

Indoor Blimp Control

Gustaf Åman



LUND
UNIVERSITY

Department of Automatic Control

MSc Thesis
TFRT-6134
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

© 2021 by Gustaf Åman. All rights reserved.
Printed in Sweden by Tryckeriet i E-huset
Lund 2021

Abstract

The purpose of this thesis is to model and develop a control law that lets an unmanned aerial vehicle of blimp type fly autonomously in three-dimensional space. There are several uses of the blimp type UAV including, but not limited to, cargo transportation and surveillance. The work done throughout the thesis includes optimization of actuator placements, modelling the blimp and simulating the process, developing a PID controller for positioning and orientation, trajectory generation, actuator effort optimization, prototype construction and real-time experiments. The results of the thesis show that with a simulation model the control sequence implementation is facilitated, but not necessarily directly convertible to the real-time prototype. They also show that faulty signal processing can lead to disturbances which propagate throughout the process. Finally, it can be stated that the system was stable enough to remain close to a point in space, however, it was not fast and stable enough to follow a trajectory and thus the goals of the thesis were not fully met.

Acknowledgements

I would like to express my deepest gratitude to all who have helped and supported me throughout this thesis. I would like start by thanking my supervisor, Professor Anders Robertsson and my two assistant supervisors, Doctoral student Marcus Greiff and Lecturer Rikard Tyllström. The assistance and knowledge you provided throughout this thesis, both theoretical and practical were key to its progression and is deeply appreciated. I also wish to thank Research engineer Alexander Pisarevskiy for all of his help and patience throughout the construction phase of the thesis. Finally, I would like to thank my friends and family for the support they gave, both through motivation and discussion.

Contents

1. Introduction	9
1.1 Overview of the Master’s Thesis	10
1.2 Goals of the Master’s Thesis	10
1.3 Delimitations	11
2. Background	13
2.1 Unmanned Aerial Vehicles	13
2.2 Modelling	14
2.3 Control Theory	28
2.4 The CrazyFlie and Lighthouse Positioning Deck	34
3. Methodology	35
3.1 Modeling the Blimp	35
3.2 Propeller Optimization	36
3.3 Simulating the Process	41
3.4 Real-time Experiments	46
4. Results	53
5. Discussion	66
5.1 Modelling the Blimp	66
5.2 Propeller Optimization	67
5.3 Simulating the Process	68
5.4 Real-time Experiments	70
6. Conclusion	73
6.1 Conclusion	73
6.2 Future Work and Development	74
Bibliography	75
A. Inertia Tensor Appendix	78
B. Weighted Least Squares Appendix	79
C. PID Appendix	83
D. Blimp data Appendix	87
E. Equations	89

1

Introduction

For hundreds of years mankind has looked up to the sky and tried to invent ways of travelling across it. Whether it is Henri Giffard's steam powered airship constructed in 1852 [Li et al., 2011], the Wright brothers' first flight in 1903 [Howard, 2013] or the first composite airliner, Boeing 787 Dreamliner, released in 2004 [Lu, 2010], mankind has not only tried but also succeeded in travelling across the skies. Given time scientists and engineers have developed several different ways of air travel, both manned and unmanned, each having its own advantages and disadvantages. In this thesis different control methods are implemented and combined with the purpose to control an unmanned aerial vehicle (UAV) of a blimp type. The implemented control methods are verified by performing real-time experiments on a prototype blimp UAV, which can be seen in Figure 1.1.



Figure 1.1: Picture of the blimp UAV used to perform real-time experiments.

A common way of controlling an UAV is to have it be autonomous, meaning that it is controlled by a computer that handles sensory input and generates control outputs that transports the UAV in a desired way. There are many ways to obtain

the required sensory data both through internally placed sensors on the UAV, and externally placed sensors which tracks the UAV from a distance. Alternatively, combinations of internal and external sensors can be used. Examples of internally placed sensors are accelerometers that measure acceleration and gyroscopes that measure angular velocity. Externally placed sensors can be a camera that reads the UAV's position through image processing. An example of a combination of internal and external sensor units is a Lighthouse positioning system, with an onboard sensor deck, that geometrically traces light emitted by an externally placed base station.

1.1 Overview of the Master's Thesis

The report is split into six different chapters; Introduction, Background, Methodology, Results, Discussion and Conclusion. Each section presents different topics and parts of the thesis and can be described as follows.

Chapter 1. Introduction, gives a brief introduction to the purpose and layout of the thesis, what the main goals are and which scientific concepts will be included.

Chapter 2, Background, summarises the theory on which the thesis is based. Here the concepts of autonomous unmanned aerial vehicles, blimp and UAV modelling, and Lighthouse positioning are described in-depth.

Chapter 3. Methodology, is where all the practical parts and steps of progression of the thesis are presented. This involves descriptions of how the literature review was performed, how the blimp UAV was modelled theoretically, how the Lighthouse positioning system was set up, how the control methods were implemented, and how the prototype blimp was constructed and tested.

Chapter 4. Results, presents and explains the relevant data that was collected during the tests performed throughout the thesis.

Chapter 5. Discussion, is where the full scope of the thesis is discussed and reflected upon. The methodology is scrutinized and both what was successful and what could be improved is presented.

Chapter 6. Conclusion, presents a summary of the work performed throughout the thesis, and the results are interpreted and evaluated to how well they meet the goals of the thesis. Suggestions for future work and development of the blimp are discussed with respect to Chapter 3.

1.2 Goals of the Master's Thesis

The main goal of the thesis is to actuate and control a blimp UAV indoors so that it can navigate safely and autonomously to coordinates and orientations in three-dimensional space (\mathbb{R}^3), provided as user input. The system should use position data provided by a Lighthouse positioning system for its controller. The main goal can be further clarified by dividing it into five intertwined sub-goals, which are listed below.

- Obtain position data from a Lighthouse positioning system.
- Numerically determine the optimal rotor placements.
- Model and simulate the blimp and design controllers for the model.
- Construct a prototype blimp UAV by using the determined rotor placements.
- Stably control the position and orientation of the blimp UAV in \mathbb{R}^3 .

The results and outcomes of this thesis are purposed to be used as a basis for further development of blimp UAVs.

1.3 Delimitations

To be able to achieve the main goal of the thesis within a feasible timeline, it is important to limit the scope of the project. It will therefore be necessary to strongly rely on previous work and existing material to avoid unnecessary development of preexisting solutions. Some of the major delimitations of the project are described in the following paragraphs.

The blimp used will be not be designed and constructed but purchased with a design that can be seen in Figure 1.2. The parameters used to describe the geometry and properties of the blimp can be found in Appendix D and will be used when modelling in Chapter 3.

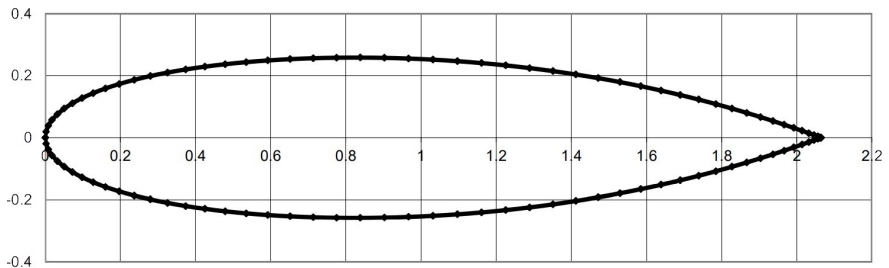


Figure 1.2: Shape of the final used blimp, expressed in meters. [Windreiter, 2020]

For actuation, sensor data collection, and communication with the designed controller, the propeller driven, 92x92x29mm (WxHxD) quadcopter developed by Bitcraze AB, called CrazyFlie 2.1 will be used. A CrazyFlie 2.1 unit is depicted in Figure 1.3, which if equipped with a Lighthouse positioning board has the additional capability of being used with a Lighthouse sensor system. This will be used

to obtain accurate position and orientation data for the UAV. To handle the Lighthouse system and control the CrazyFlie, Bitcraze AB supplies a Python application programming interface (API) containing relevant code and tutorials which will be used to base the further development on.



Figure 1.3: Picture of a CrazyFlie 2.1 [Bitcraze, 2020].

As the purpose of the blimp UAV is to be used indoors, the influence of wind will be not be considered and the air surrounding the blimp will be assumed to be stationary at all times.

2

Background

In this chapter all the relevant theoretical information used in the thesis is presented. It serves as a foundation on which the several design and implementation decisions are based. Each section of the chapter gives a general introduction to a main topic, followed by an in-depth description of selected subtopics. The purpose is to present the scientific topics in a way that is comprehensible and rewarding to the reader and at the same time focused only on the parts relevant to the thesis.

2.1 Unmanned Aerial Vehicles

As mentioned in Chapter 1, an unmanned aerial vehicle (UAV) requires no pilot sitting inside of the vehicle, and can be controlled either manually through a remote control, or autonomously by letting a computer control the vehicle. UAVs have many uses in a variety of fields, some of them being military use, cargo transportation and surveillance [Shakhatreh et al., 2019]. They come in a variety of designs which are use-case specific. Some of the more common design configurations are the quadcopter configuration, with a small body being propelled by four independent propellers, and the airplane configuration with a fuselage fitted with wings. The small body size and use of independent propellers allows the quadcopter configuration to be quick and able to move in any direction, but it constantly must spin its propellers to remain in the air which makes flight very energy-consuming. The airplane configuration is fast and can travel a longer distance due to how it generates lift, but it must move forward and roll or pitch to change direction, which makes it less agile [Paredes et al., 2017].

A solution to the large energy consumption of the quadcopter configuration, is to combine it with a buoyant blimp and make it blimp-like. A blimp or airship is a type of aircraft that makes use of lighter-than-air gases confined within its body to generate buoyancy [Li et al., 2011]. The combination would allow a UAV with a blimp-like design to create enough buoyancy to remain in the air with minimum energy input, while at the same time be able to move in all directions. However, to create the required buoyancy, the gas confining body must have sufficiently large

volume. This creates problems for navigating through narrow passages and gives rise to an increased sensitivity to air and wind disturbances in the form of gusts of wind [Li et al., 2011]. Besides being increasingly affected by gusts of wind, the increased volume also creates a larger air resistance in the form of drag forces that counteract the movement of the UAV, thus reducing its speed. This can be concluded by looking at the drag force model in Equation (2.1), where F_D is the drag force acting in the negative x -direction of the object (see Figure 2.1), ρ is the surrounding air density, V is the velocity in positive x -direction, S_h is the airship reference area, C_D is the empirically derived drag coefficient, and Λ is the airship volume [Kukillaya and Pashilkar, 2017].

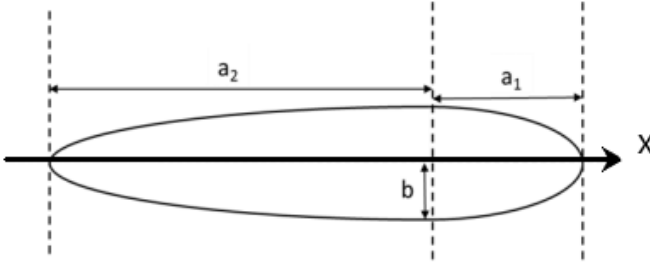


Figure 2.1: 2D shape of a double ellipse [Kukillaya and Pashilkar, 2017].

$$F_D = \frac{1}{2} \rho V^2 S_h C_D \quad (2.1a)$$

$$S_h = \Lambda^{\frac{2}{3}} \quad (2.1b)$$

2.2 Modelling

When designing the control system of an aircraft, vehicle or system in general, it is important to have an accurate mathematical model that reflects the system's dynamic properties. A proper mathematical model of the system's dynamics allows the designer to calculate and simulate how the system responds to different control inputs and parameter changes, without having to perform tests on the system itself. This allows tests to be performed quickly and without the risk of breaking the product, which reduces the amount of time required to tune the controller [Glad and Ljung, 2000]. One of the disadvantages is that for the model to be fully relevant, it must properly capture the necessary dynamics, otherwise the results will likely be off on the real system, and the designed controller might be poor. This can lead to issues since some dynamic properties are hard to anticipate and model mathematically. However, even simplified models can be informative as to give an approximate image of how the system may act.

Newton-Euler Dynamics

The basis of modelling a physical system is to capture how the system is affected by external influence. One way of doing this is to use Newton-Euler dynamics, which is a combination of different physical theorems and laws of classical mechanics that are used to describe the dynamic properties and relationships of a rigid body. This includes, among other things; the relationship between different frames of reference and different coordinate systems, how forces acting on the body result in acceleration, and how moments and inertia affect rotation [Ardema, 2005].

Frames of Reference and Coordinate Systems: The basis of using Newton-Euler dynamics is to use relevant reference frames and coordinate systems, of which there are many. Some of the most common reference frames, that are also used in this thesis are: the inertial frame denoted \cdot_G , and the body-fixed frame denoted \cdot_B . The inertial reference frame is fixed arbitrarily in \mathbb{R}^3 and the body-fixed reference frame is fixed in an object which can often move in relation to the inertial reference frame [Ardema, 2005]. Both mentioned reference frames can be described by the right-hand orthogonal unit base vectors $\{\hat{x}, \hat{y}, \hat{z}\}$. The $\hat{\cdot}$ notation means that it is a base vector that is being referred to. As for coordinate systems, two common systems are: the rectangular coordinate system and the spherical coordinate system. Rectangular coordinates describe the location of a point P in reference to the origin (O) by vector \overline{OP}_{xyz} in Equation (2.2a). In spherical coordinates, P is instead described in reference to O by vector $\overline{OP}_{r\theta\alpha}$ in Equation (2.2b). Both of these coordinate systems are illustrated in Figure 2.2 and the coordinate transformation equations relating the two are seen in Equation (2.3). Note that in Figure 2.2, the \overline{OP} vectors are not drawn but are defined as the distance between O and P followed by respective reference frame notation. Right-handed reference frames such as the one in Figure 2.2, where x is forward, y is to the left and z points upwards can be called North-West-Up (NWU) frames, which are commonly used for land-based bodies. However, when regarding the body-fixed reference frame of an airplane, it is common to use a so-called North-East-Down (NED) frame, which is shown in Figure 2.3.

$$\overline{OP}_{xyz} = x\hat{x} + y\hat{y} + z\hat{z} = (x, y, z) \quad (2.2a)$$

$$\overline{OP}_{r\theta\alpha} = (r \cdot \sin\alpha \cdot \cos\theta)\hat{x} + (r \cdot \sin\alpha \cdot \sin\theta)\hat{y} + (r \cdot \cos\alpha)\hat{z} = (r, \theta, \alpha) \quad (2.2b)$$

$$x = r \cdot \sin\alpha \cdot \cos\theta \quad (2.3a)$$

$$y = r \cdot \sin\alpha \cdot \sin\theta \quad (2.3b)$$

$$z = r \cdot \cos\alpha \quad (2.3c)$$

When using both an inertial reference frame and a body-fixed frame, it is important to determine how they are positioned in relation to each other. Introduce an inertial reference frame $\{\hat{x}_G, \hat{y}_G, \hat{z}_G\}$, that is fixed in a global origin position $O = (0, 0, 0)$ defined by a rectangular coordinate system. In the inertial frame, introduce an arbitrary rigid body with a body-fixed reference frame, $\{\hat{x}_B, \hat{y}_B, \hat{z}_B\}$ placed in its center

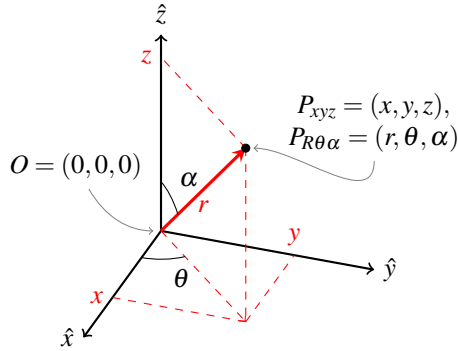


Figure 2.2: Description of how rectangular coordinates and spherical coordinates relate to each other and to the set frame of reference.

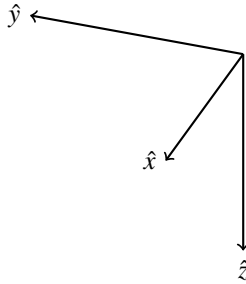


Figure 2.3: Illustration of a North-East-Down.

of gravity (CG_B). The position vector, \bar{X}_B in Equation (2.4a) consisting of the coordinates x_B , y_B and z_B describes the distance from O to CG_B which for this example is the same as the position of CG_B in reference to the inertial frame. The attitude vector, $\bar{\eta}_B$ in Equation (2.4b) describes the extrinsic Euler angle rotation of the body frame in reference to the inertial frame. Both \bar{X}_B and $\bar{\eta}_B$ are illustrated in Figure 2.4.

$$\bar{X}_B = [x_B \quad y_B \quad z_B]^T \tag{2.4a}$$

$$\bar{\eta}_B = [\phi \quad \theta \quad \psi]^T \tag{2.4b}$$

To determine the dynamics of an object in a given frame, Newton's three laws can be used, and are described as follows:

Newton's First Law states that if the sum of all forces acting on a particle or body's CG is equal to zero at any given time, then the acceleration of the particle or CG is also zero in that time instance [Ardema, 2005].

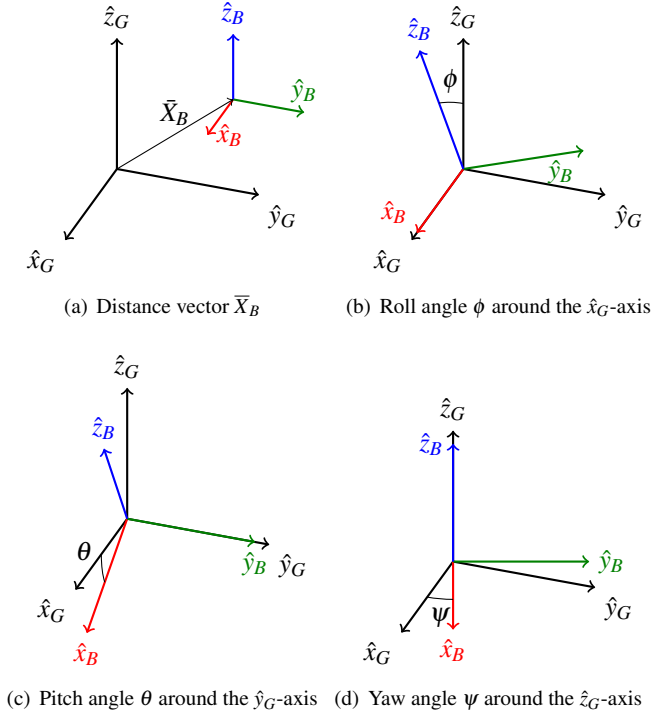


Figure 2.4: Description of the distance vector r and the attitude states ϕ , θ and ϕ .

Newton's Second Law states that the acceleration, $\bar{a}_G = [a_x^G, a_y^G, a_z^G]^T$ of a particle or body's center of mass in reference to the inertial frame is directly proportional to the sum of the forces, $\bar{F}_G = [F_{G_x}, F_{G_y}, F_{G_z}]^T$ acting on it. This can be expressed as in Equation (2.5) [Ardema, 2005].

$$\sum \bar{F}_G = m \cdot \bar{a}_G \quad (2.5)$$

Newton's Third Law states that the force exerted by one particle onto another is equal in size but opposite in direction to the force that the second exerts on the first [Ardema, 2005].

Angular Dynamics The angular dynamics of an object can be described by its attitude rates $\bar{\omega} = \dot{\eta}$ which are driven in time by the applied moments \bar{M} . The applied moment are, in turn, related to the attitude rate time derivatives, as in Equation (2.6), where \bar{I} is a positive definite inertia tensor derived from the mass and geometry properties of the object. Equations for calculating the inertia tensor can be found

in Appendix A [Ardema, 2005].

$$\bar{H} = \bar{I}\bar{\omega} \quad (2.6a)$$

$$\bar{M} = \frac{d}{dt}\bar{H} = \bar{I}\dot{\bar{\omega}} \quad (2.6b)$$

Tait-Bryan Rotations

To determine a body's acceleration in the body frame instead of the inertial frame, coordinate transformation must be performed. One way to do this is to use Tait-Bryan rotations or Tait-Bryan angles, which are a part of the Euler angles, where Euler angles are described in [Ang and Tourassis, 1987] as a way of describing orientation through consecutively rotating around the coordinate axes three times. The order in which each axis is rotated around differs and there are 12 different combinations available, with six of them using all three available axes [Ang and Tourassis, 1987]. Tait-Bryan rotations use all three available axes and are used to relate the base vectors of a body-fixed reference frame to the base vectors of an inertial reference frame. The way they are used is by looking at the trigonometric relationship between the two frames for each of the three rotation angles displayed in Figure 2.4. The projections of the inertial axes onto the rotated body-fixed axes, can for each type of rotation be written as the 3×3 matrices derived in Equation (2.7). The order in which the axes are rotated can be chosen arbitrarily as long as all three axes are included. The Tait-Bryan rotation matrices derived in Equations (2.7) and (2.8) are ordered as counter clockwise rotation around the first the \hat{x} -axis, then the \hat{y} -axis and finally the \hat{z} -axis, and are often referred to as XYZ-rotations.

$$\bar{R}_{GB}(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \quad (2.7a)$$

$$\bar{R}_{GB}(\theta) = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \quad (2.7b)$$

$$\bar{R}_{GB}(\psi) = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.7c)$$

Multiplying the three axial rotation matrices results in the complete rotational matrix in Equation (2.8), where C_i and S_i denote the cosine and sine values of the angle i , respectively. The equation describes the relationship of going from the inertial frame to the rotated body frame.

$$\begin{aligned} \bar{R}_{GB}(\phi, \theta, \psi) &= \bar{R}_{GB}(\phi) \cdot \bar{R}_{GB}(\theta) \cdot \bar{R}_{GB}(\psi) \\ &= \begin{bmatrix} C_\theta \cdot C_\psi & C_\theta \cdot S_\psi & -S_\theta \\ C_\psi \cdot S_\phi \cdot S_\theta - C_\phi \cdot S_\psi & C_\phi \cdot C_\psi + S_\phi \cdot S_\psi \cdot S_\theta & C_\theta \cdot S_\phi \\ S_\phi \cdot S_\psi + C_\phi \cdot C_\psi \cdot S_\theta & C_\phi \cdot S_\theta \cdot S_\psi - C_\psi \cdot S_\phi & C_\phi \cdot C_\theta \end{bmatrix} \quad (2.8) \end{aligned}$$

The resulting relationship for transforming acceleration in the inertial reference frame to the body-fixed reference frame, and from the body-fixed reference frame to the inertial reference frame is presented in Equation (2.9).

$$\bar{a}_B = \bar{R}_{GB}(\phi, \theta, \psi) \bar{a}_G \quad (2.9a)$$

$$\bar{a}_G = \bar{R}_{GB}^T(\phi, \theta, \psi) \bar{a}_B = \bar{R}_{BG}(\phi, \theta, \psi) \bar{a}_B \quad (2.9b)$$

Similar to this, transforming from a NWU frame to a NED frame can be described as rotating the coordinates with the rotational matrix in (2.8) by a roll angle, ϕ , of π radians. Another way to view it is to add an extra rotation of the body frame from the NWU body frame to the NED body frame. Using the first way of transformation from a NWU frame to a NED frame, the relationships seen in Equation (2.10) can be derived.

$$\bar{R}_F = \bar{R}_{GB}(\pi, 0, 0) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \quad (2.10a)$$

$$\bar{X}_{NED} = \bar{R}_F \bar{X}_{NWU} \quad (2.10b)$$

$$x_{NED} = x_{NWU} \quad (2.10c)$$

$$y_{NED} = -y_{NWU} \quad (2.10d)$$

$$z_{NED} = -z_{NWU} \quad (2.10e)$$

$$\phi_{NED} = \pi + \phi_{NWU} \quad (2.10f)$$

$$\theta_{NED} = -\theta_{NWU} \quad (2.10g)$$

$$\psi_{NED} = -\psi_{NWU} \quad (2.10h)$$

An issue with this type of coordinate transformation is that it is prone to result in mathematical matrix singularities at certain rotations. This is discussed by [Greiff, 2017], and can be seen by letting the parameters in Equation (2.8) be $(\phi, \theta, \psi) = (\phi, (n + \frac{1}{2})\pi, 0)$, which for $n \in \mathbb{N}$ leads to the following result:

$$\begin{aligned} \bar{R}_{GB}(\phi, \theta, \psi) &= \begin{bmatrix} 0 & 0 & -1 \\ C_\psi \cdot S_\phi - C_\phi \cdot S_\psi & C_\phi \cdot C_\psi + S_\phi \cdot S_\psi & 0 \\ S_\phi \cdot S_\psi + C_\phi \cdot C_\psi & C_\phi \cdot S_\psi - C_\psi \cdot S_\phi & 0 \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 & -1 \\ \sin(\phi - \psi) & \cos(\phi - \psi) & 0 \\ \cos(\phi - \psi) & -\sin(\phi - \psi) & 0 \end{bmatrix} \end{aligned} \quad (2.11)$$

Having the singular rotation matrix shown in Equation (2.11) implies that for a pitch angle where the body-fixed \hat{x} -axis points either straight up or straight down in the inertial reference frame, the roll and yaw angles will appear in pairs allowing different combinations of angles to give the same rotation matrix. The rotation matrix is ambiguous unless one of the two angles are known which leads to a reduction of the numbers of freedom by one in regions close to $\theta = \pm \frac{\pi}{2}$ [Hemingway and O'Reilly, 2018]. One way to negate this issue when considering flight is to avoid the troublesome region through careful piloting of the aircraft, thus avoiding the singularity. Other examples of avoiding the singularity are: using quaternions instead of Eu-

ler angles, and switching coordinate systems near the troublesome region [Hosier, 2018].

