, the important variable will be the standardized residuals related to the model.

The standardized residuals take the units out of residuals of the response variable (power usage, here, measured in Watts) and transforms it into a variable with standard deviation $\sigma = 1$. This means that, if the variable follows a normal distribution, we would expect around 95% of its standardized residuals to be within $\pm 2\sigma$ from the mean.

The standardized residual can be computed in various ways, one – the chosen one for this work – is:

$$\hat{\epsilon}_i^{\star} = \frac{\hat{\epsilon}_i}{S\sqrt{1-h_{ii}}}\,, \tag{12}$$

with $\hat{\epsilon}_i^{\star}$ as the standardized residual for the i$^{\text{th}}$ data point, $\hat{\epsilon}_i$ the residual for the i$^{\text{th}}$ data point, S the estimated standard deviation of the errors and

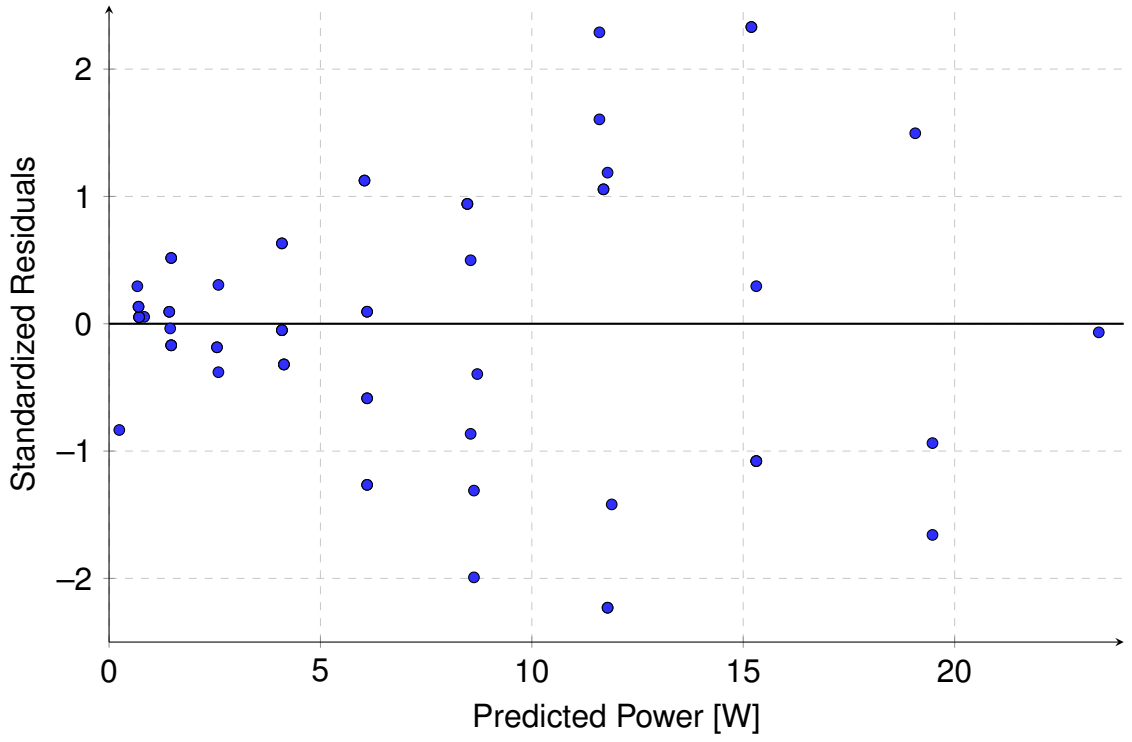
$$h_{ii} = \frac{1}{n} + \frac{(x_i-\bar{x})^2}{\sum_{j=1}^{n}(x_j-\bar{x})^2}\,, \tag{13}$$

where n is the number of data points (in this case, $n = 63$), x_k the value of the explanatory variable – here, the rotation speed – at the k$^{\text{th}}$ sample and $\bar{x}$ is the mean of the explanatory variable.

