NTNU

Norwegian University of
Science and Technology

Hover Control of Thrust Vectoring VTOL Flying Wing

Modeling, Control and Development of Test Platform with Experimental Results

Erlend Magnus Lervik Coates

Master of Science in Cybernetics and Robotics
Submission date: August 2017
Supervisor: Jan Tommy Gravdahl, ITK
Co-supervisor: Jostein Furseth, Sevendof AS

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Acknowledgements

I would like to offer a special thanks to my supervisor Professor Jan Tommy Gravdahl for making this project possible and for good advise throughout the project period. A big thank you goes to the Sevendof team, for introducing me to the wondrous world of unmanned aircraft and providing a fantastic UAV prototype test platform. I would especially like to thank Johannes Hatle Lundgaard for video editing, help with SolidWorks for chapter 3, for making a fantastic test rig for chapter 9 and for great company throughout countless hours of testing. In addition, I would like to thank The Department of Engineering Cybernetics for providing such great test facilities, enabling rapid test cycles in a safe and quiet environment.

I am grateful for valuable discussions with, among others, Jostein Furseth, Glenn Bitar, Kjetil Kjeka, Emil Scott Bale, Per Magnus Veierland and Haavard Holta. I would also like to thank Nasser Ukla for help with practical issues related to radio equipment, batteries and piloting.

Finally, I would like to thank all my friends and family. Thanks to my classmates for two of the best years of my life. Thanks to my parents, Else-Marie and Robert for love and support, and last but not least, a big thank you to my girlfriend Amalie for being so patient and supporting me during long days and busy periods of work.

Abstract

This work consists of theoretical and practical advances in the development of a fully thrust-vectoring hybrid vertical take-off and landing (VTOL) unmanned aerial vehicle (UAV). A high-fidelity dynamic simulation model is successfully implemented in Matlab and Simulink based on a 10 DOF Euler-Lagrange approach, including actuator dynamics and external aerodynamic forces. A reformulation of a previously proposed control allocation scheme is presented based on quadratic programming (QP). The proposed model and control allocation method is verified through simulation. Simulation results show that the vehicle is able to track attitude and velocity commands. The inertial properties and propeller parameters used in simulation are based on CAD-models and propeller tests of an actual prototype UAV developed by Sevendof AS. Further, a test platform for autopilot implementation, including software-in-the-loop simulation (SITL), is developed based on the open-source software frameworks PX4 and Gazebo. The implemented autopilot successfully stabilizes the vehicle in SITL simulation, and is further verified through flight tests in an indoor test rig. The flight experiments show that the proposed autopilot solution succesfully stabilizes the attitude and velocity of the prototype vehicle.

Sammendrag

Dette arbeidet tar for seg teoretiske og praktiske fremskritt på veien til å utvikle et vertikal take-off og landing (VTOL) ubemannet luftfartøy (UAV) som er kapabelt til å rette kraftvektoren produsert av propellene i alle retninger (thrust vectoring). En dynamisk simuleringsmodell er implementert i Matlab og Simulink basert på Euler-Lagrange-metoden med 10 frihetsgrader. Modellen inkluderer aktuatordynamikk og vingeaerodynamikk. En reformulering av en tidligere utviklet metode for kontrollallokering er foreslått basert på kvadratisk programmering (QP). Simuleringsmodellen med kontrollallokering er verifisert gjennom simuleringer, og resultatene viser at fartøyet følger orienterings- og hastighetskommandoer. Masser, treghetsmomenter og propellparametre brukt i simuleringene er basert på 3D-modeller og propelltester av en prototype utviklet av Sevendof AS. En testplatform for implementasjon av autopiloter, inkludert software-in-the-loop (SITL) simulering, er uviklet basert på åpen kildekode-prosjektene PX4 og Gazebo. Den implementerte autopiloten stabiliserer fartøyets orienting og hastighet i SITL simuleringer, og er videre verifisert gjennom testflyginger i en innendørs testrigg. Eksperimentene viser at den foreslåtte autopilotløsningen stabiliserer orienteringen og hastigheten til prototypen på en tilfredstillende måte.

Nomenclature

α	Angle of attack [rad]
$\bar{c}$	Mean aerodynamic chord [m]
β	Sideslip angle [rad]
$\boldsymbol{\omega}_{b/n}^{b}$	Angular velocity of body frame with respect to NED frame expressed in body [rad/s]
γ_d	Commanded twist angle [rad]
γ_l	Left twist angle [rad]
γ_r	Right twist angle [rad]
λ_d	Commanded tilt angle [rad]
λ_l	Left tilt angle [rad]
λ_r	Right tilt angle [rad]
$\mathbf{I}_b$	Main body inertia tensor about main body CG, aligned with $\{b\}$ [$\mathrm{kg} \cdot \mathrm{m}^2$]
$\mathbf{I}_t$	Total body inertia tensor about idealized body CG, aligned with $\{b\}$ [$\mathrm{kg} \cdot \mathrm{m}^2$]

$\mathbf{I}_{t_l}$ Left tilt inertia tensor about left tilt CG, aligned with $\{t_l\}$ $[\,\mathrm{kg}\cdot\mathrm{m}^2]$

$\mathbf{I}_{t_r}$ Right tilt inertia tensor about right tilt CG, aligned with $\{t_r\}$ $[\,\mathrm{kg}\cdot\mathrm{m}^2]$

$\mathbf{I}_{tw_l}$ Left twist inertia tensor about left twist CG, aligned with $\{tw_l\}$ $[\,\mathrm{kg}\cdot\mathrm{m}^2]$

$\mathbf{I}_{tw_r}$ Right twist inertia tensor about right twist CG, aligned with $\{tw_r\}$ $[\,\mathrm{kg}\cdot\mathrm{m}^2]$

$\mathbf{K}_{pQ}$ Attitude proportional gains

$\mathbf{q}$ Generalized coordinates

$\mathbf{r}_b$ NED position of main body center of mass $[\,\mathrm{m}]$

$\mathbf{R}_b^n$ Rotation matrix from NED frame to body frame

$\mathbf{R}_{t_l}^b$ Rotation matrix from body frame to left tilt frame

$\mathbf{R}_{t_r}^b$ Rotation matrix from body frame to right tilt frame

$\mathbf{R}_{tw_l}^b$ Rotation matrix from body to left twist frame

$\mathbf{R}_{tw_l}^{t_l}$ Rotation matrix from left tilt frame to left twist frame

$\mathbf{R}_{tw_r}^b$ Rotation matrix from body to right twist frame

$\mathbf{R}_{tw_r}^{t_r}$ Rotation matrix from right tilt frame to right twist frame

Ω_i Rotational speed of propeller i $[\,\mathrm{rad/s}]$

Ω_{i_d} Commanded rotational speed of propeller i $[\,\mathrm{rad/s}]$

ϕ Roll angle $[\,\mathrm{rad}]$

ψ Yaw angle $[\,\mathrm{rad}]$

ρ_a Density of air $[\,\mathrm{kg/m}^3]$

θ Pitch angle $[\,\mathrm{rad}]$

Θ_{nb} Vector of Euler angles $[\,\mathrm{rad}]$

$\{b\}$ Body reference frame

$\{n\}$ NED reference frame (inertial)

$\{t_l\}$ Left tilt frame

$\{t_r\}$ Right tilt frame

$\{tw_l\}$ Left twist frame

$\{tw_r\}$ Right twist frame

b Wingspan [m]

c Propeller torque constant

C_D Aerodynamic drag coefficient

d Propeller torque coefficient

F_t Total force [N]

F_x Force along body x-axis (surge force) [N]

F_y Force along body y-axis (sway force) [N]

F_z Force along body z-axis (heave force) [N]

F_{xz} Force in xz-plane [N]

k Propeller thrust coefficient

l_c Distance between wing leading edge and tilt rotation axis [m]

l_t Distance between body x-axis and propellers rotation axis (roll moment arm) [m]

l_{tw} Half-length of tilt arm (idealized pitch moment arm) [m]

m_b Mass of main body link [kg]

m_t Total mass [kg]

M_x Moment about body x-axis (roll moment) [Nm]

M_y Moment about body y-axis (pitch moment) [Nm]

M_z Moment about body z-axis (yaw moment) [Nm]

m_{t_l} Mass of left tilt link [kg]

m_{t_r} Mass of right tilt link [kg]

m_{tw_l} Mass of left twist link [kg]

m_{tw_r} Mass of right twist link [kg]

S Planform area of the wing [m^2]

V_a Airspeed [m/s]

Contents

Introduction **3**

1 Introduction **3**

1.1 Motivation and Background 3

1.1.1 Hybrid VTOL UAVs 5

1.1.2 The StormPetrel Airframe 5

1.1.3 Previous Work 7

1.2 Contributions 9

1.3 Overview 10

I Modeling, Identification and Control **11**

2 Modeling **13**

2.1 Kinematics 15

2.1.1 Coordinate Frames 15
2.1.2 Rotation Matrices 16
2.1.3 Coordinate Vectors 16
2.1.4 Angular Velocity 17
2.1.5 Linear Velocity 20
2.2 Dynamics 22
2.2.1 Kinetic Energy 22
2.2.2 Potential Energy 23
2.2.3 Euler-Lagrange Equations of Motion 23
2.3 Actuator Dynamics 24
2.3.1 Servo Motors 24
2.3.2 Thruster Model 25
2.4 External Forces 27
2.4.1 Propulsion Forces and Moments 29
2.4.2 Aerodynamics 34
2.5 Total Vehicle Model 38
2.6 Comments on the Equations of Motion 38

3 Parameter Identification **41**

3.1 Mass Properties 41
3.2 Thruster Parameters 43

3.2.1 Comparison With Supplier Test Data 49

4 Control System Design **51**

4.1 Attitude Control 51

4.2 Velocity Control 52

4.3 Control Allocation 53

4.3.1 Main Algorithm 53

4.3.2 Constraints 56

4.3.3 Optimization Based Formulation 58

5 Simulation Results **61**

5.1 Simulator Implementation 61

5.2 Velocity Response 62

5.3 Attitude Response 70

II Development of Test Platform **79**

6 Implementation in the PX4 Flight Stack **81**

7 Software-in-the-Loop Simulation **85**

7.1 Gazebo 85

8 Flight Tests **91**

8.1 Test Setup 91
8.2 Experimental Data 93
8.2.1 Typical Test Flight 94
8.2.2 Roll Response 105
8.2.3 Yaw Rate Response 108

Concluding Remarks **113**

9 Conclusion **113**

9.1 Summary and Conclusion 113
9.2 Future Work 116

Bibliography **121**

List of Figures **126**

List of Tables **128**

List of Code Listings **A1**

Appendix A Parameter List **A1**

Appendix B SITL Plots **A9**

Introduction

1 Introduction

This introductary chapter gives the motivation and background for this thesis, and provides an overview of the report.

1.1 Motivation and Background

During the last decades, developments in unmanned aerial vehicles (UAVs) have been tremendous, and areas of application are steadily increasing. Sevendof AS, a Trondheim based startup company, is developing a novel multipurpose UAV capable of vertical take-off and landing (VTOL) in tight spaces, operating in harsh weather conditions, and long distance flight. The vehicle concept is called StormPetrel. StormPetrel's first application will be for powerline inspections, and a cooperation is in place with TrønderEnergi Nett and norwegian IT company Powel AS. Future possible applications include search and rescue (SAR) and offshore wind turbine inspection. Figure 1.1 shows some concept illustrations.

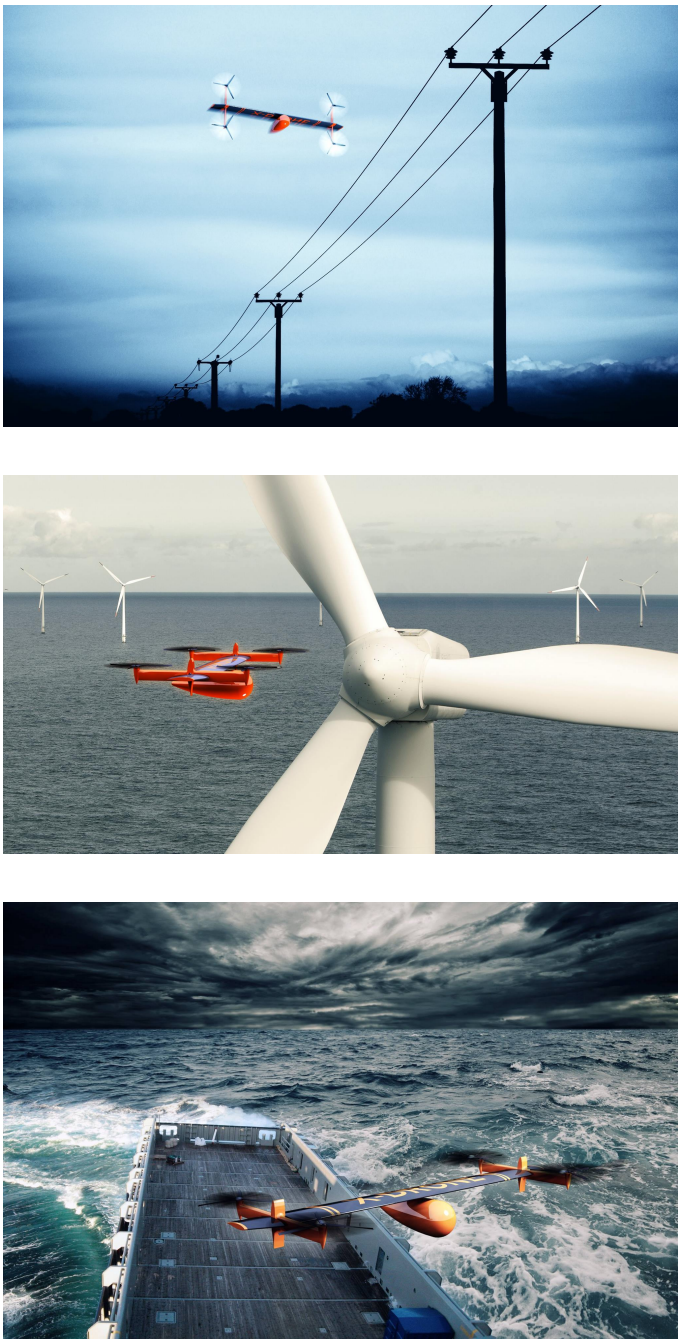


Figure 1.1: Concept illustrations, courtesy of Sevendof AS

1.1.1 Hybrid VTOL UAVs

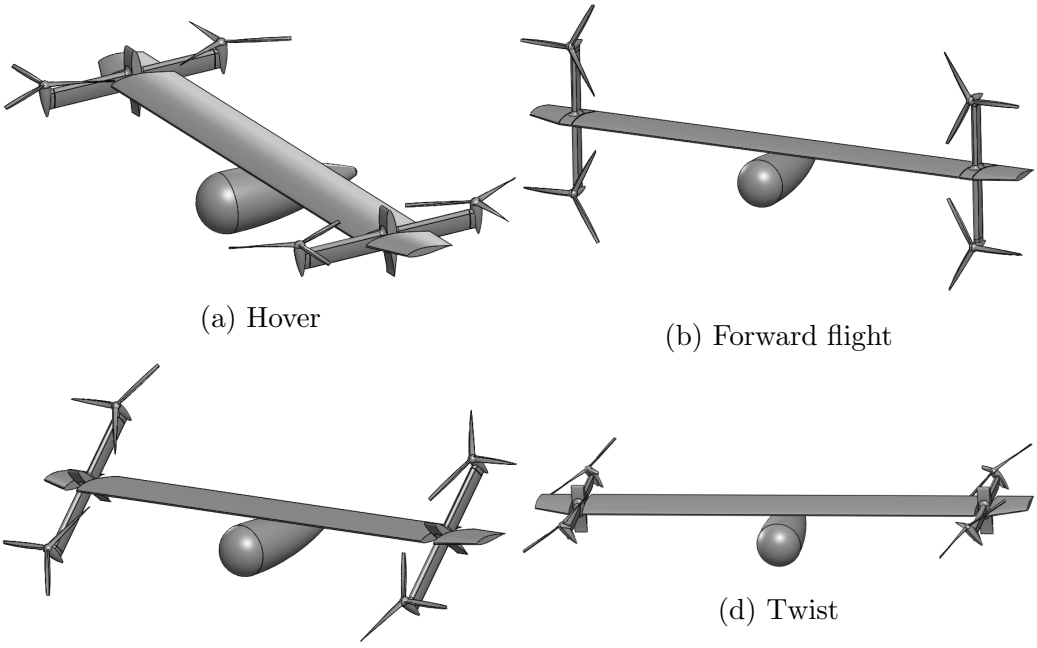
Several types of UAVs exists today. These can be broadly characterized into multirotors and fixed wing. For multirotors, propellers create the thrust force needed to hold the vehicle aloft and maneuver through the air. Fixed wings utilize a wing for lift and aerodynamic control surfaces to stabilize and maneuver the vehicle, while usually a main propeller provides thrust needed for forward motion. Unmanned vehicles that combine the characteristics of multirotors and fixed wing aircraft are usually called VTOL or *hybrid VTOL* UAVs. In this text, the latter will be preferred, as the term VTOL is sometimes also used for regular multirotors, such as quadcopters, which can also take-off and land vertically. Hybrid VTOLs utilize some sort of *transition*, or *conversion* mechanism to convert between the two modes, hover and forward flight respectively. Several types of hybrid VTOLs exist today, and these are characterized according to how the transition is carried out. These include tilt-rotors, tilt-wings and tail-sitters. A comprehensive review of hybrid VTOL platform designs, dynamic modeling and control is given in [27].

1.1.2 The StormPetrel Airframe

The StormPetrel can be characterized as a quad tilt-rotor convertiplane, or more generally, a hybrid VTOL flying wing. A tilting mechanism rotates the thrusters relative to the aircraft body when transitioning from hover to forward flight. The lift responsibility of the thrusters is gradually taken over by the wing as the vehicle gains speed. This is illustrated in figures 1.2a - 1.2c.

Two core properties that distinguish the StormPetrel concept from other designs are:

- There are no aerodynamic control surfaces such as ailerons, elevators and rudders. The vehicle attitude is stabilized by thrust differencing



(a) Hover

(b) Forward flight

(c) Transition/Tilt

(d) Twist

Figure 1.2: The StormPetrel VTOL Concept

between propellers only.

- The vehicle is capable of full *thrust vectoring*, i.e. the resulting combined thrust vector from the four propellers can be directed in any relative direction (inside operating constraints of the tilting mechanisms). This is achieved by tilting the propellers, not only in the longitudinal direction, but also sideways as shown in figure 1.2d. The former will be referred to as *tilt*, and the latter as *twist*.

- An onboard hybrid power (HPP) plant to provide long range and endurance.

The tilt and twist mechanisms are implemented using powerful servo motors inside the nacelles (seperate housings) at each side of wing, as can be seen in figure 1.3. The twist mechanism can be used to suppress side winds during forward-flight [21], and the full thrust-vectoring capabilites can be utilized during hover for translational control without tilting the vehicle, thus poten-

tially reducing the vehicle surface area being affected by horizontal winds. In [21], it is also shown how the thrust vectoring system enables the vehicle to attain an arbitrary attitude while still being able to realize force commands in any direction.

One potential downside to the design is the increased mechanical complexity and cost due to the tilting mechanisms. Another is the lack of control surfaces during transition and forward flight. As the airspeed increases, fixed pitch propellers loose efficiency, and as the vehicle relies on the propellers to produce moments for stabilization, this might cause a problem. This may be solved by the use of variable pitch propellers, but that would further increase the mechanical complexity. The design might change in the future, but further analysis and testing is needed.

As a development platform, to test autpilot designs, thrust vectoring mechanics and transition performance, Sevendof AS has developed a prototype vehicle, the MiniPetrel. Some images of the MiniPetrel prototype is shown in figure 1.3. It has the full functionality of the current StormPetrel design, but is significantly smaller than the intended final version of the UAV, is battery powered and not designed to carry payloads. It has a wingspan of 1.1 meters and a mass of roughly 5 kilograms.

This thesis continues the work on dynamic modeling, simulator implementation and control system development. In addition, the first ever thrust vectoring hover flight tests are performed using the MiniPetrel prototype aircraft.

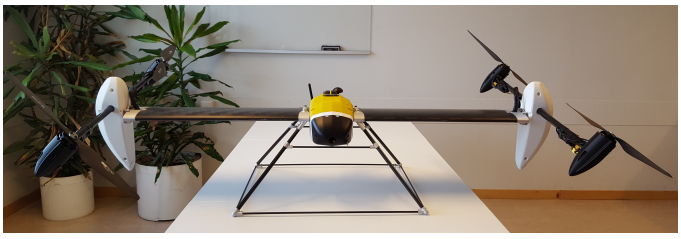
1.1.3 Previous Work

A lot of previous work have been carried out on the development of the StormPetrel platform, including several master's theses. Of particular interest to this thesis are:

- In [21], limitations of existing designs are discussed and a mathematical

(a) Hover configuration

(b) Tilt



(c) Twist

Figure 1.3: The MiniPetrel prototype

model and flight control system, including a nonlinear 6 DOF model for the wing aerodynamics, is implemented in Matlab/Simulink. The dynamic performance is investigated and verified through simulations.

- In [24], the mechanical thrust vectoring system is developed.

- In [15], suitable actuators and sensors are chosen for the MiniPetrel prototype, and a preliminary investigation into choice of autopilot platform is carried out.

1.2 Contributions

The main contributions of this work are:

- As an alternative to the model proposed in [21], a high-fidelity simulation model including actuator dynamics is developed based on the Euler-Lagrange formalism.

- Inertial properties and propeller parameters are identified for the MiniPetrel prototype.

- The control allocation scheme is reformulated as an optimization problem and together with the Euler-Lagrange equations of motion verified through simulation.

- A test platform for thrust vectoring autopilot implementation is developed based on the open-source PX4 flight stack, and verified through flight tests in a laboratory environment.

- A Gazebo model for the StormPetrel UAV is developed for software-in-the-loop (SITL) simulation and interfaced to the PX4 test platform.

1.3 Overview

This work consists parts of both theoretical and practical elements. Therefore, this thesis has been divided into two main parts. Part 1, which is the more theoretical, consists of chapters 2-5 and addresses modeling, parameter identification and control, while part 2, consisting of chapters 6-8, presents the more practical aspects of developing a test platform for implementation of control algorithms for testing on the aircraft prototype.

In chapter 2, a mathematical model for the UAV is developed based on a 10 DOF Euler-Lagrange approach. In chapter 3, some of the parameters arising in this model is estimated based on CAD-models and propeller testing. Chapter 4 addresses the issue of control system design, while part 1 is concluded by simulation results in chapter 5.

Chapter 6 shows how the open-source PX4 flight stack can be modified and customized to run custom autopilot code, including control allocation for the StormPetrel UAV. In chapter 7, a StormPetrel model is implemented in the Gazebo open-source robot simulator for SITL simulation. Chapter 8 documents flight tests of the MiniPetrel prototype using the system mentioned above.

Finally, a summary and conclusion, as well as recommendations for future work are given in chapter 9.

In addition, a video documenting propeller tests, flight experiments and SITL simulations is attached in the digital appendix.

Part I

Modeling, Identification and Control

2 Modeling

In this chapter, mathematical models describing the motion of the Storm-Petrel UAV will be derived. The Euler-Lagrange equations of motion are developed for the vehicle, including actuator dynamics and external forces due to aerodynamic effects.

The following assumptions will be used when developing the model:

Assumptions

- The UAV consists of five links that rotate relative to each other, which are rigid bodies, i.e. deformation is neglected.
- The propellers are rigid.
- The main forces and moments acting on the UAV are due to aerodynamics, propulsion and gravity.
- Ground effects are not modeled, i.e. model is not valid for take-off and landing.
- Interference between propeller wakes and wing is assumed negligible due to the relatively wide placement of the thrusters in the longitudinal direction.

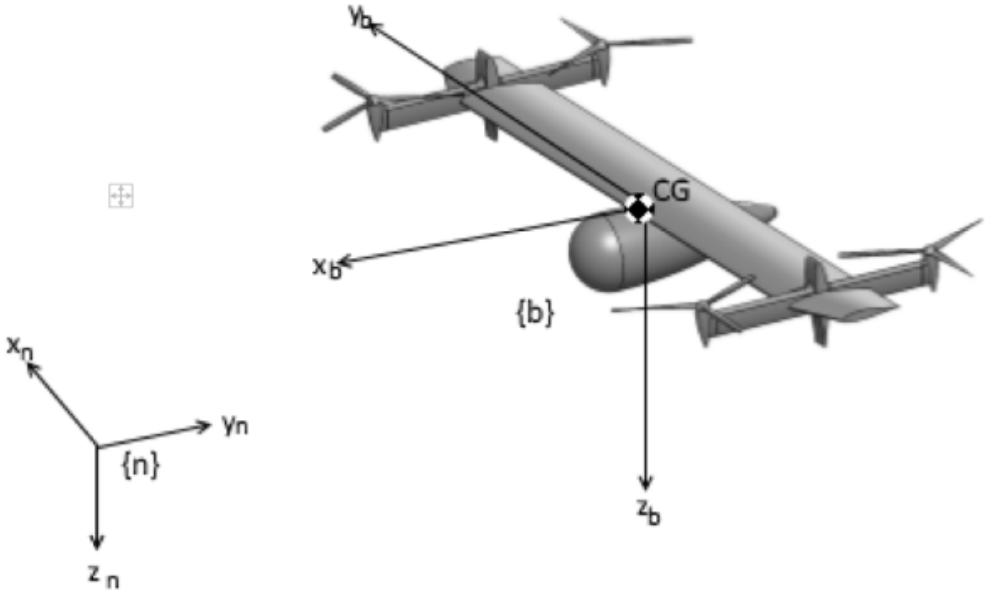


Figure 2.1: Coordinate frames

Flat Earth Navigation

We will consider motion of the UAV in a local area, with approximately constant longitude and latitude. This justifies the use of *flat earth navigation* [17]. A body-fixed frame $\{b\}$ is rigidly attached to the main body link (consisting of wing, fuselage and landing gear), with the origin at the center of mass. The position and orientation in space of $\{b\}$ will be described relative to an earth-fixed coordinate system with the x-axis pointing north, y-axis pointing east and z-axis toward the center of the earth. This is often referred to as a north-east-down (NED) frame, and will be denoted $\{n\}$. It is assumed inertial so that Newton's laws apply. These two frames are illustrated in figure 2.1. The position of the origin of $\{b\}$ relative to $\{n\}$ will be denoted $\mathbf{r}_b = \begin{bmatrix} x & y & z \end{bmatrix}^T$. The orientation (attitude) of the aircraft is uniquely defined by a rotation matrix $\mathbf{R}_b^n \in SO(3)$. The attitude can also be represented by a set of three Euler angles $\mathbf{\Theta}_{nb} = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^T$, also called roll, pitch and yaw angles.

Generalized Coordinates

Introductions to the Euler-Lagrange equations can be found in [30] and [16]. The method requires a choice of *generalized* coordinates that also are a minimal representation of the configuration of the vehicle. Here, the following choice of generalized coordinates will be used:

$$\mathbf{q} = \begin{bmatrix} x & y & z & \phi & \theta & \psi & \lambda_r & \lambda_l & \gamma_r & \gamma_l \end{bmatrix}^T \tag{2.1}$$

λ_r and λ_l are the right and left tilt angles, while γ_r and γ_l are the right and left twist angles respectively.

The angular velocity of $\{b\}$ relative to $\{n\}$ expressed in $\{b\}$ will be denoted $\boldsymbol{\omega}_{b/n}^b = \begin{bmatrix} p & q & r \end{bmatrix}^T$ and can be related to the Euler angles as follows [17]:

$$\boldsymbol{\omega}_{b/n}^b = \underbrace{\begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\phi) & \cos(\theta)\sin(\phi) \\ 0 & -\sin(\phi) & \cos(\theta)\cos(\phi) \end{bmatrix}}_{\mathbf{T}_\Theta^{-1}(\boldsymbol{\Theta}_{nb})} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{2.2}$$

In the following sections, expressions for the linear and angular velocity for each link (main body, right tilt, left tilt, right twist and left twist) will be derived in terms of the generalized coordinates in the following form:

$$\mathbf{v}_i = \mathbf{J}_{v_i}\dot{\mathbf{q}} \qquad\qquad \boldsymbol{\omega}_{i/n}^i = \mathbf{J}_{\omega_i}\dot{\mathbf{q}} \tag{2.3}$$

The Jacobian matrices $\mathbf{J}_{v_i}$ and $\mathbf{J}_{\omega_i}$ will then be used to derive the Euler-Lagrange equations based on total kinetic and potential energy.

2.1 Kinematics

2.1.1 Coordinate Frames

In addition to the body frame {b}, a coordinate frame will be rigidly attached in the center of mass of the other four links. These will be denoted $\{t_r\}$,

$\{t_l\}$, $\{tw_r\}$ and $\{tw_l\}$. In hover configuration, $\lambda_j = \gamma_j = 0$, and these frames are parallel to the body frame. The tilt angle is defined positive when tilting forward (negative rotation about body y-axis), while the twist angle is defined positive to the right (positive rotation about tilt x-axis). The propeller thrust is then directed along the z-axis of the twist frame in the negative direction.