A similar phenomenon to the singularity of the rotation matrix, is the so-called gimbal lock. A gimbal is a circular plane that rotates around an axis and are in combinations of three used in sensory equipment such as gyroscopes and inertial measurement units. When combined in groups of three, the gimbal group can be viewed as a physical manifestation of rotations using Euler angles. The way the gimbals are organized in the gimbal group can be described as interconnected. The rotations of the outermost gimbal results in rotations around the same axis for the two inner gimbals. Rotation of the middle gimbal results not only in rotation of itself but also in rotation of the innermost gimbal, while rotation of the innermost gimbal only affects itself.

Consider a fictive arrow pointing along the axis of rotation of the innermost gimbal. By rotating the three gimbals in a certain sequence, the arrow can point in any direction. However, if the middle gimbal plane was to rotate by $\pm\frac{\pi}{2}$ so that it coincides with the outermost plane, the gimbal group would lose one degree of freedom. This is known as a gimbal lock, and can be described as a physical manifestation of the mathematical singularity of a Tait-Bryan rotation matrix. Figure 2.5 shows an illustration of a gimbal group in its starting position where all planes are orthogonal to each other. If the middle blue plane in the figure was to rotate around its axis by $\pm\frac{\pi}{2}$, it would coincide with the outer red plane, thus a gimbal lock would ensue. Examples of ways of avoiding the gimbal lock phenomenon are: avoiding the troublesome region through navigation, and adding a fourth gimbal. [Hemingway and O'Reilly, 2018]

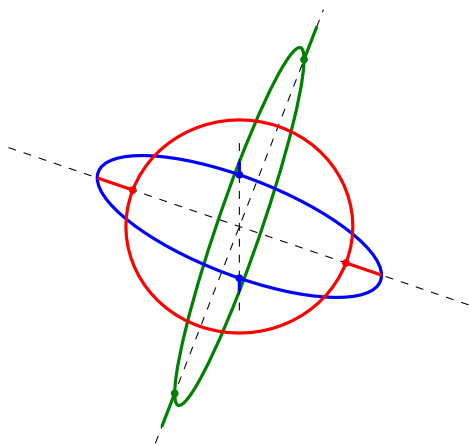


Figure 2.5: Illustration of a gimbal group in a starting position where all planes are orthogonal to each other.

Equations of Motion

To be able to use the Newton-Euler dynamics for a large body, such as a blimp, the relationship between all forces acting on the body and how its mass is dispersed must be obtained. Therefore the mass part of Newton's Second Law (2.5), must be expanded to account for the holistic view of the mass distribution of the body. As for the case of a blimp, the forces acting on it can be summed as in Equation (2.12), which has the form of Newton's Second Law (2.5) [Kukillaya and Pashilkar, 2017].

$$\sum \bar{F} = \bar{F}_d + \bar{F}_a + \bar{F}_b + \bar{F}_g + \bar{F}_c + \bar{F}_p + \bar{F}_f = \bar{M}_a \cdot \ddot{\bar{x}} \quad (2.12)$$

Each part of Equation (2.12) holds its own relevant part of the blimp dynamics and is defined by [Kukillaya and Pashilkar, 2017] as in Table 2.1, where the states are selected as the translational and rotational velocities expressed in the body frame. The further descriptions and derivations of these parts are all adapted from the work of [Kukillaya and Pashilkar, 2017], which serves as the model used in this thesis [Kukillaya and Pashilkar, 2017]. To derive the expressions for the components listed

Variable	Definition	Dimension
$\ddot{\bar{x}}$	Time derivative of state vector \bar{x} , holding the body frame's translational and rotational accelerations	6×1
\bar{M}_a	Matrix of masses and inertias	6×6
\bar{F}_d	Coriolis and centrifugal terms	6×1
\bar{F}_a	Aerodynamic forces and moments caused by the hull and fins	6×1
\bar{F}_b	Forces and moments caused by the blimp's buoyancy	6×1
\bar{F}_g	Forces and moments caused by the inertial frame gravity	6×1
\bar{F}_c	Aerodynamic forces and moments caused by control surfaces	6×1
\bar{F}_p	Forces and moments caused by the propulsive units acting on the blimp	6×1
\bar{F}_f	Forces and moments caused by the inertial moment of the fluid	6×1

Table 2.1: Definitions of the different components of the blimp's equations of motion.

in Table 2.1, [Kukillaya and Pashilkar, 2017] use a blimp shaped like an axisymmetric double ellipsoid as that in Figure 2.1. On the ellipsoid, four fins are placed, and a complete illustration of the blimp with directional axes and measurements is seen in Figure 2.6 with the measurements being described in Table 2.2 alongside other relevant measurement definitions.

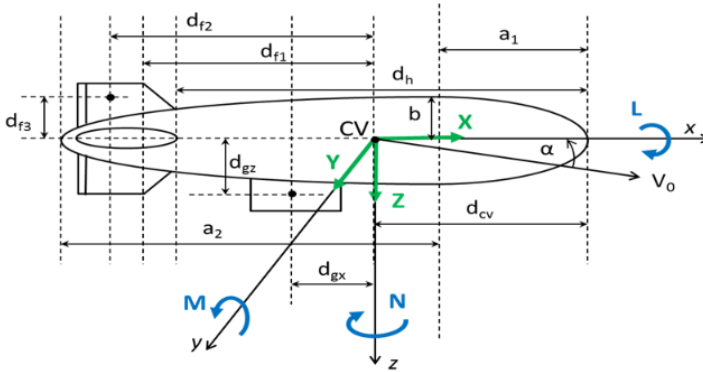


Figure 2.6: Illustration of the blimp with directional axes and measurements [Kukilaya and Pashilkar, 2017].

Variable	Definition
x, y and z	Coordinates along the \hat{x} -, \hat{y} - and \hat{z} -axes of the blimp, oriented in a North-East-Down configuration
M, N and L	The positive direction of moments acting on the \hat{x} -, \hat{y} - and \hat{z} -axes in the body frame respectively
V_0	Direction and magnitude of the blimp's velocity, defined by α and β
α	Angle of attack, being the angle between V_0 and the its projection on the $\hat{x}\hat{y}$ -plane
β	Sideslip angle, being the angle between V_0 and the \hat{x} -axis projected to the $\hat{x}\hat{y}$ -plane
$d_{f1}, d_{f2}, d_{f3}, d_{gx}, d_{gz}$	Parameters describing the placement of the rudders and fins.
a_1 and a_2	The principal semi-major axes of the front and back ellipsoid halves respectively.
CV	Center of volume of the blimp
d_{cv}	Distance from the front of the blimp to its center of volume
b	The principal semi-minor axis of the blimp being equal in the \hat{y} - and \hat{z} -directions

Table 2.2: Description of the notations used when describing the blimp geometry and properties.

Aerodynamic forces and moments (F_{a+c}) These forces and moments are the result of the blimp displacing a fluid as it travels. Each part of the blimp's surface and extremities has its own contribution and it is therefore reasonable to sum the

contributions of all parts into one extended equation for each force and moment contribution. As derived by [Kukillaya and Pashilkar, 2017], the aerodynamic force and moment vector is expressed in Equation (2.13) with its components expressed in Equation (2.14). The constants, C_{X1} , C_{Y2} etc., are the drag coefficients of the blimp's different parts and can be calculated by observing the geometry of the airship. As for the scope of this project, these coefficients will not be calculated for reasons that will be discussed in later parts of the report.

$$\bar{F}_{(a+c)} = [F_{(a+c),x} \quad F_{(a+c),y} \quad F_{(a+c),z} \quad M_{(a+c),x} \quad M_{(a+c),y} \quad M_{(a+c),z}]^T \quad (2.13)$$

$$F_{(a+c),x} = \frac{1}{2}\rho V_0^2 \left[C_{x1}\cos^2\alpha\cos^2\beta + C_{x2}\sin 2\alpha\sin\frac{\alpha}{2} + C_{x3}\sin 2\beta\sin\frac{\beta}{2} \right] \quad (2.14a)$$

$$F_{(a+c),y} = \frac{1}{2}\rho V_0^2 \left[C_{y1}\cos\frac{\beta}{2}\sin 2\beta + C_{y2}\sin 2\beta + C_{y3}\sin\beta\sin|\beta| \right. \\ \left. + C_{y4}(\delta_{RUDT} + \delta_{RUDB}) \right] \quad (2.14b)$$

$$F_{(a+c),z} = \frac{1}{2}\rho V_0^2 \left[C_{z1}\cos\frac{\alpha}{2}\sin 2\alpha + C_{z2}\sin 2\alpha + C_{z3}\sin\alpha\sin|\alpha| \right. \\ \left. + C_{z4}(\delta_{ELVL} + \delta_{ELVR}) \right] \quad (2.14c)$$

$$M_{(a+c),x} = \frac{1}{2}\rho V_0^2 \left[C_{L1}(\delta_{ELVL} - \delta_{ELVR} + \delta_{RUDB} - \delta_{RUDT}) + C_{L2}\sin\beta\sin|\beta| \right] \quad (2.14d)$$

$$M_{(a+c),y} = \frac{1}{2}\rho V_0^2 \left[C_{M1}\cos\frac{\alpha}{2}\sin 2\alpha + C_{M2}\sin 2\alpha + C_{M3}\sin\alpha\sin|\alpha| \right. \\ \left. + C_{M4}(\delta_{ELVL} + \delta_{ELVR}) \right] \quad (2.14e)$$

$$M_{(a+c),z} = \frac{1}{2}\rho V_0^2 \left[C_{N1}\cos\frac{\beta}{2}\sin 2\beta + C_{N2}\sin 2\beta + C_{N3}\sin\beta\sin|\beta| \right. \\ \left. + C_{N4}(\delta_{RUDT} + \delta_{RUDB}) \right] \quad (2.14f)$$

Coriolis and centrifugal forces and moments (F_d) These forces and moments are the outcome of the stored kinematic energy and inertia of the blimp. The force and force moment contributions are directly proportional to the translational and rotational velocity scaled by the apparent mass and apparent inertia terms that are dependant on how the mass is disposed throughout the volume. To describe these dependencies it is useful to first define the added mass matrix. In [Severholt, 2017], the forces induced by added mass and inertia are described as the pressure induced forces and moments created by the object displacing the surrounding fluid as it moves. For a body with six degrees of freedom, the added masses and inertias can

be collected into a 6x6 matrix, known as the added mass matrix, $M_{add} \in \mathbb{R}^{6 \times 6}$. The matrix coefficients are derived from the object's geometry and mass from the perspective of each degree of freedom [Kukillaya and Pashilkar, 2017]. For an axisymmetric ellipsoid, the added mass matrix can be reduced to a diagonal matrix, as shown in Equation (2.15). The coefficients are for the axisymmetric ellipsoid defined as in Equation (2.16), with the coefficient parameters described in (2.17). The influence of the added mass caused by fins and rudders is small relative to that of the ellipsoid and is thus disregarded [Kukillaya and Pashilkar, 2017].

The expression for the added mass and inertia in the ellipsoidal case can be summarized as:

$$M_{add} = \begin{bmatrix} X_{\dot{u}} & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_{\dot{v}} & 0 & 0 & 0 & 0 \\ 0 & 0 & Z_{\dot{w}} & 0 & 0 & 0 \\ 0 & 0 & 0 & L_{\dot{p}} & 0 & 0 \\ 0 & 0 & 0 & 0 & M_{\dot{q}} & 0 \\ 0 & 0 & 0 & 0 & 0 & N_{\dot{r}} \end{bmatrix} \quad (2.15)$$

where

$$X_{\dot{u}} = \frac{\alpha_0}{2 - \alpha_0} m \quad (2.16a)$$

$$Y_{\dot{v}} = \frac{\beta_0}{2 - \beta_0} m \quad (2.16b)$$

$$Z_{\dot{w}} = \frac{\gamma_0}{2 - \gamma_0} m \quad (2.16c)$$

$$L_{\dot{p}} = \frac{(b^2 - c^2)^2 (\gamma_0 - \beta_0)}{2(b^2 - c^2) + (b^2 + c^2)(\beta_0 - \gamma_0)} \frac{m}{5} \quad (2.16d)$$

$$M_{\dot{q}} = \frac{(c^2 - a^2)^2 (\alpha_0 - \gamma_0)}{2(c^2 - a^2) + (c^2 + a^2)(\gamma_0 - \alpha_0)} \frac{m}{5} \quad (2.16e)$$

$$N_{\dot{r}} = \frac{(a^2 - b^2)^2 (\beta_0 - \alpha_0)}{2(a^2 - b^2) + (a^2 + b^2)(\alpha_0 - \beta_0)} \frac{m}{5} \quad (2.16f)$$

In these equations $a = (a_1 + a_2)/2$ denotes the mean value of the principal semi-major axes of the double ellipsoid, $b = c$ denotes principal semi-minor axes of the ellipsoid, m denotes the mass of displaced fluid, e denotes the eccentricity of the ellipsoid, and the parameters α_0 and β_0 are given by:

$$\alpha_0 = \frac{(1 - e^2)}{e^3} \left[\log \frac{1 + e}{1 - e} - 2e \right] \quad (2.17a)$$

$$\beta_0 = \gamma_0 = \frac{1}{e^2} - \frac{(1 - e^2)}{2e^3} \log \frac{1 + e}{1 - e} \quad (2.17b)$$

By using the added mass matrix in Equation (2.15), the apparent mass of the blimp can be calculated. Apparent mass is the experienced mass of the object in the different directions meaning the mass of the object subtracted by the added mass caused

by the surrounding fluid. Similar to this the apparent moments of inertia can be defined as the experienced inertia of the body, thus the actual moment of inertia subtracted by the added inertia. These definitions can be seen in Equation (2.18), where m_{tot} is the total mass of the blimp, I_i are components of the inertia tensor. Due to symmetry of the blimp the tensor's off-diagonal elements J_{xy} , J_{yz} and J_{xz} are all zero.

$$m_x = m_{tot} + X_{\dot{u}} \quad (2.18a)$$

$$m_y = m_{tot} + Y_{\dot{v}} \quad (2.18b)$$

$$m_z = m_{tot} + Z_{\dot{w}} \quad (2.18c)$$

$$J_x = I_x + L_{\dot{p}} \quad (2.18d)$$

$$J_y = I_y + M_{\dot{q}} \quad (2.18e)$$

$$J_z = I_z + N_{\dot{r}} \quad (2.18f)$$

$$J_{xy} = 0 \quad (2.18g)$$

$$J_{yz} = 0 \quad (2.18h)$$

$$J_{xz} = 0 \quad (2.18i)$$

The influence that apparent mass and apparent inertia has for the different force and force moment components can be collected in a matrix known as the mass and inertia matrix M_a . [Kukillaya and Pashilkar, 2017] express this matrix for a double ellipsoidal blimp as follows.

$$M_a = \begin{bmatrix} m_x & 0 & 0 & 0 & m_{tot} \cdot a_z & 0 \\ 0 & m_y & 0 & -m_{tot} \cdot a_z & 0 & m_{tot} \cdot a_x \\ 0 & 0 & m_z & 0 & -m_{tot} a_x & 0 \\ 0 & -m_{tot} \cdot a_z & 0 & J_x & 0 & -J_{xz} \\ m_{tot} \cdot a_z & 0 & -m_{tot} \cdot a_x & 0 & J_y & 0 \\ 0 & m_{tot} \cdot a_x & 0 & -J_{xz} & 0 & J_z \end{bmatrix} \quad (2.19)$$

Using the definitions provided in (2.18), [Kukillaya and Pashilkar, 2017] derive the Coriolis and centrifugal forces and moments as follows in Equation (2.21) with the state vector from Table 2.1 defined as $\mathbf{x} = [U \ V \ W \ p \ q \ r]^T$, where the vector elements refer to the respective velocities along the body \hat{x} -, \hat{y} - and \hat{z} -axes and the respective angular velocities of the body ϕ -, θ - and ψ -Euler angles in that order. Here, the variables a_x , a_y and a_z refer to the distance from the blimp's center of gravity to its center of volume. With these definitions, [Kukillaya and Pashilkar, 2017] summarize the forces and moments caused by Coriolis and centrifugal forces as follows:

$$\bar{F}_d = [F_{d,x} \ F_{d,y} \ F_{d,z} \ M_{d,x} \ M_{d,y} \ M_{d,z}]^T \quad (2.20)$$

where

$$F_{d,x} = -m_z q W + m_y r V + m_a x (q^2 + r^2) - m_a z r p \quad (2.21a)$$

$$F_{d,y} = -m_x r U + m_z p W - m_a x p q - m_a z q r \quad (2.21b)$$

$$F_{d,z} = -m_y pV + m_x qU - ma_x rp + ma_z(p^2 + q^2) \quad (2.21c)$$

$$M_{d,x} = (J_y - J_z)qr + J_{xy}pq + ma_z(p^2 + q^2) \quad (2.21d)$$

$$M_{d,y} = (J_z - J_x)rp + J_{xz}(r^2 - p^2) + ma_x(pV - qU) - ma_z(qW - rV) \quad (2.21e)$$

$$M_{d,z} = (J_x - J_y)pq - J_{xz}qr - ma_x(rU - pW) \quad (2.21f)$$

Buoyancy and gravitational forces and moments (F_{b+g}) Apart from its actuation, the primary forces responsible for giving a blimp its lift are related to its buoyancy. The buoyancy is the lifting force acting on an object caused by the pressure of the surrounding fluid. Since the buoyancy is the result of the surrounding fluid applying pressure to the object, and that the pressure of a fluid is greater closer to the earth's core, the direction of the buoyancy will be parallel to the earth gravitational field. The buoyancy of an object can be calculated in accordance with Archimedes' principle, summarized as follows:

$$B = m_f V_f = \rho g V_f \quad (2.22)$$

where m_f is the mass of the displaced fluid, ρ is the density of the displaced fluid and V_f is the volume of the displaced fluid. With a global reference frame with the \hat{z} -axis being perpendicular to the curvature of the earth, the buoyancy will have a constant magnitude. Furthermore, the force and moment contribution in the body frame will depend on the orientation of the blimp, thus it will be dependent on the body frame Euler angles. Since gravity and buoyancy act in parallel directions, it is convenient to combine them in to a single force and force moment vector. The combined force and force moment vector is for an axisymmetric blimp with the equilibrium pitch angle θ_e derived by [Kukillaya and Pashilkar, 2017], as follows:

$$\bar{F}_{(b+g)} = [F_{(b+g),x} \quad F_{(b+g),y} \quad F_{(b+g),z} \quad M_{(b+g),x} \quad M_{(b+g),y} \quad M_{(b+g),z}]^T \quad (2.23)$$

where

$$F_{(b+g),x} = -(mg - B)\sin(\theta + \theta_e) \quad (2.24a)$$

$$F_{(b+g),y} = (mg - B)\sin(\phi)\cos(\theta + \theta_e) \quad (2.24b)$$

$$F_{(b+g),z} = (mg - B)\cos(\phi)\cos(\theta + \theta_e) \quad (2.24c)$$

$$M_{(b+g),x} = -(mga_z - Bb_z)\sin(\phi)\cos(\theta + \theta_e) \quad (2.24d)$$

$$M_{(b+g),y} = -(mga_z - Bb_z)\sin(\theta + \theta_e) - (mga_x - Bb_x)\cos(\phi)\cos(\theta + \theta_e) \quad (2.24e)$$

$$M_{(b+g),z} = (mga_x - Bb_x)\sin(\phi)\cos(\theta + \theta_e) \quad (2.24f)$$

Fluid forces and moments (F_f) Wind and waves are common examples of the motion of a fluid. As these motions intersect with an object immersed in the fluid,

a force and force moment is applied to the object. For an axisymmetric blimp, with the state vector described in Table 2.2, fluid velocity described by translational and angular components as $x_f = [U_f \ V_f \ W_f \ p_f \ q_f \ r_f]$ and coefficients as described in Equation (2.27), [Kukillaya and Pashilkar, 2017] describe the forces and moments acting on the blimp caused by fluid motion as follows:

$$\bar{F}_f = [F_{f,x} \ F_{f,y} \ F_{f,z} \ M_{f,x} \ M_{f,y} \ M_{f,z}]^T \quad (2.25)$$

where

$$F_{f,x} = \bar{m}_z q W_f - \bar{m}_y r V_f \quad (2.26a)$$

$$F_{f,y} = \bar{m}_x r U_f - \bar{m}_z p W_f \quad (2.26b)$$

$$F_{f,z} = \bar{m}_y p V_f - \bar{m}_x q U_f \quad (2.26c)$$

$$M_{f,x} = -(\bar{J}_y q_f r - \bar{J}_z q r_f) - \bar{J}_{xz} p_f q \quad (2.26d)$$

$$M_{f,y} = -(\bar{J}_z r_f p - \bar{J}_x r p_f) - \bar{J}_{xz} (r r_f - p p_f) \quad (2.26e)$$

$$M_{f,z} = -(\bar{J}_x p_f q - \bar{J}_y p q_f) + \bar{J}_{xz} q r_f \quad (2.26f)$$

where the immersed parameters are given by:

$$\bar{m}_x = \bar{m} + X_{\dot{u}} \quad (2.27a)$$

$$\bar{m}_y = \bar{m} + Y_{\dot{v}} \quad (2.27b)$$

$$\bar{m}_z = \bar{m} + Z_{\dot{w}} \quad (2.27c)$$

$$\bar{J}_x = \bar{I}_x + L_{\dot{p}} \quad (2.27d)$$

$$\bar{J}_y = \bar{I}_y + M_{\dot{q}} \quad (2.27e)$$

$$\bar{J}_z = \bar{I}_z + N_{\dot{r}} \quad (2.27f)$$

$$\bar{J}_{xz} = 0 \quad (2.27g)$$

For Equations (2.26) and (2.27), the $\bar{\cdot}$ symbol above the parameters means that the parameter refers to the fluid displaced by the immersed object, i.e., the mass and inertia tensor elements of the displaced fluid.

Propulsive forces and moments (F_p) The means of propulsion or actuation of a blimp is what powers it and allows it to move in a desired way. The way the propulsive unit generates force and force moment to the blimp is entirely dependent on the actuator. For instance, if using rotors to actuate the system, their number and configuration relative to the body frame will determine the generated propulsive forces and moments. In the following, these forces will be described as follows:

$$\bar{F}_p = [F_{p,x} \ F_{p,y} \ F_{p,z} \ M_{p,x} \ M_{p,y} \ M_{p,z}]^T \quad (2.28)$$

where the actuator configuration and how it generates forces and moments is assumed to be known.

2.3 Control Theory

Within the field there are many topics which handle different parts of control. Some of these topics are controllers, trajectory generation and optimization which will be described and used in this thesis.