We can then plot the standardized residuals in respect to the predicted values of the response variable. In this graph, in Figure 42, we want to confirm 4 main aspects:

- the residuals do not present any specific pattern, they appear random;
- the amount of positive residuals is approximately equal to the amount of negative residuals;
- the absolute value of the positive residuals is approximately equal to the absolute value of the negative residuals;
- all residuals are within the interval [-3;3].

Figure 42 – Scatter plot of standardized residuals, computed as indicated in Equation 12, in respect to the predicted values of power usage.



Source – original.

Even though the residuals do not appear to present a specific pattern, the amount and absolute value of positive and negative residuals are nearly the same and they are all within the interval of $\pm$ 3 standard deviations, it is adequate to properly test the normality of the variable. For that, there is the Shapiro-Wilk test of normality. The hypothesis for the test are as follows:

$$\begin{cases} H_0: & \text{the variable is normally distributed} \\ H_1: & \text{the variable is not normally distributed.} \end{cases}$$

Hence, the aim is not rejecting the null hypothesis in this test, in order to verify the normality of the distribution of the residuals. Using the `shapiro.test` command in **R** we get the p-value related to the test, which should be greater than the significance level in order to not reject the null hypothesis. In this case, the p-value is 0.1879 > 0.05, which allows to conclude that, at significance level of 5%, the residuals are normally distributed.

After this analysis, one can reason that the 3rd degree polynomial model for power usage over rotation speed is useful, adequate and valuable. These same steps to test the validity of the model can be applied to other data sets, related to other variables of interest, aiming to produce good models for the hybrid drone.