2.1.2 Rotation Matrices

The rotation matrix $\mathbf{R}_b^n$ can be expressed with respect to the Euler angles as follows:

$$\begin{aligned}\mathbf{R}_b^n\left(\boldsymbol{\Theta}_{nb}\right) &= \mathbf{R}_{z,\psi}\mathbf{R}_{y,\theta}\mathbf{R}_{x,\phi} \\ &= \begin{bmatrix} \mathrm{c}\psi\mathrm{c}\theta & -\mathrm{s}\psi\mathrm{c}\phi + \mathrm{c}\psi\mathrm{s}\theta\mathrm{s}\phi & \mathrm{s}\psi\mathrm{s}\phi + \mathrm{c}\psi\mathrm{c}\phi\mathrm{s}\theta \\ \mathrm{s}\psi\mathrm{c}\theta & \mathrm{c}\psi\mathrm{c}\phi + \mathrm{s}\phi\mathrm{s}\theta\mathrm{s}\psi & -\mathrm{c}\psi\mathrm{s}\phi + \mathrm{s}\theta\mathrm{s}\psi\mathrm{c}\phi \\ -\mathrm{s}\theta & \mathrm{c}\theta\mathrm{s}\phi & \mathrm{c}\theta\mathrm{c}\phi \end{bmatrix}\end{aligned} \tag{2.4}$$

where $\mathrm{s}(\cdot)$ and $\mathrm{c}(\cdot)$ is short-hand notation for $\sin(\cdot)$ and $\cos(\cdot)$ respectively.

The rotation matrices relating the tilt and twist frame to the body frame are:

$$\mathbf{R}_{t_j}^b(\lambda_j) = \mathbf{R}_{y,-\lambda_j} = \begin{bmatrix} \cos(\lambda_j) & 0 & -\sin(\lambda_j) \\ 0 & 1 & 0 \\ \sin(\lambda_j) & 0 & \cos(\lambda_j) \end{bmatrix} \qquad j = r, l \tag{2.5}$$

$$\mathbf{R}_{tw_j}^{t_j}(\gamma_j) = \mathbf{R}_{x,\gamma_j} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\gamma_j) & -\sin(\gamma_j) \\ 0 & \sin(\gamma_j) & \cos(\gamma_j) \end{bmatrix} \qquad j = r, l \tag{2.6}$$

$$\mathbf{R}_{tw_j}^b(\lambda_j, \gamma_j) = \mathbf{R}_{t_j}^b(\lambda_j)\mathbf{R}_{tw_j}^{t_j}(\gamma_j) \qquad j = r, l \tag{2.7}$$

2.1.3 Coordinate Vectors

The coordinate vectors of each center of mass can be expressed as:

$$\mathbf{r}_b = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2.8}$$

$$\mathbf{r}_{t_r} = \mathbf{r}_b + \mathbf{R}_b^n \left(\mathbf{r}_r + \mathbf{R}_{t_r}^b \mathbf{r}_{t_r} \right) \tag{2.9}$$

$$\mathbf{r}_{t_l} = \mathbf{r}_b + \mathbf{R}_b^n \left(\mathbf{r}_l + \mathbf{R}_{t_l}^b \mathbf{r}_{t_l} \right) \tag{2.10}$$

$$\mathbf{r}_{tw_r} = \mathbf{r}_b + \mathbf{R}_b^n \left(\mathbf{r}_r + \mathbf{R}_{tw_r}^b \mathbf{r}_{tw_r} \right) \tag{2.11}$$

$$\mathbf{r}_{tw_l} = \mathbf{r}_b + \mathbf{R}_b^n \left(\mathbf{r}_l + \mathbf{R}_{tw_l}^b \mathbf{r}_{tw_l} \right) \tag{2.12}$$

Here, $\mathbf{r}_l$ and $\mathbf{r}_l$ are vectors from the main body center of mass to end-of-wing datums that lie on the tilt axis and are used to express the tilt and twist centers of mass relative to. These datums are also discussed in chapter 3.

2.1.4 Angular Velocity

Derivatives of the rotation matrix between NED and body can be formulated as [16]:

$$\frac{d}{dt}\mathbf{R}_b^n = \mathbf{R}_b^n \mathbf{S}(\boldsymbol{\omega}_{b/n}^b) \tag{2.13}$$

Similarily, using the same formula:

$$\frac{d}{dt}\mathbf{R}_{t_j}^b = \mathbf{R}_{t_j}^b \mathbf{S}(\boldsymbol{\omega}_{t_j/b}^{t_j}) \qquad j = r, l \tag{2.14}$$

$$\frac{d}{dt}\mathbf{R}_{tw_j}^{t_j} = \mathbf{R}_{tw_j}^{t_j} \mathbf{S}(\boldsymbol{\omega}_{tw_j/t_j}^{tw_j}) \qquad j = r, l \tag{2.15}$$

$$\frac{d}{dt}\mathbf{R}_{tw_j}^b = \mathbf{R}_{tw_j}^b \mathbf{S}\left(\boldsymbol{\omega}_{tw_j/b}^{tw_j}\right) \qquad j = r, l \tag{2.16}$$

where the angular velocites are:

$$\boldsymbol{\omega}_{t_j/b}^{t_j} = \begin{bmatrix} 0 \\ -\dot{\lambda}_j \\ 0 \end{bmatrix} \qquad j = r, l \qquad (2.17)$$

$$\boldsymbol{\omega}_{tw_j/t_j}^{tw_j} = \begin{bmatrix} \dot{\gamma}_j \\ 0 \\ 0 \end{bmatrix} \qquad j = r, l \qquad (2.18)$$

$$\boldsymbol{\omega}_{tw_j/b}^{tw_j} = \boldsymbol{\omega}_{tw_j/t_j}^{tw_j} + \boldsymbol{\omega}_{t_j/b}^{tw_j} = \boldsymbol{\omega}_{tw_j/t_j}^{tw_j} + \left(\mathbf{R}_{tw_j}^{t_j}\right)^T \boldsymbol{\omega}_{t_j/b}^{t_j} \qquad j = r, l \qquad (2.19)$$

The angular velocites relative to NED of each link, expressed in their respective frames, can now be derived with respect to the derivatives of the generalized coordinates:

$$\boldsymbol{\omega}_{b/n}^{b} = \underbrace{\begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{T}_{\Theta}^{-1} & \mathbf{0}_{3\times4} \end{bmatrix}}_{\mathbf{J}_{\omega_b}} \dot{\mathbf{q}} \qquad (2.20)$$

$$\begin{aligned} \boldsymbol{\omega}_{t_r/n}^{t_r} &= \boldsymbol{\omega}_{t_r/b}^{t_r} + \left(\mathbf{R}_{t_r}^{b}\right)^T \boldsymbol{\omega}_{b/n}^{b} \\ &= \begin{bmatrix} 0 \\ -\dot{\lambda}_r \\ 0 \end{bmatrix} + \left(\mathbf{R}_{t_r}^{b}\right)^T \mathbf{T}_{\Theta}^{-1} \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \\ &= \underbrace{\begin{bmatrix} \mathbf{0}_{3\times3} & \left(\mathbf{R}_{t_r}^{b}\right)^T \mathbf{T}_{\Theta}^{-1} & \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix} & \mathbf{0}_{3\times3} \end{bmatrix}}_{\mathbf{J}_{\omega_{t_r}}} \dot{\mathbf{q}} \end{aligned} \qquad (2.21)$$

$$
\begin{aligned}
\boldsymbol{\omega}_{t_l/n}^{t_l} &= \boldsymbol{\omega}_{t_l/b}^{t_l} + \left(\mathbf{R}_{t_l}^{b}\right)^T \boldsymbol{\omega}_{b/n}^{b} \\
&= \begin{bmatrix} 0 \\ -\dot{\lambda}_l \\ 0 \end{bmatrix} + \left(\mathbf{R}_{t_l}^{b}\right)^T \mathbf{T}_{\Theta}^{-1} \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \\
&= \underbrace{\begin{bmatrix} \mathbf{0}_{3\times 3} & \left(\mathbf{R}_{t_l}^{b}\right)^T \mathbf{T}_{\Theta}^{-1} & \mathbf{0}_{3\times 1} & \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix} & \mathbf{0}_{3\times 2} \end{bmatrix}}_{\mathbf{J}_{\omega_{t_l}}} \dot{\mathbf{q}}
\end{aligned}
\tag{2.22}
$$

$$
\begin{aligned}
\boldsymbol{\omega}_{tw_r/n}^{tw_r} &= \boldsymbol{\omega}_{tw_r/b}^{tw_r} + \left(\mathbf{R}_{tw_r}^{b}\right)^T \boldsymbol{\omega}_{b/n}^{b} \\
&= \boldsymbol{\omega}_{tw_r/t_r}^{tw_r} + \left(\mathbf{R}_{tw_r}^{t_r}\right)^T \boldsymbol{\omega}_{t_r/b}^{t_r} + \left(\mathbf{R}_{tw_r}^{b}\right)^T \boldsymbol{\omega}_{b/n}^{b} \\
&= \begin{bmatrix} \dot{\gamma}_r \\ 0 \\ 0 \end{bmatrix} + \left(\mathbf{R}_{tw_r}^{t_r}\right)^T \begin{bmatrix} 0 \\ -\dot{\lambda}_r \\ 0 \end{bmatrix} + \left(\mathbf{R}_{tw_r}^{b}\right)^T \mathbf{T}_{\Theta}^{-1} \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \\
&= \underbrace{\begin{bmatrix} \mathbf{0}_{3\times 3} & \left(\mathbf{R}_{tw_r}^{b}\right)^T \mathbf{T}_{\Theta}^{-1} & \left(\mathbf{R}_{tw_r}^{t_r}\right)^T \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix} & \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{bmatrix}}_{\mathbf{J}_{\omega_{tw_r}}} \dot{\mathbf{q}}
\end{aligned}
\tag{2.23}
$$

$$
\begin{aligned}
\boldsymbol{\omega}_{tw_l/n}^{tw_l} &= \boldsymbol{\omega}_{tw_l/b}^{tw_l} + \left(\mathbf{R}_{tw_l}^{b}\right)^T \boldsymbol{\omega}_{b/n}^{b} \\
&= \boldsymbol{\omega}_{tw_l/t_l}^{tw_l} + \left(\mathbf{R}_{tw_l}^{t_l}\right)^T \boldsymbol{\omega}_{t_l/b}^{t_l} + \left(\mathbf{R}_{tw_l}^{b}\right)^T \boldsymbol{\omega}_{b/n}^{b} \\
&= \begin{bmatrix} \dot{\gamma}_l \\ 0 \\ 0 \end{bmatrix} + \left(\mathbf{R}_{tw_l}^{t_l}\right)^T \begin{bmatrix} 0 \\ -\dot{\lambda}_l \\ 0 \end{bmatrix} + \left(\mathbf{R}_{tw_l}^{b}\right)^T \mathbf{T}_{\Theta}^{-1} \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \\
&= \underbrace{\begin{bmatrix} \mathbf{0}_{3\times 3} & \left(\mathbf{R}_{tw_l}^{b}\right)^T \mathbf{T}_{\Theta}^{-1} & \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} & \left(\mathbf{R}_{tw_l}^{t_l}\right)^T \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix} & \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \end{bmatrix}}_{\mathbf{J}_{\omega_{tw_l}}} \dot{\mathbf{q}}
\end{aligned}
\tag{2.24}
$$

2.1.5 Linear Velocity

In the same manner, expressions for the linear velocities can be found by differentiating the coordinate vectors of each link:

$$\mathbf{v}_b = \frac{d}{dt}\mathbf{r}_b = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \end{bmatrix}}_{\mathbf{J}_{v_b}} \dot{\mathbf{q}} \tag{2.25}$$

$$\begin{aligned}
\mathbf{v}_{t_r} &= \frac{d}{dt}\mathbf{r}_{t_r} \\
&= \mathbf{v}_b + \dot{\mathbf{R}}_b^n \mathbf{r}_r + \cancelto{0}{\mathbf{R}_b^n \dot{\mathbf{r}}_r} + \dot{\mathbf{R}}_b^n \mathbf{R}_{t_r}^b \mathbf{r}_{t_r} + \mathbf{R}_b^n \dot{\mathbf{R}}_{t_r}^b \mathbf{r}_{t_r} + \cancelto{0}{\mathbf{R}_b^n \mathbf{R}_{t_r}^b \dot{\mathbf{r}}_{t_r}} \\
&= \mathbf{v}_b + \mathbf{R}_b^n \mathbf{S}(\boldsymbol{\omega}_{b/n}^b)\mathbf{r}_r + \mathbf{R}_b^n \mathbf{S}(\boldsymbol{\omega}_{b/n}^b)\mathbf{R}_{t_r}^b \mathbf{r}_{t_r} + \mathbf{R}_b^n \mathbf{R}_{t_r}^b \mathbf{S}(\boldsymbol{\omega}_{t_r/b}^{t_r})\mathbf{r}_{t_r} \\
&= \mathbf{v}_b - \mathbf{R}_b^n \mathbf{S}(\mathbf{r}_r)\boldsymbol{\omega}_{b/n}^b - \mathbf{R}_b^n \mathbf{S}(\mathbf{R}_{t_r}^b \mathbf{r}_{t_r})\boldsymbol{\omega}_{b/n}^b - \mathbf{R}_b^n \mathbf{R}_{t_r}^b \mathbf{S}(\mathbf{r}_{t_r})\boldsymbol{\omega}_{t_r/b}^{t_r} \\
&= \mathbf{v}_b - \mathbf{R}_b^n \mathbf{S}\left(\mathbf{r}_r + \mathbf{R}_{t_r}^b \mathbf{r}_{t_r}\right)\boldsymbol{\omega}_{b/n}^b - \mathbf{R}_b^n \mathbf{R}_{t_r}^b \mathbf{S}(\mathbf{r}_{t_r})\boldsymbol{\omega}_{t_r/b}^{t_r} \\
&= \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} - \mathbf{R}_b^n \mathbf{S}\left(\mathbf{r}_r + \mathbf{R}_{t_r}^b \mathbf{r}_{t_r}\right)]\mathbf{T}_\Theta^{-1} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} - \mathbf{R}_b^n \mathbf{R}_{t_r}^b \mathbf{S}(\mathbf{r}_{t_r}) \begin{bmatrix} 0 \\ -\dot{\lambda}_r \\ 0 \end{bmatrix} \\
&= \underbrace{\begin{bmatrix} \mathbf{1}_{3\times 3} & -\mathbf{R}_b^n \mathbf{S}\left(\mathbf{r}_r + \mathbf{R}_{t_r}^b \mathbf{r}_{t_r}\right)\mathbf{T}_\Theta^{-1} & \mathbf{R}_b^n \mathbf{R}_{t_r}^b \mathbf{S}(\mathbf{r}_{t_r}) \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} & \mathbf{0}_{3\times 3} \end{bmatrix}}_{\mathbf{J}_{v_{t_r}}} \dot{\mathbf{q}}
\end{aligned} \tag{2.26}$$

$$
\begin{aligned}
\mathbf{v}_{t_l} &= \frac{d}{dt}\mathbf{r}_{t_l} \\
&= \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} - \mathbf{R}_b^n \mathbf{S}\left(\mathbf{r}_l + \mathbf{R}_{t_l}^b \mathbf{r}_{t_l}\right) \mathbf{T}_\Theta^{-1} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} - \mathbf{R}_b^n \mathbf{R}_{t_l}^b \mathbf{S}(\mathbf{r}_{t_l}) \begin{bmatrix} 0 \\ -\dot{\lambda}_l \\ 0 \end{bmatrix} \\
&= \underbrace{\begin{bmatrix} \mathbf{1}_{3\times 3} & -\mathbf{R}_b^n \mathbf{S}\big(\mathbf{r}_l + \mathbf{R}_{t_l}^b \mathbf{r}_{t_l}\big) \mathbf{T}_\Theta^{-1} & \mathbf{0}_{3\times 1} & \mathbf{R}_b^n \mathbf{R}_{t_l}^b \mathbf{S}(\mathbf{r}_{t_l}) \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} & \mathbf{0}_{3\times 2} \end{bmatrix}}_{\mathbf{J}_{v_{t_l}}} \dot{\mathbf{q}}
\end{aligned}
\tag{2.27}
$$

$$
\begin{aligned}
\mathbf{v}_{tw_r} &= \frac{d}{dt}\mathbf{r}_{t_r} \\
&= \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} - \mathbf{R}_b^n \left[\mathbf{S}(\mathbf{r}_r) + \mathbf{S}(\mathbf{R}_{tw_r}^b \mathbf{r}_{tw_r})\right] \mathbf{T}_\Theta^{-1} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \\
&\quad + \mathbf{R}_b^n \mathbf{R}_{t_r}^b \mathbf{S}(\mathbf{R}_{tw_r}^b \mathbf{r}_{t_r}) \begin{bmatrix} 0 \\ \dot{\lambda}_r \\ 0 \end{bmatrix} - \mathbf{R}_b^n \mathbf{R}_{tw_r}^b \mathbf{S}(\mathbf{r}_{\mathbf{tw_r}}) \begin{bmatrix} \dot{\gamma}_r \\ 0 \\ 0 \end{bmatrix} \\
&= \underbrace{\begin{bmatrix} \mathbf{1}_{3\times 3} & -\mathbf{R}_b^n \mathbf{S}\big(\mathbf{r}_r + \mathbf{R}_{tw_r}^b \mathbf{r}_{tw_r}\big) \mathbf{T}_\Theta^{-1} & \mathbf{R}_b^n \mathbf{R}_{t_r}^b \mathbf{S}\big(\mathbf{R}_{tw_r}^{t_r} \mathbf{r}_{tw_r}\big) \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} & \mathbf{0}_{3\times 1} & -\mathbf{R}_b^n \mathbf{R}_{tw_r}^b \mathbf{S}(\mathbf{r}_{\mathbf{tw_r}}) \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} & \mathbf{0}_{3\times 1} \end{bmatrix}}_{\mathbf{J}_{v_{tw_r}}} \dot{\mathbf{q}}
\end{aligned}
\tag{2.28}
$$

$$
\begin{aligned}
\mathbf{v}_{tw_l} &= \frac{d}{dt}\mathbf{r}_{t_l} \\
&= \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} - \mathbf{R}_b^n \left[\mathbf{S}(\mathbf{r}_l) + \mathbf{S}(\mathbf{R}_{tw_l}^b \mathbf{r}_{tw_l}) \right] \mathbf{T}_\Theta^{-1} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \\
&\quad + \mathbf{R}_b^n \mathbf{R}_{t_l}^b \mathbf{S}(\mathbf{R}_{tw_l}^b \mathbf{r}_{t_l}) \begin{bmatrix} 0 \\ \dot{\lambda}_l \\ 0 \end{bmatrix} - \mathbf{R}_b^n \mathbf{R}_{tw_l}^b \mathbf{S}(\mathbf{r_{tw_1}}) \begin{bmatrix} \dot{\gamma}_l \\ 0 \\ 0 \end{bmatrix} \\
&= \underbrace{\begin{bmatrix} \mathbf{1}_{3\times 3} & -\mathbf{R}_b^n \mathbf{S}\big(\mathbf{r}_l + \mathbf{R}_{tw_l}^b \mathbf{r}_{tw_l}\big) \mathbf{T}_\Theta^{-1} & \mathbf{0}_{3\times 1} & \mathbf{R}_b^n \mathbf{R}_{t_l}^b \mathbf{S}\big(\mathbf{R}_{tw_l}^{t_l} \mathbf{r}_{tw_l}\big) \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} & \mathbf{0}_{3\times 1} & -\mathbf{R}_b^n \mathbf{R}_{tw_l}^b \mathbf{S}(\mathbf{r_{tw_1}}) \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \end{bmatrix}}_{\mathbf{J}_{v_{tw_l}}} \dot{\mathbf{q}}
\end{aligned}
\tag{2.29}
$$

2.2 Dynamics

2.2.1 Kinetic Energy

The kinetic energy of each rigid body is:

$$
K_i = \frac{1}{2} m_i \mathbf{v}_i^T \mathbf{v}_i + \frac{1}{2} \boldsymbol{\omega}_{i/n}^T \mathbf{I}_i \boldsymbol{\omega}_{i/n} \tag{2.30}
$$

m_i is the mass of link i in kg, and $\mathbf{I}_i \in \mathbb{R}^{3\times 3}$ is the symmetric and positive definite *inertia tensor* of link $i$about its center of mass and aligned with the axes of its coordinate frame.

$$
\mathbf{I} \triangleq \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}, \qquad [\,\mathrm{kg} \cdot \mathrm{m}^2] \tag{2.31}
$$

The total kinetic energy for the entire vehicle is then [30]:

$$\mathcal{K} = \sum_{i=1}^{n} K_i = \frac{1}{2}\dot{\mathbf{q}}^T \mathbf{D}(\mathbf{q})\dot{\mathbf{q}} \tag{2.32}$$

where

$$\mathbf{D}(\mathbf{q}) = \left[\sum_{\mathcal{V}} m_i \mathbf{J}_{v_i}^T(\mathbf{q})\mathbf{J}_{v_i}(\mathbf{q}) + \mathbf{J}_{w_i}^T(\mathbf{q})\mathbf{I}_i\mathbf{J}_{w_i}(\mathbf{q})\right] \tag{2.33}$$

2.2.2 Potential Energy

The potential energy of each link is

$$P_i = -m_i \mathbf{g}^T \mathbf{r}_i \tag{2.34}$$

where $\mathbf{g} = \begin{bmatrix} 0 & 0 & g \end{bmatrix}^T$ and $g = 9.81\,\mathrm{m/s}$ is the acceleration of gravity.

Again, summing the potential energy for all links gives:

$$\mathcal{P} = \sum_{\mathcal{V}} P_i = -\mathbf{g}^T \sum_{\mathcal{V}} m_i \mathbf{r}_i \tag{2.35}$$

2.2.3 Euler-Lagrange Equations of Motion

For a system of rigid bodies with kinetic energy in the form (2.32) and potential energy independent of $\dot{\mathbf{q}}$, which also holds here, the Euler-Lagrange equations can be written [30]:

$$\mathbf{D}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \bar{\boldsymbol{\tau}} \tag{2.36}$$

where $\mathbf{g}(\mathbf{q})$ is the gravity vector

$$\mathbf{g}(\mathbf{q}) = \begin{bmatrix} g_1(\mathbf{q}) & \dots & g_n(\mathbf{q}) \end{bmatrix}^T \tag{2.37}$$

$$g_k = \frac{\partial \mathcal{P}}{\partial q_i} \tag{2.38}$$

$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is called the coriolis and centrifugal matrix, and can be calculated from $\mathbf{D}(\mathbf{q})$. The $(k,j)^{th}$ element of $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ can be calculated in the following way:

$$c_{kj} = \sum_{i=1}^{n} \frac{1}{2} \left(\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k} \right) \dot{q}_i \tag{2.39}$$

where d_{ij} denotes the $(i,j)^{th}$ element of $\mathbf{D}(\mathbf{q})$.

Next, actuator dynamics will be added to the model, as well as external generalized forces $\bar{\boldsymbol{\tau}}$.

2.3 Actuator Dynamics

In total, the vehicle has eight actuators. There are four propellers, and the thrust vectoring system is controlled by four servo motors. This section covers actuator dynamics and presents equations that govern the dynamics of servo motors and thrusters.

2.3.1 Servo Motors

A servo motor can be modelled as a DC motor in series with a gear train with gear ratio $r:1$, and an integrated position feedback controller [21]. The tilt servos carry heavier loads and are thus larger and capable of producing more torque than the twist servos. For tilt, the equations are:

$$\lambda_j = \frac{1}{r_\lambda} \theta_{\lambda_j} \tag{2.40}$$

$$J_\lambda \ddot{\theta}_{\lambda_j} + B_\lambda \dot{\theta}_{\lambda_j} = u_{\lambda_j} - \frac{1}{r_\lambda} \tau_\lambda^{tot} \qquad j = r, l \tag{2.41}$$

$$u_{\lambda_j} = \text{sat} \left(k_{p_\lambda} e_{\lambda_j} + k_{d_\lambda} \dot{e}_{\lambda_j} + k_{i_\lambda} \int_0^t e_{\lambda_j}(\tau) d\tau \right) \tag{2.42}$$

$$e_{\lambda_j} = \text{sat}(\lambda_j^d) - \lambda_j \tag{2.43}$$

where $J_\lambda [\,\text{kg} \cdot \text{m}^2]$ is the sum of motor and gear moment of inertia, B_λ is a coefficient of friction, $\theta_{\lambda_j} [\,\text{rad}]$ is the motor angle, while λ_j is the output

angle after gearing. λ_j^d is the desired tilt angle commanded by the control system, and k_{p_λ}, k_{i_λ} and k_{d_λ} are proportional, integral and derivative gains respectively. τ_λ^{tot} is the load torque, which will be defined later in section 2.5. The saturation function $\text{sat}(\cdot)$ represent limits on torque and angles produced by the servo motor, and is defined as follows:

$$\text{sat}(x) = \begin{cases} x_{max} & \text{if } x \geq x_{max} \\ x_{min} & \text{if } x \leq x_{min} \\ x & \text{otherwise} \end{cases} \tag{2.44}$$

For twist, the equivalent expressions can be obtained by replacing λ with γ in the equations above.

2.3.2 Thruster Model

Thruster Dynamics

The thrusters are composed of fixed-pitch carbon fiber blade propellers driven by brushless permanent magnet AC motors controlled by electronic speed controllers (ESCs). [23].

In [21], the thruster dynamics are modelled similarly to the servo motor model (2.40) - (2.43):

$$J_p \dot{\Omega}_i = u_i - Q_{p_i} \tag{2.45}$$

$$u_i = \text{sat}\left(k_{p_{esc_i}} e_i + k_{i_{esc_i}} \int_0^t e_i(\tau) d\tau\right) \qquad i = 1, \ldots, 4 \tag{2.46}$$

$$e_i = \text{sat}(\Omega_{d_i}) - \Omega_i \tag{2.47}$$

$$Q_{m_i} = -\text{sat}(u_i) \tag{2.48}$$

J_p is the total moment of inertia of one propeller and motor, Ω_i is the signed rotational speed of propeller i, Q_{p_i} is the load torque that the propeller excerts on the motor, given by equation 2.53, $\text{sat}(u_i)$ is the limited motor

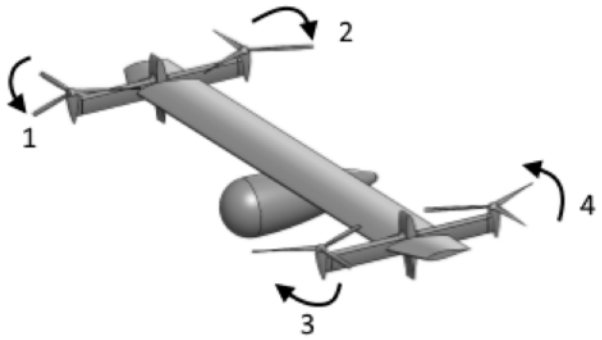


Figure 2.2: Propeller numbering and direction of rotation

torque, while Q_{m_i} is the equal and opposite reaction torque excerted by the motor on the vehicle, adhering to Newton's third law. Ω_{d_i} is the desired rotational speed commanded by the control system.

All propellers rotate in a fixed direction. The direction of rotation for each propeller as well as propeller numbering is shown in figure 2.2. The sign of Ω_i is determined using the right-hand rule about the twist frame z-axis, which is parallel to the rotation axis of the propeller. The rotational speed of clockwise (CW) rotating propellers are positive, while the opposite holds for counterclockwise (CCW) rotating propellers.

The velocity of propeller i expressed in the twist frame is given by:

$$\boldsymbol{\omega}_{p_i/tw_j}^{tw_j} = \begin{bmatrix} 0 \\ 0 \\ \Omega_i \end{bmatrix} \tag{2.49}$$

Propeller Aerodynamics

For hovering and low airspeed flight, the thrust and drag torque generated by one propeller can be modelled by [21]

$$T_p = \rho_a D^4 C_T \Omega^2 \tag{2.50}$$

$$Q_p = \rho_a D^5 C_Q \Omega |\Omega| \tag{2.51}$$

$D[\,\mathrm{m}]$ is the propeller diameter, and C_T and C_Q are nondimensional thrust and drag torque coefficients. To simplify, we will assume that the density of air is constant and equal to sea level density at 15° C, which implies $\rho_a = 1.225[\,\mathrm{kg/m^3}]$ (International Standard Atmosphere) [12], and write:

$$T_p = k\Omega^2 \tag{2.52}$$

$$Q_p = d\Omega|\Omega| \tag{2.53}$$

Dividing Q_p by T_p gives

$$\frac{Q_p}{T_p} = \frac{d\Omega|\Omega|}{k\Omega^2} = \frac{d}{k}\,\mathrm{sgn}(\Omega) \triangleq c \tag{2.54}$$

which will be used in the control allocation scheme presented in chapter 5. The parameters k and d will be determined experimentally in chapter 3.