Proportional-Integral-Derivative Control

A proportional integral derivative controller (PID) is a type of controller that generates a control signal, $u(t)$ based on the control error $e(t) = r(t) - y(t)$ where $r(t)$ and $y(t)$ are the reference and the measurements, respectively. The controller is based on multiplying the error, its integral and its derivative by different scaling coefficients. This can be described in continuous time as follows:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt} \quad (2.29)$$

where K_p , K_i and K_d are the controller gains acting on the different parts of the calculated control error [Hägglund, 2019]. Commonly the derivative part is filtered before its use and the anti-windup protection is used to limit the magnitude of the integral part.

To use the PID controller in discrete time, which is used when programming the controller, a sampling time h must be introduced. Using this sampling time, the derivative and integral parts can be numerically converted from continuous to discrete time as in Equation (2.30), resulting in the discrete time PID approximation in Equation (2.31).

$$\int_0^t e(\tau) d\tau \implies \sum_{n=1}^k \frac{h}{2} (e(n) + e(n-1)) \quad (2.30a)$$

$$\frac{de(t)}{dt} \implies \frac{e(k) - e(k-1)}{h} \quad (2.30b)$$

$$u(k) = K_p e(k) + \sum_{n=1}^k \frac{h}{2} (e(n) + e(n-1)) + K_d \frac{e(k) - e(k-1)}{h} \quad (2.31)$$

The K_p , K_i and K_d parameters must be tuned carefully to give a stable and satisfactory controller. This can be done either by iteratively testing the controller for the process it is to control, or for a computer model that represents the actual process.

An alternative way of expressing the derivative part of the PID controller is by splitting the error into its components $r(t)$ and $y(t)$. This changes the controlled part from being the rate of change of the error signal to the difference between the time derivative of a reference signal and the time derivative of the measured state. This can be further expressed as the difference between a reference signal for on the state time derivative and the state time derivative. An example to explain this is if the measured states $y(t)$ are positions and $r(t)$ are position references, then

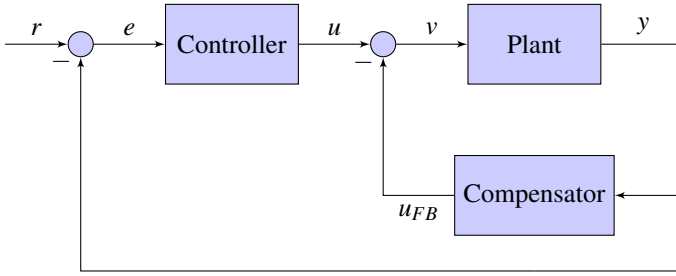


Figure 2.7: Block diagram describing how feedback compensation can be included in a control sequence.

$dy(t)/dt$ are the measured velocities and $dr(t)/dt$ are velocity references. To clarify the differences the reference, state and error derivatives will for the alternative derivative be noted as \dot{r} , \dot{y} and \dot{e} respectively. This alternative derivative is expressed in Equation (2.32) and the accompanying PID controller in Equation (2.33).

$$\frac{de(t)}{dt} = \frac{dr(t)}{dt} - \frac{dy(t)}{dt} \approx \dot{r}(k) - \dot{y}(k) = \dot{e} \quad (2.32)$$

$$u(k) = K_p e(k) + \sum_{n=1}^k \frac{h}{2} (e(n) + e(n-1)) + K_d \dot{e}(k) \quad (2.33)$$

Feedback Compensation

To account for known disturbances or unwanted behaviour that can be modelled, feedback compensation or feedback linearization can be used. The concept of feedback compensation is to remove the effect of the unwanted disturbances by creating a control signal that is equal to the modelled unwanted behaviour with a switched sign [Meng et al., 2021]. This is illustrated in Figure 2.7, where if the model is perfect, the effects of the disturbance will be cancelled letting the controlled states be strictly controlled by the designed controller. A simple example of this is to consider a helium balloon with a downwards pointing propeller attached to it. If the desired action is to maintain the balloon at a stable altitude the propeller must generate a force that is equal to the upwards lifting force generated by the balloon's buoyancy. With a disturbance model equal to the lifting force of the balloon, a feedback compensator will cancel this lifting force. If it is desired for the balloon to also change its altitude, then a different control strategy, such as a PID can be implemented which will now only act to move the balloon and not to keep its buoyancy from lifting it uncontrollably.

Trajectory Generation

To move a vehicle from point to point, the vehicle must move along some path through \mathbb{R}^3 -space. The closest path between the two points would be a straight line,

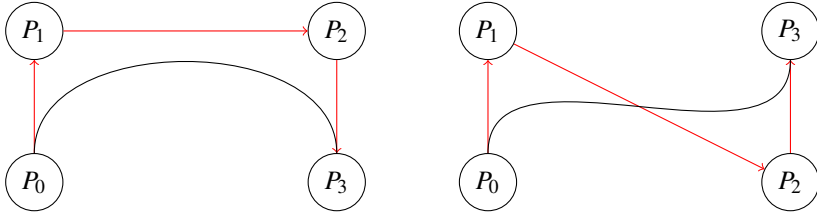


Figure 2.8: Illustration of two cubic Bezier curves generated from the same points but in different orders.

but in the event of obstacles or constraints on the terminal angle, the path must take some other shape than that of a single straight line. One way of solving this is to manually input a desired path, but a simpler way of solving it is to generate a path determined per time-slot, also known as a trajectory, through an algorithm. There are many ways in which this process can be automated. A common method for producing a curved trajectory through \mathbb{R}^2 - or \mathbb{R}^3 -space is the Bezier curve. A Bezier curve is a generated \mathbb{R}^n path that is calculated from n coordinates with a Bernstein basis polynomial of degree n and is expressed as follows [Hsu and Liu, 2020]:

$$B(t) = \sum_{i=0}^n \binom{n}{i} (1-t)^{n-i} t^i P_i \quad (2.34)$$

The curve can be described as adding the contribution of moving between straight lines between the set coordinates, with the contribution of each line being proportional to the time from start to finish, see Figure 2.8. For airships that move in \mathbb{R}^3 it is most common to calculate the Bezier curve for $n = 3$ which develops into the cubic Bezier curve expressed as follows:

$$B(t) = (1-t)^3 P_0 + 3(1-t)^2 t P_1 + 3(1-t) t^2 P_2 + t^3 P_3 \quad (2.35)$$

with its time derivative expressed as:

$$\frac{dB(t)}{dt} = 3(1-t)^2 (P_1 - P_0) + 6(1-t)t (P_2 - P_1) + 3t^2 (P_3 - P_2) \quad (2.36)$$

Worth noting is that t refers to a normalization of the time it takes from the first point to the last, thus $0 \leq t \leq 1$. By storing the calculated \mathbb{R}^3 -coordinates for equally spaced values of t , the \mathbb{R}^3 -trajectory is generated [Hsu and Liu, 2020].

Least Squares Optimization

When using a controller that measures the states of a vehicle, there might not be a clear relationship between the states and the actuation unit. For example, take a vehicle with three propulsion units, one creating thrust upwards and to the right, one that creates thrust to the right and one to that creates thrust to the left. Now say

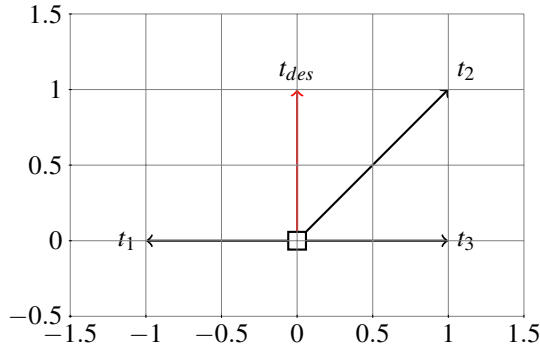


Figure 2.9: Illustration of the maximum thrust vectors of three propulsion units (black) and the desired resultant thrust vector (red).

that a controller gives as output that, to reach the desired setpoint the vehicle must generate thrust straight upwards. Also say that to conserve fuel the vehicle should use as little thrust as possible. Based on the configuration of the propulsion units, they must be actuated simultaneously but at different output effects to achieve the desired thrust, see Figure 2.9. In the figure it can be observed that optimal thrust, t_{des} , is achieved when $|t_1| = 1$, $|t_2| = \sqrt{2}$ and $|t_3| = 0$. For cases that are not as simple as the example in Figure 2.9, and when the calculations should be done iteratively, it is useful to find an algorithm that calculates the optimal thrusts automatically. One way of doing this is using constrained quadratic programming of which Least Squares (LS) optimization is a subcategory. In [Härkegård, 2002], both Sequential Least Squares (SLS) and Weighted Least Squares (WLS) algorithms are developed. The SLS problem formulation is expressed as follows:

$$u_S = \arg \min_u \left\{ \|W(u - u_p)\| : u \in \arg \min_{u_{min} \leq u \leq u_{max}} \|W_a(Bu - v)\| \right\} \quad (2.37)$$

where $\|\cdot\|$ notates the two-norm of a vector, that is the square root of the sum of quadratic vector elements [Glad and Ljung, 2000]. The WLS problem formulation is expressed as follows:

$$u_W = \arg \min_{u_{min} \leq u \leq u_{max}} \|W(u - u_p)\|^2 + \gamma \|W_a(Bu - v)\|^2 \quad (2.38)$$

The variables used in both least-squares formulations are described in Table 2.3. In words the SLS formulation in Equations (2.37) can be described as follows: The WLS formulation in Equation (2.38) can in words be described as follows:

The formulations described in Equations (2.37) and (2.38) only tell the objective of the least-squares algorithms, but not how they are to be performed. Minimizing a function is a process that requires multiple iterations to achieve a locally optimal

Find the actuator effort, u , that is as close to the desired actuator effort, u_p , in a square norm sense as possible by finding all u that also makes the control signal mapping of u as close to the desired control signal v as possible without letting u exceed a set of lower and upper limits defined by u_{min} and u_{max} .

Find the actuator effort, u , that is as close to the desired actuator effort, u_p , as possible and that makes the control signal mapping of u as close to the desired control signal v as possible without letting u exceed a set of rectangular limits defined by u_{min} and u_{max} .

Variables	Description	Dimension
u_S	Optimal sequential least-squares actuator effort	$m \times 1$
u_W	Optimal weighted least-squares actuator effort	$m \times 1$
u_p	Desired actuator effort	$m \times 1$
u	Variable actuator effort to be optimized	$m \times 1$
v	Desired control signal	$k \times 1$
B	Matrix mapping actuator effort to control signal	$k \times m$
W	Weighting matrix penalizing actuator effort differing from the desired	$m \times m$
W_a	Weighting matrix penalizing the mapped control signal from the desired	$k \times k$

Table 2.3: Description of least-squares optimization variables.

solution. One way of doing this is using Active Set algorithms. This is described briefly in [Härkegård, 2002] and in-depth by [Nocedal, 2006]. To summarize the descriptions made by [Härkegård, 2002] and [Nocedal, 2006], the algorithm can be explained as follows:

Let the least-squares problem be written on the form of $\min_u |Au - b|$, with the mapped equality constraint $Bu = v$ and the feasibility constraint $Cu \geq U$, where $C = [I \quad -I]^T$ and $U = [u_{min} \quad -u_{max}]^T$. Let W be the working set initially defined by guessing some constraints that might be active for the optimal solution. With this working set, and an attainable starting point $u_i = u_0$, see if the LS formulation is minimized, otherwise let $u_{i+1} = u_i + p_i$, where p_i is some step away from the starting point and of equal dimension as the starting point. For each active constraint acting on some index i , let $p_i = 0$. If u_{i+1} is outside of the existing constraints shorten the step of p_i by some factor, add this new constraint to the working set and try again. If u_{i+1} is within the existing constraints, calculate the Lagrange multipliers of the mapped equality constraint and the feasibility constraint. See [Nocedal, 2006, Chapter 12] for a description of Lagrange multipliers. If the Lagrange multiplier for the feasibility constraints is component-wise greater or equal to zero, it follows by the properties of a Lagrange multiplier that the optimal solution u , has been found and the current working set acting on the optimal solution is known as the active set. If the multiplier instead has negative components, update the working set to not include the constraint related to the smallest component of the Lagrange multiplier. With the updated working set, let the new u_{start} be the previous u_i and then reiterate the process by taking a step, p_i away from the starting point and so on. If no minimum has been found after $i = N$ iterations, u_N is used as the pseudo optimal u [Härkegård, 2002][Nocedal, 2006].

The exact algorithm used to solve the actuation problem is described in Appendix B, closely following the developments in [Härkegård, 2004]

Signal Processing and Filtering

The definition of a numerical state derivative in Equation (E.1) gives that as the sampling time h approaches zero, the influence of measurement noise increases, thus distorting from the value of the state derivative. To reduce this distortion the signals must be processed. A common way of processing the signals is to attempt to remove the noise from the original signal, removing its distortion effect from the derivative. Another approach is to filter or smooth the calculated derivative. To perform these tasks there are numerous types of filters, each acting in its own way with its own advantages and disadvantages [Taylor, 2012, p.151-165].

One such smoothing method is exponential smoothing, which acts on a measured time series of a desired length and finds a weighted average of the series. The exponential smoothing filter can be described as follows:

$$y_{filt}[k] = \alpha y[k] + \sum_{i=1}^{k-1} \alpha(1-\alpha)^i y[k-i] \quad (2.39)$$

where $y_{filt}[k]$ is the filtered signal at time index k , $y[k]$ is the measurement at time index k , and $\alpha \in [0, 1]$ is a weighting parameter. The weighting parameter α

is selected as how much weight the signals of each previous time index should be given. If α is near zero the most weight is given to the signals from the oldest time indexes, while α near one gives the most weight to the signals from the most recent time indexes. Instead of using the entire time series, it is sometimes only desired to use the most recent part of the time series when smoothing. This type of exponential smoothing is described as follows:

$$y_{filt}[k] = \alpha y[k] + \sum_{i=k-N}^N \alpha(1-\alpha)^i y[k-i] \quad (2.40)$$

where $N \geq 1$ is the order of the filter meaning how many time samples it acts on. An example of when the limited filter is used is when it is costly to store a lot of data and if there are restrictions on how long time the calculation can take [Hyndman et al., 2008, p.13].

2.4 The CrazyFlie and Lighthouse Positioning Deck

A CrazyFlie is a small quadcopter UAV, consisting of a PCB control board, a battery, four DC motors and four propellers, with room for accessories, such as additional sensors [Bitcraze, 2020]. One of the additional sensors is the Lighthouse positioning deck, which in combination with Lighthouse V2 base stations yield accurate position and attitude estimates of the CrazyFlie. The positioning system can in short be described as the Lighthouse V2 base station sending out planes of light which intersect with four sensors placed on the Lighthouse positioning deck. Based on the observed angles that these planes intersect with the sensors, a position relative to the Lighthouse V2 base station can be calculated [Bitcraze, 2021]. For an in-depth explanation of how the position and attitude is calculated, see the Lighthouse positioning system documentation provided by Bitcraze AB on their web page.

In accordance with the system identification study performed in [Förster, 2015-08], the generated thrust of a single CrazyFlie motor, f_i can be mapped as a polynomial function of the control signal c_i . The mapping identified in [Förster, 2015-08] is expressed as

$$f_i = 2.130295 \cdot 10^{-11} \cdot c_i^2 + 1.032633 \cdot 10^{-6} \cdot c_i + 5.4845600 \cdot 10^{-4} \quad (2.41)$$

3

Methodology

In this chapter, the practical parts and steps of progression of the master's thesis are presented. This involves descriptions of how the literature review was performed, how the blimp UAV was modelled theoretically, how the Lighthouse positioning system was set up, how the computer vision algorithms were implemented and how the blimp UAV was controlled. The chapter also describes how the system is tested.

3.1 Modeling the Blimp

The first step of the theoretical work is to create a mathematical model of a to be used blimp, and implement it in a simulation software, in this case Matlab and Simulink. The mathematical model selected for the thesis is the one derived by [Kukillaya and Pashilkar, 2017] which is described in Section 2.1. Before describing the implementation, a list of assumptions and delimitations must be made. The first assumption is that as the blimp is purposed to operate indoors, the velocity of the surrounding air is assumed to be zero, leading to the contribution of $\bar{F}_f = \bar{0}$. Secondly, it is assumed that the angle of attack α , sideslip angle β and rudder deflection angles δ_i , are all set to zero. This leaves the \bar{F}_{a+c} vector contribution as only affecting the F_x component by the size of $\rho V_0^2 C_{y1}/2$. Since the purpose of the blimp is to move slowly ($V_0 \approx 0.1m/s$), which in combination with $C_y \approx 0$, gives that this component can be ignored (see Figure 21 in [Kukillaya and Pashilkar, 2017]). Thus, \bar{F}_{a+c} reduces to a vector of zeros. This leaves the contribution of the F_p , $F_{(b+g)}$ and F_D vectors where the propulsion vector F_p will be described in Section 3.2.

With the data for the intended blimp seen in Figure D.1 in Appendix D, the coefficients in Equation (2.17) becomes:

$$\begin{aligned} a_1 &= 1.341m, a_2 = 1,490m \implies a = 2.831m \\ b &= c = 0.472/2 = 0.236 \\ m_{dis} &= (2/3) \cdot \pi \cdot \rho_{air} \cdot (a_1 + a_2) \cdot b^2 = 0.4045 \end{aligned}$$

$$\begin{aligned}
 e &= 0.9965 \\
 \alpha_0 &= 0.0307 \\
 \beta_0 = \gamma_0 &= 0.9847
 \end{aligned}$$

With these coefficients and a placeholder variable for the total weight of the airship, $m_{airship}$ the added mass matrix in Equation (2.15) with the components in Equation (2.16) become:

$$M_{add} = \begin{bmatrix} 0.0063 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.3923 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.3923 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5950 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5950 \end{bmatrix} \quad (3.1)$$

Before the mass and inertia matrix in Equation (2.19) can be introduced, all components that are to be attached to the blimp must be determined to their position and mass. However, after the components' masses and inertias are introduced and the mass and inertia matrix is calculated, the simulation sequence can be set up. A sketch of the simulation implementation is illustrated in Figure 3.1. In the figure, contribution from F_p , F_d and $F_{(b+g)}$ are summed before they are multiplied by the inverse of the mass and inertia matrix, M_a^{-1} , producing the body state accelerations. The body accelerations are both integrated and sent back to the F_d block, and rotated from the body frame to the global frame. In the global frame, they are integrated twice producing the global positions which are sent back to the $F_{(b+g)}$ block. This sequence is what is iterated through and thus simulated. The F_p blocks contribution will be described in Section 3.2.

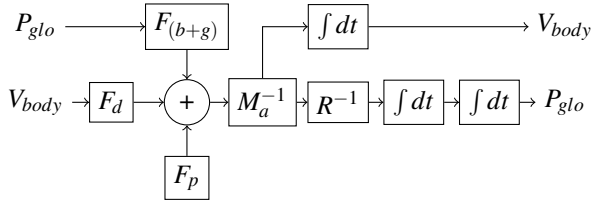


Figure 3.1: Illustration of the steps of the simulation sequence.

3.2 Propeller Optimization

An important part of controlling a vehicle is the placement of the actuators. For a blimp-like UAV with propellers as actuators, both the number, direction and placement of these propellers are relevant. For example, by placing a propeller in a way

that coincides with the \hat{x} -axis allows for propulsion forwards or backwards, but does not allow the generation of any moment around the axes which is needed to create rotation. By instead placing the propeller parallel to the \hat{x} -axis but translated from it, both propulsion and rotation are created, but it requires an additional propeller to counteract the rotation if only propulsion is desired. Since the goals of the thesis, listed in Section 1.2 includes being able to control both the position and the rotation of the UAV, it is important to be able to maintain six degrees of freedom. To achieve this the propellers must be placed so that forces can be created in all directions as well as being able to generate moments around all three axes.

Definition of Forces and Moments

The force vector, F , in Figure 3.2, is made up of three component forces, F_x , F_y and F_z , each being parallel to the \hat{x} -, \hat{y} - and \hat{z} -coordinate axes respectively, which is illustrated in Figure 3.3. To calculate the size of these three components, the angles α and θ are introduced, where α represents the angle between the force vector F and the \hat{z} -axis, and θ the angle between the \hat{x} -axis and F_{xy} , which is given by projecting F onto the $\hat{x}\hat{y}$ -plane. Using these notations and definitions, the resulting force components arising from a force applied in a NWU body frame at the coordinates $[x_p, y_p, z_p]$ are:

$$F = F_{xy} + F_y = F_x + F_y + F_z \quad (3.2a)$$

$$F_{xy} = F \cdot \sin(\alpha) \quad (3.2b)$$

$$F_x = F_{xy} \cdot \cos(\theta) = F \cdot \cos(\theta) \cdot \sin(\alpha) \quad (3.2c)$$

$$F_y = F_{xy} \cdot \sin(\theta) = F \cdot \sin(\theta) \cdot \sin(\alpha) \quad (3.2d)$$

$$F_z = F \cdot \cos(\alpha) \quad (3.2e)$$

As illustrated in Figure 3.4, by translating the component forces so that their base intersect the $\hat{x}_B\hat{y}_B$ -, $\hat{x}_B\hat{z}_B$ - and $\hat{y}_B\hat{z}_B$ -planes, the resulting moments acting in the body frame become as follows:

$$M_x = F_z \cdot y_p - F_y \cdot z_p \quad (3.3a)$$

$$M_y = F_x \cdot z_p - F_z \cdot x_p \quad (3.3b)$$

$$M_z = F_y \cdot x_p - F_x \cdot y_p \quad (3.3c)$$

The subequations in Equation (3.2) are derived using a NWU reference frame, and must be transformed if they are to be used in a NED frame. This can be done by comparing the two reference frames which are illustrated in Figure 3.5 and noticing that the two frames have flipped \hat{y} - and \hat{z} - axes making the signs of these coordinates switch, resulting in the following equations:

$$x_p^{NED} = x_p^{NWU} \quad (3.4a)$$

$$y_p^{NED} = -y_p^{NWU} \quad (3.4b)$$

$$z_p^{NED} = -z_p^{NWU} \quad (3.4c)$$

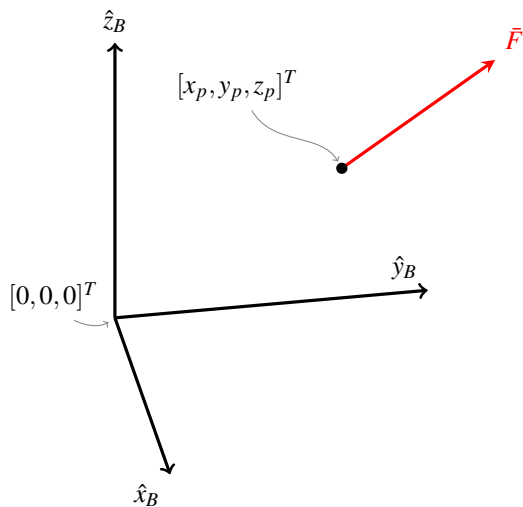


Figure 3.2: An arbitrary force vector \bar{F} placed in the body frame coordinate system $[\hat{x}_B, \hat{y}_B, \hat{z}_B]$, originating from the arbitrary coordinates $[x_p, y_p, z_p]$.

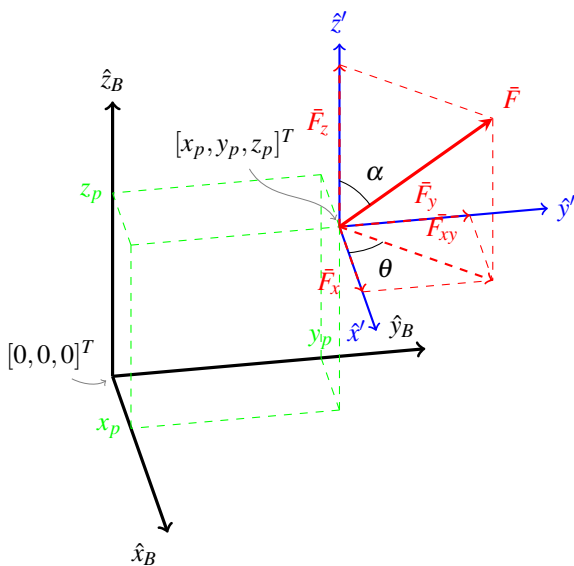


Figure 3.3: The force vector \bar{F} with an added coordinate system $[\hat{x}', \hat{y}', \hat{z}']$ with describing angles, projection lines and the projected \hat{x} -, \hat{y} - and \hat{z} -component forces of \bar{F} .