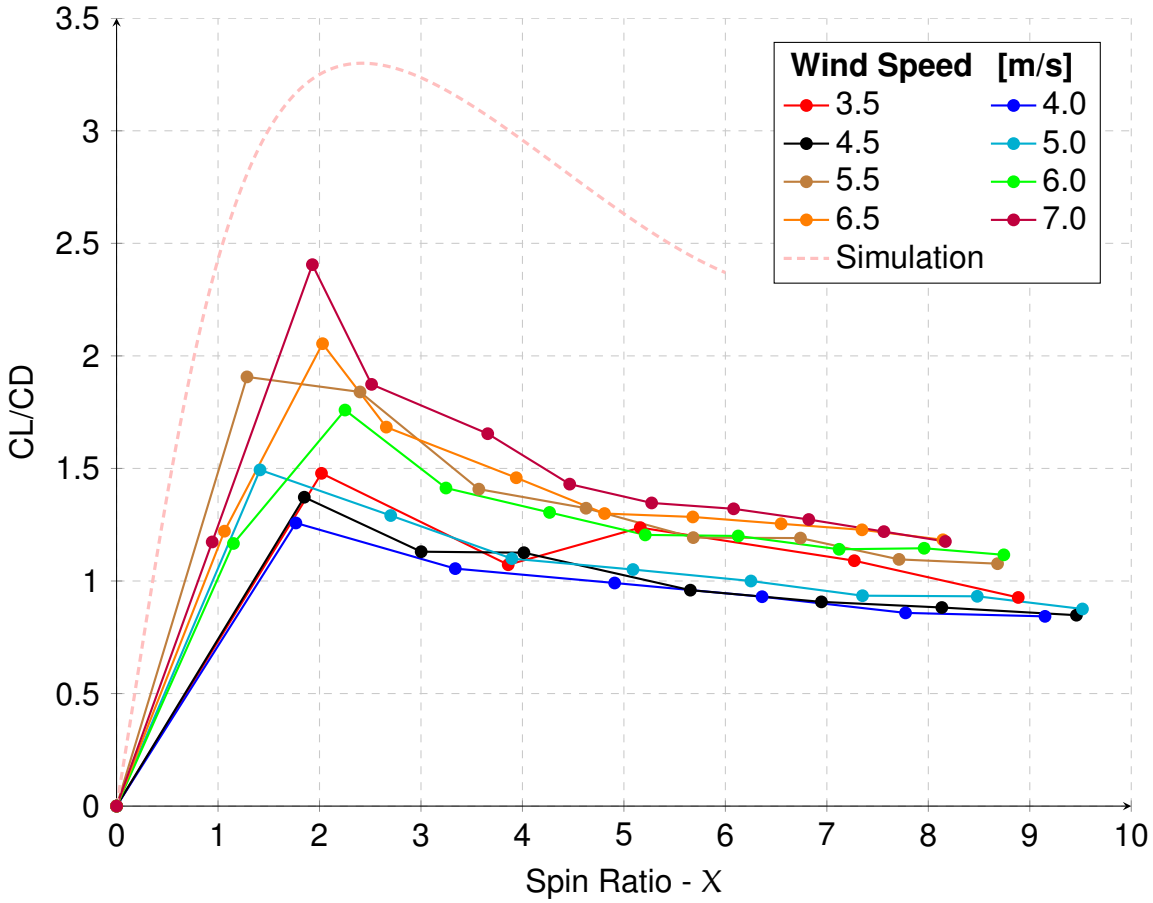
5.3 LIFT AND DRAG

For the coefficients of lift and drag, as expected, the wind speed was an important variable, related to the Reynolds number, that is known to affect the dynamic behaviour of wings and alike. Given the limitations of the setup for the tests in the laboratory and the measuring instruments, the amount of data gathered under different circumstances was limited. The measurements took place under 8 different wind speeds – from 3.5 m/s to 7.0 m/s, taking steps of 0.5 m/s –, but only around 7 samples per wind speed, trying to distribute them evenly until either the maximum speed of the driver of the motors or the maximum spin ratio desired was reached.

The maximum spin ratio desired was set at 10, mostly because the experiments led to a lot of data between 1 and 9 and the reference polynomial models, of 4^{th} order for C_L and 3^{rd} for C_D, are based on data in a range of 1 to 6, for the most part. Another reason is that the expected (and later observed) behaviour indicated that the maximum lift over drag ratio $\left(\frac{C_L}{C_D}\right)$ should be around spin ratio of 2.

In Figure 43 we can see a graph showing the lift over drag ratio, taken from the measurements during the lab tests, grouped by wind speed. The values are plotted alongside the polynomial approximation relative to lift over drag ratio on the first simulation, in Simulink.

Figure 43 – Experimental data for lift over drag ratio, per wind speed (solid), and originally simulated polynomial (dashed).



Source – original.

As already commented, we can see the peak of lift over drag ratio very close to spin ratio of 2, in most wind speeds. Since this maximum point is related to a more efficient flight, its estimation and the control of the spin ratio around it are crucial for the development of a proper controller for the hybrid drone.

5.3.1 Lift Model

After analyzing the data on lift coefficient along with the reference model, it was decided to use 3rd degree polynomial functions to capture its behaviour, one for each wind speed. Table 4 presents the coefficients for each obtained model, indicating the respective wind speed and the $\mathbf{R^2}$, computed as presented in Equation 11. For each group of coefficients, the polynomial can be constructed as:

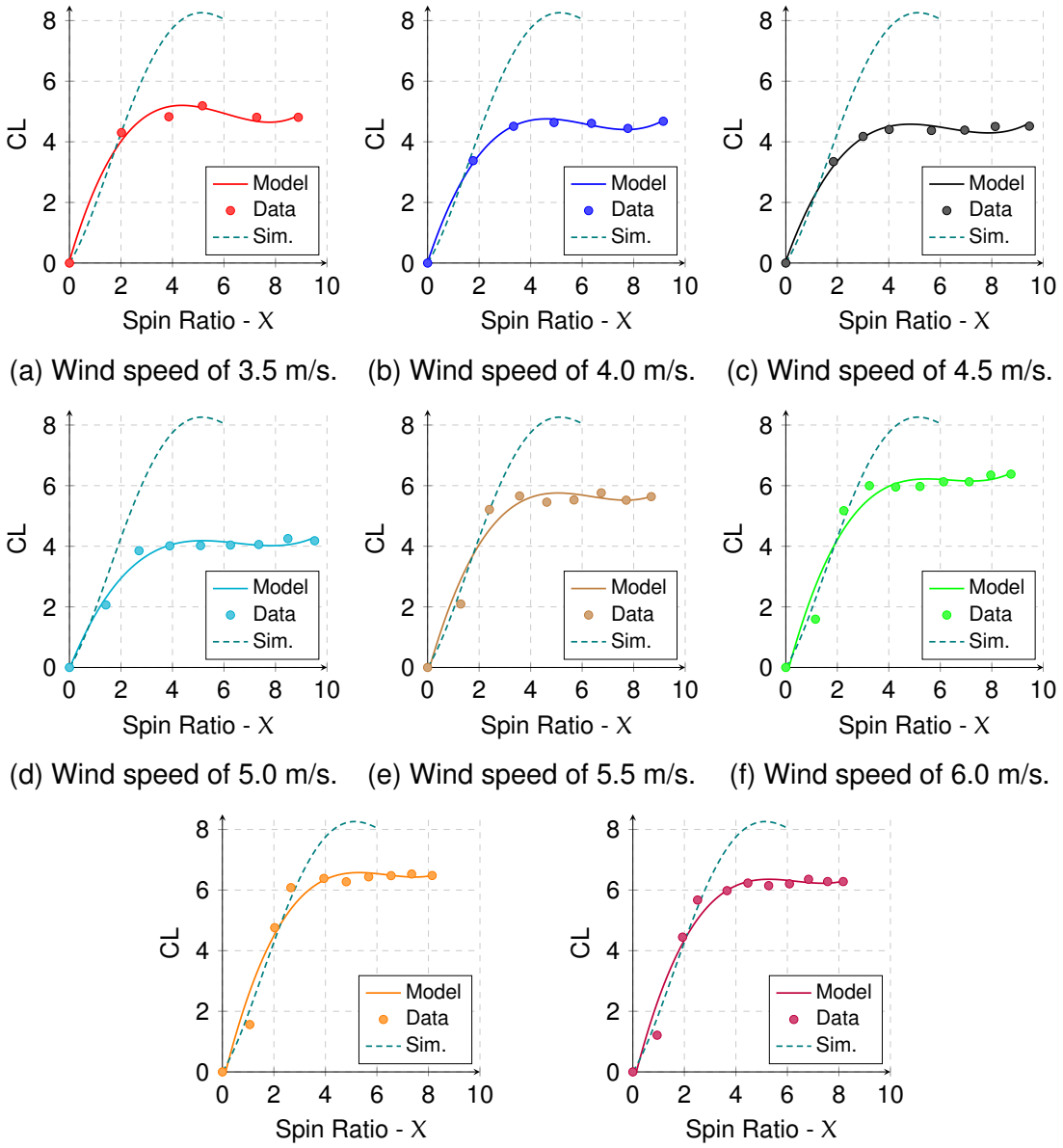
$$C_L(X) = b_3 \cdot X^3 + b_2 \cdot X^2 + b_1 \cdot X + b_0 \ . \tag{14}$$

Table 4 – Coefficients of each 3rd degree polynomial for coefficient of lift (C_L) in respect to spin ratio(X).

Wind Speed [m/s]	b_3	b_2	b_1	b_0	$\mathbf{R^2}$
3.5	0.02843	-0.51739	2.89271	0.06872	0.9891
4.0	0.02397	-0.44307	2.55887	0.02870	0.9985
4.5	0.02069	-0.39373	2.35503	0.05988	0.9932
5.0	0.01722	-0.33612	2.09466	-0.04032	0.9839
5.5	0.02595	-0.49658	3.03231	-0.23378	0.9601
6.0	0.02722	-0.51828	3.22907	-0.38934	0.9562
6.5	0.02904	-0.55684	3.45653	-0.42473	0.9596
7.0	0.02886	-0.54856	3.38068	-0.44957	0.9634

Though limited in number of observations, which reduces the strength of the conclusions to be drawn, all the models presented pass the ANOVA test, given the same level of significance of 5%. In Figure 44 we can see all 8 models, in ascending order of wind speed, along with the data points used to generate them and the polynomial model used during the first simulations.

Figure 44 – Polynomial models (solid) for lift coefficient based on experimental data (points). Originally simulated polynomial for comparison (dashed).



(a) Wind speed of 3.5 m/s. (b) Wind speed of 4.0 m/s. (c) Wind speed of 4.5 m/s.

(d) Wind speed of 5.0 m/s. (e) Wind speed of 5.5 m/s. (f) Wind speed of 6.0 m/s.