2.4 External Forces

As previously stated, it is assumed that the main forces and moments acting on the UAV are due to aerodynamics, propulsion and gravity. Gravity is accounted for in the Euler-Lagrange equations through the potential energy term. In addition, the reaction torques and gyroscopic terms that arise when the vehicle links accelerate and rotate relative to each other are accounted for. We need however, to explicitly add the effects of rotating and thrust generating propellers, as well as the aerodynamic effects generated by the airflow generated by vehicle motion.

First we will combine the servo model (2.40) - (2.43) with the Euler-Lagrange equations given by (2.36) in a similar way as in [30]. By multiplying (2.41) with r_λ and inserting (2.40) we get:

$$r_\lambda^2 J_\lambda \ddot{\lambda}_{t_j} + r_\lambda^2 B_\lambda \dot{\lambda}_{t_j} = r_\lambda u_{\lambda_j} - \tau_{\lambda_j}^{tot} \tag{2.55}$$

We continue by dividing the load torque $\tau_{\lambda_j}^{tot}$ into two components, one given by the relevant term from $\bar{\boldsymbol{\tau}}$ in equation (2.36), and one representing added

terms due to propulsion moments:

$$\tau_{\lambda_j}^{tot} = \tau_{\lambda_{EL_j}} + \tau_{\lambda_j} \tag{2.56}$$

If we repeat this for all tilt and twist angles and define

$$\mathbf{J} = \begin{bmatrix} \mathbf{0}_{6\times6} & \mathbf{0}_{6\times4} \\ \mathbf{0}_{4\times6} & \mathbf{\Gamma} \end{bmatrix} \tag{2.57}$$

$$\mathbf{\Gamma} = \text{diag}\left(r_\lambda^2 J_\lambda, r_\lambda^2 J_\lambda, r_\gamma^2 J_\gamma, r_\gamma^2 J_\gamma\right) \tag{2.58}$$

$$\mathbf{B} = \begin{bmatrix} \mathbf{0}_{6\times6} & \mathbf{0}_{6\times4} \\ \mathbf{0}_{4\times6} & \mathbf{\Lambda} \end{bmatrix} \tag{2.59}$$

$$\mathbf{\Lambda} = \text{diag}\left(r_\lambda^2 B_\lambda, r_\lambda^2 B_\lambda, r_\gamma^2 B_\gamma, r_\gamma^2 B_\gamma\right) \tag{2.60}$$

we get by combining (2.36) and (2.55) - (2.60):

$$\mathbf{M}\left(\mathbf{q}\right)\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{B}\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau} \tag{2.61}$$

where $\mathbf{M}\left(\mathbf{q}\right) = \mathbf{D}\left(\mathbf{q}\right) + \mathbf{J}$ and

$$\boldsymbol{\tau} = \begin{bmatrix} \mathbf{f}^n \\ \mathbf{m}^b \\ r_\lambda u_{\lambda_r} - \tau_{\lambda_r} \\ r_\lambda u_{\lambda_l} - \tau_{\lambda_l} \\ r_\gamma u_{\gamma_r} - \tau_{\gamma_r} \\ r_\gamma u_{\gamma_l} - \tau_{\gamma_l} \end{bmatrix} \tag{2.62}$$

$\mathbf{f}^n \in \mathbb{R}^3$ is a vector of forces acting on the vehicle, expressed in $\{n\}$, while $\mathbf{m} = \begin{bmatrix} M_x & M_y & M_z \end{bmatrix}^T \in \mathbb{R}^3$ are the moments acting on the vehicle, expressed in $\{\text{b}\}$.

In the following, forces and moments generated by propulsion and aerodynamics will be covered, and the elements of (2.62) will be defined. The elements of the force vector $\mathbf{f}^n$ and the moment vector $\mathbf{m}^b$ will be derived in $\{b\}$

and split up into propulsion and aerodynamic components in the following way:

$$\mathbf{f}^n = \mathbf{R}_b^n \mathbf{f}^b = \mathbf{R}_b^n \left(\mathbf{f}_{prop}^b + \mathbf{f}_{aero}^b \right) \tag{2.63}$$

$$\mathbf{m}^b = \mathbf{m}_{prop}^b + \mathbf{m}_{aero}^b \tag{2.64}$$

The elements of $\mathbf{f}^b$ will be denoted $\mathbf{f}^b = \begin{bmatrix} F_x & F_y & F_z \end{bmatrix}^T$

2.4.1 Propulsion Forces and Moments

Propeller Force

The combined thrust of all four propellers produce a total force acting on the vehicle, where the direction is governed by the tilt and twist servo angles:

$$\mathbf{f}_{p_i}^{tw_j} = \begin{bmatrix} 0 \\ 0 \\ -T_{p_i} \end{bmatrix} \tag{2.65}$$

$$\mathbf{f}_{prop}^b = \sum_{i=1}^{4} \mathbf{f}_{p_i}^b = \sum_{i=1}^{4} \mathbf{R}_{tw_j}^b \mathbf{f}_{p_i}^{tw_j} \tag{2.66}$$

Moment Arms

To get expressions for the moments generated by thrust differencing, the moment arms are needed:

$$\mathbf{r}_{p_1}^b = -\mathbf{r}_{b_c} + \begin{bmatrix} -l_c + l_{tw}\cos(\lambda_r) \\ l_t \\ l_{tw}\sin(\lambda_r) \end{bmatrix} \qquad \mathbf{r}_{p_2}^b = -\mathbf{r}_{b_c} + \begin{bmatrix} -l_c - l_{tw}\cos(\lambda_r) \\ l_t \\ -l_{tw}\sin(\lambda_r) \end{bmatrix}$$
$$\mathbf{r}_{p_3}^b = -\mathbf{r}_{b_c} + \begin{bmatrix} -l_c + l_{tw}\cos(\lambda_l) \\ -l_t \\ l_{tw}\sin(\lambda_l) \end{bmatrix} \qquad \mathbf{r}_{p_4}^b = -\mathbf{r}_{b_c} + \begin{bmatrix} -l_c - l_{tw}\cos(\lambda_l) \\ -l_t \\ -l_{tw}\sin(\lambda_l) \end{bmatrix} \tag{2.67}$$

$\mathbf{r}_{p_i}$ is vector pointing from the main body center of mass to propeller i, $\mathbf{r}_{b_c}$ is the main body center of mass relative to the middle of the wing leading edge, l_t is the length between the body x-axis and the propellers, $l_t w$ is half the length of the tilt arms, and l_c is the distance between the leading edge and the tilt rotation axis along the body x-axis.

Thrust Differencing

The moments acting on the body due to thrust differencing are:

$$\mathbf{m}_T^b = \sum_{i=1}^{4} \mathbf{r}_{p_i}^b \times \mathbf{f}_{p_i}^b = \sum_{i=1}^{4} \mathbf{S}\left(\mathbf{r}_{p_i}^b\right) \mathbf{f}_{p_i}^b \tag{2.68}$$

Reaction Torque

The reaction torque from thruster motor i expressed in the appropriate twist frame is:

$$\mathbf{m}_{m_i}^{tw_j} = \begin{bmatrix} 0 \\ 0 \\ Q_{m_i} \end{bmatrix} \tag{2.69}$$

where Q_{m_i} is given by (2.48).

Torque Differencing

Differences in reaction torques wil generate a net moment on the vehicle defined by:

$$\mathbf{m}_m^b = \sum_{i=1}^{4} \mathbf{R}_{tw_j}^b \mathbf{m}_{m_i}^{tw_j} \tag{2.70}$$

Gyroscopic Moment

Due to the rotating masses of the propellers, the aircraft body will experience a gyroscopic moment when rotating [23]. Decomposed in the body frame, this is given by:

$$\mathbf{m}_{\text{gyro}}^b = \sum_{i=1}^{4} \mathbf{R}_{tw_j}^b \left(\boldsymbol{\omega}_{tw_j/n}^{tw_j} \times \mathbf{h}_{p_i}^{tw_j} \right) = -\sum_{i=1}^{4} \mathbf{R}_{tw_j}^b \left(\mathbf{S}(\mathbf{h}_{p_i}^{tw_j}) \boldsymbol{\omega}_{tw_j/n}^{tw_j} \right) \tag{2.71}$$

where

$$\mathbf{h}_{p_i}^{tw_j} = \mathbf{I}_p \boldsymbol{\omega}_{p_i/tw_j}^{tw_j} \tag{2.72}$$

is the angular momentum vector of propeller i in frame $\{tw_j\}$. Assuming $\mathbf{I}_p$ is diagonal (negligible products of inertia), we get:

$$\mathbf{h}_{p_i}^{tw_j} = \begin{bmatrix} 0 \\ 0 \\ J_p \Omega_i \end{bmatrix} \tag{2.73}$$

where J_p is the total moment of inertia of the propeller and motor rotating masses about their rotation axis. $\boldsymbol{\omega}_{tw_j/n}^{tw_w}$ can be calculated according to

$$\begin{aligned} \boldsymbol{\omega}_{tw_j/n}^{tw_j} &= \boldsymbol{\omega}_{tw_j/b}^{tw_j} + \boldsymbol{\omega}_{b/n}^{tw_j} \\ &= \boldsymbol{\omega}_{tw_j/b}^{tw_j} + (\mathbf{R}_{tw_j}^b)^T \boldsymbol{\omega}_{b/n}^b \end{aligned} \tag{2.74}$$

Without thrust-vectoring (fixed quad-mode), $\mathbf{R}_{tw_j}^b = \mathbf{1}_{3\times 3}$, $\boldsymbol{\omega}_{tw_j/b}^{tw_j} = \mathbf{0}$ and equation (2.71) reduces to the special case

$$\mathbf{m}_{\text{gyro}}^b = \sum_{i=1}^{4} \boldsymbol{\omega}_{b/n}^b \times \mathbf{h}_{p_i}^b = J_p \sum_{i=1}^{4} \begin{bmatrix} q\Omega_i \\ -p\Omega_i \\ 0 \end{bmatrix} \tag{2.75}$$

which has a similar form to the gyroscoping moment presented in [18] and [23]. Due to pairs of counter-rotating propellers, the gyroscoping moment vanishes when operating with equal tilt and twist on both sides, and when all propeller speeds are equal.

Total Propulsion Forces and Moments

Summing up the effects of thrust differencing, torque differencing and gyroscopic effects gives the total propulsion moment acting on the body:

$$\mathbf{m}^b_{prop} = \mathbf{m}^b_T + \mathbf{m}^b_m + \mathbf{m}^b_{\text{gyro}} \tag{2.76}$$

In addition, the tilt and twist servo motors will feel a load torque given by:

$$\tau_{\gamma_r} = \begin{bmatrix} -1 & 0 & 0 \end{bmatrix} \sum_{i=1}^{2} -\mathbf{S}(\mathbf{h}^{tw_r}_{p_i})\boldsymbol{\omega}^{tw_r}_{tw_r/n} \tag{2.77}$$

$$\tau_{\gamma_l} = \begin{bmatrix} -1 & 0 & 0 \end{bmatrix} \sum_{i=3}^{4} -\mathbf{S}(\mathbf{h}^{tw_l}_{p_i})\boldsymbol{\omega}^{tw_l}_{tw_l/n} \tag{2.78}$$

$$\tau_{\lambda_r} = \begin{bmatrix} 0 & -1 & 0 \end{bmatrix} \sum_{i=1}^{2} \left[-\mathbf{R}^{t_r}_{tw_r}\mathbf{S}(\mathbf{h}^{tw_r}_{p_i})\boldsymbol{\omega}^{tw_r}_{tw_r/n} + \mathbf{S}\left(\begin{bmatrix} (-1)^{i+1}l_{tw} \\ 0 \\ 0 \end{bmatrix} \right) \mathbf{R}^{t_r}_{tw_r}\mathbf{f}^{tw_r}_{p_i} + \mathbf{R}^{t_r}_{tw_r}\mathbf{m}^{tw_r}_{m_i} \right] \tag{2.79}$$

$$\tau_{\lambda_l} = \begin{bmatrix} 0 & -1 & 0 \end{bmatrix} \sum_{i=3}^{4} \left[-\mathbf{R}^{t_l}_{tw_l}\mathbf{S}(\mathbf{h}^{tw_l}_{p_i})\boldsymbol{\omega}^{tw_l}_{tw_l/n} + \mathbf{S}\left(\begin{bmatrix} (-1)^{i+1}l_{tw} \\ 0 \\ 0 \end{bmatrix} \right) \mathbf{R}^{t_l}_{tw_l}\mathbf{f}^{tw_l}_{p_i} + \mathbf{R}^{t_l}_{tw_l}\mathbf{m}^{tw_l}_{m_i} \right] \tag{2.80}$$

Some illustrations of how moments are generated in hover configuration are given in figures 2.3 - 2.5.

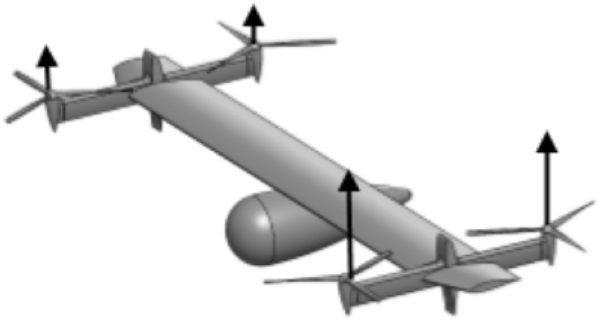


Figure 2.3: Generation of roll moment

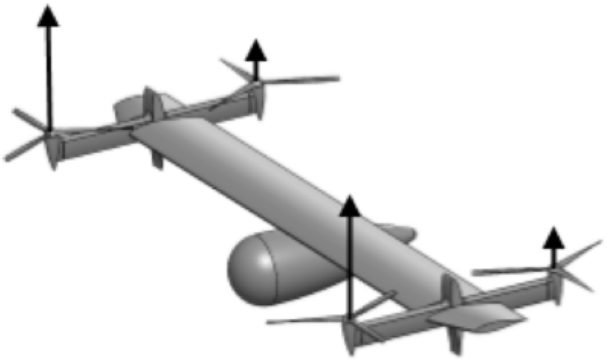


Figure 2.4: Generation of pitch moment

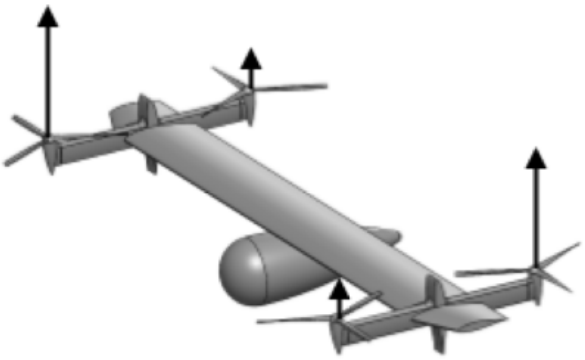


Figure 2.5: Generation of yaw moment

2.4.2 Aerodynamics

Brief introductions to aerodynamics in the context of aircraft control can be found in [14] and [18]. Comprehensive treatments of the subject can be found in [12] and [13]. For simplicity, we will in this section not consider the effects of tilt and twist on the aerodynamic forces and moments, and mainly consider the wing structure of the main body.

Basic Quantities

When an aircraft moves through the air, a pressure distribution is generated around the body. This can be modelled as a resultant force and a moment about a suitable point. The resultant force is commonly decomposed in elements parallel and perpendicular to the airflow:

$$\mathbf{f}_{aero}^{s} = \begin{bmatrix} -F_D \\ F_Y \\ -F_L \end{bmatrix} \tag{2.81}$$

F_L is the *lift* force and F_D is the *drag* force. The lift force is referenced upwards, while the drag force always acts opposing the longitudinal relative airflow. The relative velocity of the aircraft through the air, decomposed in the body frame, can be calculated as

$$\mathbf{v}_r^b = \mathbf{v}^b - \mathbf{v}_w^b = \begin{bmatrix} u_r \\ v_r \\ w_r \end{bmatrix} \tag{2.82}$$

where $\mathbf{v}_w^b$ is the velocity of the wind. This will not be covered here, but several models exist, including *Gauss-Markov* models and the *von Karmen* turbulence spectrum [14]. From this we can calculate

$$V_a = \sqrt{u_r^2 + v_r^2 + w_r^2} \tag{2.83}$$

$$\alpha = \tan^{-1}\left(\frac{w_r}{u_r}\right) \tag{2.84}$$

$$\beta = \sin^{-1}\left(\frac{v_r}{V_a}\right) \tag{2.85}$$

V_a is the total *airspeed*, α is the *angle of attack*, while β is the *sideslip angle*.

The lift and drag forces of (2.81) is defined in a coordinate frame with its axes following the longitudinal relative airflow, called the *stability frame*. $\mathbf{f}^s_{aero}$ can be rotated to the body frame by the angle of attack:

$$\mathbf{f}^b_{aero} = \mathbf{R}_{x,-\alpha}\mathbf{f}^s_{aero} \tag{2.86}$$

$$\mathbf{R}_{x,-\alpha} = \begin{bmatrix} \cos(\alpha) & 0 & -\sin(\alpha) \\ 0 & 1 & 0 \\ \sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix} \triangleq \mathbf{R}^b_s(\alpha) \tag{2.87}$$

In this context, the aerodynamic forces and moments are mainly dependent on vehicle geometry, α, β, total airspeed and the angular velocity $\boldsymbol{\omega}^b_{b/n}$ [14]. The forces and moments are commonly expressed as:

$$F_L = \frac{1}{2}\rho_a V_a^2 S C_L(\alpha, q) \tag{2.88}$$

$$F_D = \frac{1}{2}\rho_a V_a^2 S C_D(\alpha, q) \tag{2.89}$$

$$F_Y = \frac{1}{2}\rho_a V_a^2 S C_Y(\beta, p, r) \tag{2.90}$$

$$l = \frac{1}{2}\rho_a V_a^2 S b C_l(\beta, p, r) \tag{2.91}$$

$$m = \frac{1}{2}\rho_a V_a^2 S \bar{c} C_m(\alpha, q) \tag{2.92}$$

$$n = \frac{1}{2}\rho_a V_a^2 S b C_n(\beta, p, r) \tag{2.93}$$

$$\mathbf{m}^b_{aero} = \begin{bmatrix} l & m & n \end{bmatrix}^T \tag{2.94}$$

S is the *planform area* of the wing, $\bar{c}$ is the *mean chord length*, while b is the *wingspan* of the UAV. C_L, C_Y, C_D, C_l, C_m and C_n are nondimensional aerodynamic coefficients for lift force, lateral force, drag force, roll moment, pitch moment, and yaw moment respectively. These coefficients are commonly expressed as linear functions based on a first order Tayler series approximation. See [14] for details. In [21], a 6-DOF nonlinear model is developed using Newtonian flow theory on a flat plate, in combination with

blending and interpolation to be valid for a large range of angles of attack and sideslip, as well as account for nonlinear effects such as stall.

We will group the aerodynamical coefficients into longitudinal and lateral coefficients, and in the following state the nonlinear versions.

Longitudinal coefficients

The longitudinal aerodynamic coefficients are defined as

$$C_L = \Gamma_{long}(\beta)\left[(1-\sigma(\alpha))\left[C_{L_0}+C_{L_\alpha}\alpha\right]+\sigma(\alpha)\left[C_{L_{NL}}\text{sign}(\alpha)\sin^2(\alpha)\cos(\alpha)\right]\right] + C_{L_q}\frac{\bar{c}}{2V_a}q \tag{2.95}$$

$$C_D = \Gamma_{long}(\beta)\left[C_{D_0}+(1-\sigma(\alpha))\kappa\left[C_{L_0}+C_{L_\alpha}\alpha\right]^2+\sigma(\alpha)\left[C_{D_{NL}}\text{sign}(\alpha)\sin^3(\alpha)\right]\right] + C_{D_q}\frac{\bar{c}}{2V_a}q \tag{2.96}$$

$$C_m = \Gamma_{long}(\beta)\left[(1-\sigma(\alpha))\left[C_{m_0}+C_{m_\alpha}\alpha\right]+\sigma(\alpha)\left[C_{m_{NL}}\text{sign}(\alpha)\sin^2(\alpha)\right]\right] + C_{m_q}\frac{\bar{c}}{2V_a}q \tag{2.97}$$

Lateral coefficients

The lateral aerodynamic coefficients are given by

$$C_Y = C_{Y_0}+(1-\sigma(\beta))C_{Y_\beta}\beta+\sigma(\beta)\text{sign}(\beta)\Gamma_Y(\beta) + C_{Y_p}\frac{b}{2V_a}p+C_{Y_r}\frac{b}{2V_a}r \tag{2.98}$$

$$C_l = C_{l_0} + (1 - \sigma(\beta))C_{l_\beta}\beta + \sigma(\beta)\text{sign}(\beta)\Gamma_l(\beta) + C_{l_p}\frac{b}{2V_a}p + C_{l_r}\frac{b}{2V_a}r \tag{2.99}$$

$$C_n = C_{n_0} + (1 - \sigma(\beta))C_{n_\beta}\beta + \sigma(\beta)\text{sign}(\beta)\Gamma_n(\beta) + C_{n_p}\frac{b}{2V_a}p + C_{n_r}\frac{b}{2V_a}r \tag{2.100}$$

$\sigma(u)$ is a sigmoid function used to blend between the linear and nonlinear domains, $\Gamma_{long}(\beta)$ represents a Catmul-Rom cubic spline interpolation between one and zero used to compensate for the reduction in longitudinal forces and moment as β increases. For the lateral coefficients, $\Gamma_{(\cdot)}(\beta)$ is used similarly to interpolate between the linear region and a constant defining the coefficient at ninety degrees sideslip. For details, see [21].

Total Aerodynamic Forces and Moments

A suitable point to define the moments about is the *aerodynamic center*. About this point, the pitch moment do not vary with angle of attack [13]. For an airfoil, this is approximately located at the quarter-chord point, and we will here use this as an approximation for the aerodynamic center for the entire vehicle. We then need to add the moments resulting from the moment arm between this point and the center of mass given by the cross-product rule $\mathbf{m} = \mathbf{r} \times \mathbf{f}$.

The total aerodynamic forces and moments acting upon the vehicle can be

summarized as:

$$\mathbf{f}^b_{aero} = \frac{1}{2}\rho_a V_a^2 S \mathbf{R}^b_s(\alpha) \begin{bmatrix} -C_D \\ C_Y \\ -C_L \end{bmatrix} \tag{2.101}$$

$$\mathbf{m}^b_{aero} = \frac{1}{2}\rho_a V_a^2 S \begin{bmatrix} bC_l \\ \bar{c}C_m \\ bC_n \end{bmatrix} + \mathbf{r}^b_{AC} \times \mathbf{f}^b_{aero} \tag{2.102}$$

where $\mathbf{r}^b_{AC}$ is the location of the aerodynamic center relative to center of mass, and the aerodynamic coefficients are given by (2.95) - (2.100).

2.5 Total Vehicle Model

The total vehicle model can be summarized as

$$\mathbf{D}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \bar{\boldsymbol{\tau}} \tag{2.103}$$

$$\boldsymbol{\tau} = \begin{bmatrix} \mathbf{f}^n \\ \mathbf{m}^b \\ r_\lambda u_{\lambda_r} - \tau_{\lambda_r} \\ r_\lambda u_{\lambda_l} - \tau_{\lambda_l} \\ r_\gamma u_{\gamma_r} - \tau_{\gamma_r} \\ r_\gamma u_{\gamma_l} - \tau_{\gamma_l} \end{bmatrix} \tag{2.104}$$

2.6 Comments on the Equations of Motion

The choice of Euler angles as generalized coordinates indroduce a singularity in the model. More specifically, the derivatives of the Euler angles, given by (2.2) are not defined at $\theta = \pm\frac{\pi}{2}$. This singularity is discussed in detail in [17]. In [16], it is shown how the Euler-Lagrange equations for a single

rotating body are equivalent to the rotational part of the Newton-Euler equations, which is singularity-free and independent of kinematic terms, and can be transformed to the latter after a lot of algebraic substitutions. This will not be attempted here, as the equations include large 10×10 matrices with lots of complicated trigonometric terms involving $\mathbf{\Theta}_{nb}$. Of the exact same reason, the elements of these matrices are not shown here, but rather calculated using the Matlab Symbolic Math Toolbox and used for code generation of Matlab functions used in Simulink.

It would be nice however, to get explicit expressions, in closed form, free of singularities for a multi-body problem such as this. In [16], methods based on the calculus of variations are presented, including the Euler-PoincarÃ© equation that do not require generalized coordinates. In [19], singularity-free dynamic equations are developed for spacecraft-manipulator systems using quasi-coordinates and Lie theory. Spacecraft-manipulator systems are not unlike the StormPetrel UAV if one considers the main body as the vehicle, and each tilt-twist pair as a two-link manipulator with thrusters as end-effectors. These possibilities should be investigated further.

3 Parameter Identification

The models derived in chapter 2 contain a lot of physical parameters. For simulation and control design purposes, these should reflect reality as close as possible. In addition, several of these are used in the control allocation method described in chapter 5. This chapter presents an effort to identify some of these, and serves as a basis for the simulation and flight tests parts in this thesis. First the mass properties of the vehicle is estimated based on CAD models. Second a thrust stand and dynamometer is used to estimate propeller aerodynamic parameters.

3.1 Mass Properties

SolidWorks [9] is a 3D computer-aided design (CAD) and engineering program for Microsoft Windows. Sevendof has used this to develop 3D design models of the MiniPetrel prototype. In addition to 3D geometry of parts, materials and assembly definitions, SolidWorks also enables the user to calculate mass properties from models. Based on known density of standard materials as well as user-provided mass and moments of inertia, the software can numerically calculate total mass, moments of inertia and center of mass for each model.

If we, for now, ignore the rotating masses of propellers and motors, the MiniPetrel UAV consists of five main bodies that rotate with respect to the inertial frame and with respect to each other. These are the main body (wing, fuselage and landing gear), right tilt, left tilt, right twist and left twist. We consider them rigid bodies and all have some reference frame rigidly attached, namely the body frame and the respective tilt and twist frames. Modeling, including descriptions of the reference frames involved is covered in chapter 2.

If we attach a frame $\{0\}$ rigidly to any rigid body, and calculate the inertia tensor $\mathbf{I}_0$ about some point with respect to this frame, then $\mathbf{I}_0$ is invariant under any motion of this body [30]. If $\mathbf{R}$ is the rotation matrix that transforms coordinates from frame $\{0\}$ to another rigidly attached frame $\{1\}$, i.e. $\mathbf{p}^1 = \mathbf{R}\mathbf{p}^0$, then the inertia tensor with respect to the new frame will be:

$$\mathbf{I}_1 = \mathbf{R}\mathbf{I}_0\mathbf{R}^T \tag{3.1}$$

Equation (3.1) is very useful when the reference frames used in our dynamic model does not match the coordinate systems defined in the SolidWorks design model. The mass properties tool by default outputs the inertia matrices in three forms:

- About the center of mass aligned with the principle axes of inertia.
- About the center of mass aligned with the output coordinate system.
- About the origin of and aligned with the output coordinate system.

For simplicity, the second form above was used. This enables us to use an output coordinate system placed at properly defined datums that do not move as the model is tweaked. As new parts are added the center of mass might change. Then we just calculate the center of masses again relative to these datums. The calculated inertia tensors were then transformed to the relevant coordinate frames using equation (3.1). For the main body, this datum is located at the leading edge of the wing, in the center of the lateral

axis. For tilt and twist, it is located at the end of the wing in the tilt rotation axis, on the right and left sides respectively.

The procedure used involves isolating the parts of each rigid body and then invoking the mass properties tool for each. To save time, some symmetry considerations were applied, more specifically symmetry about the xz-plane. Since the left and right tilt frames are parallel (the same applies to twist), the mass distribution is mirrored about the xz-plane going through the center of mass of each body. By inspecting the definition of the inertia tensor given by equation (3.2) [16], it is easy to see that this only leads to a sign flip of the products of inertia (off-diagonal terms) containing y. Similarly, the sign of the y-coordinate of the center of mass changes. Because of this, the procedure could be carried out for tilt and twist on the left side only and then flipping the signs of the relevant components to get properties of the right side.

$$\mathbf{I}_b = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} = \int_b \begin{bmatrix} y^2+z^2 & -xy & -xz \\ -xy & x^2+z^2 & -yz \\ -xz & -yz & x^2+y^2 \end{bmatrix} dm \tag{3.2}$$

Tables 3.1-3.3 lists the parameters gathered. Just to be clear, the inertia tensors are taken about each center of mass and aligned with the respective coordinate frames defined in chapter 2, while the coordinates of the centers of mass are relative to the listed reference points along the axes of the same mentioned frames.