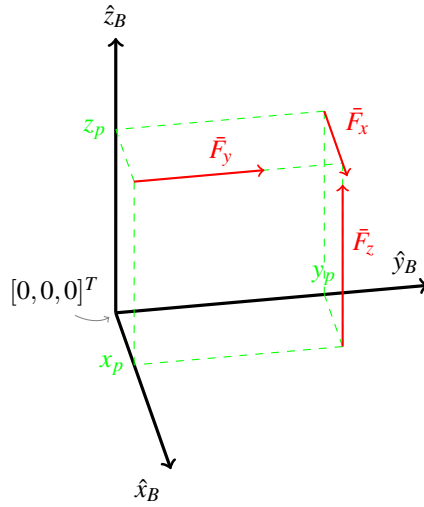


Figure 3.4: The component forces vectors \bar{F}_x , \bar{F}_y and \bar{F}_z , translated parallelly to intersect with the $\hat{y}\hat{z}$ -, $\hat{x}\hat{z}$ - and $\hat{x}\hat{y}$ -plane respectively.

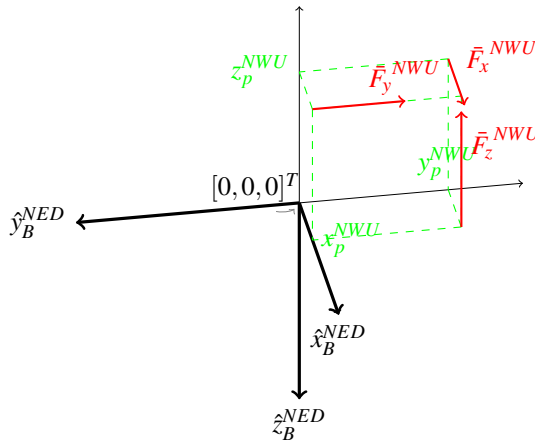


Figure 3.5: Illustration of how forces from a NWU frame relate to a NED frame.

Another observation is that the forces along the axes can be transformed by the rotation matrix in Equation (2.8) with $R(\pi, 0, 0)$, resulting in the following equation:

$$\begin{bmatrix} F_x^{NED} \\ F_y^{NED} \\ F_z^{NED} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} F_x^{NWU} \\ F_y^{NWU} \\ F_z^{NWU} \end{bmatrix} = \begin{bmatrix} F_x^{NWU} \\ -F_y^{NWU} \\ -F_z^{NWU} \end{bmatrix} \quad (3.5)$$

The final observation from Figure 3.5, is that the moments in the NED frame are calculated as follows:

$$\begin{bmatrix} M_x^{NED} \\ M_y^{NED} \\ M_z^{NED} \end{bmatrix} = \begin{bmatrix} F_z^{NWU} \cdot y_p^{NWU} - F_y^{NWU} \cdot z_p^{NWU} \\ F_x^{NWU} \cdot x_p^{NWU} - F_z^{NWU} \cdot z_p^{NWU} \\ F_x^{NWU} \cdot y_p^{NWU} - F_y^{NWU} \cdot x_p^{NWU} \end{bmatrix} \quad (3.6)$$

Using the equations derived in this subsection, in combination with Equation (2.5) gives that the resulting force from n actuators, attached to a body can be described as the sum of each actuator's contribution defined by its position described with x_p, y_p and z_p and its direction described by α and θ . The same logic is valid for the resulting force moment acting on the body. These two sums are expressed as follows:

$$\bar{F}_p^{Tot} = \sum_i^n \bar{F}_{p,i} \quad (3.7a)$$

$$\bar{M}_p^{Tot} = \sum_i^n \bar{M}_{p,i} \quad (3.7b)$$

Propeller Placements

To examine how well a set of actuators, or for this thesis propellers, can create a desired combination of forces and moments, numerical tests must be performed. This is done by first creating a propeller configuration vector defining eight propellers by positions and directions as a row vector. This vector is in turn used to extract the data of each propeller and through Equation (3.2) calculate its force and moment contributions which are stored in the 6×8 matrix, G_p . This matrix consists of the force and moment contribution of the eight propellers and is expressed as follows:

$$G_p = \begin{bmatrix} F_x^1 & \dots & F_x^8 \\ F_y^1 & \dots & F_y^8 \\ F_z^1 & \dots & F_z^8 \\ M_x^1 & \dots & M_x^8 \\ M_y^1 & \dots & M_y^8 \\ M_z^1 & \dots & M_z^8 \end{bmatrix}, \in \mathbb{R}^{6 \times 8} \quad (3.8)$$

In the matrix, each column holds the contribution of an individual propeller, with the row elements representing the size of the propeller's force and moment contribution. By multiplying G_p with a 8×1 thrust vector T , holding the normalized thrust amplitude of each propeller as its row elements, the resulting body force and moments, \bar{F}_B , are calculated as follows:

$$\bar{F}_B = G_p \bar{T} \quad (3.9)$$

Using Equation (3.9) with T as a variable, optimization using a nonlinear solver can be performed to evaluate how well a propeller configuration is able to generate desired the forces and moments. The minimization problem is solved using the

Matlab tool `fmincon`, and can be described as minimizing the two-norm of the difference between a desired force vector, \bar{F}_{des} , and a generated force vector, \bar{F}_{gen} , with an extra term penalizing the two-norm of the thrust by a factor λ , see below.

$$\min_{T \in \mathbb{R}^8} \quad \|F_{des} - G_p \cdot T\|^2 + \lambda \cdot \|T\|^2 \quad (3.10a)$$

$$\text{subject to} \quad [1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1]^T \geq T \quad (3.10b)$$

$$[0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]^T \leq T \quad (3.10c)$$

$$\lambda = 0.01 \quad (3.10d)$$

Thus, the solver finds the optimal thrust combination for the propeller configuration set, so that minimum thrust effort is required. The output of the solver is the optimal thrust and the calculated residual or cost.

By solving this problem for a variety of realistic forces and moments for a large number of propeller configurations, the residuals of each iteration can be collected for each propeller configuration. With the total residual associated with each configuration a comparison between different propeller configuration can be made. The configuration with the least total residual is the optimal configuration. The desired vectors of forces and moments are split into two, testing the forces and moments separately. The separate vectors are defined by setting up two ellipsoids of which the three-dimensional boundary positions represent $[F_x \ F_y \ F_z]^T$ and $[M_x \ M_y \ M_z]^T$, respectively. Thus, when testing the forces, $F_{des} = [F_x \ F_y \ F_z \ 0 \ 0 \ 0]^T$ and when testing the moments, $F_{des} = [0 \ 0 \ 0 \ M_x \ M_y \ M_z]^T$. Setting up the propeller configurations is an iterative process that combines testing with intuition, only limited by that the propellers must be able to be attached to the intended blimp, described by Figure D.1 in Appendix D. First, a few configurations were set up by thinking intuitively on how good placements might be and before they were run through the optimization algorithm. The residuals were then analyzed and both the desired force vectors and generated force vectors were plotted. Using these results more configurations were set up in a better way, and then optimized. This process was iterated until a configuration that seemed satisfactory was obtained, and the propulsion vector used in the blimp model can be described as $F_p = G_p \cdot T$. The force and force moment ellipsoids were sized with semi-major and semi-minor axes as described in Table 3.1.

3.3 Simulating the Process

This section functions as a description of how the modelled blimp and the obtained propeller configuration were implemented in Simulink. Using the model descriptions of the control sequence implementation and trajectory generation are also presented.

Axis	Maximum Force [N]	Maximum Force Moment [Nm]
\hat{x}	[-1,2]	[-0.4,0.4]
\hat{y}	[-1,1]	[-1.2,1.2]
\hat{z}	[-1,1]	[-1.6,1.6]

Table 3.1: Sizes of the maximum tested Forces and moments.

Component Placements

To obtain the total mass of the blimp and its inertia, all additional components must be included. Based on the propeller configuration obtained from the propeller optimization, x , y and z coordinates of the eight propellers are given. The masses of these propellers are set as the summed weight of the DC motor, propeller and some approximated weight of a motor holder. The two CrazyFlie boards that are used to control the motors are given coordinates so that they are placed on the top and bottom of the blimp, centered in the body frame \hat{x} -direction. The weights of the boards are set as the total CrazyFlie weight minus the weight of four motor and propeller pairs. The inertias are then calculated using Equation (A.1) and summated. The total mass of the blimp is set as the combined weight of all components and the blimp's own weight. Note that the inertia of the blimp itself and the helium within are also taken into account, and calculated using the same equations. With the components placed and both the mass and inertia of the complete blimp acquired, M_a , can be calculated and implemented in the Simulink model visualized in Figure 3.1.

Control Sequence

To control the blimp, a sequence of actions and calculations are performed. For this purpose a combination of a PID controller, feedback compensation and LS optimization is used. The sequence is described by five blocks in Figure 3.6. The *PID*-block represents the PID controller, the *Compensator*-block represents the feedback compensation, the *LS* block represents the least-squares optimizer, the *B*-block represents the $\mathbb{R}^{6 \times 8}$ propeller configuration, and the *Blimp*-block represents the modelled blimp dynamics. The states used in the control sequence are measured and collected in $y \in \mathbb{R}^{12}$ and are defined as follows:

$$y = \begin{bmatrix} \bar{x} \\ \dot{\bar{x}} \end{bmatrix} \quad (3.11a)$$

$$\bar{x} = [x_B \quad y_B \quad z_B \quad \phi_B \quad \theta_B \quad \psi_B]^T \quad (3.11b)$$

$$\dot{\bar{x}} = [\dot{x}_B \quad \dot{y}_B \quad \dot{z}_B \quad \dot{\phi}_B \quad \dot{\theta}_B \quad \dot{\psi}_B]^T \quad (3.11c)$$

Additionally, $r \in \mathbb{R}^{12}$, is the reference, $e \in \mathbb{R}^{12}$ is the state error, $v_{PID} \in \mathbb{R}^6$ is the PID control signal, $v_{FB} \in \mathbb{R}^6$ is the feedback compensation signal, $U_W \in \mathbb{R}^8$ is the optimized actuator efforts and $v_{gen} \in \mathbb{R}^6$ are the generated forces.

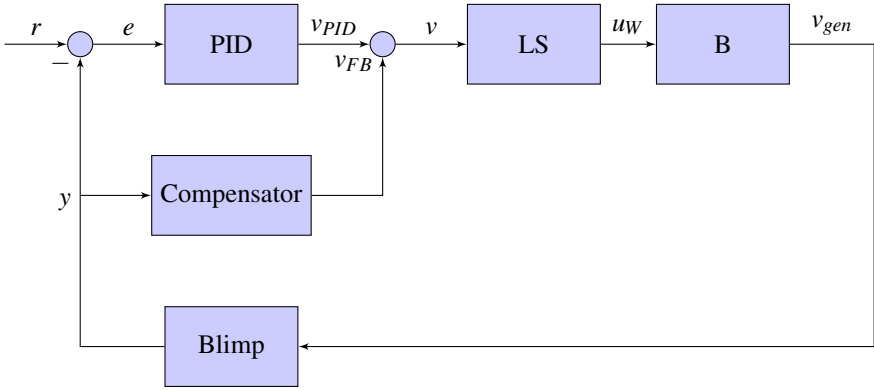


Figure 3.6: Block diagram describing the control sequence used to control the simulated blimp.

Tuning the PID: When tuning the PID parameters in Equation (2.29), the LS part is left out, letting $v_{gen} = v$ in Figure 3.6. The process is then simulated iteratively for different references and parameter values, until satisfactory results are obtained. For the purpose of this thesis, satisfactory results are defined as the modelled blimp being able to follow ramp trajectories of slope 0.2 meters per second for the x , y and z states, a slope of 0.1 meters per second for the θ and ϕ states, and of slope 0.5 radians per second for the ψ state. When the parameters are tuned for the different ramp trajectories, the blimp is also tested for following sinusoidal trajectories of frequencies 0.2 radians per second and amplitude 1.0 meters for the x , y and z states, and of frequencies 0.2 radians per second and amplitude π for the ϕ , θ and ψ state. If the trajectory following is deemed satisfactory for the sinusoidal trajectories as well, the final PID parameters are found. If not, the parameters are altered until the sinusoidal trajectories are properly followed, then the ramp curves are tested again for the new parameters, and if the ramp and sine tests are passed the final PID parameters are found.

Implementing the Least-Squares Optimization: Once the PID controller is properly tuned, least-squares optimization is implemented to compute the actuator signals. To do this, firstly B in Figure 3.6 is set to be the selected propeller configuration matrix defined as $G_P \in \mathbb{R}^{6 \times 8}$ in Equation (3.8). Secondly the LS algorithm is implemented and tested in the simulations. The algorithm is selected as the weighted least-squares method, and is strictly based on the algorithm designed by [Härkegård, 2004], visible in Appendix B. In the algorithm, u_{max} and u_{min} are set to match the limits used for the propeller configuration optimization seen in Equation (3.10d), the algorithm parameter B is set to be equal to the B used to define the 6×8 configuration matrix, and the parameter v is the same as the 6×1 control signal vector v calculated by the PID controller. The initial working set W , the initial actuator effort

u , and desired actuator effort u_{des} , are set to be empty 6×1 , 8×1 and 8×1 vectors, respectively. However, each time the LS optimization finds an optimal actuator effort, the connected working set becomes the active set, and it and the optimal actuator efforts are passed as the initial parameters for the next optimization iteration. The maximum number of iterations are arbitrarily set to 30, and can be decreased or increased if any deadlines are missed or the calculated effort is deemed sub-optimal. The other algorithm parameters are left empty, thus letting the weighting matrices W and W_a in Table 2.3 be 6×6 and 8×8 identity matrices.

Once the LS optimizer is implemented, it is placed in the control sequence, after the PID controller, and followed by a multiplication by the propeller configuration matrix B . To make sure that the generated actuator efforts represent the desired outputs i.e., $v_{gen} = sat(v)_{vmin}^{vmax}$ similar tests as for tuning the PID controller are run. However, before these tests can be performed, it is noted that the desired forces and moments represented by the PID control signal $v_{PID} \in \mathbb{R}^6$ are all expressed in the global frame, while the LS-block optimizes in reference to the body frame where B is defined. To account for this, v_{PID} , is multiplied by the rotational matrix defined in Equation (2.8) before the signal is passed to the LS block.

To make the generated actuator efforts less volatile some modifications are made to the weighted least-squares algorithm presented in [Härkegård, 2004]. The modifications include letting the parameters u_{max} and u_{min} be constant, set as column vectors of ones and column vectors of zeros, respectively. Another modification of the original WLS algorithm is that the control signal derivatives are also penalized. This is achieved by adding a numerical derivative term, where, u_{old} , is the control signal of the previous iteration, to the WLS problem formulation in Equation (2.38), resulting in the following formulation:

$$u_W = \arg \min_{u_{min} \leq u \leq u_{max}} \|W(u - u_p)\|^2 + \gamma \|W_a(Bu - v)\|^2 + \|W_d(u - u_{old})/dt\|^2 \quad (3.12)$$

where $W_d \in \mathbb{R}^{8 \times 8}$ is a diagonal penalty matrix acting on the WLS derivative term, and chosen to have equal size in each diagonal element.

Feedback Compensation

As a complement to the PID controller, the modelled dynamics of the blimp can be taken into account and compensated for. As described in Chapter 2.3, this consists of subtracting a force and moment vector, \bar{F}_{FB} , from the calculated PID control signal. The size of \bar{F}_{FB} is equal to the size of the contributions of the modelled equations of motion in Equation (2.12). By definition, the equations of motion are expressed in the blimp body frame and must therefore first be rotated to the global frame before they are subtracted from the PID control signal. This is done by multiplying them with the transposed rotation matrix, as in Equation (2.9b). When the PID control signal and the feedback compensation signal are both expressed in the global frame, the subtraction is performed and the difference is transformed to the body frame by multiplying with the regular rotational matrix in Equation (2.8).

Trajectory Generation

With the blimp's dynamics modelled, control signals generated, and through LS optimization converted to actuator efforts, and then mapped back to generated forces, it becomes relevant to generate a trajectory for the blimp to follow. When traveling in \mathbb{R}^3 , it is often desired to travel with the front of the blimp being parallel to the direction of where the blimp is going. The trajectory must therefore not only consist of a x , y and z components, but ϕ , θ and ψ , components as well. By setting a delimitation of having the blimp's ϕ and θ angles be zero, only the x , y , z and ψ components remain. The x , y and z , components are calculated using the cubic Bezier curve in Equation (2.35) for a number of time steps and storing the values in a trajectory matrix. Similarly, the respective state time derivatives are calculated as in Equation (2.36) and stored in a trajectory derivative matrix. To calculate the ψ component at each time step the geometrical definition of the Euler angle must be reviewed. With ϕ and θ both equal to zero, the yaw angle, ψ , can be described as lying in the $\hat{x}\hat{y}$ -plane, regardless of the z coordinate, which is depicted in Figure 3.7. Based on the definition of ψ in Figure 3.7, ψ can be calculated as in Equation (E.2) with $\psi = \text{atan}(dy/dx)$. Its time derivative, $d\psi/dt$ can be calculated by Equation (E.1) as $d\psi(t_1)/dt = (\psi(t_1) - \psi(t_0))/(t_1 - t_0)$, where t_0 and t_1 are the previous and current time measurements, respectively. Due to the properties of the trigonometric functions and their inverses, it is not unambiguous in what quadrant the calculated angle lies. An alternative is to use the $\text{atan2}(x, y)$ function available in Matlab and Python [MathWorks, 2021]. The function calculates the angle as $\text{atan}(y/x)$ but wraps the calculated angle to $-\pi \leq \psi \leq \pi$, making it continuous between $[-\pi, \pi]$, see Figure 3.8. However, to make the calculated references continuous when the yaw angles crosses the negative \hat{x} -axis and the positive \hat{x} -axis, making a full lap, a series of conditions and changes are made. These conditions can be seen in Table 3.2 and the resulting angle is set as $\psi_{next} = \psi_{calc} + 2\pi \cdot n$, where the integer n is altered according to the conditions in the table $n = n_{prev}$ otherwise.

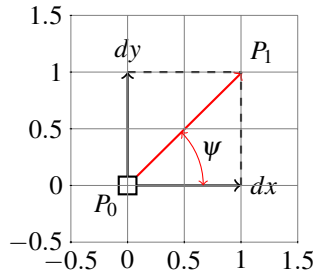


Figure 3.7: Illustration of how ψ is defined in the $\hat{x}\hat{y}$ plane.

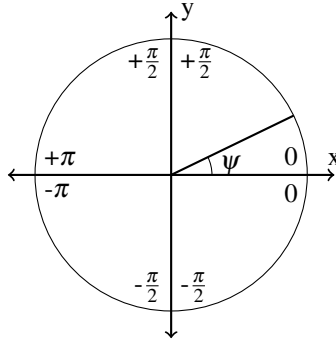


Figure 3.8: Illustration of how $\text{atan2}(x, y)$ is mapped to the unit circle.

From (ψ_{prev})	To (ψ_{calc})	Previous Lap (n_{prev})	Next Lap (n)
$\psi_{prev} < -0.95\pi + 2\pi \cdot n$	$\psi_{calc} \geq 0.95\pi$	<i>N/A</i>	$n = n_{prev} - 1$
$\psi_{prev} > 0.95\pi + 2\pi \cdot n$	$\psi_{calc} \leq -0.95\pi$	<i>N/A</i>	$n = n_{prev} + 1$
$\psi_{prev} > 0.95\pi + 2\pi \cdot n$	$\psi_{calc} \leq 0.95\pi$	<i>N/A</i>	$n = n_{prev} - 1$
$\psi_{prev} < -0.95\pi + 2\pi \cdot n$	$\psi_{calc} < -0.95\pi$	$n_{prev} \leq -1$	$n = n_{prev} + 1$

Table 3.2: Table of conditions and changes used to make ψ_{calc} as calculated in [MathWorks, 2021], continuous in time.

3.4 Real-time Experiments

Now that the blimp has been modelled, propeller placements been determined, PID controller designed, and least-squares optimization algorithm implemented, the physical blimp is to be controlled. Initially, some preparatory steps are performed, followed by multiple intertwined experimental steps performed iteratively to continually make adjustments and improvements. A block diagram that describes the control sequence used for the physical blimp is illustrated in Figure 3.9 and a sequence describing how the CrazyFlie and PC interact is illustrated in Figure 3.10.

Retrieving and Plotting Sensor Data

The first step is to be able to obtain position and attitude data. For this purpose, a CrazyFlie quadcopter equipped with a Lighthouse positioning deck and in radio communication with a host PC is used. As described in Section 2.4, the Lighthouse positioning deck gives accurate global frame position readings and body frame attitude readings, which through predefined Python methods can be retrieved and used. To verify the validity of these readings a CrazyFlie quadcopter is equipped with a Lighthouse deck, and a GUI thread is coded in Python, purposed to plot and print the retrieved position and attitude data at a frequency of 5 Hz. With the script available

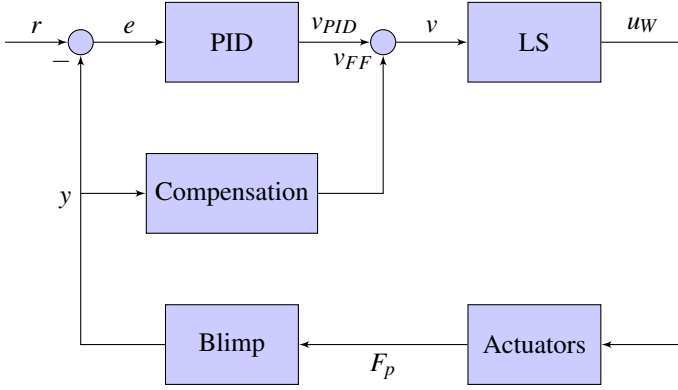


Figure 3.9: Block diagram describing the control sequence used to control the physical blimp.

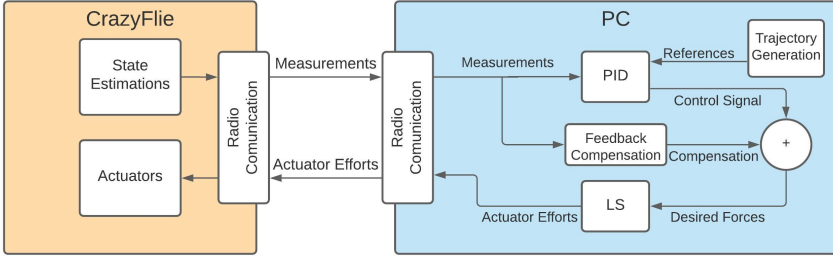


Figure 3.10: Sequence describing how of the components interact.

and a communication link established between the CrazyFlie and the host PC, the CrazyFlie is manually moved around in \mathbb{R}^3 and tilted around the three body axes, while simultaneously observing the plotted data. If the plotted data agrees with the estimated position and attitude of the CrazyFlie, the sensor data retrieval and plotting is deemed satisfactory. Since the simulated model assumed all sensor readings were measured in the blimp's CG the retrieved sensor data must be translated from the CrazyFlie's position on the blimp's surface to the blimp's CG. This is done by assuming that the position of the CrazyFlie relative to the CG in the body frame is $\bar{\mathbf{X}}_{\text{diff}}^B = (x_{\text{diff}}, y_{\text{diff}}, z_{\text{diff}})$ which is fixed in time. The CG position in the global frame can thus be calculated as:

$$\bar{\mathbf{X}}_{\text{CG}}^G = \bar{\mathbf{X}}_{\text{CF}}^G + \bar{\mathbf{R}}_{\text{BG}} \bar{\mathbf{X}}_{\text{diff}}^B \quad (3.13)$$

The attitude of the CG is assumed to be altered only by small offset angles approximated from the CrazyFlies's mounting position and calculated as:

$$\bar{\boldsymbol{\eta}}_{\text{CG}}^B = \bar{\boldsymbol{\eta}}_{\text{CF}}^B + \bar{\boldsymbol{\eta}}_{\text{diff}}^B \quad (3.14)$$

Preparing the Control Sequence

The next step of the preparations is to implement the designed PID controller alongside with the LS optimizer. As follows by the PID definition in Equation (2.33) the translational and angular velocities must be included. This is achieved numerically by using the retrieved position and attitude data and calculating their time derivatives as in Equation (E.1), where Δy is the difference between the current state and its previous reading, and Δt is the time difference between the two readings. The time difference or sampling time was selected as the shortest time possible without missing deadlines, chosen as 0.09s.