(g) Wind speed of 6.5 m/s. (h) Wind speed of 7.0 m/s.

Source – original.

It is interesting to note from the models presented that the correlation between the wind speed and the coefficient of lift does not appear to be linear. That can be visualized as there are few variations from 44a to 44c, then a “step-down” in 44d followed by a considerable increase from 44e to 44h. In order to make further analysis, more data would be necessary, as this was not possible given the test setup and the limitations of the hardware used. However, these models present a first look at the dynamics between an important control variable – the spin ratio – and an essential element from the hybrid drone system – the lift force.

Also important to emphasize the difference between the original model used in

simulation and experimental data and models, more specifically for lower wind speeds and higher spin ratio. Even though the original model has a significant higher slope between spin ratio of around 3 and 5, the models agree, mostly, on the lower spin ratio region. It becomes clear that for a more thorough model, the wind speed must be taken into account.

5.3.2 Drag Model

The model for coefficient of drag followed a similar approach to that of the lift coefficient. However, given its characteristics, smaller degree polynomials were used in some cases, since the dispersion and overall trend of the data points could be described with a simpler model. The decision weighted the capacity of the model to describe the data while taking into account the possible loss of information in simpler models, but penalizing more complex models for the possibility of *overfitting*.

Table 5 presents the coefficients of each model, given the different wind speeds, and also the value of the respective coefficient of determination ($\mathbf{R^2}$). To construct the polynomial model for C_D, we use the same structure from Equation 14. One thing it is important to clarify is that, given the lack of proper data points for wind speed of 3.5 m/s, the measurements above spin ratio of 10 were kept, exceptionally.

Table 5 – Coefficients of each 2nd or 3rd degree polynomial for coefficient of drag (C_D) in respect to spin ratio (X).

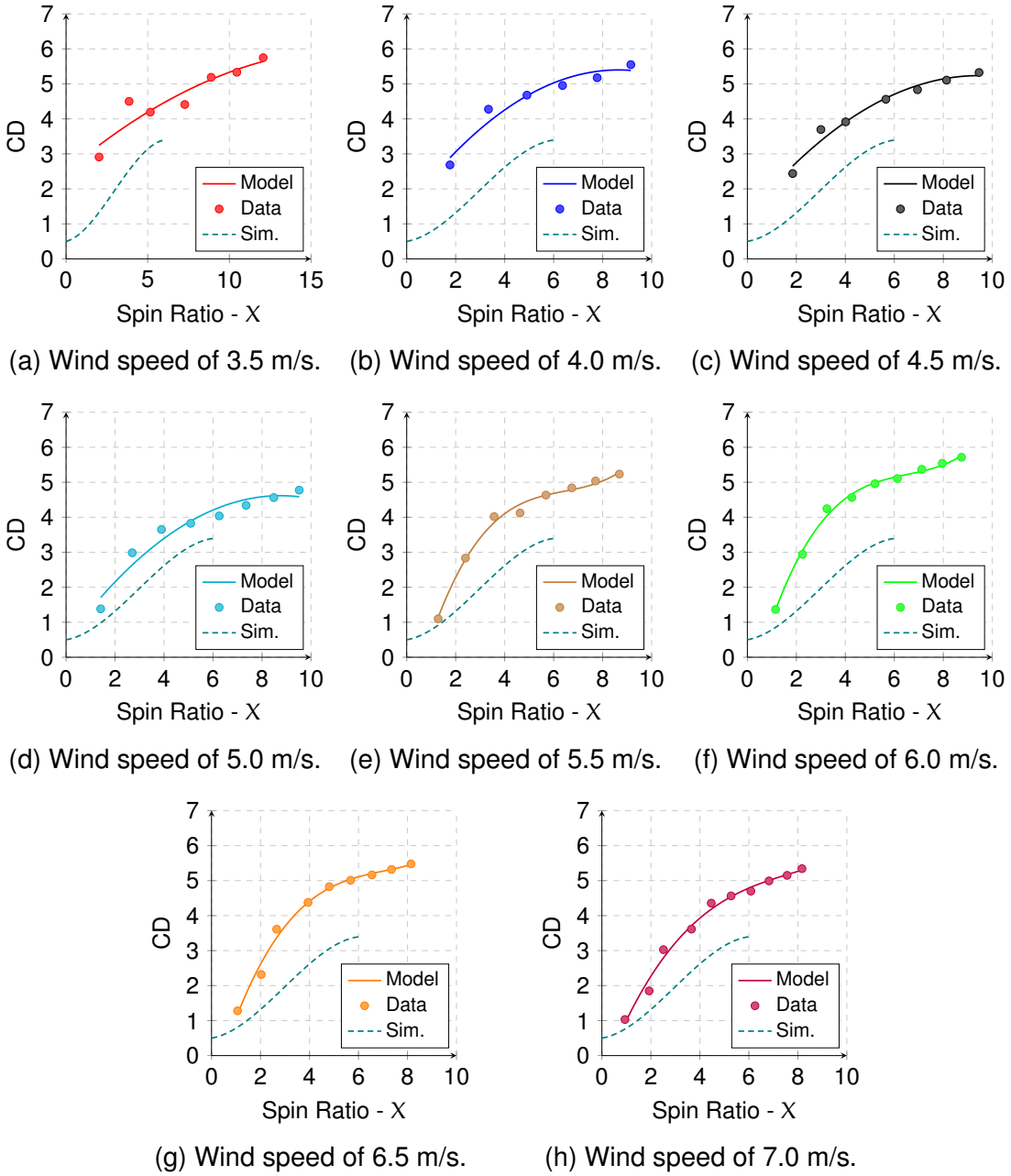
Wind Speed [m/s]	b_3	b_2	b_1	b_0	$\mathbf{R^2}$
3.5	0	-0.01113	0.39528	2.49223	0.8688
4.0	0	-0.05301	0.91648	1.43744	0.9475
4.5	0	-0.04476	0.84611	1.23758	0.9710
5.0	0	-0.05400	0.94566	0.47680	0.9460
5.5	0.02114	-0.40854	2.76855	-1.78902	0.9923
6.0	0.01974	-0.38707	2.67655	-1.24516	0.9954
6.5	0.01523	-0.32075	2.39875	-1.02596	0.9894
7.0	0.00872	-0.20460	1.81314	-0.61022	0.9878