3.2 Thruster Parameters

To measure key variables of the thruster system, a thrust stand and dynamometer from RCbenchmark [7] was used. Figure 3.1 shows the Series 1580 Thrust Stand and Dynamometer with propeller, motor and ESC mounted on a table for testing. The thrust stand measures thrust, torque, voltage, current, vibration and rotational speed. It provides a USB interface and software for calibration, data-logging and script automation. Load cells measures

Body	*Symbol*	*Mass* [kg]
Main body	m_b	2.5
Tilt right	m_{t_r}	0.477
Tilt left	m_{t_l}	0.477
Twist right	m_{tw_r}	0.914
Twist left	m_{tw_l}	0.914
Total	m_t	5.2820

Table 3.1: Mass of each rigid body

Body	*Center of mass* [m]			*Reference point*
Main body	$\begin{bmatrix} -0.086 & 0.000 & 0.029 \end{bmatrix}^T$			Middle of leading edge
Tilt right	$\begin{bmatrix} 0.003 & 0.040 & 0.011 \end{bmatrix}^T$			Tilt axis at right end of wing
Tilt left	$\begin{bmatrix} 0.003 & -0.040 & 0.011 \end{bmatrix}^T$			Tilt axis at left end of wing
Twist right	$\begin{bmatrix} 0.000 & 0.030 & -0.035 \end{bmatrix}^T$			Tilt axis at right end of wing
Twist left	$\begin{bmatrix} 0.000 & -0.030 & -0.035 \end{bmatrix}^T$			Tilt axis at left end of wing

Table 3.2: Center of mass for each rigid body

Body		I_{xx}	I_{yy}	I_{zz}	I_{xy}	I_{xz}	I_{yz}
Main body	$\mathbf{I}_b$	0.080000	0.022370	0.090500	0.000000	0.000000	0.000000
Tilt right	$\mathbf{I}_{t_r}$	0.000893	0.000861	0.000321	-0.000008	0.000009	-0.000045
Tilt left	$\mathbf{I}_{t_l}$	0.000893	0.000861	0.000321	0.000008	0.000009	0.000045
Twist right	$\mathbf{I}_{tw_r}$	0.001047	0.057189	0.056479	0.000000	0.000000	0.000094
Twist left	$\mathbf{I}_{tw_l}$	0.001047	0.057189	0.056479	0.000000	0.000000	-0.000094
Total	$\mathbf{I}_t$	1.084700	0.154300	1.211800	0.000000	-0.001300	0.000000

Table 3.3: Inertia tensor for each rigid body $[\,\mathrm{kg} \cdot \mathrm{m}^2]$

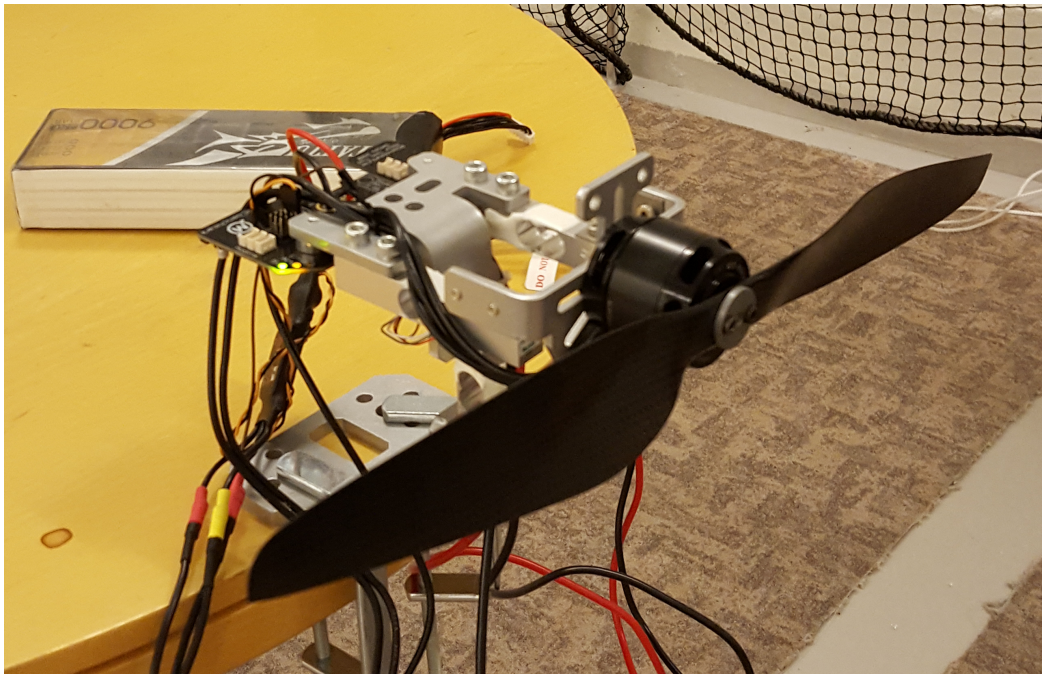


Figure 3.1: Thrust stand and dynamometer

	Manufacturer	*Model*	*Price*
Motor	T-Motor	U5 KV400	$125.9
Propeller	T-Motor	P15x5	$55.9/pair
ESC	T-Motor	AIR 40A	$39.99

Table 3.4: Components used in tests

thrust and torque, while rotational speed is measured electrically with a RPM probe soldered to one of the three phases output from the ESC. Table 3.4 lists the components used in the tests.

The RCbenchmark software stores logged data as CSV files which was imported to MATLAB for curve fitting. Figures 3.2 - 3.5 shows the resulting fitted curves plotted against measured data of propeller speed, torque, thrust, voltage and throttle. The measured data is fitted against the models (2.53) - (2.54). The current MiniPetrel prototype does not have ESCs with RPM control. Therefore a mapping from throttle and battery voltage to propeller speed is sought after, in the following form:

$$\Omega = f(\delta_t, V) \tag{3.3}$$

$$\delta_t = \frac{PWM - PWM_{min}}{PWM_{max} - PWM_{min}} \tag{3.4}$$

A good fit was achieved for the model

$$\Omega = f(\delta_t, V) = a(\delta_t V)^2 + b(\delta_t V) \tag{3.5}$$

Equation (3.5) is inverted in the control allocation implementation to convert speed commands to PWM outputs. All estimated parameters are summarized in table 3.5.

When running the propeller, an issue was noticed with the speed measurements at around 400 - 500 rad/s. These measurements were treated as outliers during curve fitting, and are quite evident in the presented figures.

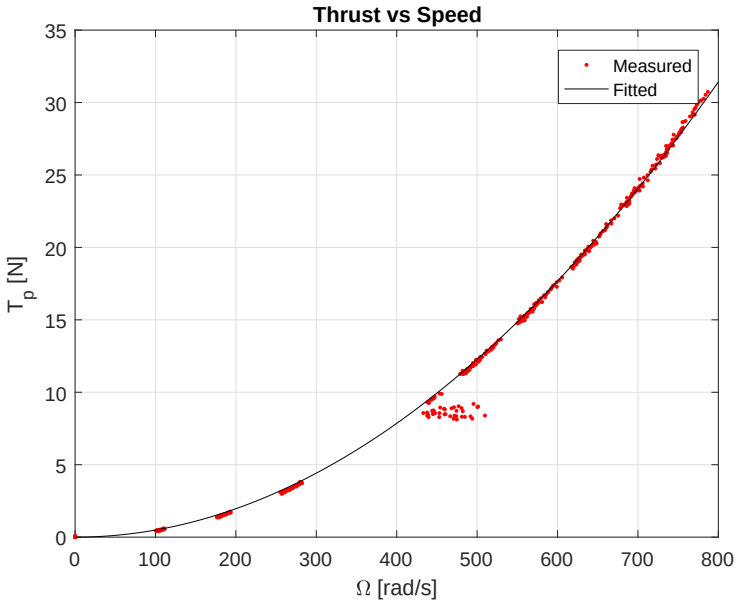


Figure 3.2: Thrust as a function of speed

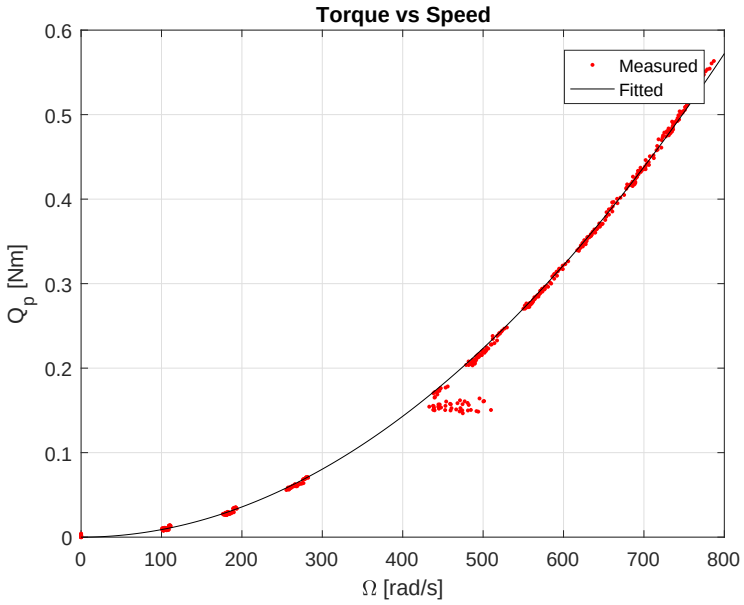


Figure 3.3: Torque as a function of speed

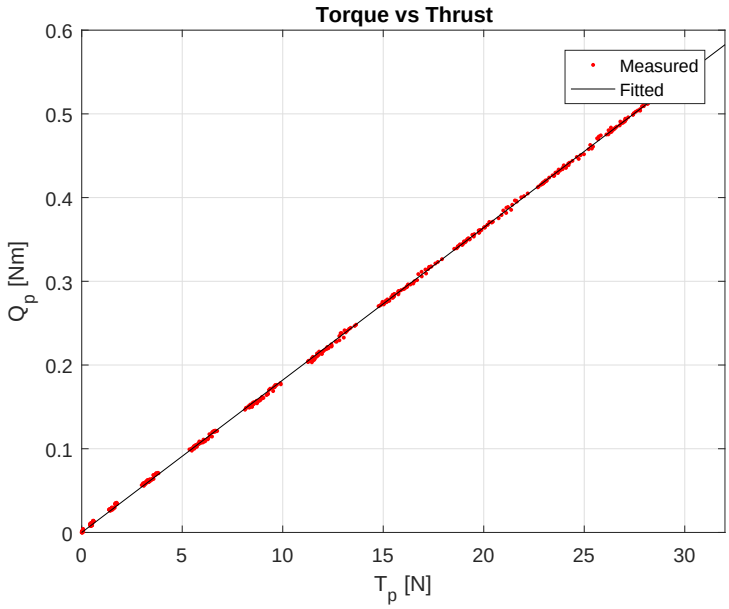


Figure 3.4: Torque as a function of thrust

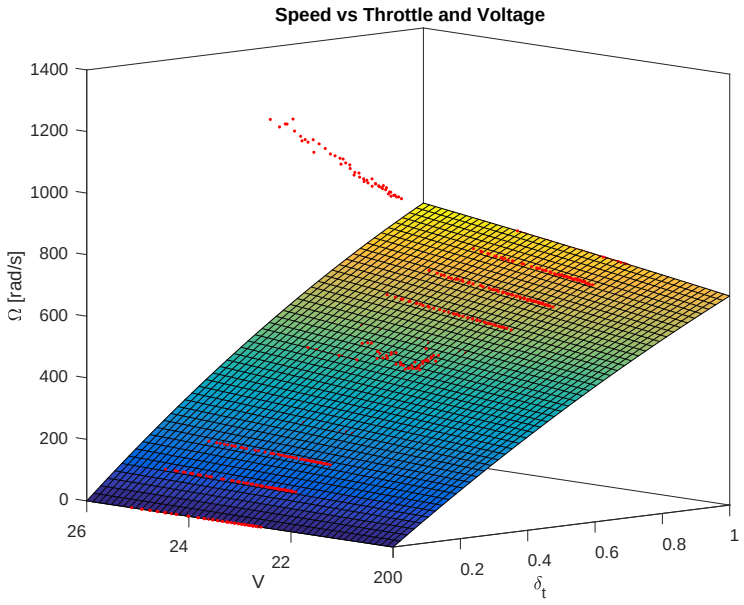


Figure 3.5: Speed as a function of throttle and voltage

Parameter	*Description*	*Value*	*Unit*
k	Thrust constant	$4.91e-5$	$\mathrm{kg}\cdot\mathrm{m}$
d	Torque constant	$8.94e-7$	$\mathrm{kg}\cdot\mathrm{m}^2$
c	Torque to thrust ratio	0.01821	m
a	Quadratic speed gain	-0.3321	N/A
b	Linear speed gain	40.6	N/A

Table 3.5: Thrust parameters

Throttle	*Speed* [rad/s]	*Thrust* [N]	*Voltage* [V]
0.50	439.82	9.797	22.2
0.65	544.54	14.61	22.2
0.75	596.90	18.63	22.2
0.85	649.26	21.77	22.2
1.00	680.68	24.32	22.2

Table 3.6: Supplier test data

3.2.1 Comparison With Supplier Test Data

Table 3.6 shows test data provided by the supplier [11] for the same combination of motor, ESC and propeller. This data is useful to verify the experimental results, and is plotted against the fitted data in figure 3.6, which shows a relatively good match. No torque data was available on the T-motor website for the components tested.

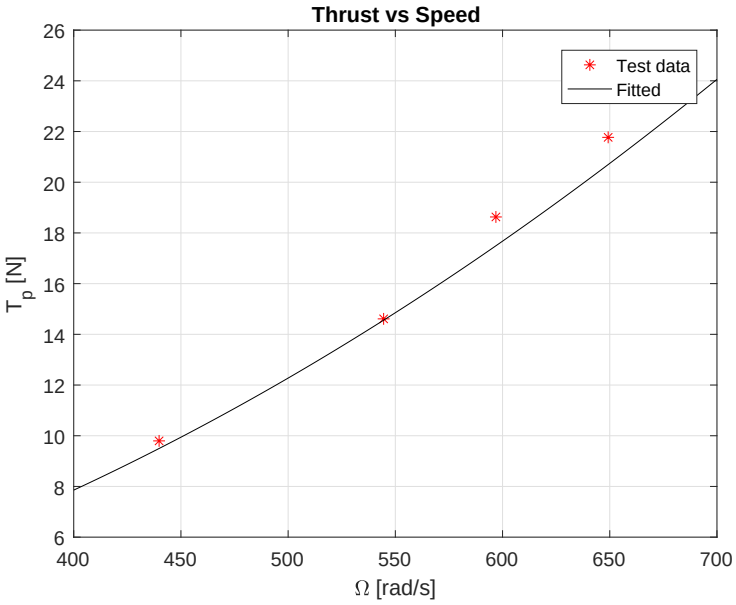


Figure 3.6: Comparison with supplier test data

4 Control System Design

A basic overview of the StormPetrel control system is shown in figure 4.1. Velocity and attitude setpoints are generated by either manual control or a guidance system for e.g. waypoint and path following. Based on these setpoints, and navigation data, velocity and attitude controllers generate force and moment commands that are converted into actuator commands in the control allocation module.

A control system for the StormPetrel UAV has been developed in detail in [21], including detailed simulation studies and performance analysis. This chapter presents the main parts of the StormPetrel control system relevant for hover control, including controllers for attitude and velocity, as well as methods for control allocation. Control allocation is given the most attention, and some modifications are proposed.

4.1 Attitude Control

The following quaternion attitude controller is from [17] with an added integral term. It is used in the Matlab simulations discussed in chapter 5.

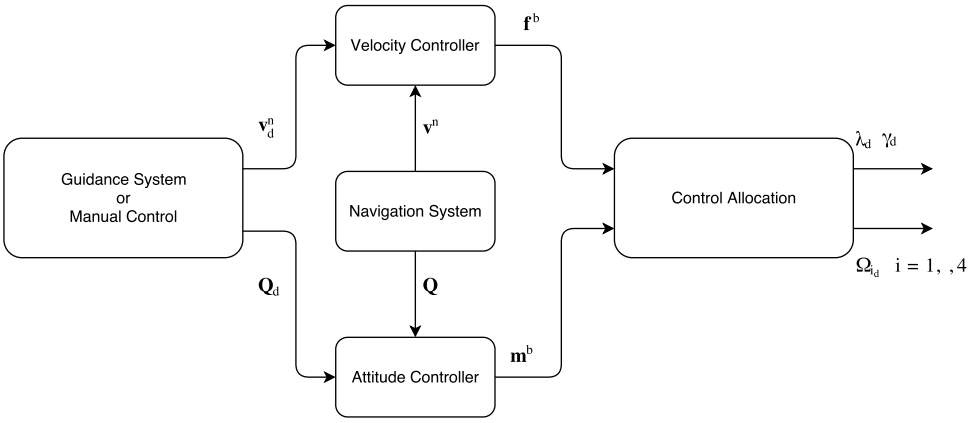


Figure 4.1: Block diagram of StormPetrel control system

$$\mathbf{m}^b = -\mathbf{K}_{p_Q}\tilde{\boldsymbol{\epsilon}} - \mathbf{K}_{d_Q}\boldsymbol{\omega} - \mathbf{K}_{i_Q}\int_0^t \tilde{\boldsymbol{\epsilon}}(\tau)d\tau \tag{4.1}$$

$$\tilde{\boldsymbol{\epsilon}} = \begin{bmatrix}\tilde{\epsilon}_1 & \tilde{\epsilon}_2 & \tilde{\epsilon}_3\end{bmatrix}^T \tag{4.2}$$

$$\tilde{\mathbf{Q}} = \mathbf{Q} \otimes \mathbf{Q}_d^* = \begin{bmatrix}\tilde{\eta} & \tilde{\epsilon}_1 & \tilde{\epsilon}_2 & \tilde{\epsilon}_3\end{bmatrix}^T \tag{4.3}$$

$\otimes$ and $*$ denotes quaternion multiplication and conjugate respectively.

4.2 Velocity Control

The velocity controller used is a PI controller with gravity compensation formulated in $\{n\}$:

$$\mathbf{f}^n = -\begin{bmatrix}0\\0\\m_t g\end{bmatrix} - \mathbf{K}_{p_v}\mathbf{e}_v^n - \mathbf{K}_{i_v}\int_0^t \mathbf{e}_v^n(\tau)d\tau \tag{4.4}$$

$$\mathbf{e}_v^n = \mathbf{v}_b^n - \mathbf{v}_d^n \tag{4.5}$$

The resulting control forces can be rotated to the body frame using the following transformation.

$$\mathbf{f}^b = (\mathbf{R}_b^n)^T \mathbf{f}^n \tag{4.6}$$

This controller is applied in both the simulations presented in chapter 5, and the flight tests discussed in chapter 8.

4.3 Control Allocation

Figure 4.2 shows a block diagram of the control allocation module. The commanded forces and moments output from the controllers are converted into actuator outputs, which include desired propeller speeds, tilt and twist angles. The diagram also shows how saturation statuses can be fed back to the controllers for use in e.g. anti wind-up schemes. The battery status is also shown as an input. With the battery voltage available, (3.5) can be inverted to calculate PWM outputs. This is applied in the flight tests of chapter 8. The simulation model used for chapter 5 however, uses the speed-regulated thruster models presented in chapter 2.

A comprehensive survey on control allocation methods, including optimization-based and fault-tolerant methods, is given in [22]. Adaptive methods are discussed in [31].

4.3.1 Main Algorithm

The main algorithm, first presented in [21] is given below. To simplify, tilt and twist angles are kept equal on both sides. Originally, the tilt coordinate frame was defined such that $\lambda = 0$ during forward flight, with λ increasing when tilting backwards (positive rotation about body y-axis). To reflect the changes in coordinate frames, introduced in chapter 2, the control allocation equations have been updated accordingly.

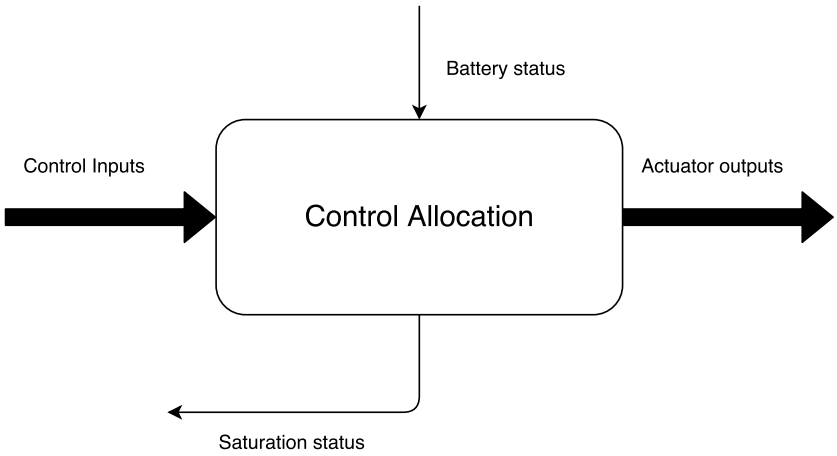


Figure 4.2: Control allocation block diagram

The propeller forces, decomposed in the body frame is given by:

$$\mathbf{f}^b = \mathbf{R}_{tw}^b(\lambda, \gamma) \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} \sum_{i=1}^{4} T_{p_i} \tag{4.7}$$

The magnitudes of the force in the xz-plane and the total force vector can be calculated as follows.

$$F_{xz} = \sqrt{F_x^2 + F_z^2} \tag{4.8}$$

$$F_t = \sqrt{F_x^2 + F_y^2 + F_z^2} = \sqrt{F_{xz}^2 + F_y^2} = \sum_{i=1}^{4} T_{p_i} \tag{4.9}$$

Expanding (4.7) gives:

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \begin{bmatrix} \cos(\gamma)\sin(\lambda) \\ \sin(\gamma) \\ -\cos(\lambda)\cos(\gamma) \end{bmatrix} F_t \tag{4.10}$$

Dividing F_x by F_z gives:

$$\frac{F_x}{F_z} = \frac{\cos(\gamma)\sin(\lambda)F_t}{-\cos(\lambda)\cos(\gamma)F_t} = \frac{\sin(\lambda)}{-\cos(\lambda)} \implies \lambda_d = \operatorname{atan2}(F_x, -F_z) \tag{4.11}$$

Substituting (4.10) into (4.8) gives

$$F_{xz} = \sqrt{F_x^2 + F_z^2} = \sqrt{\cos^2(\gamma)\sin^2(\lambda)F_t^2 + \cos^2(\lambda)\cos^2(\gamma)F_t^2} \tag{4.12}$$

$$= \sqrt{\cos^2(\gamma)\left(\sin^2(\lambda) + \cos^2(\lambda)\right)F_t^2} = \sqrt{\cos^2(\gamma)F_t^2} = \cos(\gamma)F_t \tag{4.13}$$

Furthermore:

$$\frac{F_y}{F_{xz}} = \frac{\sin(\gamma)F_t}{\cos(\gamma)F_t} = \tan(\gamma) \implies \gamma_d = \text{atan2}\left(F_y, F_{xz}\right) \tag{4.14}$$

The calculation of desired tilt and twist angles based on force commands can be summarized as:

$$\lambda_d = \text{atan2}\left(F_x, -F_z\right) \tag{4.15}$$

$$\gamma_d = \text{atan2}\left(F_y, F_{xz}\right) \tag{4.16}$$

The moment commands due to propeller thrust and torque differencing are:

$$\mathbf{m}^b = \sum_{i=1}^{4}\left[\mathbf{r}_{p_i}^b(\lambda) \times \mathbf{R}_{tw}^b\left(\lambda,\gamma\right)\mathbf{f}_{p_i}^{tw} + \mathbf{R}_{tw}^b\left(\lambda,\gamma\right)\mathbf{m}_{m_i}^{tw}\right] \tag{4.17}$$

$$= \sum_{i=1}^{4}\left[\mathbf{r}_{p_i}^b(\lambda) \times \mathbf{R}_{tw}^b\left(\lambda,\gamma\right)\mathbf{f}_{p_i}^{tw} + \mathbf{R}_{tw}^b\left(\lambda,\gamma\right)c_i\mathbf{f}_{p_i}^{tw}\right]$$

$$= \sum_{i=1}^{4}\left[T_{p_i}\underbrace{\left(\mathbf{r}_{p_i}^b(\lambda) \times \mathbf{R}_{tw}^b\left(\lambda,\gamma\right)\begin{bmatrix}0\\0\\-1\end{bmatrix} + \mathbf{R}_{tw}^b\left(\lambda,\gamma\right)c_i\begin{bmatrix}0\\0\\-1\end{bmatrix}\right)}_{\mathbf{a}_i(\lambda,\gamma)}\right]$$

$$= \sum_{i=1}^{4}\left[T_{p_i}\mathbf{a}_i\left(\lambda,\gamma\right)\right]$$

with

$$\mathbf{a}_i\left(\lambda,\gamma\right) = \begin{bmatrix}a_{x_i}\\a_{y_i}\\a_{z_i}\end{bmatrix} \tag{4.18}$$

and

$$c_i = -c \qquad i = 1, 4 \tag{4.19}$$

$$c_i = \quad c \qquad i = 2, 3 \tag{4.20}$$

Combining the previous equations for total force and moments, we get:

$$\begin{bmatrix} F_t \\ M_x \\ M_y \\ M_z \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 1 & 1 & 1 \\ a_{x_1} & a_{x_2} & a_{x_3} & a_{x_4} \\ a_{y_1} & a_{y_2} & a_{y_3} & a_{y_4} \\ a_{z_1} & a_{z_2} & a_{z_3} & a_{z_4} \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} T_{p_1} \\ T_{p_2} \\ T_{p_3} \\ T_{p_4} \end{bmatrix} \implies \begin{bmatrix} T_{p_1} \\ T_{p_2} \\ T_{p_3} \\ T_{p_4} \end{bmatrix} = \mathbf{A}^{-1} \begin{bmatrix} F_t \\ M_x \\ M_y \\ M_z \end{bmatrix} \tag{4.21}$$

From this the desired propeller speeds can be calculated according to:

$$\Omega_i = -\sqrt{\frac{T_{p_i}}{k}} \qquad i = 1, 4 \tag{4.22}$$

$$\Omega_i = \sqrt{\frac{T_{p_i}}{k}} \qquad i = 2, 3 \tag{4.23}$$

4.3.2 Constraints

To provide sane input and output values, as well as avoiding a negative number under the square root when calculating propeller speed, the following constraints are used:

$$F_x = \begin{cases} \text{sgn}(F_{x_d}) F_{x_{LIM}} & \text{if } |F_{x_d}| > F_{x_{LIM}} \\ F_{x_d} & \text{otherwise} \end{cases} \tag{4.24}$$

$$F_y = \begin{cases} \text{sgn}(F_{y_d}) F_{y_{LIM}} & \text{if } |F_{y_d}| > F_{y_{LIM}} \\ F_{y_d} & \text{otherwise} \end{cases} \tag{4.25}$$

$$F_z = \begin{cases} 0 & \text{if } F_{z_d} > 0 \\ -F_{z_{LIM}} & \text{if } F_{z_d} < -F_{z_{LIM}} \\ F_{z_d} & \text{otherwise} \end{cases} \tag{4.26}$$

$$M_x = \begin{cases} \text{sgn}(M_{x_d}) M_{x_{LIM}} & \text{if } |M_{x_d}| > M_{x_{LIM}} \\ M_{x_d} & \text{otherwise} \end{cases} \tag{4.27}$$

$$M_y = \begin{cases} \text{sgn}(M_{y_d}) M_{y_{LIM}} & \text{if } |M_{y_d}| > M_{y_{LIM}} \\ M_{y_d} & \text{otherwise} \end{cases} \tag{4.28}$$

$$M_z = \begin{cases} \text{sgn}(M_{z_d}) M_{z_{LIM}} & \text{if } |M_{z_d}| > M_{z_{LIM}} \\ M_{z_d} & \text{otherwise} \end{cases} \tag{4.29}$$

$$F_{xz} = \begin{cases} F_{xz_{MIN}} & \text{if } F_{xz_d} < F_{xz_{MIN}} \\ F_{xz_d} & \text{otherwise} \end{cases} \tag{4.30}$$

$$\lambda = \begin{cases} \lambda_{MAX} & \text{if } \lambda_d > \lambda_{MAX} \\ \lambda_{MIN} & \text{if } \lambda_d < \lambda_{MIN} \\ \lambda_d & \text{otherwise} \end{cases} \tag{4.31}$$

$$\gamma = \begin{cases} \gamma_{MAX} & \text{if } \gamma_d > \gamma_{MAX} \\ \gamma_{MIN} & \text{if } \gamma_d < \gamma_{MIN} \\ \gamma_d & \text{otherwise} \end{cases} \tag{4.32}$$

$$T_{p_{MAX}} = k\Omega^2_{MAX} \tag{4.33}$$

$$F_t = \begin{cases} F_{t_{MAX}} & \text{if } F_{t_d} > F_{t_{MAX}} \\ F_{t_d} & \text{otherwise} \end{cases} \tag{4.34}$$

$$F_{t_{MAX}} = 4T_{p_{MAX}} \tag{4.35}$$

$$T_p = \begin{cases} 0 & \text{if } T_{p_d} < 0 \\ T_{p_d} & \text{otherwise} \end{cases} \tag{4.36}$$

4.3.3 Optimization Based Formulation

An alternative to the simple method above is to formulate an optimization problem. The advantage of this is that constraints on propeller speeds can be incorporated into the problem formulation, and balanced moments can be achieved. This is not always the case if some of the propellers reaches their minimum or maximum speed. To solve this, slack variables are introduced, that allow to boost, or reduce the total thrust a little bit to ensure that commanded moments are realized. Inequality constraints on the slack variables can be introduced to limit the allowed thrust boost/reduction.