An issue with this type of velocity approximation, is that when the denominator tends towards zero, small differences in the nominator lead to large differences in the calculated velocity, making the approximation susceptible to noise and signal outliers. To avoid this a third-order exponential smoothness filter is implemented. With the full state vector y defined in Equation (3.11), the PID controller can be implemented in Python as follows by the pseudocode in Algorithm 1 available in Appendix C. To test that the code is valid, a known input is sent through the controller and its output is observed. If the output agrees with what is expected the controller is deemed satisfactory.

Just as in the simulation model, the PID controller is complemented with feedback compensation based on the equations of motion of the modelled blimp. As the equations of motion are expressed in the body frame they must be converted to the global frame before they are subtracted from the PID control signal. Whether or not feedback compensation is beneficial to the control sequence depends on how well the modelled blimp agrees with the dynamics of the real blimp. Tests must be performed to determine if the control sequence is better with or without feedback compensation.

Following the PID and feedback compensation blocks in the control sequence is first a global to body frame transformation before the least-squares optimization block. For the LS block the same modified weighted least-squares algorithm as described by the WLS formulation in Equation (3.12) is used, only converted from Matlab code to Python code. To test the LS block a similar test to that for the simulated LS algorithm is performed, where a known force and force moment vector is sent into the LS block and then the resulting actuator efforts are multiplied by the propeller configuration matrix and the generated forces observed. If the desired forces and moments agree with the generated ones, the least-squares optimization block is deemed satisfactory.

Actuating the Propellers

The CrazyFlie boards that actuate the motors and it is to these units that the optimized actuator effort signals are sent. The signals are sent as individual motor power parameters which takes an integer value, $c_i \in [0, 65535]$ as input and converts it to an angular velocity of the motor. The resulting thrust value of the individual motor

is a function of the integer value c_i is seen in Equation (2.41). However, to find the c_i value that represents the desired actuator effort, the inverse function must be found. This inverse function can be derived as follows:

$$\begin{aligned}
 f_i &= 2.130295 \cdot 10^{-11} \cdot c_i^2 + 1.032633 \cdot 10^{-6} \cdot c_i + 5.4845600 \cdot 10^{-4} \\
 \implies 0 &= 2.130295 \cdot 10^{-11} \cdot c_i^2 + 1.032633 \cdot 10^{-6} \cdot c_i + 5.4845600 \cdot 10^{-4} - f_i \\
 \implies 0 &= c_i^2 + \frac{1.032633 \cdot 10^{-6}}{2.130295 \cdot 10^{-11}} \cdot c_i + \frac{5.4845600 \cdot 10^{-4} - f_i}{2.130295 \cdot 10^{-11}} \\
 \implies c_i &= -\frac{51631.65}{2.130295} \pm \sqrt{\left(\frac{51631.65}{2.130295}\right)^2 - \frac{5.4845600 \cdot 10^{-4} - f_i}{2.130295 \cdot 10^{-11}}} \quad (3.15)
 \end{aligned}$$

Realize the Propeller Configuration

In the preceding chapters, the propeller configuration has been purely theoretical and consisting only of a mathematical matrix mapping of the actuator effort values to their respective force and force moment contribution in the blimp's body frame. To be able to place the propellers and motors in the proper configuration on the blimp, a motor holder was used. With the blimp geometry data displayed in Figure D.1 in Appendix D and the propeller placements in Figure 4.1, two types of motor holders were designed as CAD parts, one version for placing in the front of the blimp and one for placing in the back. Due to the symmetry of the desired propeller placements the two motor holder versions were identical in all aspects except for how the base was tilted. The tilt angle was set to imitate the angle between the body \hat{x} -axis of the blimp and the line tangent to the blimp's surface horizontally in the desired \hat{x} coordinates. On top of the tilted base a shaft was placed so that it was perpendicular to the body \hat{x} -axis. At the top end of the shaft, two cylindrical holders were placed at angles that match the α and θ angles of the desired propeller placements. The finished CAD parts were then 3D-printed and tested to see if they matched the weight requirements set by the buoyancy of the blimp, the dimensional requirements set by the tangential angle at which they are placed, and the dimensional requirements set by the size of the motors that they hold. If the holders matched the requirements, a matching set of holders were 3D-printed and used for the real-time experiments on the blimp. If not, the dimensions and design were reworked until the requirements were met.

With the designed motor holders, the motors themselves are modified. This is done by extending their connecting wires so that they are long enough to reach the CrazyFlie boards that power them. Additionally, holders for the CrazyFlie boards are designed as CAD parts with the purpose of making sure that the CrazyFlie boards remain properly fixed in the body frame even as the blimp moves. The CrazyFlie holders, are designed with a base curved to follow the curvature of the blimp, and a platform cast after the CrazyFlie board on top. To keep the CrazyFlie board fixed on the platform, lids fitted with screws are attached. It is now time to inflate the blimp and mount the 3D-printed holders, CrazyFlie boards, motors and

connecting wires. Due to some delays in getting the properly sized blimp, a smaller blimp with less buoyancy is initially used. As the smaller blimp is not buoyant enough to support the added components, a chord is attached to the blimp's top allowing some testing to be done before the larger more buoyant blimp arrives. The smaller blimp used for the initial testing is presented in Figure 3.11.



Figure 3.11: Picture of the smaller blimp used for the initial testing, with all components attached.

Testing and Tuning the Blimp

As previously mentioned, the initial testing is due to delays performed on a smaller blimp than intended. The smaller blimp is not buoyant enough to lift the required equipment and is therefore suspended in the air by a thin cord placed so that the blimp attitude angles are all close to zero while in equilibrium. The cord placement is for a blimp that is symmetric in the $\hat{x}\hat{y}$ plane illustrated in Figure 3.12 where MG is the net gravitational force of the blimp, B is the net buoyancy force and T is the cord lifting force. For a blimp geometry resulting in the center of gravity and center of buoyancy not coinciding, the cord must be placed so that equilibrium is achieved when $\delta = 0$. An expression for the placement is derived as follows:

$$F_x = T \cdot \sin\delta \quad (3.16a)$$

$$F_z = T \cdot \cos\delta + B - MG \quad (3.16b)$$

$$M_y = MG \cdot a + B \cdot b - T \cdot \sin\delta \cdot c - T \cdot \cos\delta \cdot d \quad (3.16c)$$

where

$$\begin{aligned}
 [F_z = 0] &\implies T \cdot \cos\delta = B - MG \implies T = \frac{B - MG}{\cos\delta} \\
 [M_y = 0] &\implies 0 = MG \cdot a + B \cdot b - T \cdot \sin\delta \cdot c - T \cdot \cos\delta \cdot d \\
 [T = \frac{B - MG}{\cos\delta}] &\implies 0 = MG \cdot a + B \cdot b - \frac{B - MG}{\cos\delta} \cdot \sin\delta \cdot c - \frac{B - MG}{\cos\delta} \cdot \cos\delta \cdot d \\
 [\tan\delta = \frac{\sin\delta}{\cos\delta}] &\implies 0 = MG \cdot a + B \cdot b - (B - MG) \cdot \tan\delta \cdot c - (B - MG) \cdot d \\
 [\delta = 0] &\implies d = MG \cdot a + B \cdot b - (B - MG) \cdot 0 \cdot c - (B - MG) \cdot d \\
 &\implies d = \frac{MG \cdot a + B \cdot b}{B - MG} \tag{3.17}
 \end{aligned}$$

For the blimp used, this position was instead determined through trial and error, resulting in the test blimp setup shown in Figure 3.11.

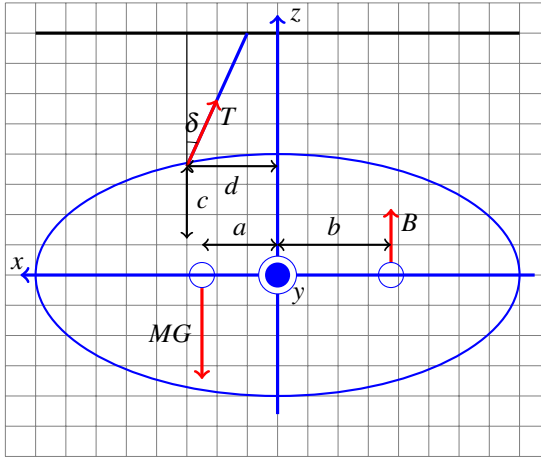


Figure 3.12: Illustration how a cord attached to the blimp affects its dynamics.

With the test blimp setup ready, a preliminary PID controller can be tuned. Since the model equations are based on the geometric data for a larger blimp, their respective feedback compensation components are excluded from the preliminary control sequence. The mechanics of having a cord attached to the blimp implies that if it moves away from its initial equilibrium position, additional forces and moments are introduced in the body frame which are viewed as disturbances. Attempts can be made to model these disturbances and include them in the feedback compensation block in order to remove their influence.

Due to additional complications, the originally intended blimp that was delayed, was destroyed. To solve this a substitute blimp is used, specified by the data in Figure D.2 in Appendix D. With the previously printed propeller holders and CrazyFlie

boards, the new blimp is assembled and can be viewed in Figure 3.13. To compensate for the extra buoyancy of the final blimp that was not needed to lift all the components mounted on it a ballast was placed on a cord on the blimp's bottom side. It was placed so that the CrazyFlie's orientation remains close to zero while in equilibrium. The ballast was shaped like a cup which enabled weights to be placed inside of it until equilibrium was achieved. The CrazyFlie-to-CG offset was visually approximated as $\bar{X}_{CG}^G = (-0.11, 0, -0.28)$ (m) and $\bar{\eta}_{CG} = (-1.5, 2, 0)$ (deg). With this blimp the system is tested iteratively and its PID parameters and feedback compensation components are tuned until a controller that can stably make the system reach position setpoints is obtained.



Figure 3.13: Picture of the final blimp with all components attached.

4

Results

In this chapter the results obtained during the thesis will be presented and given context. The form of presentation will vary between tables and plots.

Propeller Placements

The desired propeller placements obtained from the optimization algorithm described in Section 3.2 are presented in Table 4.1, where the x , y and z variables are coordinates in the body frame, and the α and θ variables describe spherical coordinate directions. Illustrations of the placements and the calculated residuals are presented in Figure 4.1

<i>Variables</i>	<i>P1</i>	<i>P2</i>	<i>P3</i>	<i>P4</i>	<i>P5</i>	<i>P6</i>	<i>P7</i>	<i>P8</i>
x	0.9	-0.9	-0.9	0.9	-0.9	0.9	0.9	-0.9
y	-0.3	-0.3	-0.3	-0.3	0.3	0.3	0.3	0.3
z	0	0	0	0	0	0	0	0
α	$-\frac{2\pi}{6}$	$-\frac{4\pi}{6}$	$-\frac{2\pi}{6}$	$-\frac{4\pi}{6}$	$\frac{2\pi}{6}$	$\frac{4\pi}{6}$	$\frac{2\pi}{6}$	$\frac{4\pi}{6}$
θ	$\frac{3\pi}{8}$	$\frac{5\pi}{8}$	$\frac{5\pi}{8}$	$\frac{3\pi}{8}$	$\frac{3\pi}{8}$	$\frac{5\pi}{8}$	$\frac{5\pi}{8}$	$\frac{3\pi}{8}$

Table 4.1: Table of propeller configuration parameters expressed in spherical coordinates.

Simulation PID Tuning

With the desired propeller configuration and Simulink model simulations were performed for both ramp and sinusoidal references. Simulated state responses to sinusoidal references for each state are presented in Figures 4.2–4.7. Their ramp reference counterparts are presented in Appendix C. The PID parameters obtained through iterative testing and what was used in the seen sinusoidal responses are presented in Table 4.2.

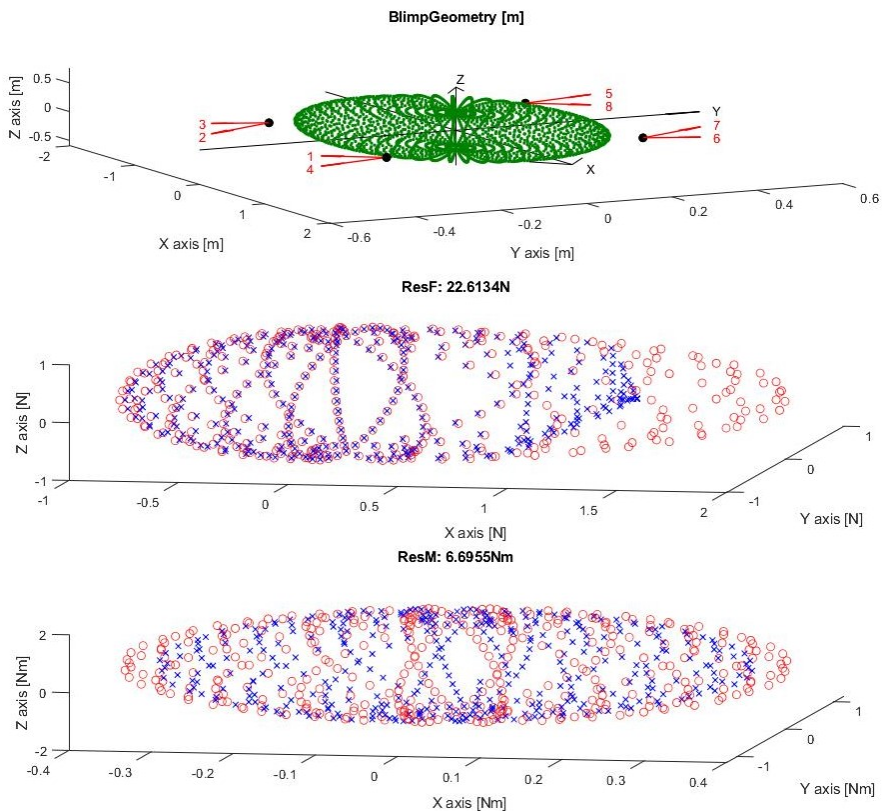


Figure 4.1: Calculated desired propeller placements for the original blimp.

	K_P	K_I	K_D
x	5	2	1
y	5	2	1
z	15	5	5
ϕ	2	0	2
θ	3	0	2
ψ	5	1	1

Table 4.2: Obtained PID parameters for the continuous-time simulated system.

Trajectory Generation

The final generated test trajectory is presented along with the simulated state responses in Figure 4.8, with the coordinates and attitudes used to generate the trajectory presented in Table 4.3. A three-dimensional illustration of the trajectory and

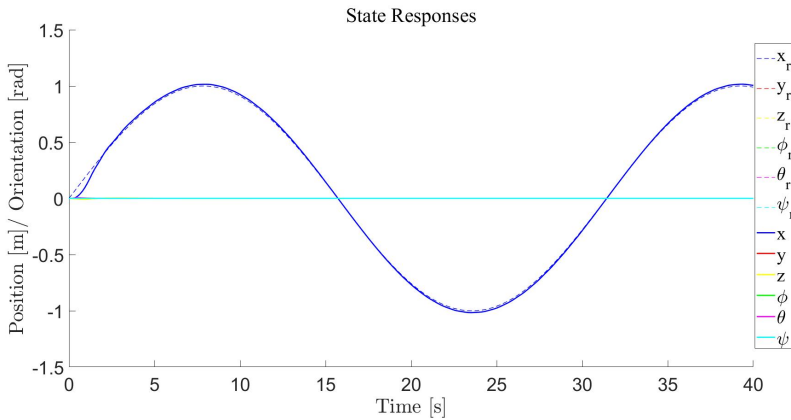


Figure 4.2: Simulated state responses to a sinusoidal reference for state x .

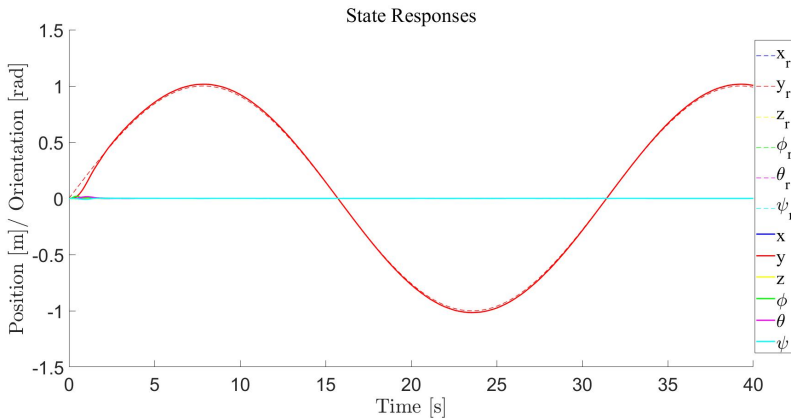


Figure 4.3: Simulated state responses to a sinusoidal reference for state y .

the state responses is presented in Figure 4.9.

Least-Squares Optimization

The validity of the least-squares optimization is tested by running the simulating including the LS-block with the generated trajectory as reference, which is presented in Figure 4.10. A comparison of the desired force generated by the controller and the actuated force generated by the LS block is presented in Figure 4.11 where the F_x components are compared. The influence of the added numerical derivative term in the WLS formulation is presented and compared in Figures 4.12 and 4.13.

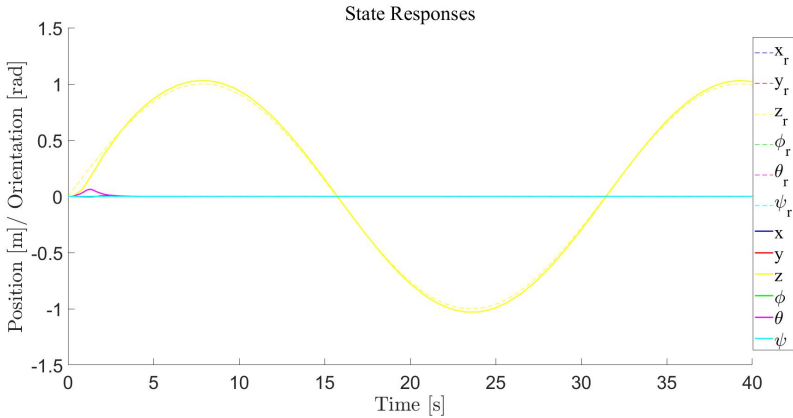


Figure 4.4: Simulated state responses to a sinusoidal reference for state z .

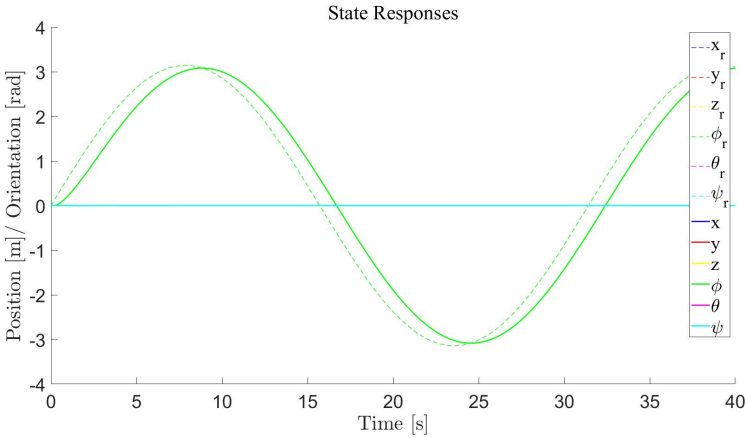


Figure 4.5: Simulated state responses to a sinusoidal reference for state ϕ .

Real-time Experiments

As the originally intended blimp was damaged a substitute blimp with different geometry had to be used. This led to that the originally derived optimal propeller placements could not be used and new ones had to be selected. The parameters describing the final propeller placements are presented in Table 4.4, and illustrations of the placements and the calculated residuals are presented in Figure 4.15. Running the simulation with data for the new blimp gave the state responses presented in Figure 4.16 and a comparison between the desired and generated forces is presented in Figure 4.17.

With the final prototype blimp, seen in Figure 3.13 constructed and the Python

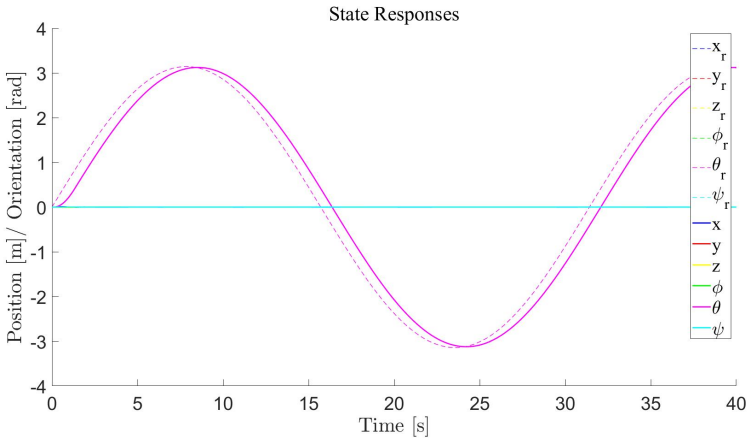


Figure 4.6: Simulated state responses to a sinusoidal reference for state θ .

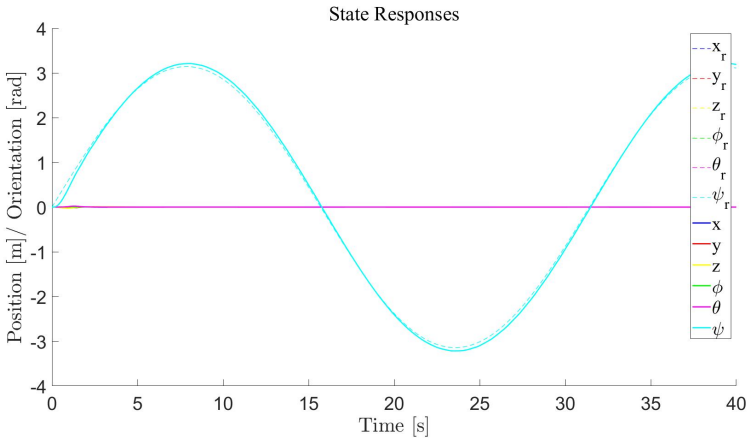


Figure 4.7: Simulated state responses to a sinusoidal reference for state ψ .

control sequence implemented, the system was tested by placing the blimp arbitrarily on the floor and then controlled to reach a set of constant setpoints. During the tests plots of the position and attitude states, the desired and generated forces, the desired and generated moments, and the actuator efforts were created. As the PID parameters obtained while simulating did not work for the prototype, new ones had to be found, which required numerous iterations. The final parameters are presented in Table 4.5 and plots for a selected trial run are presented in Figures 4.18 to 4.22 respectively. The trial was selected as the one that gave the visually most stable system. The desired forces and moments are defined as the output signal produced

	x	y	z	ϕ	θ	ψ
P_0	0	0	0	0	0	0
P_1	0	0	0	0	0	$\text{atan2}(P_2^y - P_0^y, P_2^x - P_0^x)$
P_2	-2	2	1	0	0	$\frac{3\pi}{2}$
P_3	-2	-2	0	0	0	0
P_4	2	-2	-1	0	0	$\frac{\pi}{3}$
P_5	2	2	0	0	0	$\frac{4\pi}{3}$

Table 4.3: Points used to generate the trajectory.

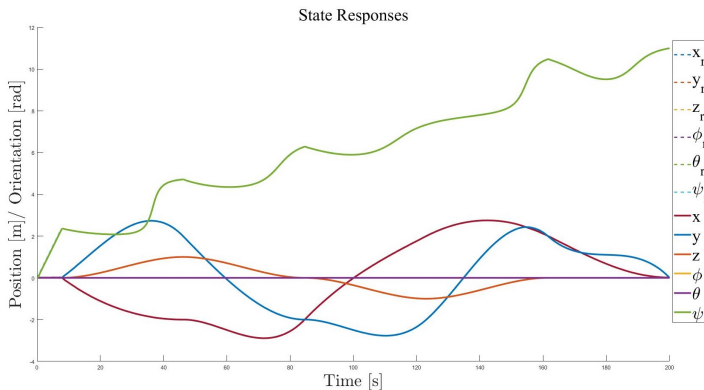


Figure 4.8: Simulated state responses for a reference trajectory not using the LS-block.