In the presented configuration, all models for coefficient of drag also pass the ANOVA test, with the same level of significance of 5%. Similarly, though, the number of observations is a clear limitation for the relevance of the models. In Figure 45, all 8 models are presented along with the data points, for each value of wind speed, and the original polynomial used in the first simulation. The first model, 45a is the only one with different scale on X-axis.

Again, the original model strays away from the experimental data and models, but here the differences are much more evident for all values of spin ratio. Even for low

spin ratio the experimental data shows a considerable gap, differing from the lift models comparison. This serves to emphasize the importance of new, updated simulations that uses the gathered data in order to enhance the mathematical models.

Figure 45 – Polynomial models (solid) for drag coefficient based on experimental data (points). Original polynomial for comparison (dashed).



(a) Wind speed of 3.5 m/s. (b) Wind speed of 4.0 m/s. (c) Wind speed of 4.5 m/s.

(d) Wind speed of 5.0 m/s. (e) Wind speed of 5.5 m/s. (f) Wind speed of 6.0 m/s.

(g) Wind speed of 6.5 m/s. (h) Wind speed of 7.0 m/s.

Source – original.

Again, it is hinted that the correlation between coefficient of drag and the wind speed is not linear, or at least not between large intervals of wind speed. Also, though some models are 2nd degree polynomials, it would be interesting to, by gathering more data, fit 3rd degree models for all wind speeds, since it seems to fit very well and can

more easily guarantee an ever increasing drag.

More evidently in the model represented in 45d, the quadratic fit seems to not follow the trend of the last data points, instead leaning towards its maximum value, while the data suggests a continuous increase. This fact reinforces the limited reliance we can have on the results, and that the models should not be much extrapolated beyond their respective intervals. Still, more evidently for the higher wind speeds, the models give a reasonable start for the modelling work that needs to be integrated with the control architecture for the hybrid drone.