To simplify the problem, pre-calculated tilt and twist angles are used, as calculated above, while the propeller speeds are calculated from propeller thrust as before. The following quadratic program (QP) is proposed:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{Q}_x\mathbf{x} + \mathbf{c}_x^T\mathbf{x} + \frac{1}{2}\mathbf{s}^T\mathbf{Q}_s\mathbf{s} + \mathbf{c}_s^T\mathbf{s} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} = \mathbf{a} + \mathbf{s} \\ & \mathbf{c} \geq \mathbf{x} \geq \mathbf{b} \geq \mathbf{0} \\ & \mathbf{e} \geq \mathbf{s} \geq \mathbf{d} \end{aligned} \tag{4.37}$$

$\mathbf{s}$ are slack variables for realized forces and moments. $\mathbf{Q}_s$ can be used to prioritize roll and pitch over yaw and total thrust, as this is important to stabilize the vehicle.

$$\mathbf{x} = \begin{bmatrix} T_{p_1} & T_{p_2} & T_{p_3} & T_{p_4} \end{bmatrix}^T \tag{4.38}$$

$$\mathbf{a} = \begin{bmatrix} F_t & M_x & M_y & M_z \end{bmatrix}^T \tag{4.39}$$

$$\mathbf{b} = \begin{bmatrix} T_{p_{MIN}} & T_{p_{MIN}} & T_{p_{MIN}} & T_{p_{MIN}} \end{bmatrix}^T \tag{4.40}$$

$$\mathbf{c} = \begin{bmatrix} T_{p_{MAX}} & T_{p_{MAX}} & T_{p_{MAX}} & T_{p_{MAX}} \end{bmatrix}^T \tag{4.41}$$

$$\mathbf{d} = \begin{bmatrix} -\text{max reduction} & -\infty & -\infty & -\infty \end{bmatrix}^T \tag{4.42}$$

$$\mathbf{e} = \begin{bmatrix} \text{max increase} & \infty & \infty & \infty \end{bmatrix}^T \tag{4.43}$$

This method is used in the simulator implementation and simulation results shown in chapter 5. Details on how to convert this to standard form ready for solving using the Matlab function quadprog, can be seen in the file "controlalloc.m" in the digital appendix. The first step is to gather the propeller thrusts and slack variables in a common vector:

$$\mathbf{z} = \begin{bmatrix} \mathbf{x}^T & \mathbf{s}^T \end{bmatrix}^T = \begin{bmatrix} T_{P_1} & T_{P_2} & T_{P_3} & T_{P_4} & s_{F_T} & s_{M_x} & s_{M_y} & s_{M_z} \end{bmatrix}^T \tag{4.44}$$

and then reformulate the problem in terms of this.

A detailed treatment on numerical optimization, including quadratic programming, can be found in [25].

5 Simulation Results

The simulation model developed in chapter 2 has been implemented in Matlab and Simulink. As a first step to verify the derived model, and the alternative control allocation scheme proposed in chapter 4, this chapter briefly presents some simulation results and discusses some aspects related to the implementation.

5.1 Simulator Implementation

A simulator is implemented in Matlab/Simulink based on the Euler-Lagrange equations of motion derived in chapter 2, the identified inertial and thruster parameters from chapter 3, and the flight control system presented in chapter 4. Most of the remaining unknown parameters are reused from [21]. Matlab function files for the Euler-Lagrange system matrices are generated from the symbolic expressions and used in the simulator. Better performance and a more efficient simulation should be strived for in the future. The current implementation is very demanding due to the complicated kinematic terms introduced by Euler angles as generalized coordinates. This is especially true for the coriolis and centrifugal matrix $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$. Symbolic simplification, optimized code generation and compilation were all attempted unsuccessfully. The slow simulation time makes it difficult to tune controllers and

improve performance, as well as doing dynamic exploration of the system. However, simulation results showing attitude and velocity command tracking are presented in the next section, thus verifying both the proposed model and optimization-based control allocation scheme to some extent. Further verification of the model should be done in the future.

The implemented Simulink models, Matlab init scripts, control allocation function, symbolic Euler-Lagrange matrices and full parameter list can be found in the attached digital appendix. The solver used for simulation is the ode45 variable step solver. No wind disturbances were applied during simulation, and first order low-pass filters were used as reference filters.

5.2 Velocity Response

Figures 5.1 - 5.14 show the response to a series of steps in desired NED velocities. Figure 5.1 shows how the vehicle velocities converge to the setpoints. Figure 5.8 shows how the vehicle attitude is stabilized during the maneuver. Plots of other relevant variables, including force commands, tilt and twist angles and propeller speeds are also included for reference.

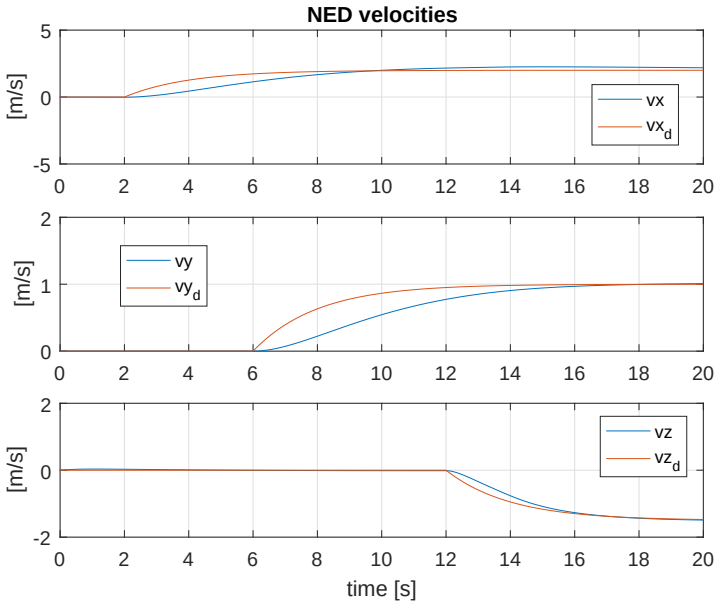


Figure 5.1: NED velocities

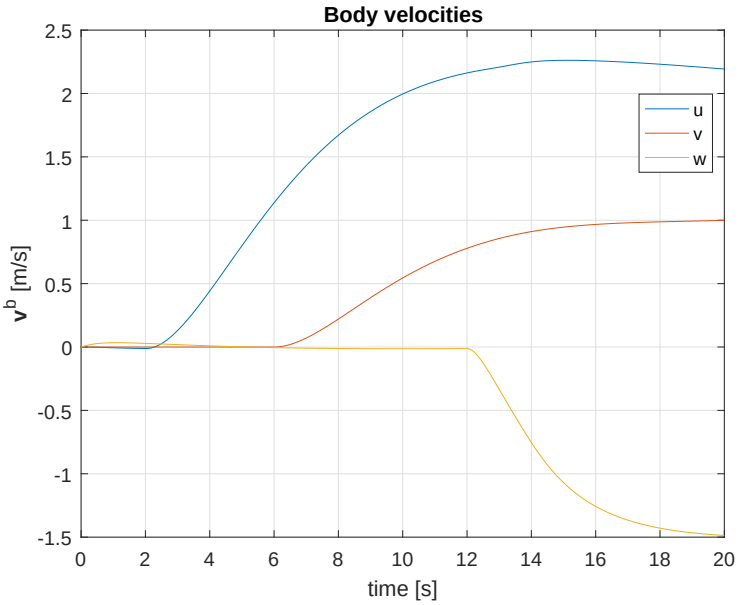


Figure 5.2: Body velocities

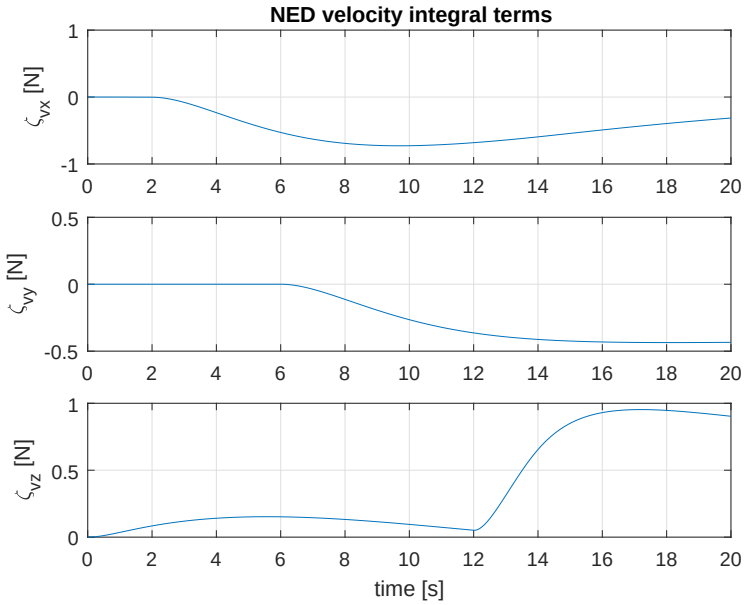


Figure 5.3: Integrals of NED velocity errors

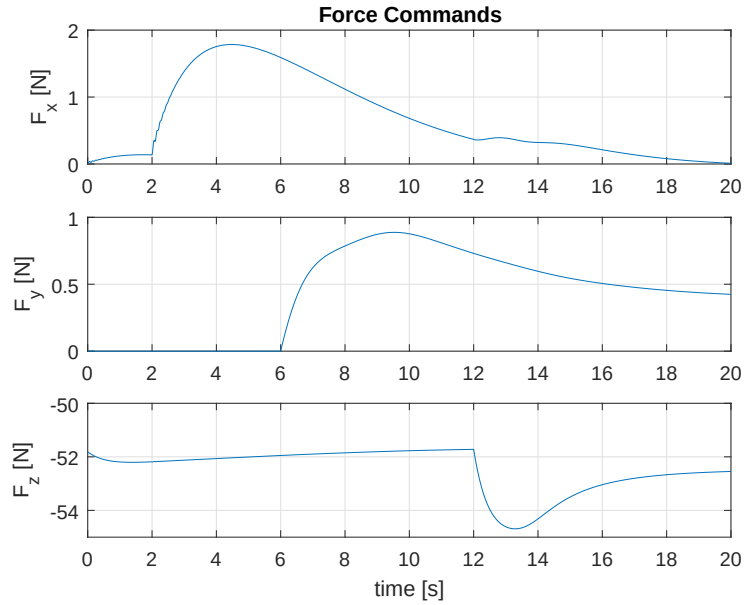


Figure 5.4: Force commands

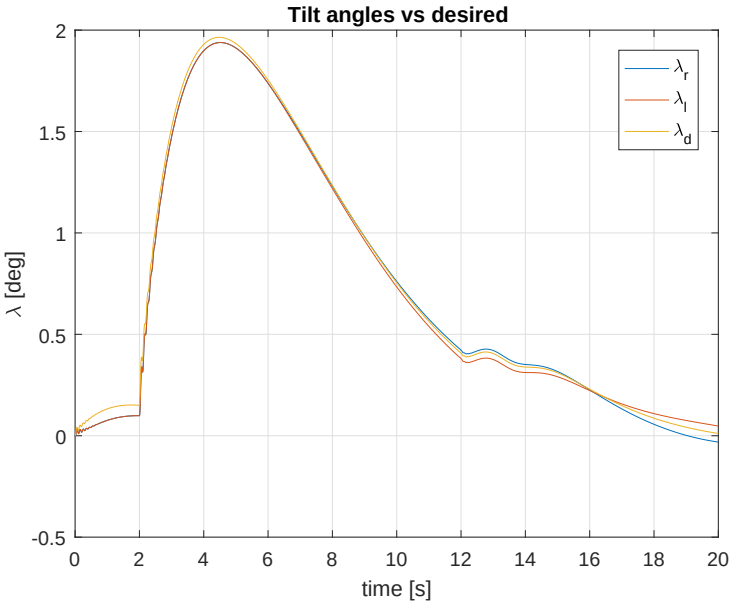


Figure 5.5: Tilt angle

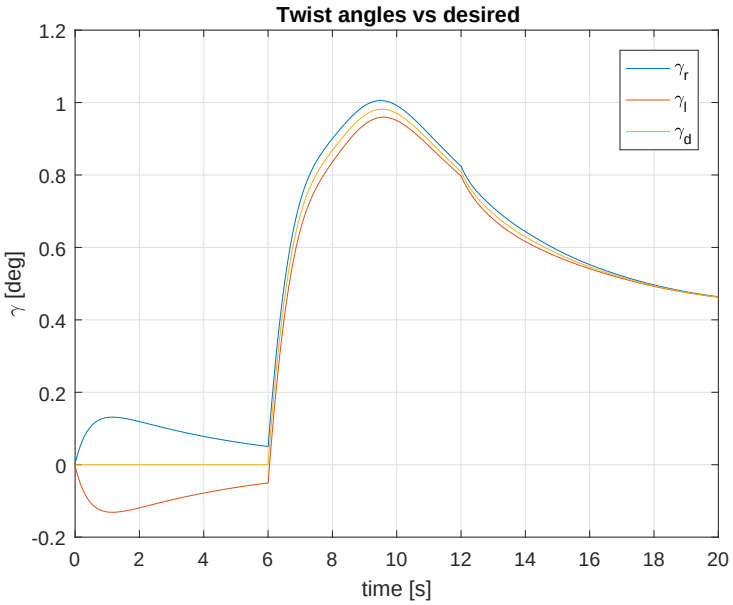


Figure 5.6: Twist angle

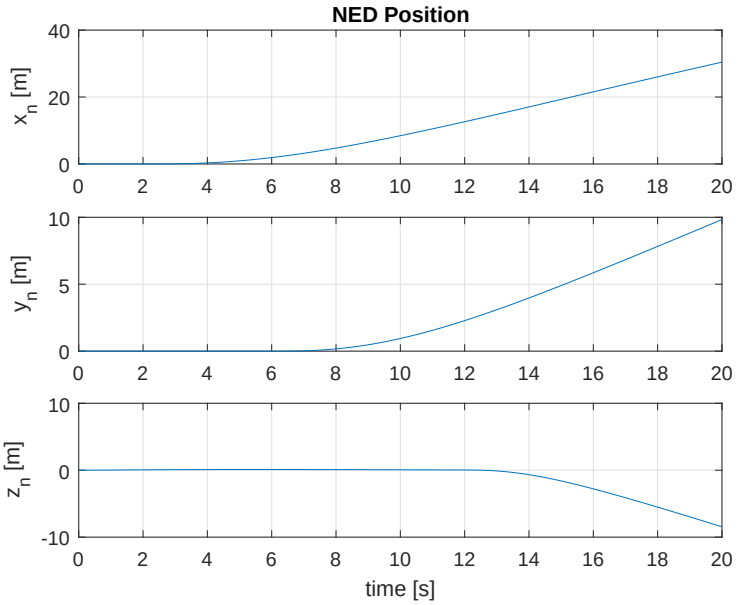


Figure 5.7: NED position

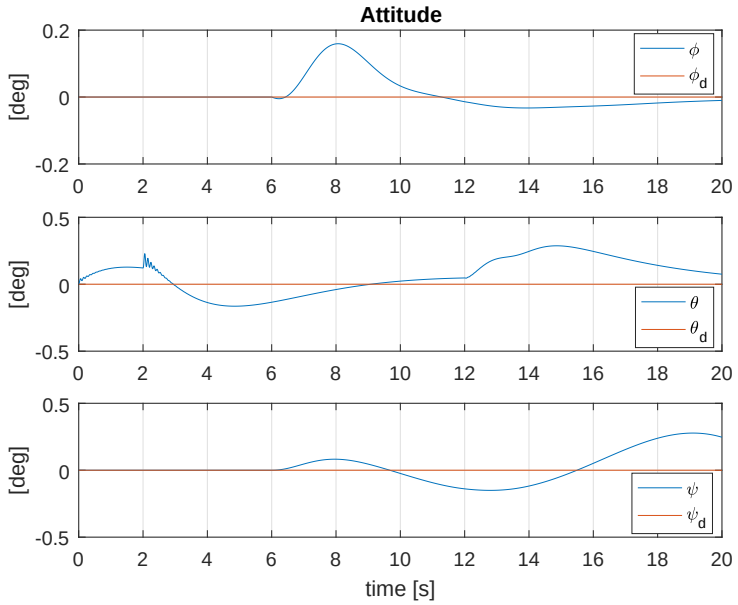


Figure 5.8: Attitude

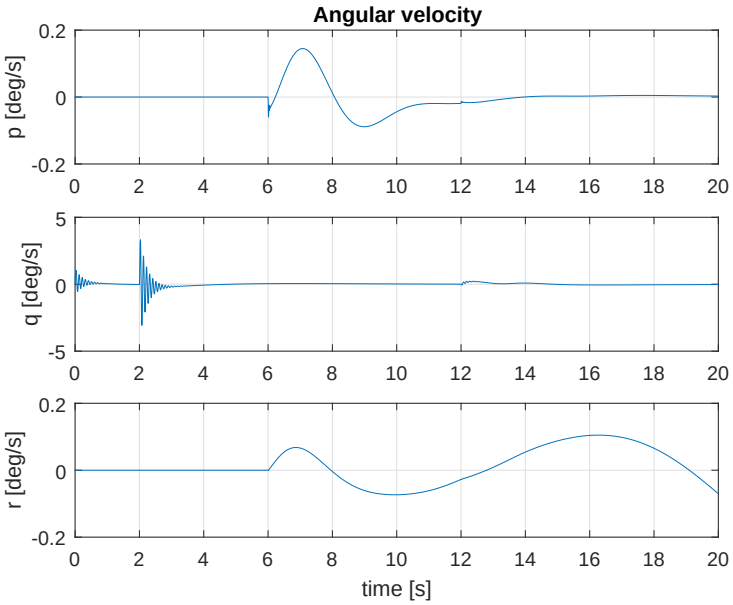


Figure 5.9: Angular velocities

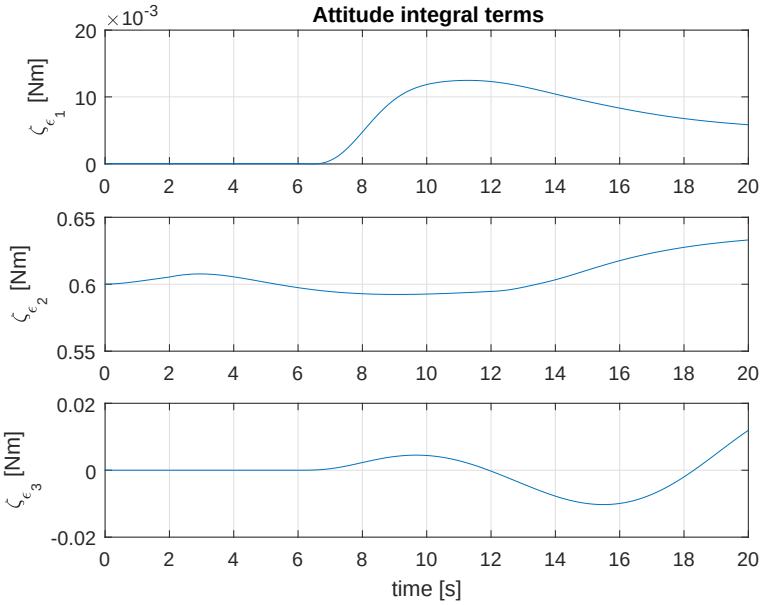


Figure 5.10: Integrals of attitude error

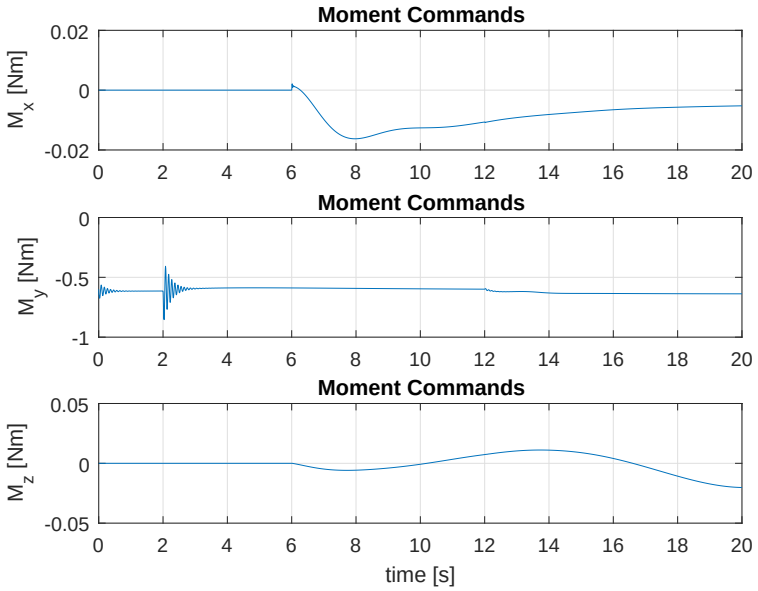


Figure 5.11: Moment commands

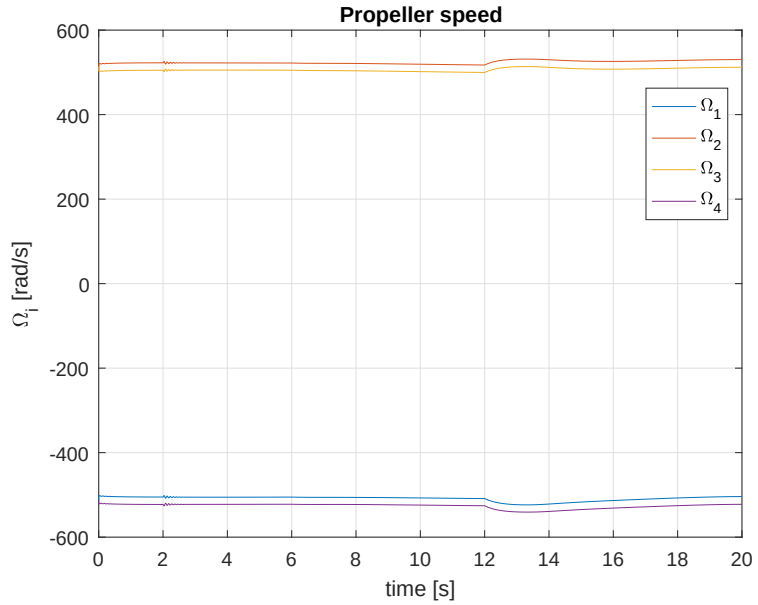


Figure 5.12: Propeller speed

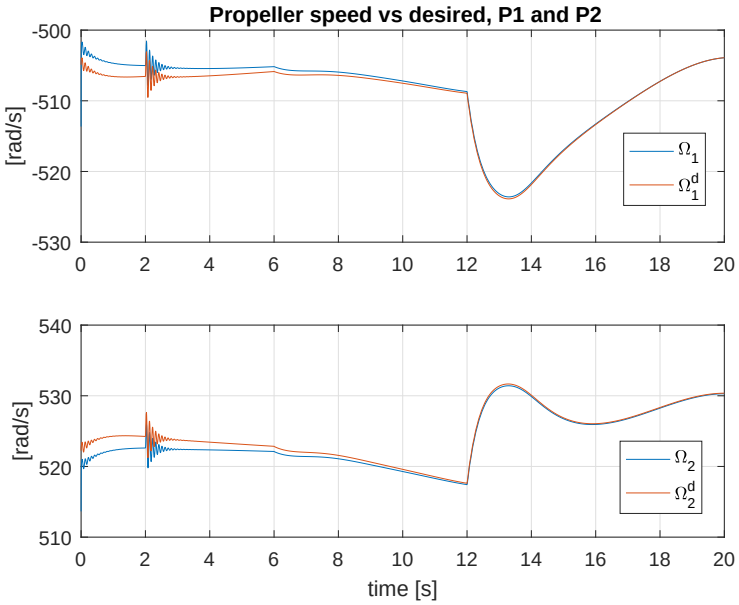


Figure 5.13: Propeller speed, propellers 1 and 2

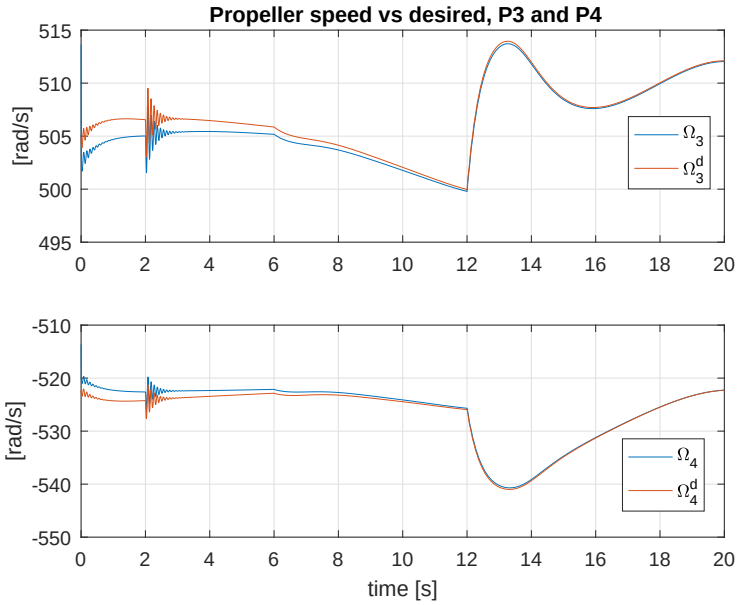


Figure 5.14: Propeller speed, propellers 3 and 4

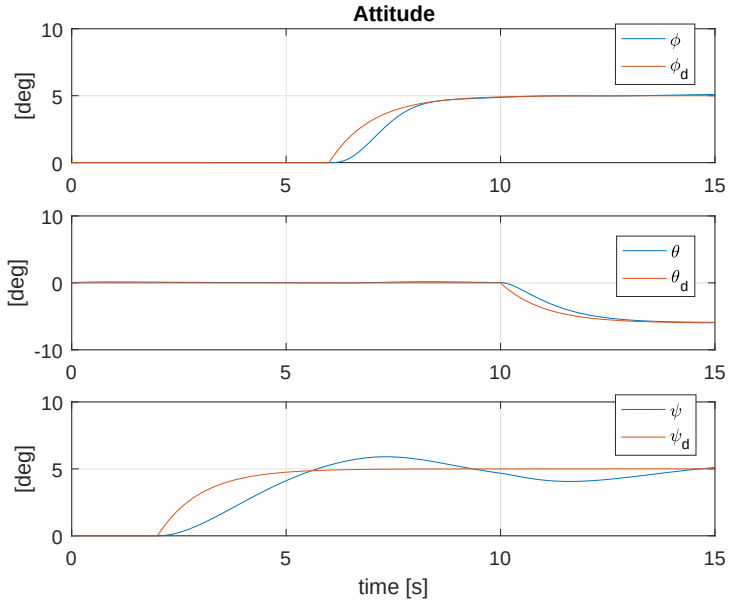


Figure 5.15: Attitude

5.3 Attitude Response

Figures 5.15 - 5.28 show the response to a series of steps in desired attitude. Figure 5.15 show that the vehicle attitude converges to the desired attitude. Figure 5.19 shows that the vehicle velocity is stabilized during the maneuver. Plots of other relevant variables, including force and moment commands, tilt and twist angles and propeller speeds are included for reference.