Variables	$P1$	$P2$	$P3$	$P4$	$P5$	$P6$	$P7$	$P8$
x	0.52	-0.52	-0.52	0.52	-0.52	0.52	0.52	-0.52
y	-0.36	-0.36	-0.36	-0.36	0.36	0.36	0.36	0.36
z	0	0	0	0	0	0	0	0
α	$-\frac{2\pi}{6}$	$-\frac{4\pi}{6}$	$-\frac{2\pi}{6}$	$-\frac{4\pi}{6}$	$\frac{2\pi}{6}$	$\frac{4\pi}{6}$	$\frac{2\pi}{6}$	$\frac{4\pi}{6}$
θ	$\frac{3\pi}{8}$	$\frac{5\pi}{8}$	$\frac{5\pi}{8}$	$\frac{3\pi}{8}$	$\frac{3\pi}{8}$	$\frac{5\pi}{8}$	$\frac{5\pi}{8}$	$\frac{3\pi}{8}$

Table 4.4: Table of propeller configuration parameters expressed in spherical coordinates.

by the PID and feedback compensation blocks transformed into the body frame, and the generated forces and moments are defined as the optimized actuator efforts calculated by the LS-block multiplied from the left by the propeller configuration matrix with its parameters presented in Table 4.4.

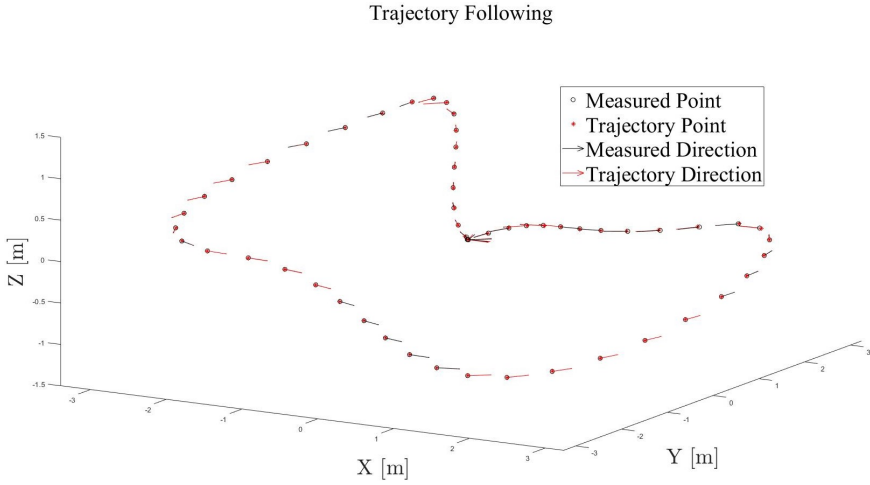


Figure 4.9: 3D illustration of the reference trajectory defined in Table 4.3 and followed in Figure 4.8, and the simulated responses.

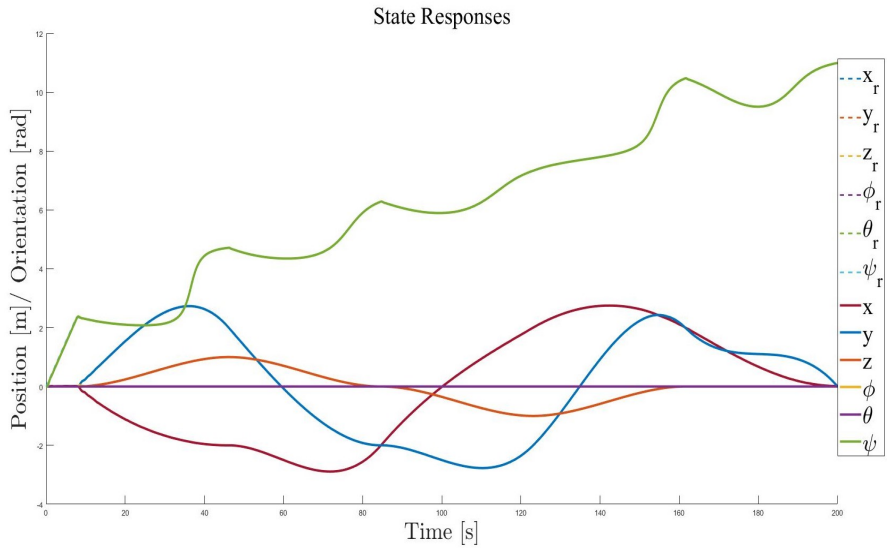


Figure 4.10: Simulated state responses for a reference trajectory using the LS-block.

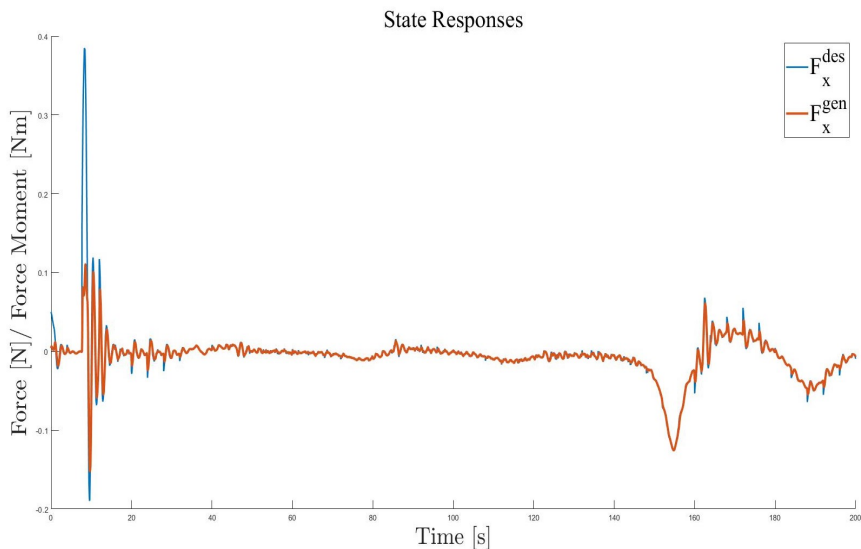


Figure 4.11: Simulated comparison of the forces sent from the controller and the forces generated from the LS-block using the data for the original blimp.

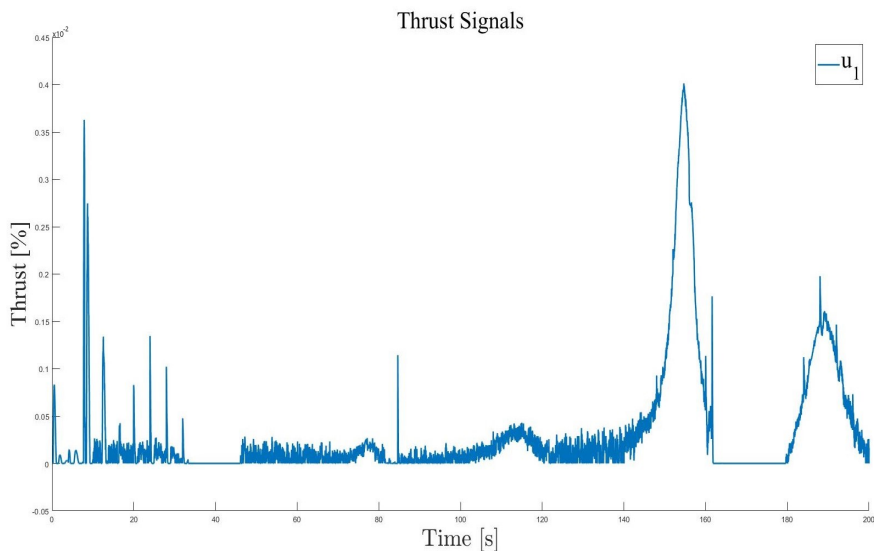


Figure 4.12: Plot of a thrust signal when not penalizing the numerical thrust derivative in Equation (3.12), with feedback compensation.

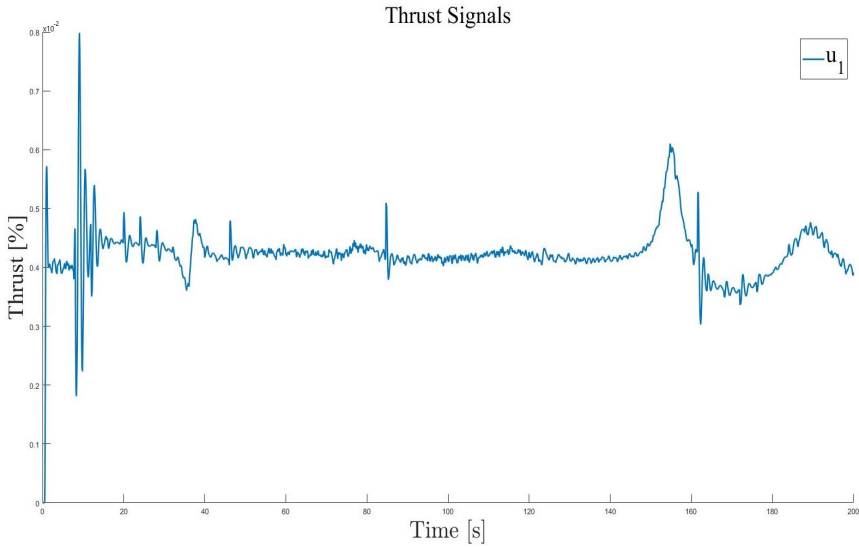


Figure 4.13: Plot of a thrust signal when penalizing the numerical thrust derivative in Equation (3.12), with feedback compensation.

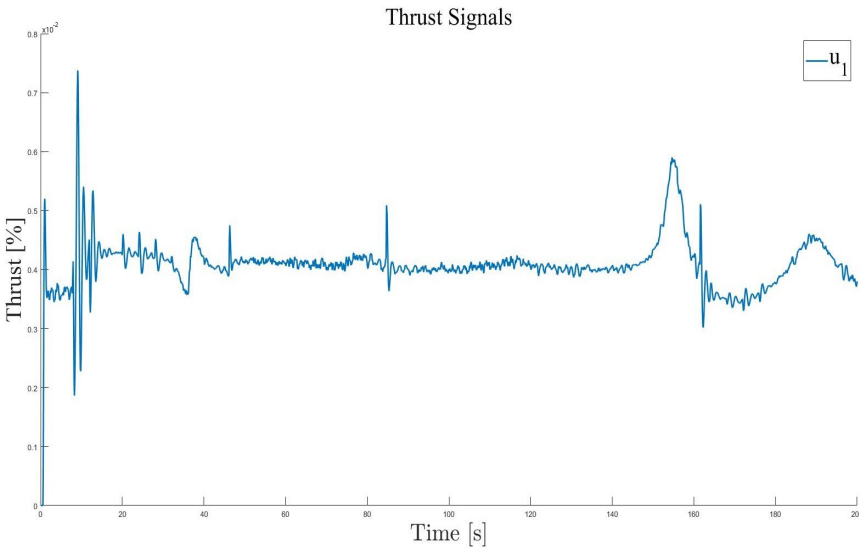


Figure 4.14: Plot of a thrust signal when penalizing the numerical thrust derivative in Equation (3.12), without feedback compensation.

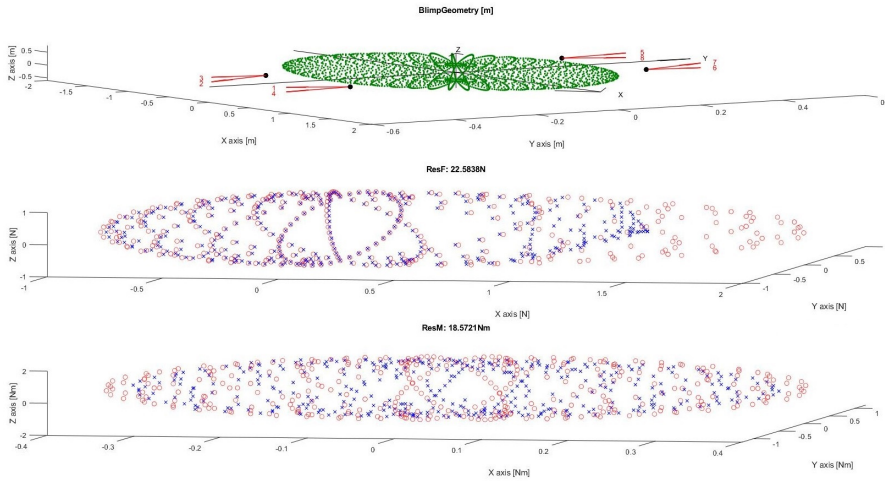


Figure 4.15: Calculated desired propeller placements for the final blimp.

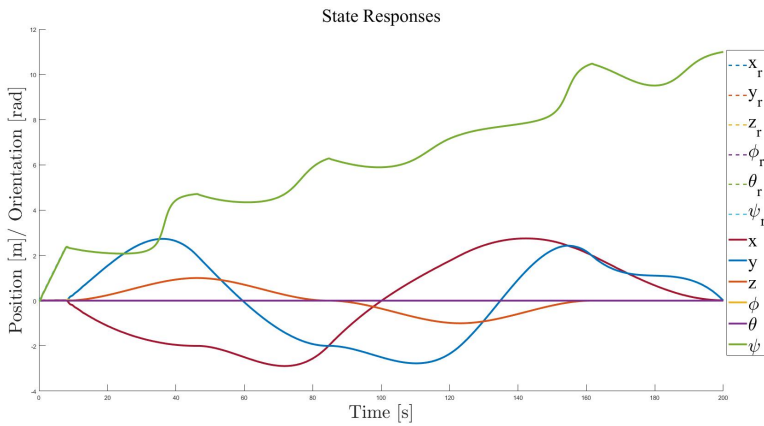


Figure 4.16: Simulated state responses using the data for the final blimp.

	K_P	K_I	K_D
x	0.07	0.05	0.099
y	0.15	0.1	0.05
z	0.25	0.05	0.25
ϕ	0.15	0.05	0.05
θ	0.5	0.05	0.07
ψ	0.16	0.06	0.06

Table 4.5: PID parameters obtained for the prototype blimp with a sampling time of 0.09 seconds.

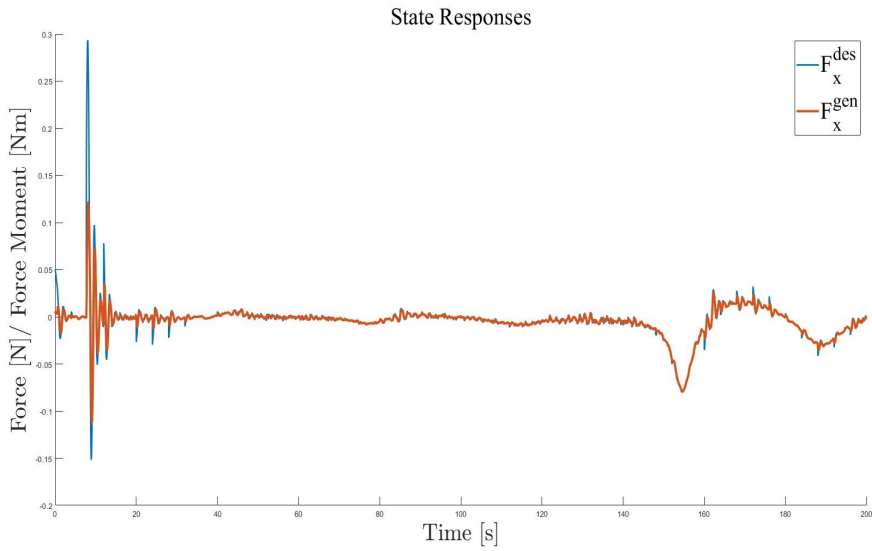


Figure 4.17: Simulated comparison of the forces sent from the controller and the forces generated from the LS block using the data for the final blimp.

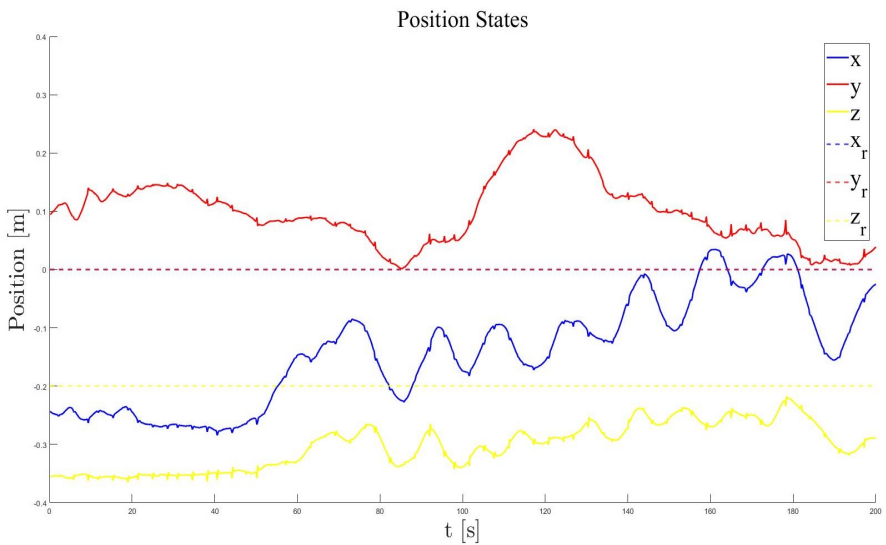


Figure 4.18: Experimentally obtained position measurements for a trial run.

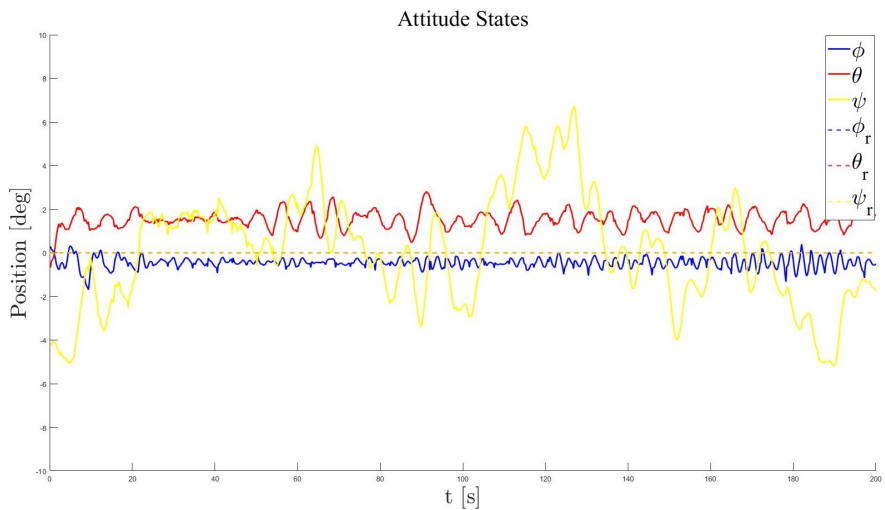


Figure 4.19: Experimentally obtained attitude measurements for a trial run.

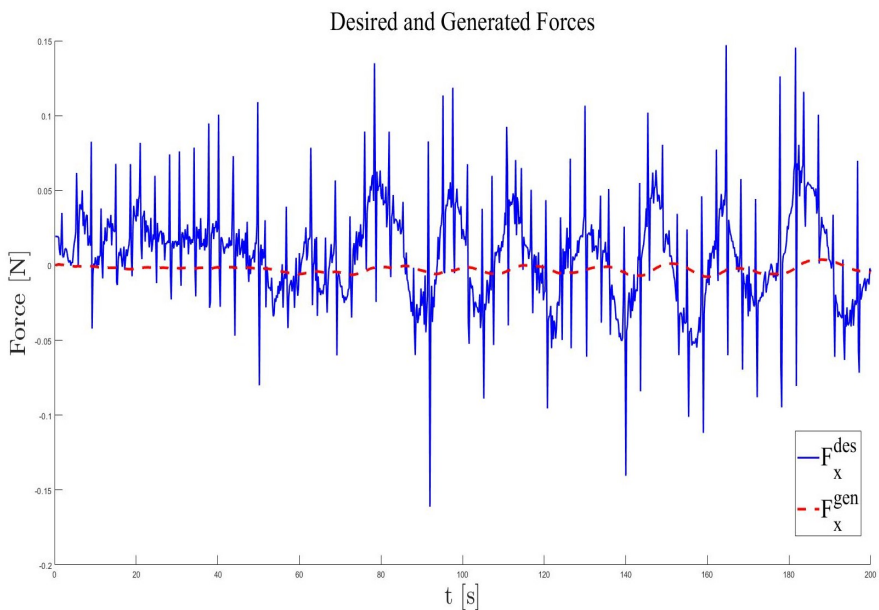


Figure 4.20: Experimentally obtained desired and generated forces in the body \hat{x} -direction.

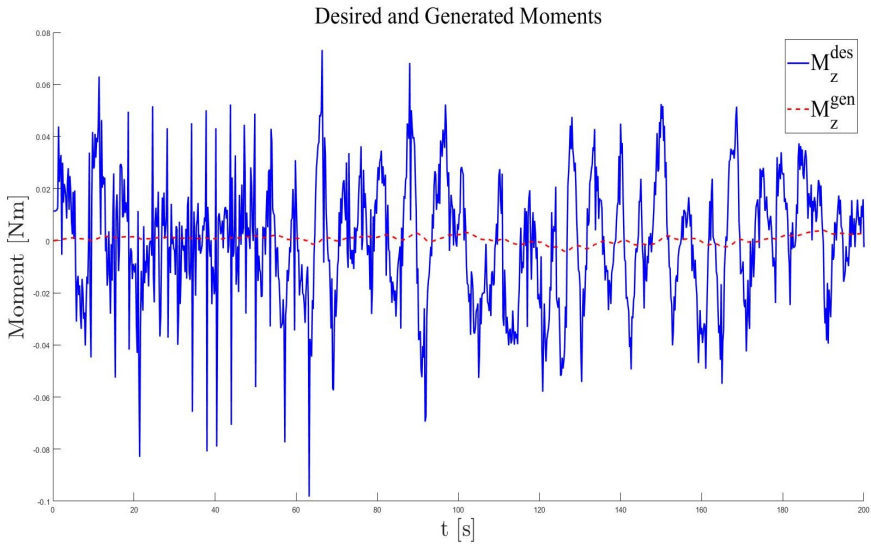


Figure 4.21: Experimentally obtained desired and generated moments around the body \hat{z} -axis.

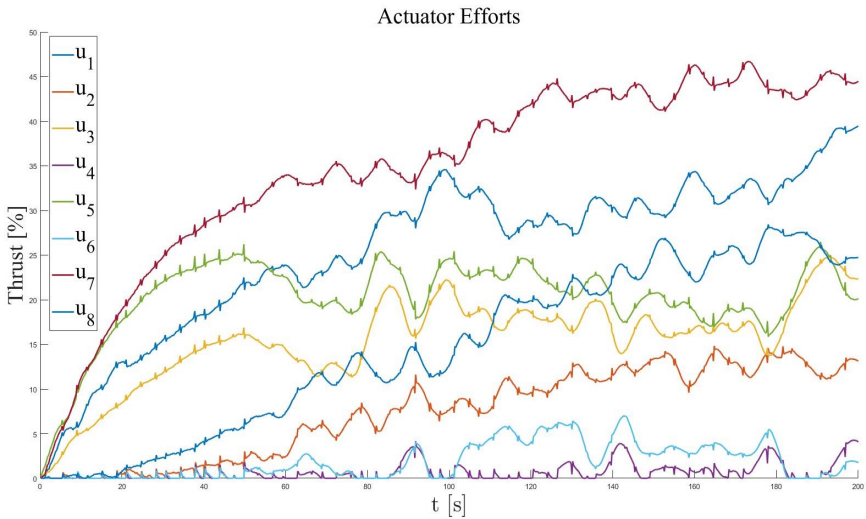


Figure 4.22: Experimentally obtained thrust signals expressed in percentage of maximum thrust.

5

Discussion

In this chapter, the work performed in the thesis will be reviewed and discussed. This includes discussing the choice of methodology, implementation and the results obtained. In addition, suggestions for future work and development will be presented based on observations made and experiences gained during the thesis.

5.1 Modelling the Blimp

The first point that can be made regarding the choice of model is that the shape of both the initial testing blimp and the final used blimp differed from the shape used in [Kukillaya and Pashilkar, 2017] to derive the equations of motion. The differences come in two forms. Firstly, the physical blimp did not have rudders and fins, and secondly, the physical blimp was not purely shaped like a double ellipsoid since its back part was not as rounded as the shape used in the model.