6 CONCLUSION AND FUTURE STEPS

After all the work done during 6 months, the project advanced greatly on the path towards the its goals, even though during the internship a lot of unforeseen problems arose, leading to delays and changes in the short-term road map. Still, it also lead us to improve in fields that were needed, even if not predicted.

The changes proposed and simulated in this work will require new control and estimation architecture for the hybrid drone in order to optimize its functionalities, which will also be a major concern for possible future steps as adapting a standard commercial drone for the same application, with the design of a module that can be added to an already functional drone. These topics are of interest for the project but are not discussed in this thesis. Another topic that needs addressing is the gyroscopic torques due to the rotation of the drone during the operation of the Magnus wings. These torques might have significant magnitude, and could be modelled using geometric aspects of the wing and *feedforward* controllers could be used in order to diminish their effects on flight control.

The simulations under Simulink provided a reasonable impression on the behaviour of the hybrid drone, whilst emphasizing some key problems that needed to be tackled. The results indicated a promising perspective on the overall efficiency and autonomy of the hybrid drone. Furthermore, a new feature, the Gazebo simulator – model, plugin and communication structure –, now allows for more realistic simulations on different environments, while using parameters and controllers directly from the real drones. This should grow more important as the project evolves, even more considering a detachable module for the wings, and it can play a key role in the exhibition of the state of the project, for interested parts.

On trajectory generation, a new algorithm allows for smooth trajectories related to real drone constraints, taking a first step into a very relevant issue related to the drone world. Another point that would fit for future work on the subject would be a "local" time-stretching algorithm, aiming at expanding the time vector only around the violations on the constraints, instead of along the whole vector, as the current version, emphasizing the efficiency of the approach.

Next, we have preliminary lab tests in the wind tunnel, that also gave important data on the real world variables and issues related to the system. With data from the physical system and updated models, the simulators should be more accurate when using similar setups, providing meaningful results on the topic. As commented, new tests could use the data already gathered and increase even more the reliability of the models in order to evolve the whole simulation and control structures. Gathering data related to the drag forces generated by the lateral wind (as defined in Chapter 2) could also enhance the model and improve the understanding of the physical aspects of the

Magnus wings.

For a more comprehensive work, future steps could include a review on the control structure of the simulators, updating the architecture to properly, and thoroughly, take into account the new effects related to the wing. That is also valid for the estimating strategies, since the somewhat controlled environment of the laboratory, during tests, provides more data than what is currently available during flight missions. The most important step would be the wind estimator, since it impacts greatly on the control of the spin ratio. Afterwards, these updates could be tested in real world scenarios and then improved.

Overall, the development has taken big steps towards a working prototype, and the team is now much more aware of the favorable aspects as much as the issues of the current system, which will certainly shape the decisions on the next phases. This thesis plays an important role in defining some of these aspects and issues, and provided background for new advances in the field.

REFERENCES

AZUR DRONES. **Autonomous Aerial Surveillance**. [S.l.: s.n.]. Available from: https://www.azurdrones.com/. Visited on: 5 July 2020.

CHEN, Zaibin; JIA, Hongguang. Design of Flight Control System for a Novel Tilt-Rotor UAV. **Hindawi-Complexity**, Mar. 2020. Available from: https://doi.org/10.1155/2020/4757381.

DJI. **Mavic Air 2 User Manual v1.0**. [S.l.], May 2020a. v1.0. Available from: https://dl.djicdn.com/downloads/Mavic_Air_2/20200511/Mavic_Air_2_User_Manual_v1.0_en.pdf. Visited on: 5 July 2020.

DJI. **Phantom 4 Pro**. [S.l.: s.n.]. Available from: https://www.dji.com/br/phantom-4-pro. Visited on: 5 July 2020.

DRONE INDUSTRY INSIGHTS. **Yearly Drone Market Investments 2008 - 2019**. Hamburg, Germany: [s.n.], Mar. 2020.

FURRER, Fadri; BURRI, Michael; ACHTELIK, Markus; SIEGWART, Roland. Robot Operating System (ROS): The Complete Reference (Volume 1). In: ed. by Anis Koubaa. [S.l.]: Springer International Publishing, 2016. RotorS—A Modular Gazebo MAV Simulator Framework, p. 595–625. ISBN 978-3-319-26054-9. DOI: 10.1007/978-3-319-26054-9_23. Available from: http://dx.doi.org/10.1007/978-3-319-26054-9_23.