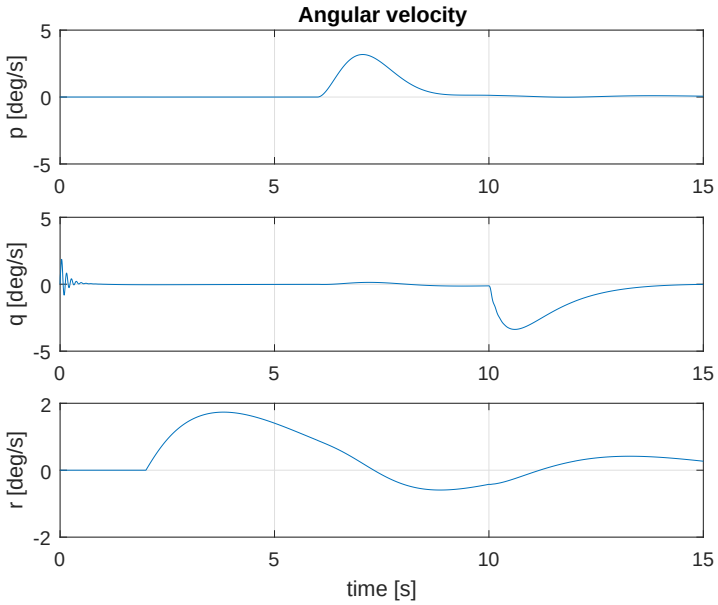


Figure 5.16: Angular velocities

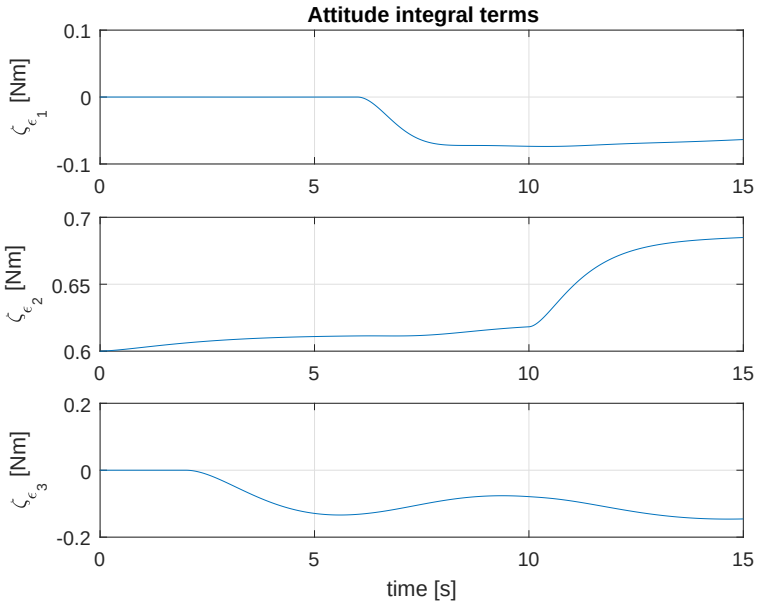


Figure 5.17: Integrals of attitude error

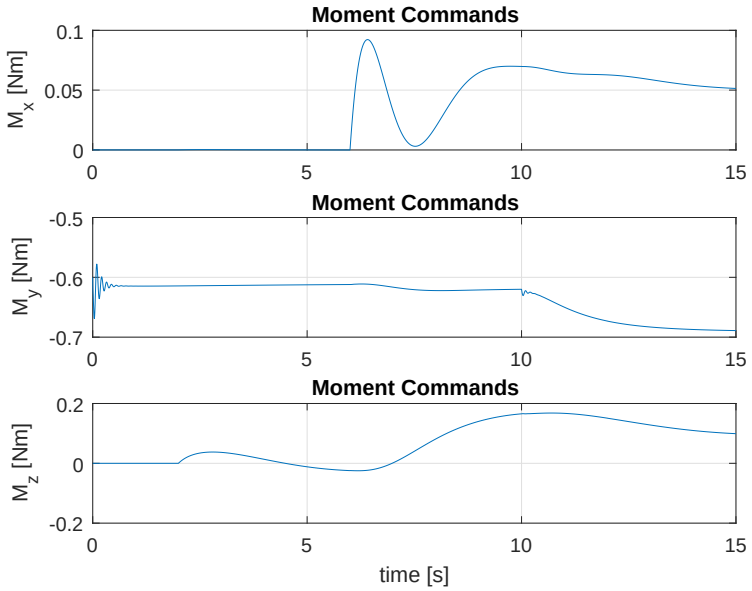


Figure 5.18: Moment commands

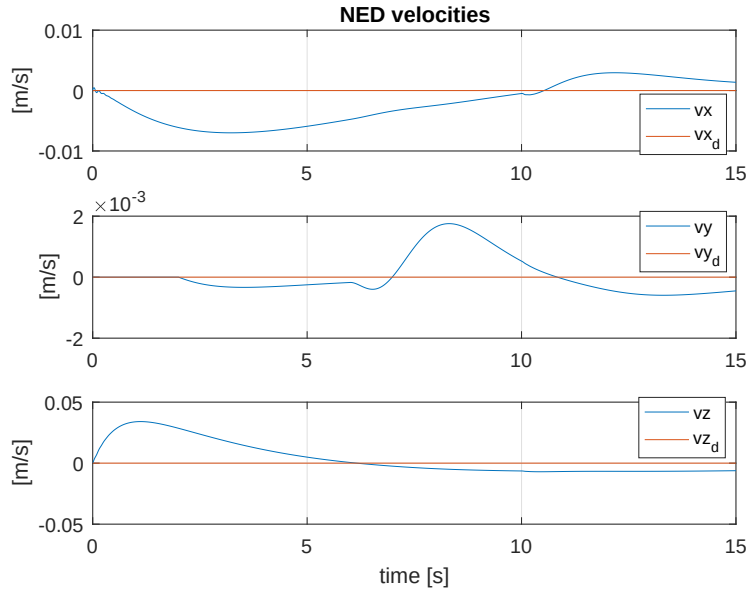


Figure 5.19: NED velocities

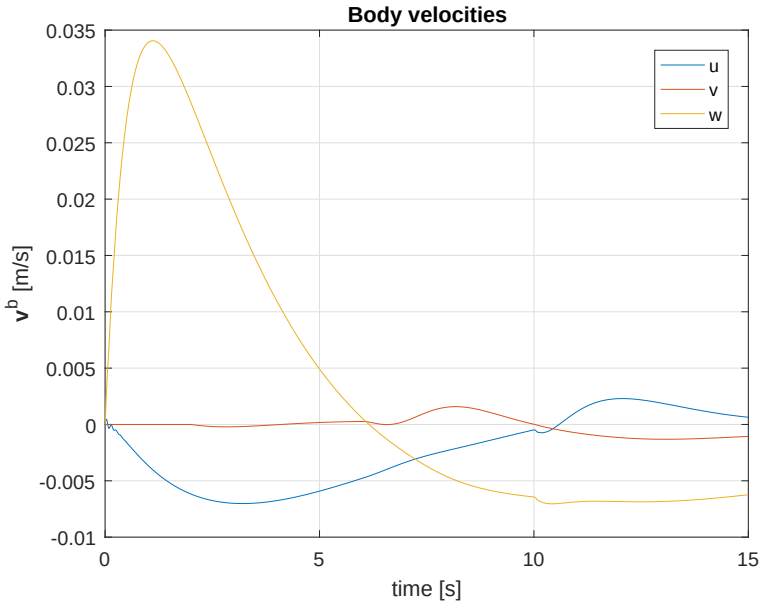


Figure 5.20: Body velocities

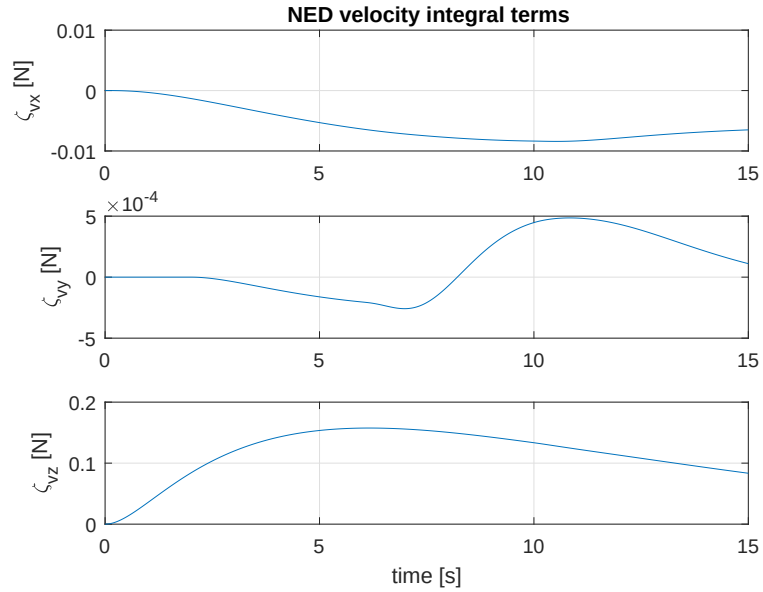


Figure 5.21: Integrals of NED velocity errors

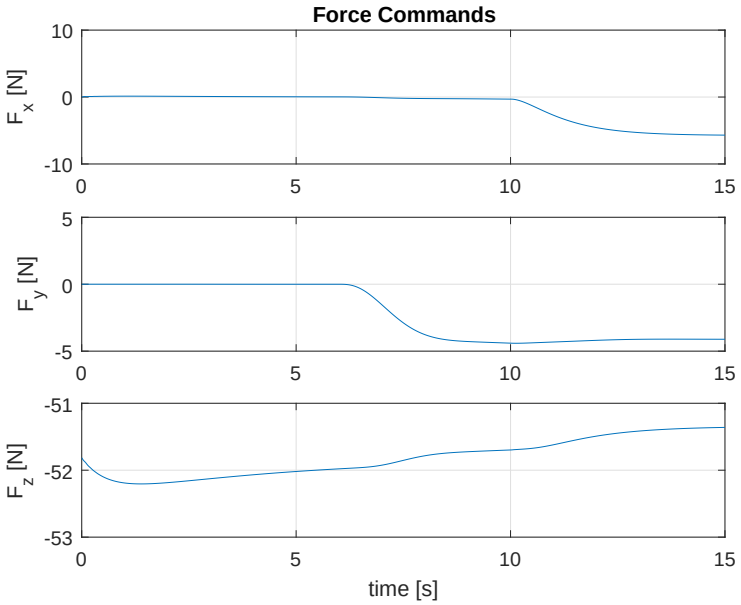


Figure 5.22: Force commands

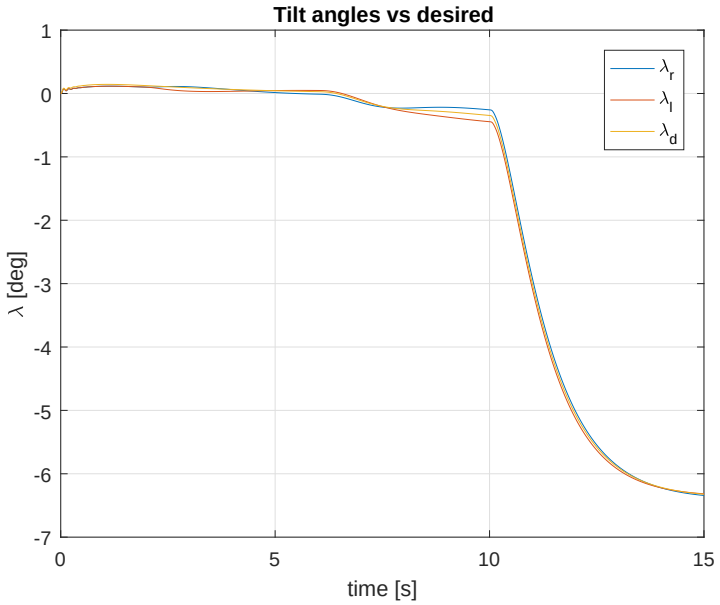


Figure 5.23: Tilt angle

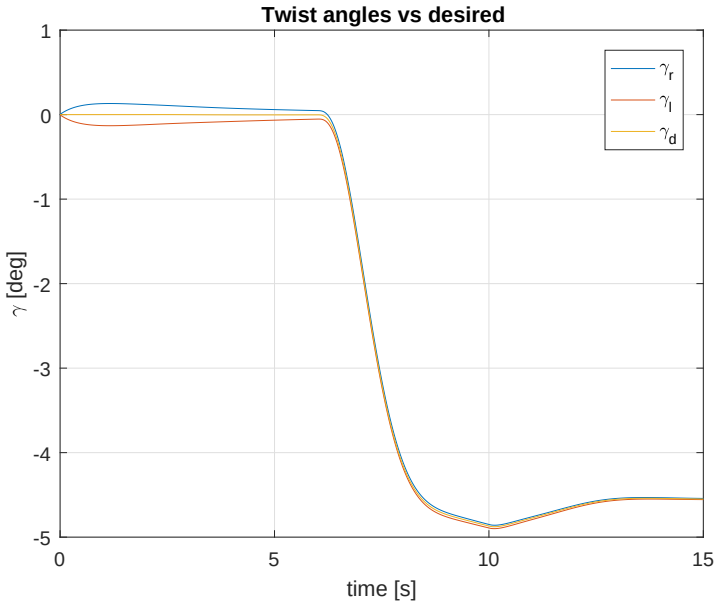


Figure 5.24: Twist angle

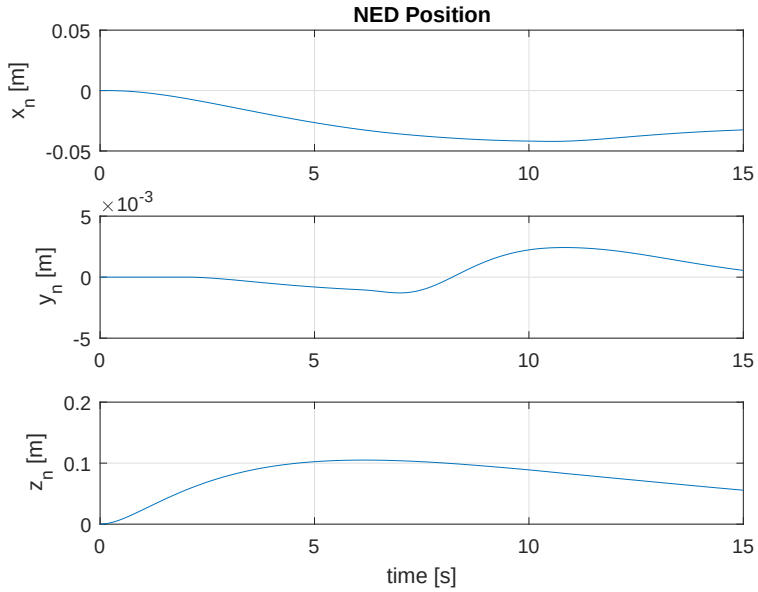


Figure 5.25: NED position

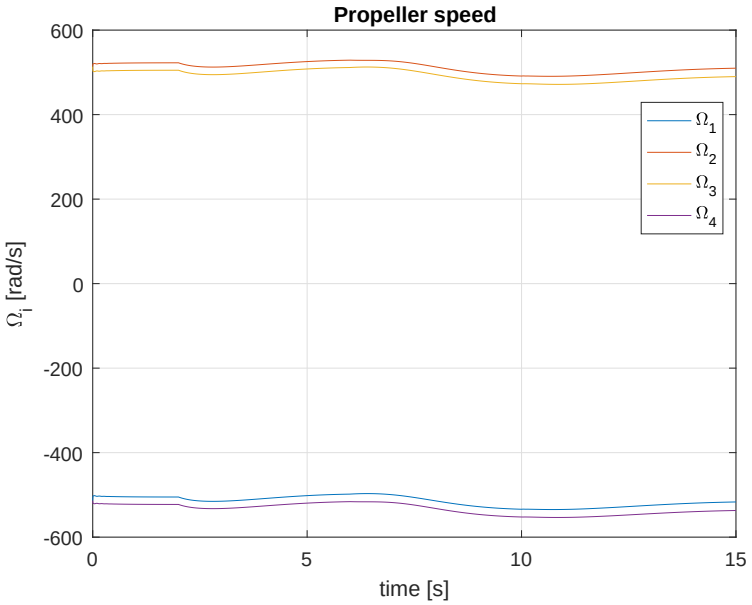


Figure 5.26: Propeller speed

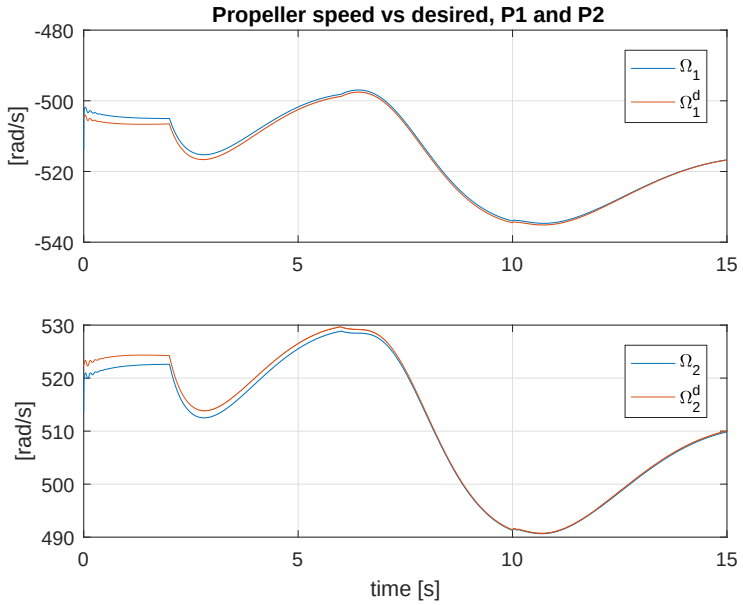


Figure 5.27: Propeller speed, propellers 1 and 2

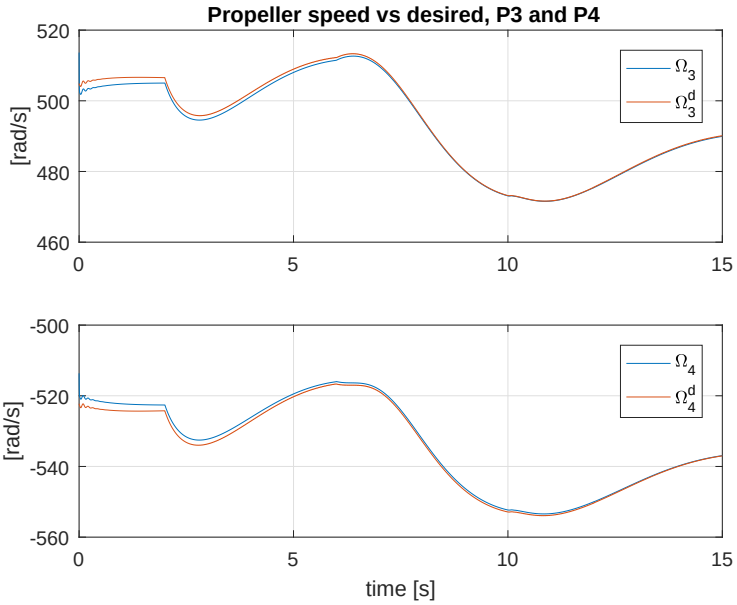


Figure 5.28: Propeller speed, propellers 3 and 4

Part II

Development of Test Platform

6 Implementation in the PX4 Flight Stack

An implementation of the flight control system presented in chapter 4 has been developed based on the PX4 open-source flight stack [6]. PX4 provides support for several hardware platforms, including the Pixhawk [4]. An image of the Pixhawk 2.1 flight controller installed on the MiniPetrel prototype is shown in figure 6.1.

The PX4 framework provides basic functionality including RC control input, communication with ground control stations using the MAVLINK protocol, hardware drivers, logging, a parameter system and an extended Kalman filter for state estimation. This has enabled the author to focus on software controller implementation, instead of basic utilities as mentioned above. Figures 6.2 and 6.3 shows screenshots of the QGroundControl GUI, which is the default ground control station for use with PX4.

The PX4 flight stack is divided into modules that runs as separate threads in the operating system, which communicate using uORB messages, which are implemented using a publish/subscribe pattern. The author has spent a significant amount of time familiarizing with the framework and how to write custom code. A lot of useful information is found in the PX4 developers guide [5], and most of the work is based on this information, but a lot of time had to be spent manually searching through code. Because of lack of time, and focus on thoroughly documenting the flight experiment results,

Figure 6.1: Pixhawk flight controller

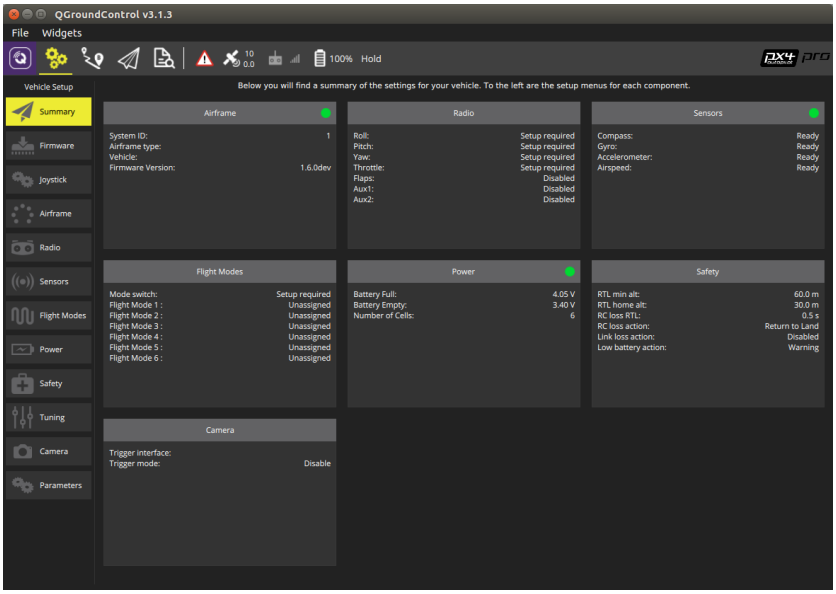


Figure 6.2: QGroundControl

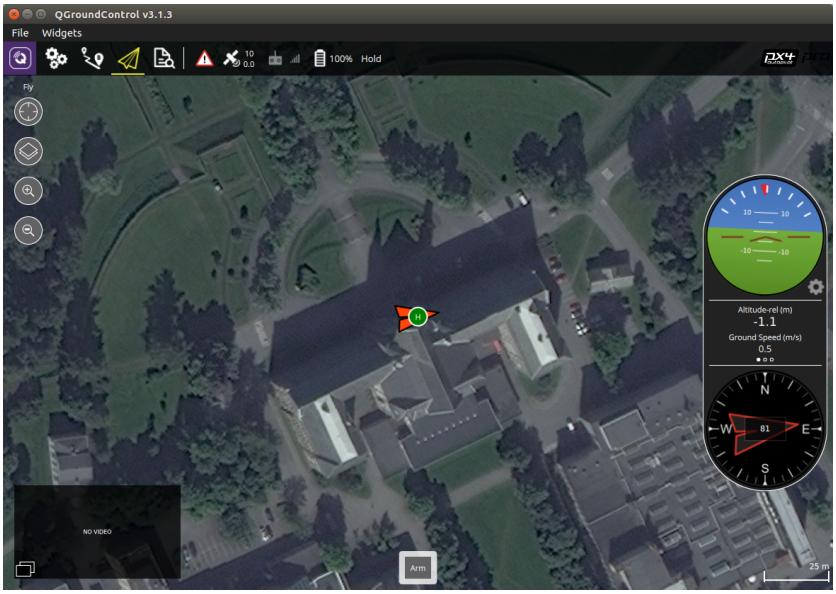


Figure 6.3: QGroundControl

the documentation of the implementation has had to suffer. The main parts of the custom code is attached in the digital appendix. In shortness, it consists of two custom modules, one for velocity control (wrongly named pos control) and one for control allocation. In addition, custom uORB message definitions, build scripts, init scripts, logger configurations and several minor changes to existing code has been made.

7 Software-in-the-Loop Simulation

To test the implemented autopilot software, a *Software-in-the-loop* (SITL) simulation has been developed using the Gazebo robot simulator. In a SITL simulation, the flight code runs on a computer, e.g. in Ubuntu Linux, in contrary to *Hardware-in-the-Loop* (HIL) where the flight code runs on the actual autopilot hardware being used on the aircraft. Both SITL and HIL simulations are useful for preflight testing when developing UAVs. This chapter provides a brief introduction to Gazebo, and documents the procedure and presents the main components involved in interfacing a SITL simulation with PX4. In addition, plots of logged simulation data is included for verification of the custom PX4 code.

Some examples of SITL and HIL related to guidance, navigation and control of UAVs in the literature can be found in [26] and [28]. HIL is also treated in [29]. Some examples of commercial resources, including specialized real-time simulation hardware are [3], [2] and [10].

7.1 Gazebo

Gazebo [1] is an open-source robot simulator including a graphics engine, ODE solver, collision detection and possibilities for sensor simulation. Models

are defined using an XML extension called SDF [8]. In Gazebo, plugins are used to extend functionality, and write custom code for e.g. aerodynamic models or communication with external applications. An excerpt of a model definition, consisting of links, joints and plugins, is given below.

```
<sdf version='1.5'>
  <model name='stormpetrel'>
    <pose>0 0 0.23 0 0 0</pose>
    <link name='base_link'>
    ⋮
    </link>
    <link name='base_link'>
    ⋮
    </link>
    ⋮
    <joint name='tiltarm_left_joint' type='revolute'>
    ⋮
    </joint>
    <joint name='tiltarm_left_joint' type='revolute'>
    ⋮
    </joint>
    ⋮
    <plugin name='rotors_gazebo_imu_plugin' filename='↩
   librotors_gazebo_imu_plugin.so'>
    ⋮
    </plugin>
    ⋮
    <static>0</static>
  </model>
</sdf>
```

Listing 7.1: SDF excerpt

A SDF model definition has been made for the StormPetrel UAV, with ge-

ometry from CAD-models and the parameters identified in chapter 3. The simulation is interface to a SITL instance of PX4 for SITL simulations. Plugins for thruster models and UDP communication with PX4 have been used from the RotorS simulator [20]. The full SDF model implementation is attached in the digital appendix.

Screenshots of running simulation runs, as well as detailed images of the developed model is shown in figures 7.1 - 7.3. Video of running simulations are shown in the video attached in the digital appendix, parameters used are summarized in appendix A, while plots of logged data are provided in appendix B.

The results show that the implemented custom autopilot succesfully stabilizes the vehicle in simulations, with good maneuverability.