The first difference reduces the complexity of the aerodynamic force vector in Equation (2.14) and thus its actual force and moment contributions, supporting the choice to exclude it from the model used when simulating. However, the aerodynamic force vector's exclusion will have an influence on the quality of the simulations. The extent of this is according to its definitions directly proportional to the squared blimp velocity and the direction it travels. For slow-moving blimps purposed to have the side slip angle, β , and the angle of attack, α be zero, the the aerodynamic force vector will be small and only acting in the body \hat{x} -direction.

The second difference introduces a source of error which must be considered when comparing the simulated and real system. The difference will affect both how the geometrical parameters used in the added mass matrix in Equation (2.17) are calculated, and how the inertia components are calculated for the blimp itself. Consequently, this affects the values of the mass and inertia matrix components in Equation (2.19) which in turn affect how the Coriolis and centrifugal forces and moments are calculated in Equation (2.21). The shape differences are thus influential both to the Coriolis and centrifugal forces and moments, and to the added mass and iner-

tia matrix, which is what in the simulations is used to relate the applied forces and moments to the body accelerations.

A second point regarding the choice of model that can be made, is about the forces and moments caused by wind disturbances that were excluded in the model. With the delimitation that the velocity of the surrounding fluid is zero at all times made in Section 1.3, the forces and moments caused by fluid motion equal to zero by default. This assumption can be valid for rooms with little to no movement of any objects in the vicinity of the blimp, but not for crowded rooms. This claim is based on experiences that were obtained when performing experiments with the blimp, where for example if the blimp was sitting still on the floor and a person walked by, the blimp would move in a way that coincided with the direction the person was walking in.

5.2 Propeller Optimization

Optimizing the propeller placements was a step that took place early in the thesis, when only a blimp was chosen and in parallel with the implementation of the mathematical model. As in consequence, assumptions regarding both the limits and requirements of the generated forces and moments had to be made. The size of the force and moment ellipsoids used when performing the optimization algorithm were intuitively set up only to be able to compare how the individual configuration performed in relation to each other. As Table 3.1 indicates, the forces tested for along the positive \hat{x} -axis were larger than in all other directions since the blimp was purposed to mainly move forward in its body frame, with some translational adjustments being able to be made in the other directions. The same table states that the moments tested for around the \hat{x} -axis were the smallest followed by around the \hat{y} -axis and then \hat{z} -axis which had the greatest test moments. The blimp was purposed to have its ϕ and θ angles remain level during positional movement, but to allow some angle changes if desired. As for the size of the thrust vector generated by each propeller, this too was unknown at this point in time. The force was thus normalized to act between 0 and 1N, which can be interpreted as acting at a percentage of the propeller's actual maximum force.

Using a combination of rectangular and spherical coordinates to describe the propeller configurations worked well, the rectangular coordinates described the location of the propeller in relation to the body frame, and the spherical coordinates described its direction and the thrust it generates in each body frame direction. The reason it worked well is that it provided an understandable way of describing the propellers position and orientation that was also easy to make changes to. After each batch of configurations was tested, the configuration parameters of the top candidates were examined and altered to see how different variants performed.

The cost function in Equation (3.10d), was selected to mainly penalize the difference between the desired forces and moments and the generated forces and mo-

ments, and secondly the size of the calculated optimal thrust signals. The reason for this was that the main purpose of the propellers is to actuate in a way so that they collectively generate forces and moments that resemble their desired counterparts. If the size of these signals could be reduced while retaining the appropriate collective forces and moments, less energy would be required, which should also be taken into account. However at a much lower priority since the main goal is to stabilize and control the blimp. Having the optimization algorithm collect the residuals of each test as well as plot how well the generated forces and moments matched the desired ones was beneficial in the sense that it gave direct feedback to how each configuration performed and how it might be lacking.

The results obtained from the performed optimization are presented in Table 4.1 and in Figure 4.1. The chosen configuration might seem poor in the sense that it is unable to generate the maximal desired force in the \hat{x} -direction. However, the configuration was successful in generating close to all required moments. The property of being able to generate the desired moments was in this thesis more important than generating the maximum force in the \hat{x} -direction. This statement is based on that with a smaller force in the \hat{x} -direction when traveling forward the blimp will still accelerate to its desired velocity, even though slowly. But to also follow a trajectory and keep the front of the blimp pointing in the direction of the desired velocity, it should be able to rotate at a high pace.

5.3 Simulating the Process

The Matlab Simulink simulation was used mainly to verify that the designed PID controller, feedback compensation, LS optimization and trajectory generation worked in theory. The initial ambition was that the simulations and the results obtained from them would be relevant enough to be directly applied to the physical process. The first issue to be presented with this ambition is that the simulations were run in continuous time without consideration for the influence of sampling time.

The sequence of actions performed at each time step of the simulation is seen in the simulation sequence depicted in Figure 3.1, where F_p is equal to the signal, v_{gen} in the block diagram seen in Figure 3.6. In the same block diagram, the y -signal consists of the blimp's x -, y - and z -position states in the global frame, its ϕ -, θ - and ψ -attitude states, and the time derivatives of both the positions and attitudes. Additionally, the body frame translational and angular velocities are included as they are part of the equations of motion, used in the feedback compensator.

Both the ramp and sinusoidal reference tests performed to tune the PID were useful in that they provided initial verification that the controller was good enough to have the simulated blimp follow a rudimentary trajectory. Worth noting is that the introduction of feedback compensation was made at an early stage of the PID tuning to facilitate obtaining a stable system. By viewing the sinusoidal tests seen

in Figures 4.2-4.7 and at the same time consider the PID parameters in Table 4.2 some observations can be made. Firstly, not having integral action for the ϕ and θ parts gives a stable system with a steady-state error, which is expected. Secondly, it can be seen at the start of the x -, y - and z - plots that there are some intertwining effects that create movement of states that are supposed to remain still. This effect is however only brief and is compensated for after approximately 3 seconds.

The trajectory generation algorithm took a number of iterations and tests before it was deemed successful. This was caused by how the ψ -angle, which in the simulation is defined on $(-\infty, \infty)$ passed through $\pm\pi$ for which the $\text{atan2}(y, x)$ command is not continuous. The conditions in Table 3.2 were derived through trial and error, and their limit values were chosen intuitively to allow all conditions to be handled. The conditions were deemed good enough when the trajectory curve was continuous. With the generated trajectory the system was tested to see if the trajectory was slow enough for the blimp to be able to follow, which, according to Figures 4.8 and 4.9, it is successful at. It should be noted that since the conditions were derived through trial and error, it is not certain that they cover all scenarios and might have to be complemented.

As mentioned previously in the thesis, the logic that converts the desired forces and moments calculated by the controller and transformed to the body frame, is the least-squares optimization block. With the LS block sending thrust signals that are converted to generated forces and moments, the system was simulated giving the result seen in Figure 4.10. These results agree with those of not using the LS block. The penalty factor acting on the numerical derivative term added to the WLS formulation in Equation (3.12) is also a parameter that can be tuned. The purpose of increasing this penalty is to obtain a thrust signal that changes slower, allowing delays and slow actuator dynamics to catch up. By testing the system for different factors, a penalty of $15/dt$ was selected, where dt is the time between samples. A comparison between a thrust signal acting with and without this penalty can be seen in Figures 4.12 and 4.13. In the figures it is clear that the signal oscillates more when not penalizing the derivative, though on the downside, the signal is consistently greater with the penalty. To choose which of the two types is preferred, their implicit consequences must be considered. A greater thrust signal will lead to a higher energy consumption, while a more oscillating thrust signal will put greater demands on the actuation speed of the system. If the actuators are too slow to react to the given signal, it is likely that the oscillations in the signal will spread to the dynamics of the blimp. With this in mind and that it for this thesis was more important to obtain a stable system than a fast one, the effects of using the penalty are preferred. To test how well the system performed without the feedback compensation that was introduced at an early stage a similar simulation was run without it, resulting in Figure 4.14, which shows that the results are very similar. By comparing this figure with Figure 4.13 it can be seen that the system performed just as well without feedback compensation. A likely reason for this is that when moving

slowly the components of the Coriolis and centrifugal forces and moment vector, (F_d), used for the feedback compensation is small.

5.4 Real-time Experiments

Converting the control sequence designed in Matlab to Python required some adjustments since not all commands and data structures were readily available. To account for these differences the Numpy Python library was used. Since the two control sequence implementations differed there could be some performance differences that for the experiments went unnoticed. Another difference from the simulations is the introduction of a sampling time chosen as 0.09 seconds. This was implemented by comparing the system time before and after each iteration. If there was time left until the next sample the sequence would wait. This prevented the controller from acting continuously on the system and made the dynamics between each sample go unnoticed. To make the system more like the simulated continuous one the sample time was reduced as much as possible while still having time to perform the computations required for each iteration.

To obtain the position data from the Lighthouse positioning system, the CrazyFlie placed on the blimp's upper surface was calibrated for the used experiment area. This was done using the software provided by Bitcraze AB. With the CrazyFlie calibrated it was placed on the blimp so that it was aligned with the blimp's \hat{x} -axis, symmetric in its \hat{y} -direction and level in the $\hat{x}\hat{y}$ -plane. However, due to these placements being estimated visually a source of error is introduced here. Since the control sequence is defined for all states being measured in the blimp's CG, the measured signals had to be transformed to their CG counterparts by Equation (3.13) and (3.14). As these offsets were visually estimated as well, an additional source of error is introduced.

The motor holders that were designed and 3D-printed had to be redesigned several times until the design was finalized. One aspect that was optimized was the motor holding cylinders that required a diameter that gave a fit that allowed the motor to be inserted without excessive force but once inserted would remain in place. Another design consideration was to reduce the weight of the holder without sacrificing the sturdiness of the print.

Due to different circumstances regarding the delivery and quality of the blimps used, three different blimps were used throughout the project. The one that was intended to be used, described in Figure D.1, the smaller one that was used for the initial testing, seen in Figure 3.11, and the one that was used for the final prototype, seen in Figure 3.13. The blimp intended to be used was selected for its buoyancy which was deemed appropriate to lift the attached components. However there was some delay in its shipping, and to be able to perform tests while waiting, a smaller one was used which did not have enough buoyancy. To compensate for the lack of buoyancy of the smaller blimp a cord was placed on its upper surface, positioned so

that the blimp remained level while not being actuated. This initial test setup enabled some testing to be performed, but since the cord itself introduced disturbances it was clear that it would be hard to develop an optimal controller using this setup. It did, however, give an opportunity to test the validity of the Python implementation of the control sequence.

When the originally intended blimp was delivered there was an accident where the blimp broke down due to a combination of improper handling and flaws in its material. To be able to continue testing a third and final blimp was used, with the properties described in Figure D.2. As the parameters for this blimp differed from what had been used in both the propeller optimization and the simulations, the propeller configuration was adjusted and re-optimized, and then the simulation data was updated and run again. The results of these updates are seen in Figures 4.15, 4.16 and 4.17, which show that the adjusted propeller configuration gives similar force residuals but larger moment residuals than the original configuration. They also show that the configuration has an equally good trajectory following, and the comparison between the desired and generated forces is similar to before. Since the motor holders were not designed for the final blimp there was some offset that kept them from having a perfect fit. However, once placed on the blimp the fit was deemed good enough through visual inspection. As the motor holders were manually placed on the blimp each holder might have been placed at an offset from its intended position, making the LS optimization compute the actuation signal for a configuration that differs from the real one.

To compensate for the extra buoyancy of the final blimp a ballast was used. By introducing this new mass to the blimp its center of gravity was lowered, which created some changes that were not accounted for in the simulations. Thus, the simulation data were adjusted for the added mass before the feedback compensation parameters of the Python counterpart were updated.

The real-time experiments were performed by placing the blimp still on the floor and then running the control sequence with constant references. These tests were iterated and the PID parameters and feedback compensation contributions were tuned until the system was deemed stable enough. As there were $3 \times 6 = 18$ PID parameters to tune the process was time-consuming and with room for error. As the tests were run and the control signals plotted, it was noticed that the measurement signal was spiky at random times which can be seen in Figure 4.18. These spikes propagated to the numerical derivatives and were difficult to filter out. As a consequence, the feedback compensation that utilizes the body frame velocities became unreliable and had to be scaled down. Similarly, the desired force calculated in the PID and sent to the LS optimization was spiky, as seen in Figure 4.20. As the LS optimization was designed to penalize the time derivative of the actuation action, this spiky signal was difficult to follow with the generated forces as seen in Figures 4.20 and 4.21. As presented in Figures 4.18 and 4.19 these spikes created positional disturbances. For example at time ~ 95 where a large spike disturbance appeared and moved all position states away from the reference. However, the same figures show

that despite these disturbances and spikes the system was stable though oscillating and not fully reaching its references during the 200 seconds long test. From Figure 4.18 it can be deduced that it takes approximately 50 seconds before the blimp is able to lift from the floor, which is likely caused by the derivative penalty of the LS optimizer. This claim is supported by Figure 4.22, which has smoothly increasing thrust signals up until the 50-second mark, after which the signals begin to change more rapidly. For the system to be faster these signals should already from the start be close to what they are at the 50 second mark, but this might lead to a more unstable system.

6

Conclusion

Before conclusions are drawn, the work performed in this thesis will be summarized. At first, a mathematical blimp model adapted from the work in [Kukillaya and Pashilkar, 2017] was implemented in Matlab Simulink. The data for the model were based on the geometry data for a desired blimp with some simplifications. Secondly, an optimization algorithm was derived to find the optimal actuator solution, and for the specified blimp a propeller configuration was obtained. Thirdly, the modelled and actuated blimp was controlled by a combination of PID control and feedback compensation, followed by a least-squares optimization block adapted from the work of [Härkegård, 2004]. Fourthly, with the controlled modelled blimp a trajectory generation algorithm was implemented and tested until the generated trajectory was continuous and that the simulated blimp could follow it. Finally, with the control sequence outlined in the simulations, a physical blimp prototype was created, tested and tuned until it could be controlled in a crude but stable way.

6.1 Conclusion

The first conclusion that can be drawn from the thesis is that it is useful to perform simulations on the system before it is tested in real-time since it allows the control sequence to be tested and tuned for different parameters quickly. The same benefit applies when developing trajectory generation, where conditions must be tested frequently. However, for the simulation to be fully relevant for the real-time implementation, its conditions should match those of the physical system, both in terms of data for the model and for signal discretization.

A second conclusion is that the propeller optimization algorithm gave a good theoretical estimation of which propeller configurations would be successful.

A third conclusion to be drawn is that even a prototype that is not perfectly matching the specifications and models can be useful to develop controller structures and to find general limits and disturbances of the system. The dynamics might behave differently than expected but it gives something to work with and can be improved upon as the testing progresses. It should however be mentioned that having

a poor prototype makes the experimental work much harder since the data gained from the simulations are by default invalid.

A fourth conclusion that regards to the real-time experiments is that the filter used for the numerical derivatives was not enough. It was clear that unwanted spikes and noise got through and greatly influenced the rest of the control. Alternatively to using a better filter, a positioning system with a smoother signal could be used to a similar effect.

A fifth conclusion is that the penalization of the derivative term in the LS block was useful at slowing down the rate of change applied to the actuators, which was useful for the otherwise twitchy blimp, likely caused by its calculated derivative state. It is unclear if the penalty was too great and the twitchy behaviour was over-compensated for since the comparison between generated and desired forces show that they are not similar at all.

The sixth and final conclusion is that by combining all the work done it is clear that the full goal of autonomously navigating in 3D space was not reached. However, since the implemented controller was able to reach some sort of stability while the blimp was in the air even though it was slow, it is my opinion that the results of this thesis were close to its goal.

6.2 Future Work and Development

With the experiences gained throughout the thesis and a vision for how the system could be improved some suggestions for future work and development will be made.

To improve the quality of the simulations and the model four main suggestions are made. The first one is to use the actual shape of the used blimp when calculating the different shape parameters. The second one is to perform drag coefficient experiments to be able to include the aerodynamical forces and moments in the model, as well as including a disturbance model to handle wind and fluid motion. The third suggestion is to consider and measure the limits of the actuator, mainly as to which rate of change they can handle. The fourth one is to properly discretize the signal entering the control sequence so that the simulated PID tuning can be closer to the real one.

As for the propeller optimization algorithm, it can be suggested that before it is executed, the maximum force of the actuator is included, as well as considering the actual forces and moments that will be required from the configuration.

When dealing with the prototype blimp, more care should be placed in making it as close to the simulated system as possible. This includes being exact with the position and orientation of all placed components and making sure that the components do not move in between or during tests. Another part when dealing with the real-time experiments is to make sure that the signal measurements are as good as possible and that any unwanted behaviour is properly filtered. This can greatly help to reduce the amount of issues that appear throughout the process.

Bibliography

- Ang, M.H. and V.D. Tourassis (1987). “Singularities of euler and roll-pitch-yaw representations.” *IEEE Transactions on Aerospace and Electronic Systems, Aerospace and Electronic Systems, IEEE Transactions on, IEEE Trans. Aerosp. Electron. Syst* **AES-23**:3, pp. 317–324. ISSN: 0018-9251. URL: <http://ludwig.lub.lu.se/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=edsee&AN=edsee.4104344&site=eds-live&scope=site>.
- Ardema, Mark D. (2005). *Newton-Euler Dynamics*. Springer, Boston, MA, USA. ISBN: 9780387232751. URL: <http://ludwig.lub.lu.se/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=cat02271a&AN=atoz.ebs285140e&site=eds-live&scope=site>.
- Bitcraze (2020). *Datasheet Crazyflie 2.1*. Crazyflie 2.1. Rev. 1.
- Bitcraze (2021). *The lighthouse positioning system*. URL: <https://www.bitcraze.io/documentation/repository/crazyflie-firmware/master/functional-areas/lighthouse/> (visited on 2021-05-27).
- Eissing, Johannes (2019). *Gertler 4621 Shape spreadsheet*. 4621 Shape.
- Förster, Julian (2015-08). *System Identification of the Crazyflie 2.0 Nano Quadcopter*. en. MA thesis. ETH Zurich, Zurich. DOI: 10.3929/ethz-b-000214143.
- Glad, Torkel and Lennart Ljung (2000). *Control theory. [Elektronisk resurs] multivariable and nonlinear methods*. Taylor Francis, London ; New York. ISBN: 9780748408771. URL: <http://ludwig.lub.lu.se/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=cat07147a&AN=lub.6608505&site=eds-live&scope=site>.
- Greiff, Marcus (2017). *Modelling and Control of the Crazyflie Quadrotor for Aggressive and Autonomous Flight by Optical Flow Driven State Estimation*. Master’s Thesis TFRT-6026. Department of Automatic Control, Lund University, Sweden.

- Hägglund, Tore (2019). *Reglerteknik AK Föreläsningar*. Tech. rep. Lunds Tekniska Högskola, Institutionen för Reglerteknik, Lund. URL: <https://www.control.lth.se/fileadmin/control/Education/EngineeringProgram/FRTF05/forel.pdf>.
- Härkegård, Ola (2002). *Efficient Active Set Algorithms for Solving Constrained Least Squares Problems in Aircraft Control Allocation*. Tech. rep. Linköpings universitet, Institutionen för Systemteknik, Reglerteknik, Linköping. ISRN: LiTH-ISY-R-2426. URL: <http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-55681>.
- Härkegård, Ola (2004). *WLS_ALLOC Control allocation using weighted least squares*. Accessed: 2021-07-2. URL: http://research.harkegard.se/qcat/qcat/wls%5C_alloc.html.
- Hemingway, Evan G. and Oliver M. O'Reilly (2018). "Perspectives on euler angle singularities, gimbal lock, and the orthogonality of applied forces and applied moments." *Multibody System Dynamics* **44**:1, pp. 31–56. ISSN: 13845640. URL: <http://ludwig.lub.lu.se/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=edb&AN=131073331&site=eds-live&scope=site>.
- Hosier, David (2018). *Avoiding Gimbal Lock in a Trajectory Simulation*. Tech. rep. ARMET-TR-17051. ARDEC, METC, RDAR-MEF-A Picatinny Arsenal United States. URL: <https://apps.dtic.mil/sti/pdfs/AD1055301.pdf>.
- Howard, Fred (2013). *Wilbur and Orville: a biography of the Wright brothers*. Courier Corporation, Mineola, New York, USA.
- Hsu, Ting-Wei and Jing-Sin Liu (2020). "Design of smooth path based on the conversion between 3 spline and bezier curve". In: *2020 American Control Conference (ACC)* (Denver, CO, USA). 1-3 July 2020, pp. 3230–3235. DOI: 10.23919/ACC45564.2020.9147319.
- Hyndman, Rob J., Ralph Snyder, service) SpringerLink (Online, Anne Koehler, and Keith Ord (2008). *Forecasting with exponential smoothing. [electronic resource] the state space approach*. Springer series in statistics. Springer-Verlag Berlin Heidelberg. ISBN: 9783540719168. URL: <http://ludwig.lub.lu.se/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=cat07147a&AN=lub.5561825&site=eds-live&scope=site>.
- Kukillaya, Raghavendra and Abhay Pashilkar (2017). *Simulink model development, validation and analysis of high altitude airship*. Tech. rep. PD-FMC/2017/1000. National Aerospace Laboratories, Flight Mechanics and Control Division, Bangalore, India. DOI: 10.13140/RG.2.2.11844.22400.
- Li, Yuwen, Meyer Nahon, and Inna Sharf (2011). "Airship dynamics modeling: a literature review". *Progress in aerospace sciences* **47**:3, pp. 217–239.
- Lu, Benjamin (2010). "The boeing 787 dreamliner designing an aircraft for the future". *Journal of Young Investigators*, ISSN: 1539 **4026**, p. 34.

- Månsson, Jonas and Patrik Nordbeck (2019). *Endimensionell analys*. Vol. 1. 9. Studentlitteratur AB, Lund. ISBN: 978-91-44-05610-4.
- MathWorks (2021). *Atan2*. URL: <https://se.mathworks.com/help/matlab/ref/atan2.html> (visited on 2021-06-09).
- Meng, Xiangxiang, Haisheng Yu, Jie Zhang, Tao Xu, Herong Wu, and Kejia Yan (2021). “Disturbance observer-based feedback linearization control for a quadruple-tank liquid level system”. *ISA Transactions*. Online ahead of print. ISSN: 0019-0578. DOI: <https://doi.org/10.1016/j.isatra.2021.04.021>. URL: <https://www.sciencedirect.com/science/article/pii/S0019057821002184>.
- Nocedal, Jorge. (2006). *Numerical optimization / Jorge Nocedal, Stephen J. Wright*. Springer. ISBN: 0-387-30303-0. URL: <http://ludwig.lub.lu.se/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=conedsqd5&AN=edsnuk.vt1s000950872&site=eds-live&scope=site>.
- Paredes, J. A., C. Saito, M. Abarca, and F. Cuellar (2017). “Study of effects of high-altitude environments on multicopter and fixed-wing uavs’ energy consumption and flight time”. In: *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*. 20-23 Aug. 2017. Xi’an, China, pp. 1645–1650. DOI: 10.1109/COASE.2017.8256340.
- Severholt, Josefine (2017). *Generic 6-DOF Added Mass Formulation for Arbitrary Underwater Vehicles based on Existing Semi-Empirical Methods*. Tech. rep. ISSN: 1651-7660 ; 2017:42. KTH, Skolan för teknikvetenskap (SCI), Stockholm. URL: <http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-211170>.
- Shakhatreh, Hazim, Ahmad H. Sawalmeh, Ala Al-Fuqaha, Zuochao Dou, Eyad Al-maita, Issa Khalil, Noor Shamsiah Othman, Abdallah Khreishah, and Mohsen Guizani (2019). “Unmanned aerial vehicles (uavs): a survey on civil applications and key research challenges”. *IEEE Access* 7, pp. 48572–48634. DOI: 10.1109/ACCESS.2019.2909530.
- Taylor, Fred J. (2012). *Digital filters. [Elektronisk resurs] principles and applications with MATLAB*. IEEE series on digital and mobile communication: 12. Wiley-IEEE Press, Hoboken, New Jersey, USA. ISBN: 9780470770399. URL: <http://ludwig.lub.lu.se/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=cat07147a&AN=lub.5863556&site=eds-live&scope=site>.
- Windreiter (2020). *Airship Envelope Datasheet*. Identifier: SB-181-200. Düsseldorf, Germany.