GUPTA, Yashank; DUMON, Jonathan; HABLY, Ahmad. Modeling and control of a Magnus effect-based airborne wind energy system in crosswind maneuvers. **IFAC-PapersOnLine**, v. 50, n. 1, p. 13878–13885, 2017. 20th IFAC World Congress. ISSN 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2017.08.2205. Available from: http://www.sciencedirect.com/science/article/pii/S2405896317328768.

HABLY, Ahmad; DUMON, Jonathan; SMITH, Garret; BELLEMAIN, Pascal. Control of a Magnus Effect-Based Airborne Wind Energy System. In: **Airborne Wind Energy - Advances in Technology Development and Research**. Ed. by Roland Schmehl. [S.l.]: Springer, 2018. P. 277–301.

INDUSTRIAL SKYWORKS. **Complete Drone Inspection Solutions**. [S.l.: s.n.]. Available from: https://industrialskyworks.com/drone-inspections-services/. Visited on: 5 July 2020.

INTEL. **Drone Light Shows**. [S.l.: s.n.]. Available from: https://www.intel.com/content/www/us/en/technology-innovation/aerial-technology-light-show.html. Visited on: 5 July 2020.

LYU, X.; GU, H.; WANG, Y.; LI, Z.; SHEN, S.; ZHANG, F. Design and implementation of a quadrotor tail-sitter VTOL UAV. In: 2017 IEEE International Conference on Robotics and Automation (ICRA). [S.l.: s.n.], 2017. P. 3924–3930. Available from: https://doi.org/10.1109/ICRA.2017.7989452.

MONTCEL, Thibaut Tezenas Du. **Évitement d'obstacles pour quadrirotors en utilisant un capteur de profondeur**. 2019. PhD thesis – Université Grenoble Alpes. Available from: https://tel.archives-ouvertes.fr/tel-02534987.

OPEN SOURCE ROBOTICS FOUNDATION. **SDFormat**. [S.l.: s.n.]. Available from: http://sdformat.org/. Visited on: 15 Sept. 2020.

PX4 DRONE AUTOPILOT. **Gazebo for MAVLink SITL and HITL**. [S.l.: s.n.]. Available from: https://github.com/PX4/sitl_gazebo/tree/master/models. Visited on: 23 Sept. 2020.

SEKI, Masatoshi. **ERB**. [S.l.: s.n.]. Available from: https://www.commandlinux.com/man-page/man1/erb.1.html. Visited on: 23 Sept. 2020.

SWISS POST. **Swiss Post drone logistics**. [S.l.: s.n.]. Available from: https://www.post.ch/en/about-us/innovation/innovations-in-development/drones. Visited on: 5 July 2020.

THE R FOUNDATION. **The R Project for Statistical Computing**. [S.l.: s.n.]. Available from: https://www.r-project.org/. Visited on: 4 Nov. 2020.

THOM, Alexander. Effect of Discs on the Air Forces on a Rotating Cylinder. **Aeronautical Research Committee Reports and Memoranda**, v. 1623, 1934. Available from: https://reports.aerade.cranfield.ac.uk/handle/1826.2/1421.

TRAUT, Michael; GILBERT, Paul; WALSH, Conor; BOWS, Alice; FILIPPONE, Antonio; STANSBY, Peter; WOOD, Ruth. Propulsive power contribution of a kite and a Flettner rotor on selected shipping routes. **Applied Energy**, v. 113, p. 362–372, 2014. ISSN 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2013.07.026. Available from: http://www.sciencedirect.com/science/article/pii/S0306261913005928.

TYTLER, Charles. **Modeling Vehicle Dynamics**. [S.l.: s.n.]. Available from: https://charlestytler.com/modeling-vehicle-dynamics-euler-angles/. Visited on: 12 Sept. 2020.

UAV-IQ PRECISION AGRICULTURE. **Biological Control by Drone**. [S.l.: s.n.]. Available from: https://www.uaviq.com/en/home/. Visited on: 5 July 2020.