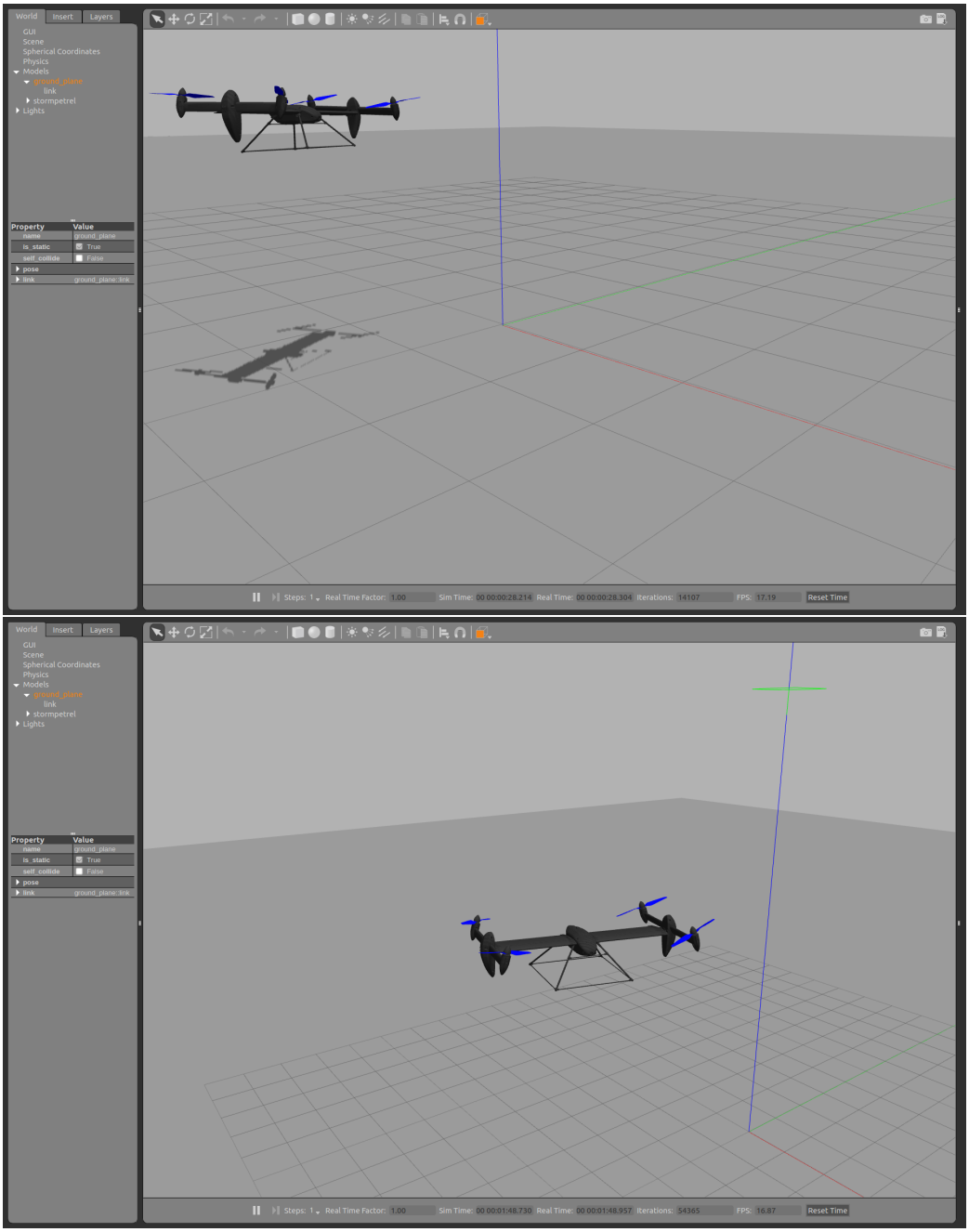


Figure 7.1: Software-in-the-loop simulation with Gazebo

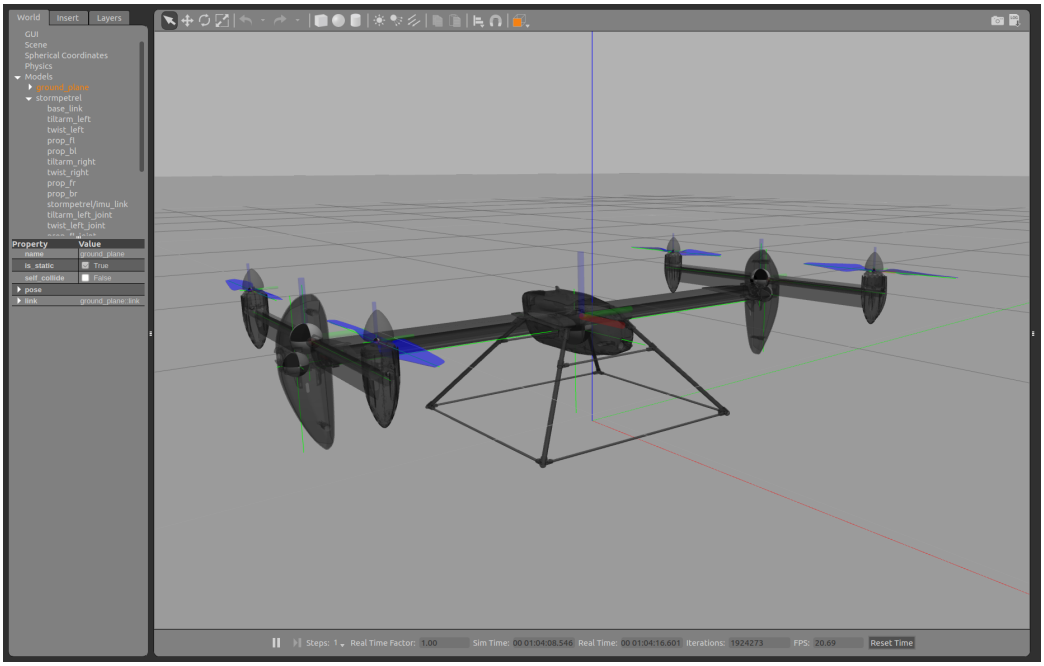


Figure 7.2: Link frames and centers of mass

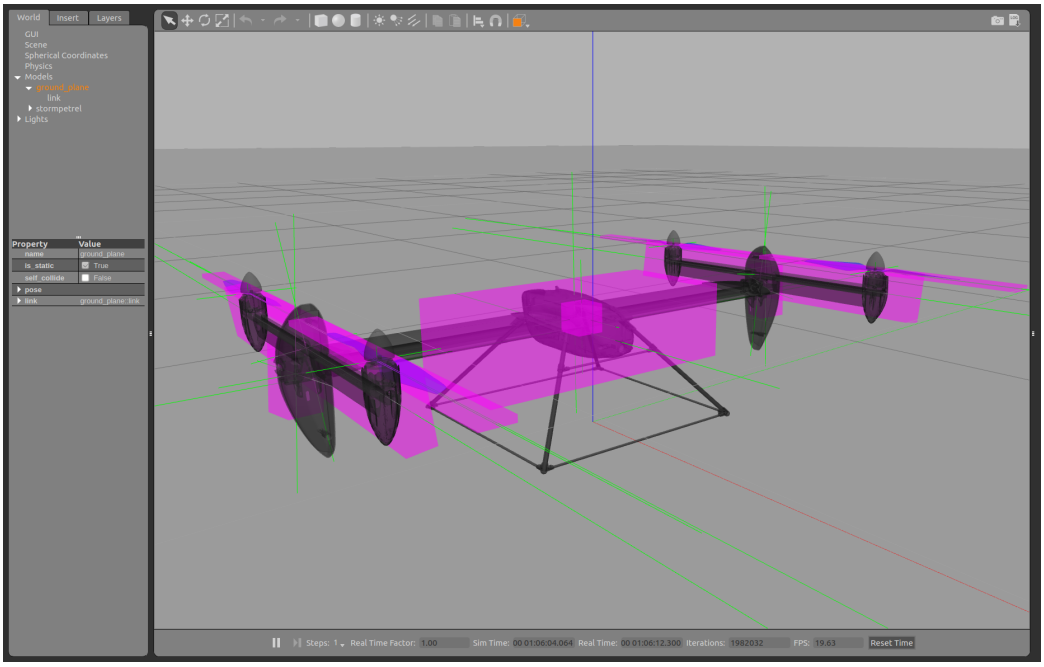


Figure 7.3: Inertia

8 Flight Tests

This chapter presents the results of flight tests carried out with the MiniPetrel prototype based on the implementation discussed in chapter 6. The laboratory test setup is explained, and plots of log data are presented and discussed.

8.1 Test Setup

To test the PX4 implementation of the custom flight code, and to do initial tuning of controllers in a safe way, an indoor test cell was made in cooperation between Sevendof AS and the Department of Engineering Cybernetics. The test cell consists of a net-enclosed area of approximately 6 x 8 x 6 meters. On the inside, the vehicle is suspended from a rope. The other end of the rope is attached to a crane with a counterweight on the outside. The counterweight and the natural damping of the crane hook provides damping when shutting off power in the air. The MiniPetrel prototype hanging in the rig is shown in figure 8.1. To abort flight if something wrong should happen, e.g. unstable attitude responses, a killswitch was mapped to the RC controller, effectively shutting off power to the outputs if engaged.

There is also a rope fastened underneath the vehicle, to manually restrain the motion if needed. A downside to these safety measures is that the ropes

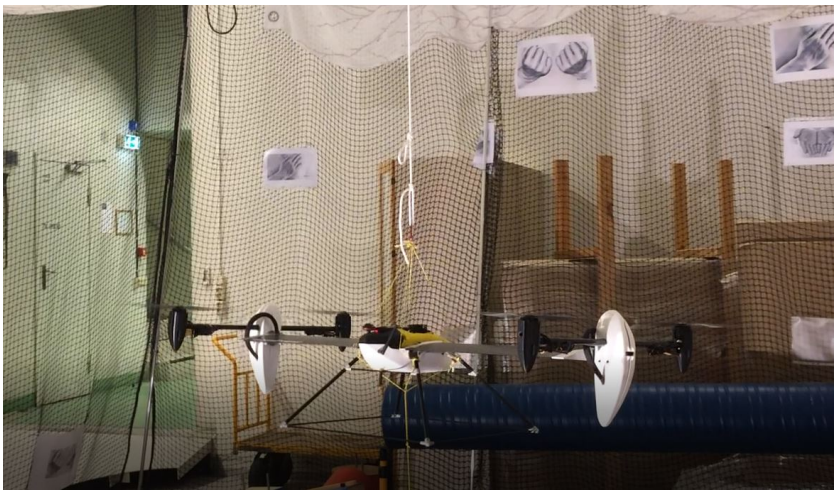


Figure 8.1: Test Rig

Figure 8.2: Hovering

affect the dynamics of the UAV by forces acting on the body of the vehicle, and the effective weight of the UAV when suspended is lower than actual. As the vehicle ascends, approaching the ceiling, the propellers carry all of the weight and the effects on the dynamics are small unless operating at the borders of the cell.

Since the test cell is located indoors, and no indoor positioning system (IPS) has been implemented, the tests are based on IMU attitude, angular rates and velocity estimates only. This means that the position has to be stabilized by manual velocity setpoints.

The mass of the MiniPetrel prototype including all equipment used during testing turned out to be a little higher than what is described in chapter 3. The total mass is approximately 6.5 kg, which is reflected in the plots in the next section.

8.2 Experimental Data

The main challenges when performing experiments were due to three causes:

- The reduced weight of the vehicle during take-off caused by the rope.
- Ground and wall effects due to powerful propellers in a tight space.
- Drifting velocity estimates due to lack of indoor positioning system.

The velocity and attitude controllers were tuned experimentally until satisfactory performance was achieved. In this context, that means a controlled, stable hover with adequate translational control.

To be able to control the position of the vehicle manually in such a limited amount of space, the velocity gains had to be quite low. In this way, aggressive responses to disturbances due to ground and wall effects was avoided. In addition, because of the reduced effective weight during take-off, the gravity

feedforward term of the velocity controller was reduced significantly. The integral term then has to do the rest of the job.

Figure 8.2 shows the MiniPetrel during a stable hover. Roll and pitch was succesfully stabilized, and the velocity and yaw rate controllers succesfully dampened the vehicle motion in the test cell. In addition, manual yaw and horizontal translation maneuvers were performed in a controlled way.

The lack of positioning system (GPS does not work indoors) caused the velocity estimates used for feedback to drift over time. This can cause the velocity estimate to have quite large magnitudes, even though the vehicle is actually standing still. Most of time, this was not a problem, as the position had to be controlled manually, but sometimes, the horizontal velocity estimates drifted beyond the range of velocity setpoints mapped from the stick. These test flights had to be aborted, with the killswitch engaged. The velocity estimate issue also has implication on the logged data shown below, as the plotted velocities does not necessarily represent the actual motion of the vehicle. Overall however, the system performed well.

A video is attached in the digital appendix which includes footage of the flight experiments.

The following sections present plots of experimental data gathered using the PX4 logger module. A SanDisk Extreme U3 32GB microSD card was used with the Pixhawk to provide high bandwidth logging with low write time and few dropouts. The generated log files were converted into CSV files and imported into Matlab to generate figures.

8.2.1 Typical Test Flight

Figures 8.3 - 8.18 show the results of a typical test flight. The most important parameters used for all results shown in this chpater are shown in tables A.4 - A.6 in appendix A.

Take-Off and Hover

The commanded forces can be seen in figure 8.6. The vertical force command, F_z starts off at approximately the value of the gravity feedforward term, which was set to 20 N. It then ramps up to a value greater than the weight of the vehicle ($6.5 \cdot 9.81 \approx 64$ N) when ascending, before settling at approximately this value during constant altitude hover. The mentioned ramp-up is due to the z-velocity integral summing up as can be seen in figure 8.8. The z velocity integral was set quite low to enable a smooth controlled take-off.

Horizontal Velocity

Figures 8.3 and 8.4 show how the horizontal velocity commands are tracked. The velocity controller perform quite well, where the vehicle velocities follow the mean of the command, but the excessive stick motion is dampened out by the relatively low gains. Figure 8.7 show the commanded tilt and twist angles, which are fluctuating about ± 3 deg to stabilize the horizontal velocity.

Attitude Stabilization

The roll and pitch angles are shown in figure 8.9. The roll and pitch angles are succesfully stabilized, with the roll angle ± 2.5 deg about zero. The pitch angle is inside ± 5 deg, but there seems to be a positive offset during take-off. The author believes that this offset is due to moments caused by the string fastened around the wing to attach to the test rig rope. Also notice the big fluctuation at about 45-50 seconds. This can be attributed to the large change in tilt angle seen at about 45 seconds in figure 8.7. In general, tilt and twist accelerations cause pitch and roll moment disturbances respectively on the main body. There is especially large nonlinear coupling between tilt and pitch. The effects of twist are of less concern, as the vehicles moment of inertia about the roll axis is much larger than about the pitch axis. In addition the twist servo torques are smaller due to the fact that they lack

the large moment arm of the tilting motion.

Figure 8.11 shows how the vehicle accurately tracks yaw rate commands. Figure 8.10 shows the roll and pitch rates vs setpoints, and it is apparent that there is some room for improvement. The cascade structure of the PX4 attitude controller makes it quite difficult to tune this inner loop. The outer loop could be disabled to tune the inner first, but this is unpractical and risky with the current test setup. The angular velocity integrals are shown in figure 8.13.

Control Allocation

The commanded propeller speeds and PWM outputs are shown in figures 8.14 and 8.15. The thrust boost, reduction and moment scaling applied during the this test flight is shown in figure 8.17. When disregarding the final seconds after landing, it is clear that none of these mechanisms are activated. This might suggest that they are not needed, but further testing with more aggressive maneuvers and wind conditions should be carried out before making any final conclusions regarding this. It is also interesting to notice the sample times of the control allocation module shown in figure 8.18, which is approximately 4 ms on average.