A

Inertia Tensor Appendix

In this appendix the equations used to calculate the symmetric positive definite inertia tensor of an object are presented [Ardema, 2005].

$$I_{xx} = \sum_m m_i (y_i^2 + z_i^2) \quad (\text{A.1a})$$

$$I_{yy} = \sum_m m_i (x_i^2 + z_i^2) \quad (\text{A.1b})$$

$$I_{zz} = \sum_m m_i (x_i^2 + y_i^2) \quad (\text{A.1c})$$

$$I_{xy} = \sum_m m_i x_i y_i = I_{yx} \quad (\text{A.1d})$$

$$I_{yz} = \sum_m m_i y_i z_i = I_{zy} \quad (\text{A.1e})$$

$$I_{xz} = \sum_m m_i x_i z_i = I_{zx} \quad (\text{A.1f})$$

$$\bar{I} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} \quad (\text{A.2})$$

$$I_{xx} = \int_m (y^2 + z^2) dm \quad (\text{A.3a})$$

$$I_{yy} = \int_m (x^2 + z^2) dm \quad (\text{A.3b})$$

$$I_{zz} = \int_m (x^2 + y^2) dm \quad (\text{A.3c})$$

$$I_{xy} = \int_m xy dm = I_{yx} \quad (\text{A.3d})$$

$$I_{yz} = \int_m yz dm = I_{zy} \quad (\text{A.3e})$$

$$I_{xz} = \int_m xz dm = I_{zx} \quad (\text{A.3f})$$

B

Weighted Least Squares Appendix

In this appendix the weighted Least Squares algorithm derived in [Härkegård, 2004] is shown. In the thesis this algorithm was used and modified to include a numerical time derivative term in the cost formulation.

```
function [u,W,iter] = wls_alloc(B,v,umin,umax,Wv,Wu,ud,gam,u,W,
    imax)

% WLS_ALLOC - Control allocation using weighted least squares.
%
% [u,W,iter] = wls_alloc(B,v,umin,umax,[Wv,Wu,ud,gamma,u0,W0,
%   imax])
%
% Solves the weighted, bounded least-squares problem
%
%   min ||Wu(u-ud)||^2 + gamma ||Wv(Bu-v)||^2
%
%   subj. to   umin <= u <= umax
%
% using an active set method.
%
% Inputs:
% -----
% B      control effectiveness matrix (k x m)
% v      commanded virtual control (k x 1)
% umin   lower position limits (m x 1)
% umax   upper position limits (m x 1)
% Wv     virtual control weighting matrix (k x k) [I]
% Wu     control weighting matrix (m x m) [I]
% ud     desired control (m x 1) [0]
% gamma  weight (scalar) [1e6]
% u0     initial point (m x 1)
% W0     initial working set (m x 1) [empty]
% imax   max no. of iterations [100]
%
```

Appendix B. Weighted Least Squares Appendix

```
% Outputs:
% -----
% u      optimal control
% W      optimal active set
% iter  no. of iterations (= no. of changes in the working set +
%       1)
%
%                               0 if u_i not saturated
% Working set syntax: W_i = -1 if u_i = u_min_i
%                               +1 if u_i = u_max_i
%
% See also: WLSC_ALLOC, IP_ALLOC, FXP_ALLOC, QP_SIM.

% Number of variables
m = length(u_min);

% Set default values of optional arguments
if nargin < 11
imax = 100; % Heuristic value
[k,m] = size(B);
if nargin < 10, u = (u_min+u_max)/2; W = zeros(m,1); end
if nargin < 8,  gam = 1e6;          end
if nargin < 7,  ud = zeros(m,1); end
if nargin < 6,  Wu = eye(m);      end
if nargin < 5,  Wv = eye(k);      end
end

gam_sq = sqrt(gam);
A = [gam_sq*Wv*B ; Wu];
b = [gam_sq*Wv*v ; Wu*ud];

% Initial residual.
d = b - A*u;
% Determine indices of free variables.
i_free = W==0;

% Iterate until optimum is found or maximum number of iterations
% is reached.
for iter = 1:imax
% -----
% Compute optimal perturbation vector p.
% -----

% Eliminate saturated variables.
A_free = A(:,i_free);
% Solve the reduced optimization problem for free variables.
p_free = A_free\d;
% Zero all perturbations corresponding to active constraints.
p = zeros(m,1);
% Insert perturbations from p_free into free the variables.
p(i_free) = p_free;

% -----
% Is the new point feasible?
```



```

% -----
u_opt = u + p;
infeasible = (u_opt < umin) | (u_opt > umax);

if ~any(infeasible(i_free))

% -----
% Yes, check for optimality.
% -----

% Update point and residual.
u = u_opt;
d = d - A_free*p_free;
% Compute Lagrangian multipliers.
lambda = W.*(A'*d);
% Are all lambda non-negative?
if lambda >= -eps
% / ----- |
% | Optimum found, bail out. |
% \ ----- /
return;
end

% -----
% Optimum not found, remove one active constraint.
% -----

% Remove constraint with most negative lambda from the
% working set.
[lambda_neg, i_neg] = min(lambda);
W(i_neg) = 0;
i_free(i_neg) = 1;

else

% -----
% No, find primary bounding constraint.
% -----

% Compute distances to the different boundaries. Since
% alpha < 1
% is the maximum step length, initiate with ones.
dist = ones(m,1);
i_min = i_free & p<0;
i_max = i_free & p>0;

dist(i_min) = (umin(i_min) - u(i_min)) ./ p(i_min);
dist(i_max) = (umax(i_max) - u(i_max)) ./ p(i_max);

% Proportion of p to travel
[alpha, i_alpha] = min(dist);
% Update point and residual.
u = u + alpha*p;

```

Appendix B. Weighted Least Squares Appendix

```
d = d - A_free*alpha*p_free;  
  
% Add corresponding constraint to working set.  
W(i_alpha) = sign(p(i_alpha));  
i_free(i_alpha) = 0;  
  
end  
  
end
```

C

PID Appendix

Here a pseudocode example of the PID Python implementation is presented to give an overview of this part of the control sequence. Following the pseudocode are the state responses obtained when simulating for a ramp state trajectory.

Algorithm 1 Pseudocode of the Python PID implementation

```
1: ref, refD = getRef()           ▷ 6x1 vectors of references
2: statePrev, ePrev, intPrev = zeroVector()   ▷ 6x1 vectors of zeros
3: h = getSampTime()             ▷ Sample time expressed in seconds
4: function PID (ref, refD, statePrev, ePrev, intPrev, h )
5:     Kp, Ki, Kd = getKp(), getKi(), getKd()
6:     state, t0 = getState(), getTime()
7:     velState = (state-statePrev)/h
8:     velFilt = filter(velState)
9:     e = ref - state
10:    eD = refD - velFilt
11:    eI = intPrev+0.5*h*(e+ePrev)
12:    vPID = Kp*e+Ki*eI+Kd*eD
13:    statePrev, errorPrev, intPrev = state, e, eI
14:    return vPID, statePrev, errorPrev, intPrev
15: end function
```

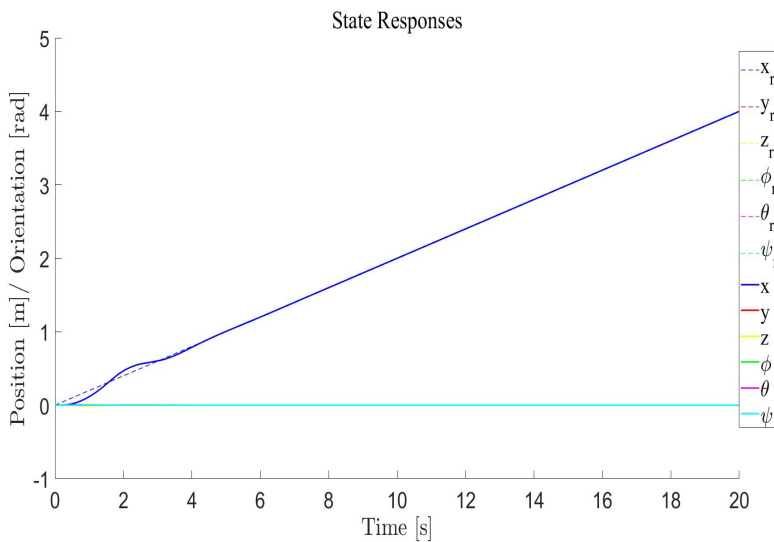


Figure C.1: Simulated state responses to a ramp reference for state x .

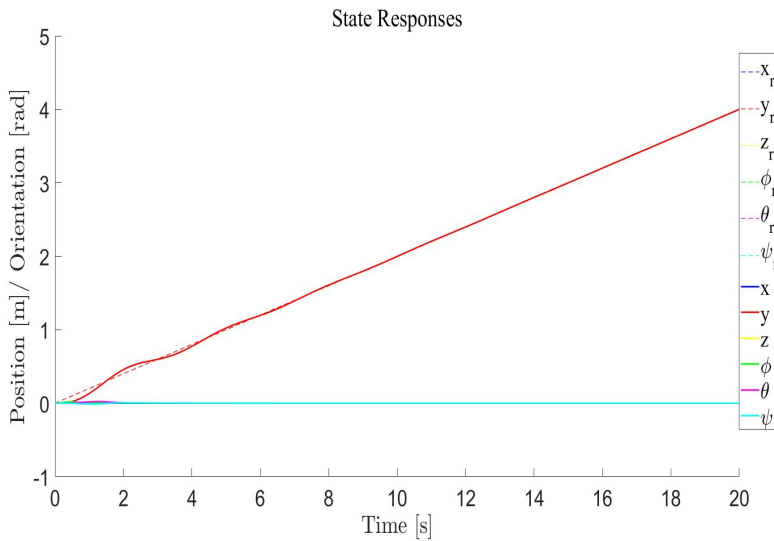


Figure C.2: Simulated state responses to a ramp reference for state y .

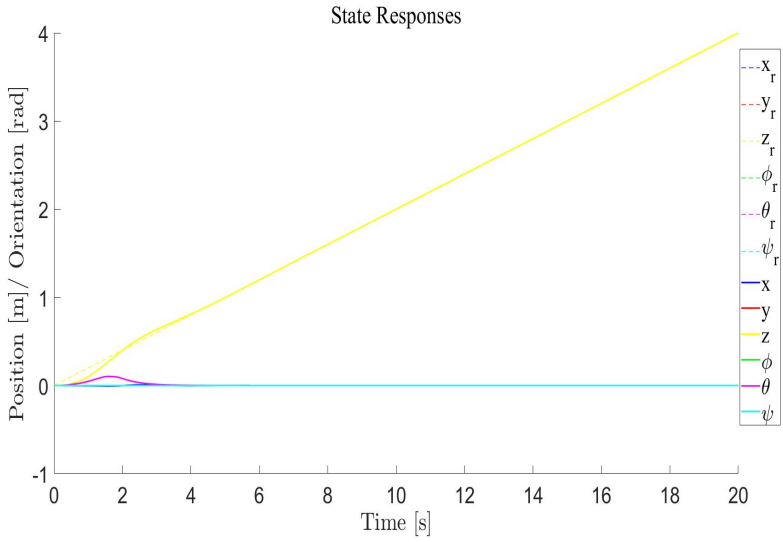


Figure C.3: Simulated state responses to a ramp reference for state z .

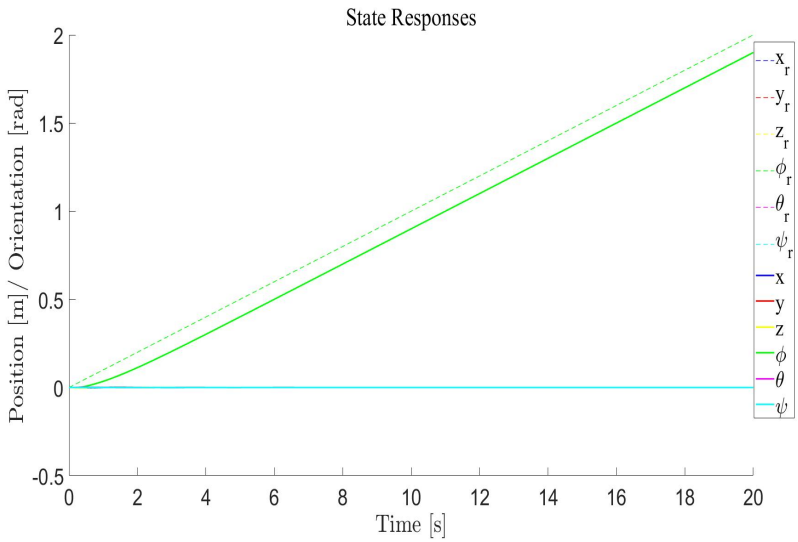


Figure C.4: Simulated state responses to a ramp reference for state ϕ .

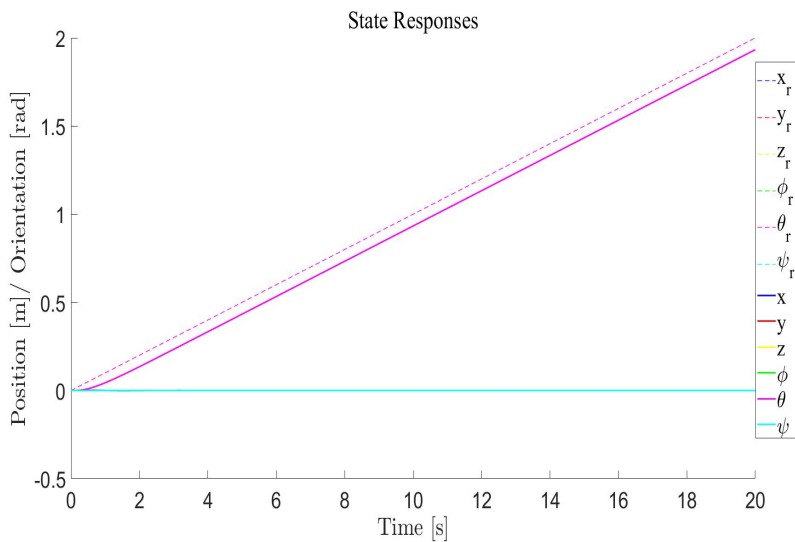


Figure C.5: Simulated state responses to a ramp reference for state θ .

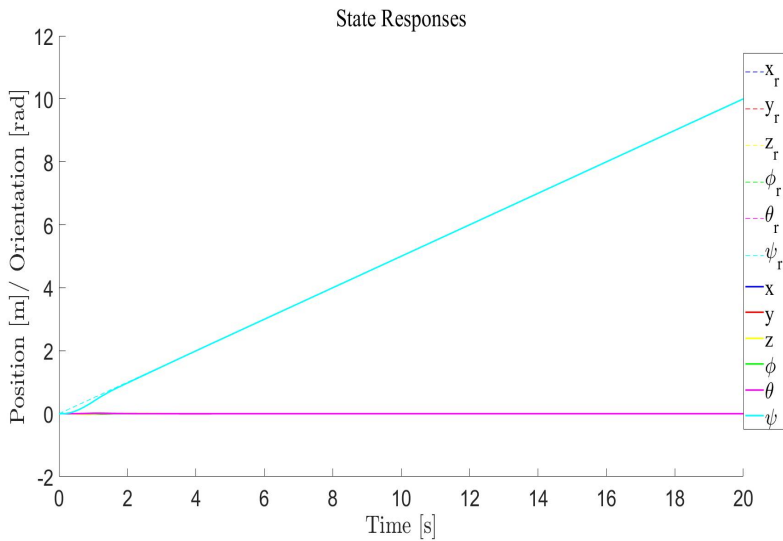
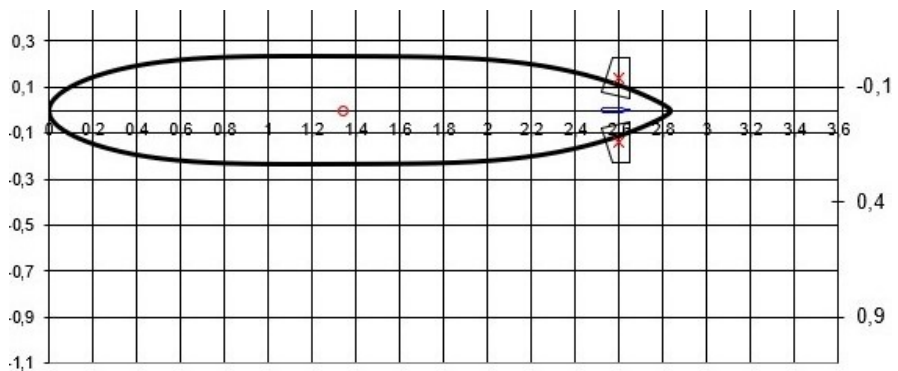


Figure C.6: Simulated state responses to a ramp reference for state ψ .

D

Blimp data Appendix

In this appendix the data and shape of the originally intended blimp is presented followed by the data for final blimp.



Cb	0,5655 [-]	Block Coefficient
V	0,3563 [m ³]	Volume
L	2,831 [m]	Length
D	0,472 [m]	Diameter
V(integrated)	0,339851 [m ³]	Volume
S	3,532 [m ²]	Surface Area
Cs	0,84 [-]	Surface Coefficient
rhoS	0,057 [kg/m ²]	
mS	0,201 [kg]	Envelope Weight
lift	0,155 [kg]	
lambda	0,619047023	

Figure D.1: Originally intended blimp geometry data. Source: [Eissing, 2019].

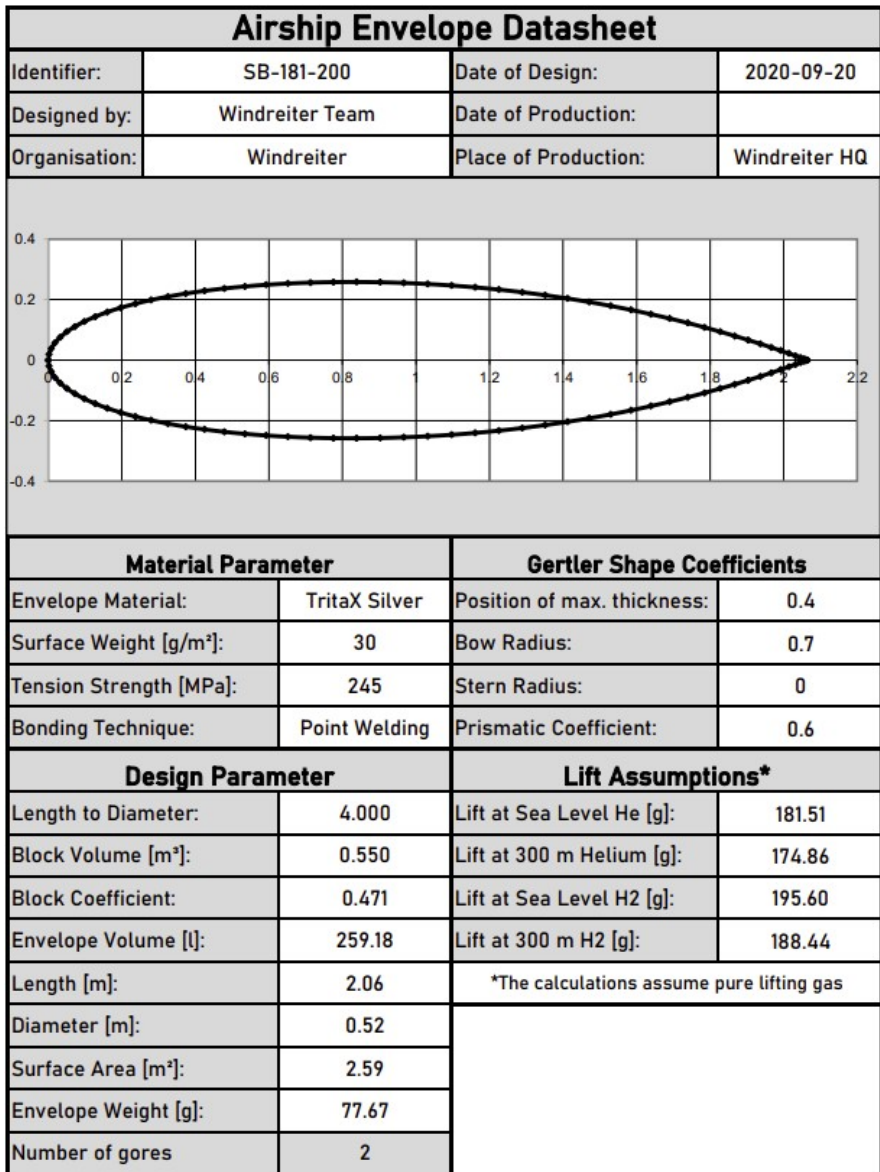


Figure D.2: Final Blimp geometry data. Source: [Windreiter, 2020].

E

Equations

This section serves to collect general mathematical and physical equations that are used variously throughout the thesis. These equations are regarded as general knowledge and will therefore not be described in-depth.

Definition of the derivative: [Månsson and Nordbeck, 2019, p. 205, 206]

$$\frac{\Delta y}{\Delta x} = \lim_{h \rightarrow 0} \frac{f(a+h) - f(a)}{h} \quad (\text{E.1})$$

Trigonometric functions: [Månsson and Nordbeck, 2019, p. 138, 157, 158]

$$y = \cos(\alpha) = \frac{a}{c} \quad (\text{E.2a})$$

$$y = \sin(\alpha) = \frac{b}{c} \quad (\text{E.2b})$$

$$y = \tan(\alpha) = \frac{b}{a} \quad (\text{E.2c})$$

$$\alpha = \cos^{-1}(y) = \text{acos}(y), \quad -1 \leq y \leq 1, \quad 0 \leq \alpha \leq \pi \quad (\text{E.2d})$$

$$\alpha = \sin^{-1}(y) = \text{asin}(y), \quad -1 \leq y \leq 1, \quad -\frac{\pi}{2} \leq \alpha \leq \frac{\pi}{2} \quad (\text{E.2e})$$

$$\alpha = \tan^{-1}(y) = \text{atan}(y), \quad y \in \mathbb{R}, \quad -\frac{\pi}{2} \leq \alpha \leq \frac{\pi}{2} \quad (\text{E.2f})$$

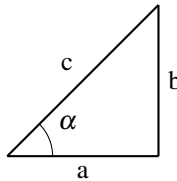


Figure E.1: Right angled triangle used to describe the trigonometric functions [Månsson and Nordbeck, 2019, p. 138].

Lund University Department of Automatic Control Box 118 SE-221 00 Lund Sweden		<i>Document name</i> MASTER'S THESIS
		<i>Date of issue</i> June 2021
		<i>Document Number</i> TFRT-6134
<i>Author(s)</i> Gustaf Åman		<i>Supervisor</i> Rikard Tyllström, School of Aviation, Lund University, Sweden Marcus Greiff, Dept. of Automatic Control, Lund University, Sweden Anders Robertsson, Dept. of Automatic Control, Lund University, Sweden Rolf Johansson, Dept. of Automatic Control, Lund University, Sweden (examiner)
<i>Title and subtitle</i> Indoor Blimp Control		
<i>Abstract</i> <p>The purpose of this thesis is to model and develop a control law that lets an unmanned aerial vehicle of blimp type fly autonomously in three-dimensional space. There are several uses of the blimp type UAV including, but not limited to, cargo transportation and surveillance. The work done throughout the thesis includes optimization of actuator placements, modelling the blimp and simulating the process, developing a PID controller for positioning and orientation, trajectory generation, actuator effort optimization, prototype construction and real-time experiments. The results of the thesis show that with a simulation model the control sequence implementation is facilitated, but not necessarily directly convertible to the real-time prototype. They also show that faulty signal processing can lead to disturbances which propagate throughout the process. Finally, it can be stated that the system was stable enough to remain close to a point in space, however, it was not fast and stable enough to follow a trajectory and thus the goals of the thesis were not fully met.</p>		
<i>Keywords</i>		
<i>Classification system and/or index terms (if any)</i>		
<i>Supplementary bibliographical information</i>		
<i>ISSN and key title</i> 0280-5316		<i>ISBN</i>
<i>Language</i> English	<i>Number of pages</i> 1-89	<i>Recipient's notes</i>
<i>Security classification</i>		