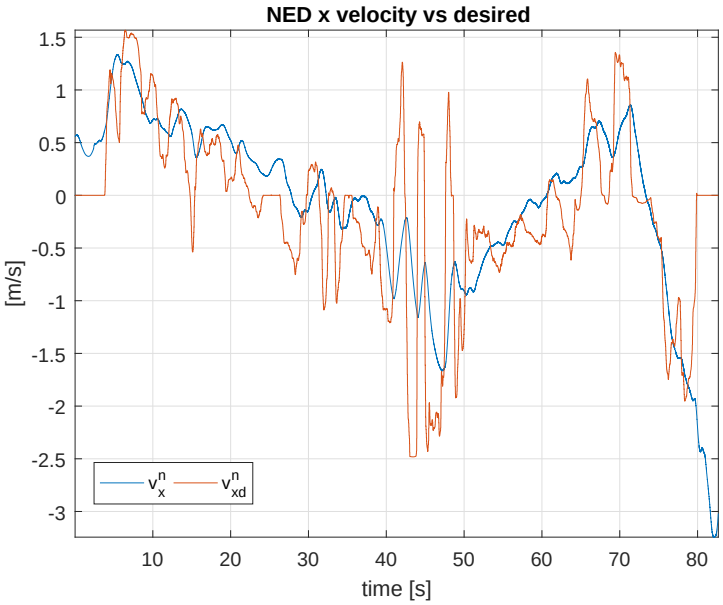


Figure 8.3: NED x velocity

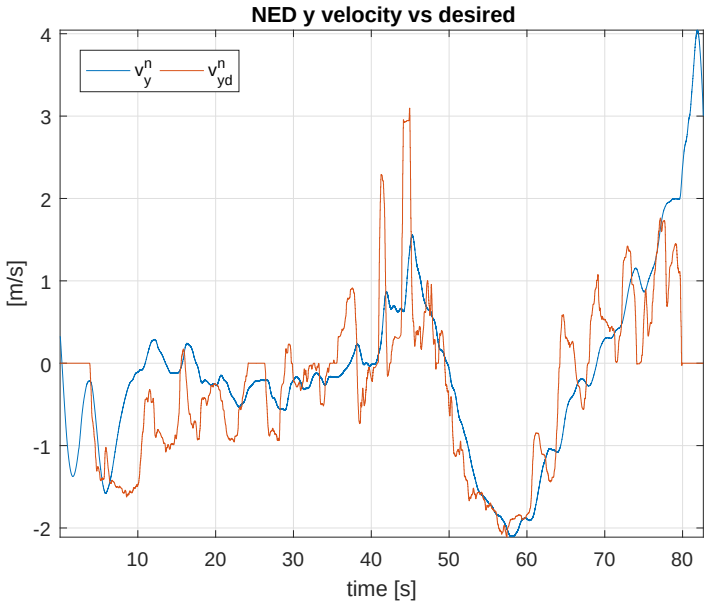


Figure 8.4: NED y velocity

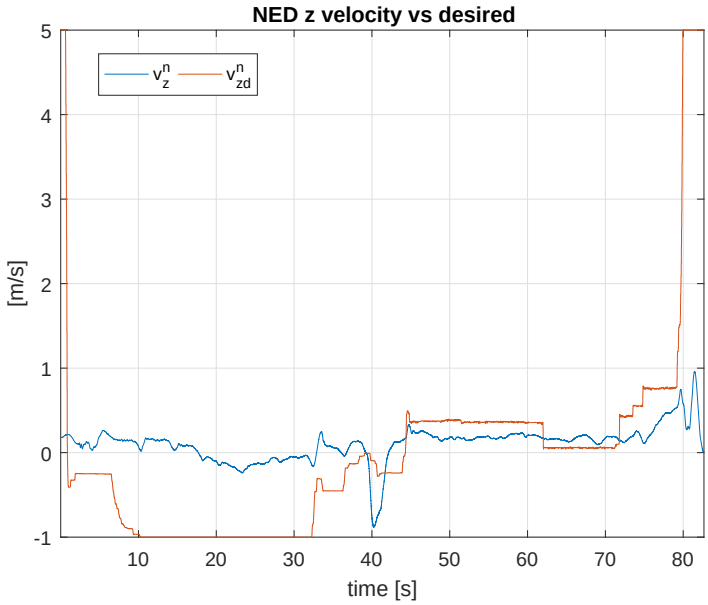


Figure 8.5: NED z velocity

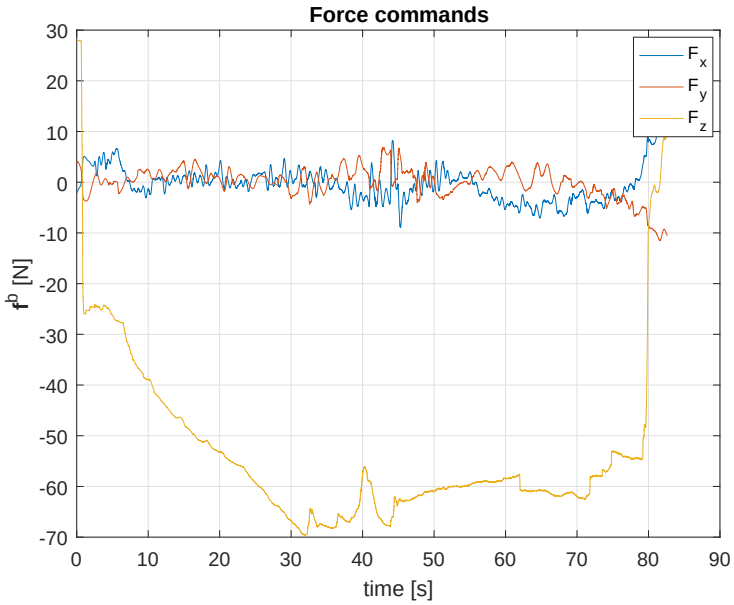


Figure 8.6: Force commands

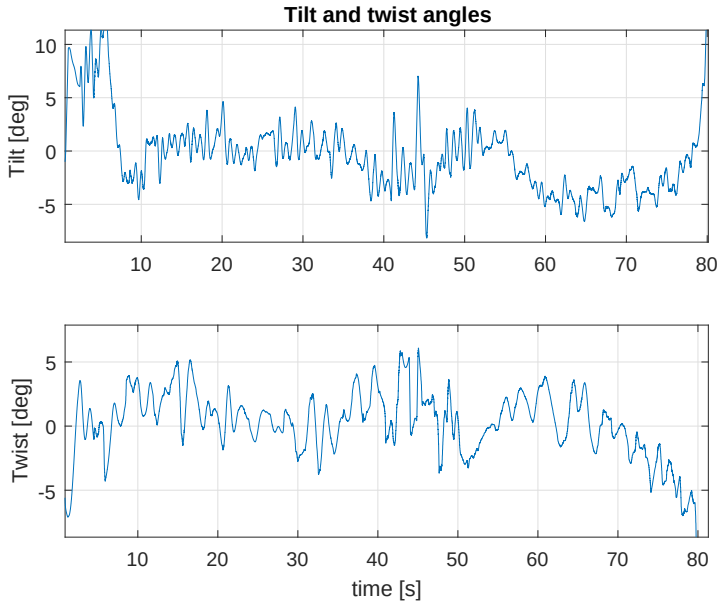


Figure 8.7: Tilt and twist commands

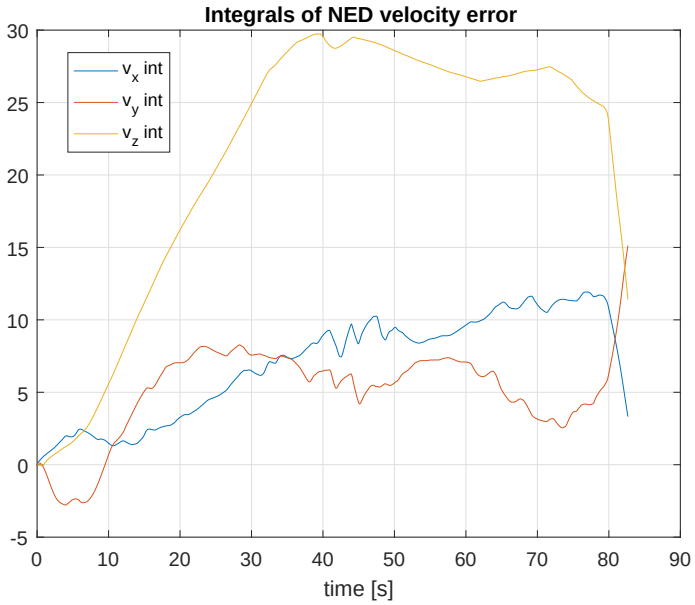


Figure 8.8: Velocity integrals

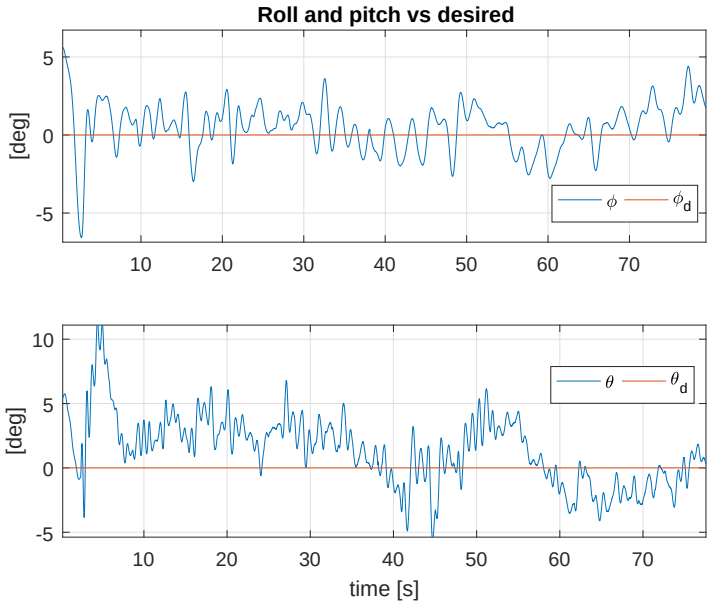


Figure 8.9: Roll and pitch angles

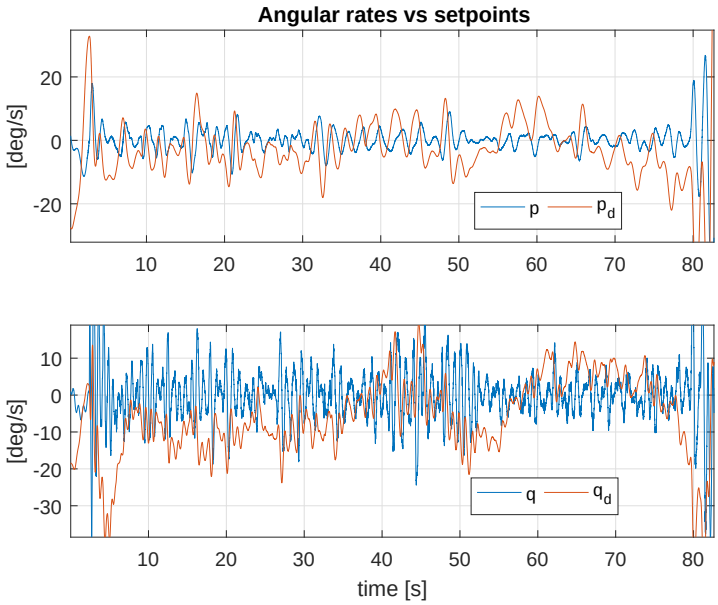


Figure 8.10: Roll and pitch rates

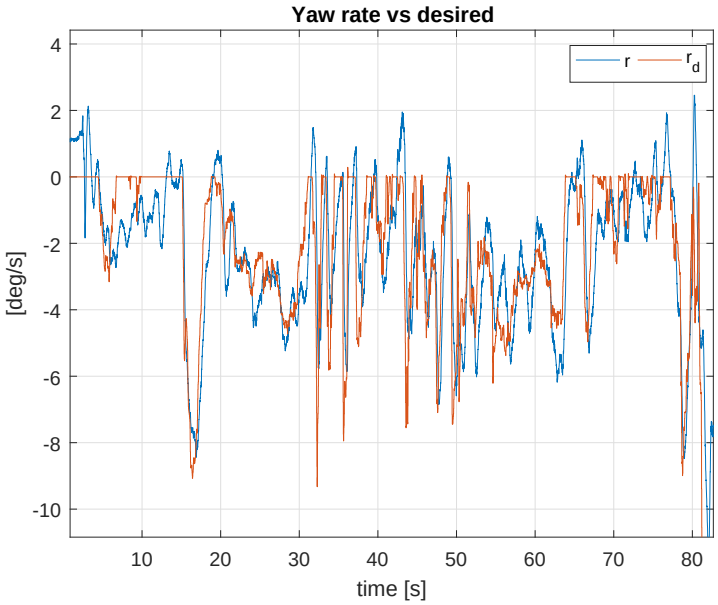


Figure 8.11: Yaw rate

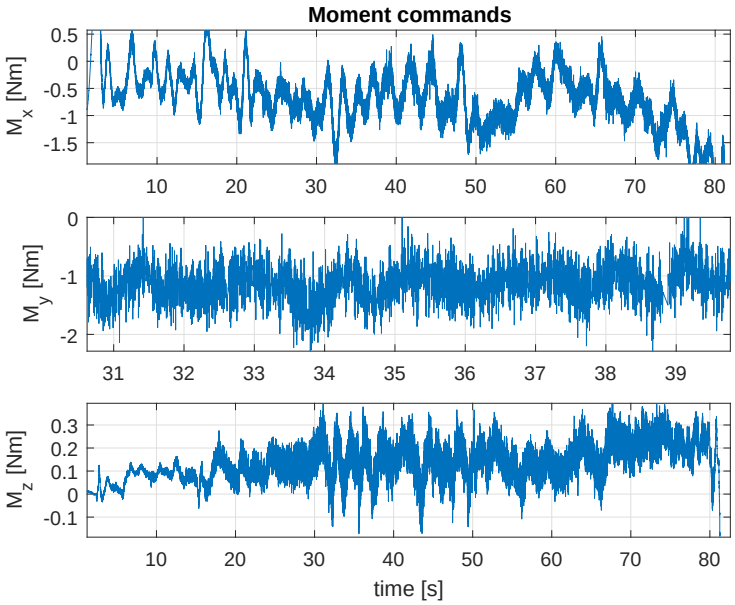


Figure 8.12: Moment commands

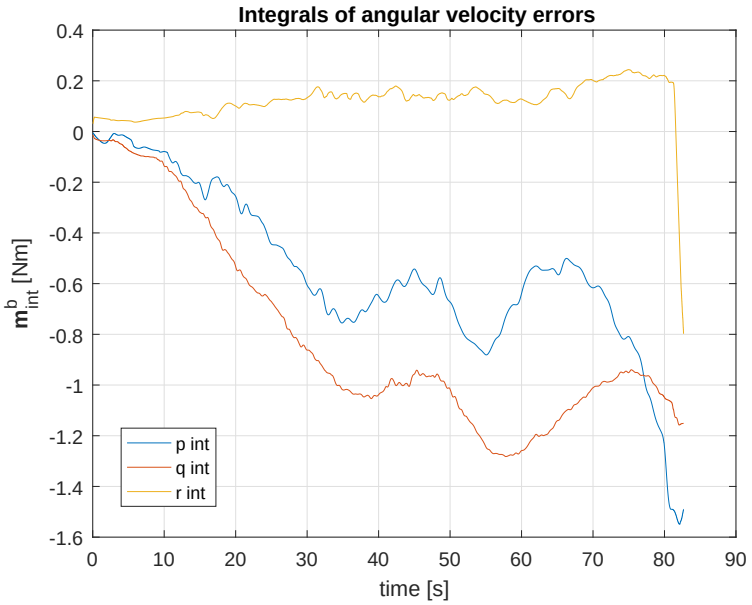


Figure 8.13: Angular velocity integrals

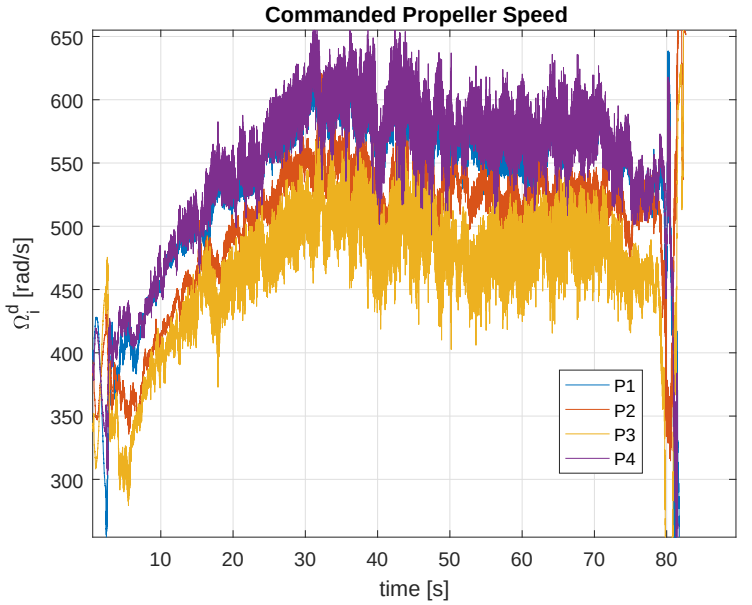


Figure 8.14: Commanded propeller speed

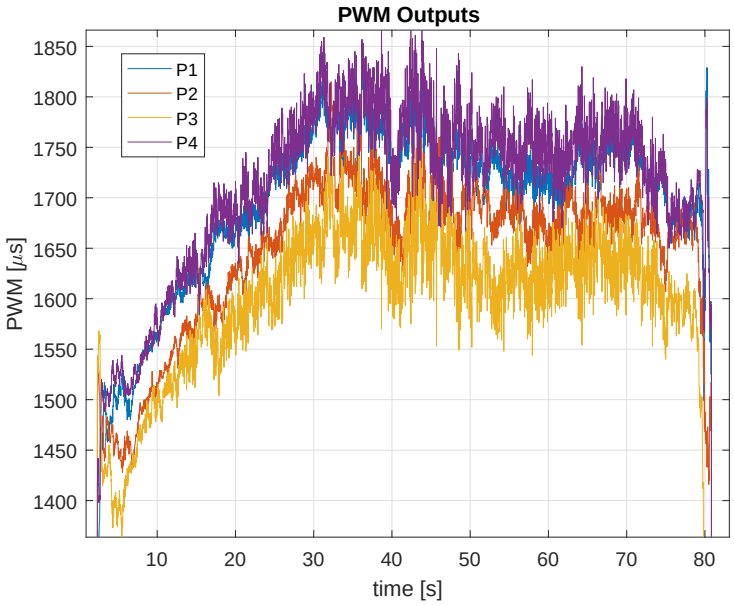


Figure 8.15: PWM outputs

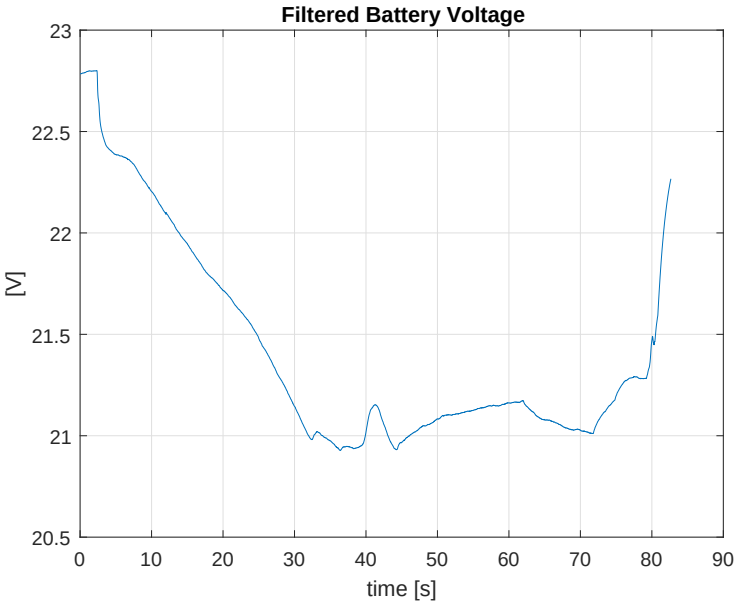


Figure 8.16: Voltage

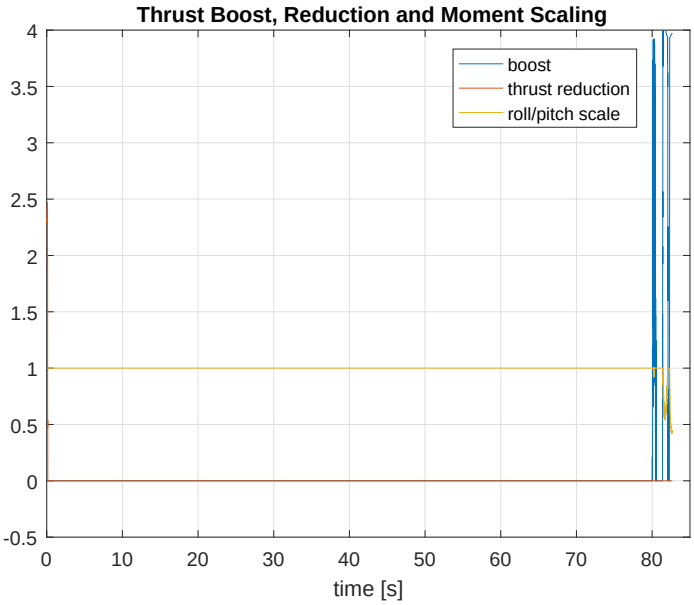


Figure 8.17: Thrust boost, reduction and moment scaling

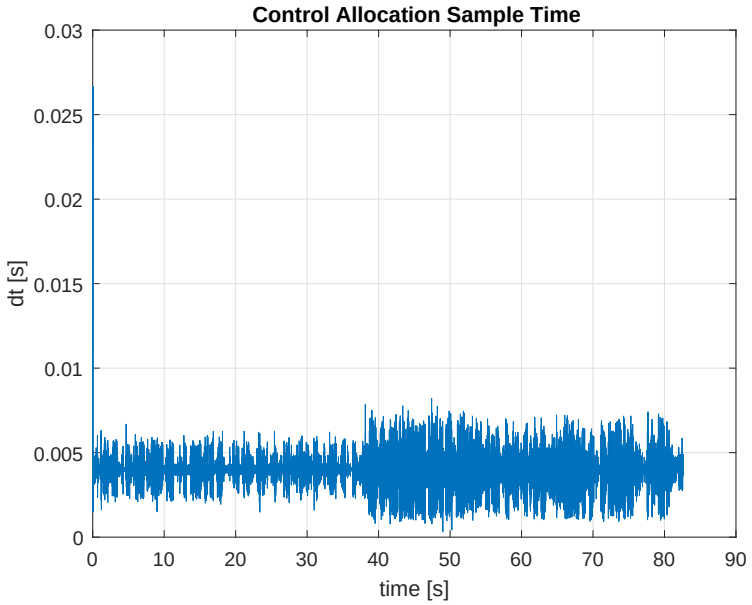


Figure 8.18: Control allocation sample time

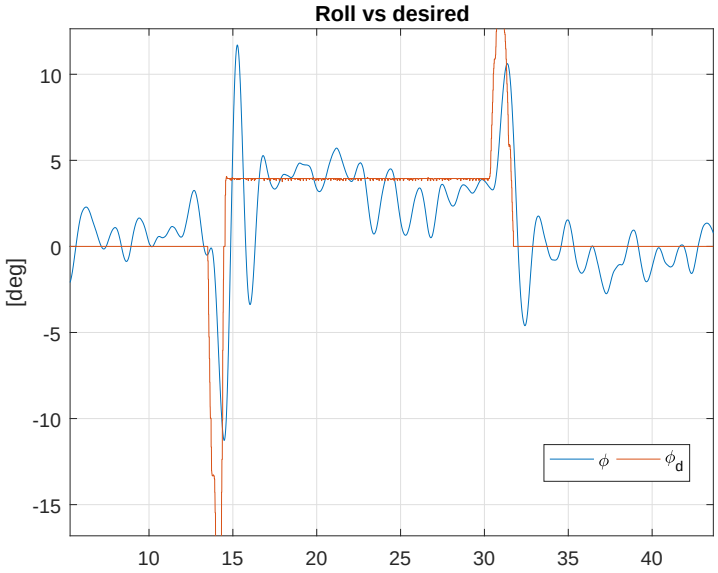


Figure 8.19: Roll response

8.2.2 Roll Response

Figures 8.19 - 8.22 show a zoomed in view of a series of steps in desired roll angle. Figure 8.19 shows that the roll angle converges to the setpoint quite quickly, but the response is underdamped with large overshoots. This should be tuned further in future flight tests. Figure 8.21 show how the twist angle compensates to hold zero velocity, and is approximately equal to the opposite of the roll angle.

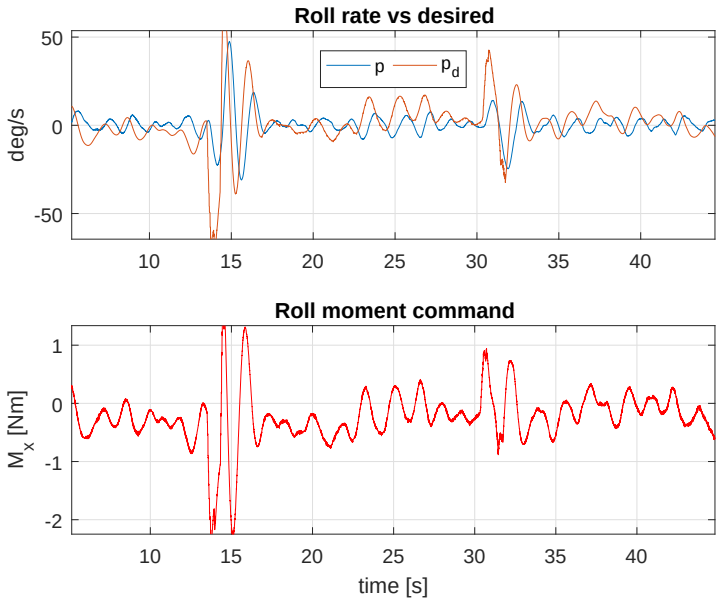


Figure 8.20: Roll rate and moment command

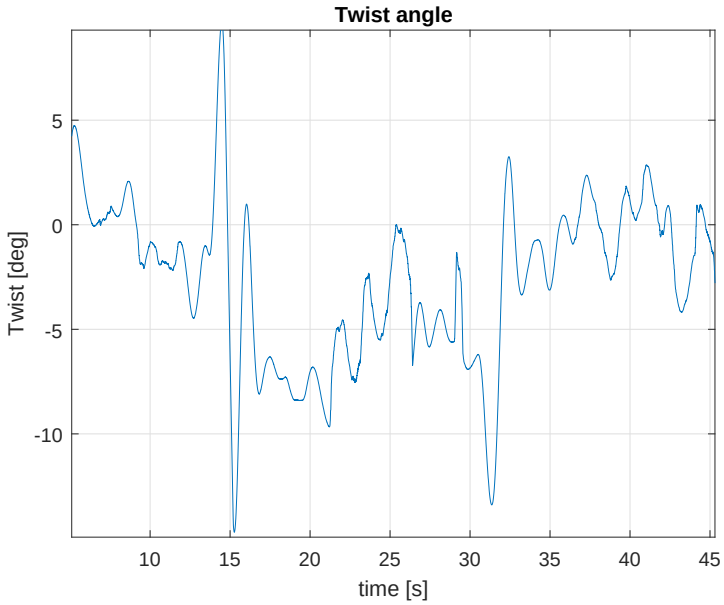


Figure 8.21: Commanded twist angle

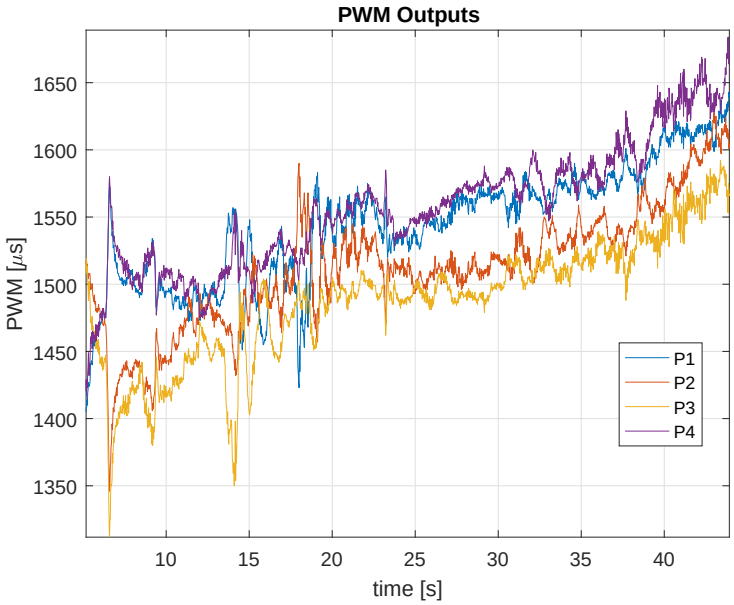


Figure 8.22: PWM outputs

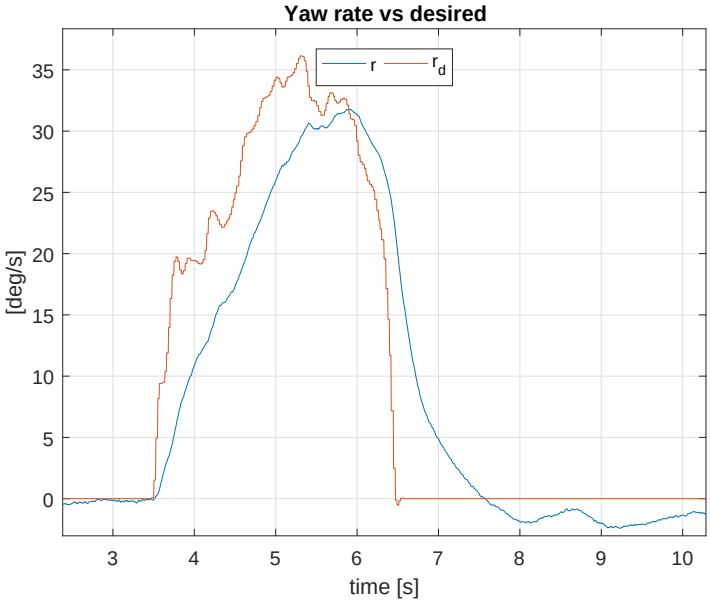


Figure 8.23: Yaw rate response

8.2.3 Yaw Rate Response

Figures 8.23 - 8.25 show a zoomed in view of a yaw rate response. Again, command tracking works well, and it is especially interesting to see how the diagonal pairs of propellers work in opposite magnitude to generate the yaw moment, shown in figure 8.24.

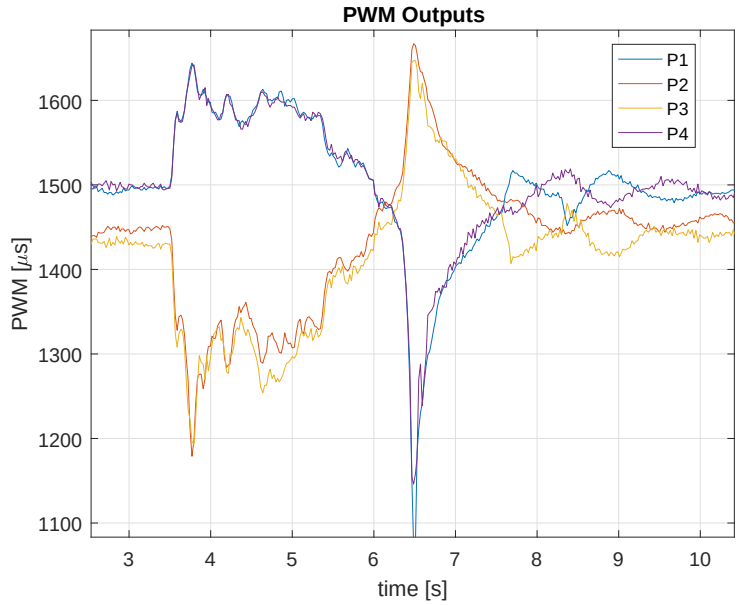


Figure 8.24: PWM outputs

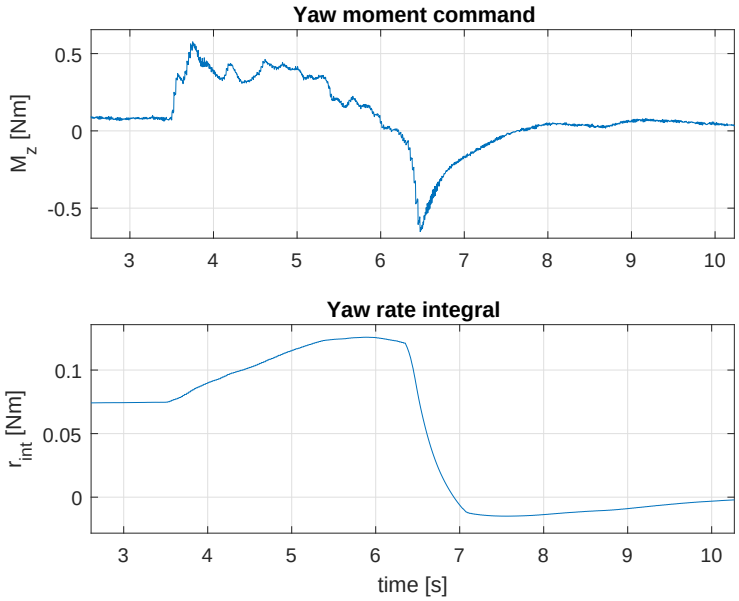


Figure 8.25: Yaw moment command and yaw rate integral

Concluding Remarks

9 Conclusion

In this chapter, a summary, conclusions and suggestions for future work are given.

9.1 Summary and Conclusion

This thesis consists of two main parts. In the first part, issues related to modeling, identification and control of the StormPetrel UAV are covered. The second part discusses autopilot implementation, which is verified through SITL simulation and prototype flight tests.

First, an alternative mathematical model is presented, based on a 10 DOF Euler-Lagrange approach. Expressions for the kinetic and potential energy of the vehicle is derived, and the system matrices are calculated from this using the Matlab Symbolic Math Toolbox. Actuator dynamics and aerodynamic effects are added to the model as external forces. The downside of this approach is that the need for generalized coordinates and the choice of Euler angles as attitude representation introduces singularities in the model. For most practical use-cases however, this is not a problem, as the singularities are outside of the normal operating conditions of the UAV. If failure-situations need to be simulated, other approaches has to be used. Alternatives are discussed

and suggested for future work. It is also shown how a simpler control design model can be derived for hover flight, based on certain assumptions.

As a basis for simulation and control design, inertial properties of the prototype are estimated based on CAD-models, and thruster parameters are estimated based on measured RPM, force and torque using a propeller test stand and dynamometer. A basic control system is presented, including attitude controller, velocity controller and control allocation. The basic control allocation method from [21] is modified by introducing a more convenient choice of reference frames, and it is attempted to introduce additional robustness using two different approaches. The first approach uses a series of heuristics, inspired by the multirotor "mixers" used in PX4 flight stack, while the second formulates an optimization problem, which is solved using quadratic programming. The first approach is deployed in the flight tests, while the second is implemented in the Matlab simulations.

A simulator is implemented in Matlab/Simulink based on the Euler-Lagrange equations of motion mentioned above, the identified inertial and thruster parameters, and the flight control system presented in chapter 4. Most of the remaining unknown parameters are reused from [21]. Matlab function files for the Euler-Lagrange system matrices are generated from the symbolic expressions and used in the simulator. Better performance and a more efficient simulation should be strived for in the future. The current implementation is very demanding due to the complicated kinematic terms introduced by Euler angles as generalized coordinates. This is especially true for the coriolis and centrifugal matrix $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$. Symbolic simplification, optimized code generation and compilation were all attempted unsuccessfully. The slow simulation time makes it difficult to tune controllers and improve performance, as well as doing dynamic exploration of the system. However, simulation results showing attitude and velocity command tracking are presented, thus verifying both the proposed model and optimization-based control allocation scheme to some extent. Further verification of the model should be done in the future.

In part 2, the velocity controller and heuristics-based control allocation scheme

is implemented using the open-source PX4 flight stack. An introduction to PX4 is given, as well as the basics of writing custom code. To test the custom software, a model for the MiniPetrel UAV has been developed in the Gazebo robot simulator for SITL simulation. Plots given in appendix B, as well as the video attached in the digital appendix show simulation runs where the implemented autopilot system succesfully stabilizes both the attitude and velocity of the vehicle. Further work should be done to make the simulation more realistic, and the thruster dynamics, servo models and nonlinear aerodynamics presented in chapter 2 should be implemented as plugins. In addition, the same system could potentially be used for HIL simulation.

Finally, the implemented custom autopilot code is tested using the MiniPetrel prototype in an indoor test rig. The flight experiment results show, not only that the proposed solution works, but that the thrust vectoring concept in general, as well as the current mechanical solution shows great promise. The main challenge seems to be a tight nonlinear closed-loop coupling between the pitch controller of the vehicle body and the tilt motion of the propellers. This arises due to a reaction torque acting on the body about the pitch axis, when the tilt servos accelerate. The pitch controller compensates, which again causes a load on the tilt servo due to thrust differences between the fore and aft propellers. When the pitch angle changes, the commanded tilt angles will again change due to the rotation matrix used in the velocity controller. This cannot be completely avoided, even if the velocity controller is based on body-velocities instead, as the rotation matrix will then be needed in the guidance system instead. This has to be solved by careful tuning of attitude controllers, tilt servos, as well as limiting tilt accelerations. The current implementation uses the default multirotor attitude controller provided by the PX4 framework, which provides the control allocation module with moment commands. This uses as cascade structure, with an outer loop providing angular velocity setpoints to an inner rate loop. These cascaded loops are quite difficult to tune, and the presented flight plots show that the angular velocity command tracking is not very good. Further work should therefore focus on an implementation of a custom attitude controller, e.g. a version of (4.1). The next natural steps would then be to take the flight tests

outdoors, with GPS and more space available, and implement a position hold loop.

9.2 Future Work

Some recommendations for future work are summarized below:

- Further verification and dynamic exploration of the proposed Euler-Lagrange model and alternative control allocation scheme should be performed. To enable this, the current simulator implementation should be made more efficient.
- An alternative singularity-free formulation should be strived for, e.g. based on the calculus of variations or inspired by [19].
- Work on estimating the aerodynamic parameters of the vehicle should be performed, e.g. wind-tunnel testing or CFD (computational fluid dynamics) simulations.
- The servo motor models, thruster dynamics and nonlinear aerodynamics should be implemented as Gazebo plugins.
- Implement a HIL testing framework based on the developed SITL model.
- Implement a custom attitude controller in the PX4 flight stack.
- Implement a position controller and conduct further flight experiments outdoors with GPS and more space.

Bibliography

[1] Gazebo robot simulator. http://gazebosim.org/. Accessed: Jul 23, 2017.

[2] MathWorks sil and pil simulations. https://se.mathworks.com/help/ecoder/ug/about-sil-and-pil-simulations.html. Accessed: Jul 25, 2017.

[3] National Instruments hardware-in-the-loop (hil) simulation. http://www.ni.com/en-no/innovations/aerospace-defense/hardware-in-the-loop.html. Accessed: Jul 25, 2017.

[4] Pixhawk open hardware project. https://pixhawk.org/. Accessed: Jul 25, 2017.

[5] PX4 development guide. https://dev.px4.io/en/. Accessed: Jul 23, 2017.

[6] PX4 professional autopilot. http://px4.io/. Accessed: Jul 25, 2017.

[7] RCbenchmark series 1580 thrust stand and dynamometer. https://www.rcbenchmark.com/dynamometer-series-1580/. Accessed: Jul 18, 2017.

[8] SDF describe your world. `http://sdformat.org/`. Accessed: Jul 25, 2017.

[9] SolidWorks 3d cad software. `http://www.solidworks.com/`. Accessed: Jul 19, 2017.

[10] Speedgoat real-time simulation and testing. `https://www.speedgoat.com/`. Accessed: Jul 25, 2017.

[11] T-MOTOR motor description. `http://www.rctigermotor.com/html/2013/Power-Type_0928/91.html`. Accessed: Jan 12, 2017.

[12] John Anderson. *Introduction to Flight*. McGraw-Hill Education, 2015.

[13] John Anderson. *Fundamentals of Aerodynamics*. McGraw-Hill Education, 2017.

[14] Randal W. Beard and Timothy W. McLain. *Small Unmanned Aircraft*. Princeton University Press, 2012.

[15] Glenn Bitar. *Development of Flight Control System for VTOL UAV MiniPetrel*. Specialization project, Department of Engineering Cybernetics, NTNU, 2017.

[16] Olav Egeland and Tommy Gravdahl. *Modeling and Simulation for Automatic Control*. Marine Cybernetics, 2002.

[17] T.I. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley & Sons, Ltd, 2011.

[18] T.I. Fossen. *Mathematical Models for Control of Aircraft and Satellites*. Department of Engineering Cybernetics, NTNU, 2013.

[19] Pål J. From, Kristin Ytterstad Pettersen, and Jan T. Gravdahl. *Singularity-Free Dynamic Equations of Spacecraft-Manipulator Systems*. Acta Astronautica 69, p. 1057-1065, 2011.

[20] Fadri Furrer, Michael Burri, Markus Achtelik, and Roland Siegwart. *Robot Operating System (ROS): The Complete Reference (Volume 1)*,

chapter RotorS—A Modular Gazebo MAV Simulator Framework, pages 595–625. Springer International Publishing, Cham, 2016.

[21] Jostein Furseth. *Modeling and Control of VTOL Flying Wing with Thrust Vectoring*. Master's thesis, Department of Engineering Cybernetics, NTNU, 2015.

[22] Tor A. Johansen and Thor I. Fossen. *Control Allocation - A Survey*. Automatica, Volume 49, Issue 5, Pages 1087-1103, 2013.

[23] Waqas Khan and Meyer Nahon. *Toward an Accurate Physics-Based UAV Thruster Model*. IEEE/ASME Transactions on Mechatronics, vol. 18, no. 4, August 2013, 2013.

[24] Johannes Hatle Lundgaard. *Development of Thrust Vectoring System for StormPetrel*. Master's thesis, Norwegian University of Science and Technology, 2016.

[25] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 2006.

[26] Romulo Rodrigues, Rafael C. B. Sampaio, A. Pedro Aguiar, and Marcelo Becker. *FVMS Software-in-the-Loop Flight Simulation Experiments: Guidance, Navigation and Control*. IEEE/ASME Transactions on Mechatronics, vol. 18, no. 4, August 2013, 2014.

[27] Adnan S. Saeed, Ahmad Bani Younes, Shafiqul Islam, Jorge Dias, Lakmal Seneviratne, and Guowei Cai. *A Review on the Platform Design, Dynamic Modeling and Control of Hybrid UAVs*. 2015 International Conference on Unmanned Aircraft Systems (ICUAS), 2015.

[28] V. M. Sineglazov and S.O. Dolgorukov. *Dynamic Hardware-in-the-loop UAV Ground Testing System*. 2015 IEEE 3rd International Conference Actual Problems of Unmanned Aerial Vehicles Developments (APUAVD) Proceedings, 2015.

[29] Asgeir J. Sørensen. *Marine Control Systems - Propulsion and Motion Control of Ships and Ocean Structures, Lecture Notes*. Department of

Marine Technology, Norwegian University of Science and Technology, 2013.

[30] Mark W. Spong, Seth Hutchinson, and M. Vidyasagar. *Robot Modeling and Control*. John Wiley & Sons, Inc., 2006.

[31] Johannes Tjønnås and Tor A. Johansen. *Adaptive Control Allocation*. Automatica, Volume 44, Issue 11, Pages 2754-2765, 2008.

List of Figures

1.1 Concept illustrations, courtesy of Sevendof AS 4

1.2 The StormPetrel VTOL Concept 6

1.3 The MiniPetrel prototype 8

2.1 Coordinate frames 14

2.2 Propeller numbering and direction of rotation 26

2.3 Generation of roll moment 33

2.4 Generation of pitch moment 33

2.5 Generation of yaw moment 33

3.1 Thrust stand and dynamometer 45

3.2 Thrust as a function of speed 47

3.3 Torque as a function of speed 47

3.4 Torque as a function of thrust 48

3.5 Speed as a function of throttle and voltage 48

3.6 Comparison with supplier test data 50

4.1 Block diagram of StormPetrel control system 52

4.2 Control allocation block diagram 54

5.1 NED velocities 63

5.2 Body velocities 63

5.3 Integrals of NED velocity errors 64

5.4 Force commands 64

5.5 Tilt angle 65

5.6 Twist angle 65

5.7 NED position 66

5.8 Attitude 66

5.9 Angular velocities 67

5.10 Integrals of attitude error 67

5.11 Moment commands 68

5.12 Propeller speed 68

5.13 Propeller speed, propellers 1 and 2 69

5.14 Propeller speed, propellers 3 and 4 69

5.15 Attitude 70

5.16 Angular velocities 71

5.17 Integrals of attitude error 71

5.18 Moment commands 72

5.19 NED velocities 72

5.20 Body velocities 73

5.21 Integrals of NED velocity errors 73

5.22 Force commands 74

5.23 Tilt angle 74

5.24 Twist angle 75

5.25 NED position 75

5.26 Propeller speed 76

5.27 Propeller speed, propellers 1 and 2 76

5.28 Propeller speed, propellers 3 and 4 77

6.1 Pixhawk flight controller 82

6.2 QGroundControl 83

6.3 QGroundControl 83

7.1 Software-in-the-loop simulation with Gazebo 88

7.2 Link frames and centers of mass 89

7.3 Inertia 89

8.1 Test Rig 92

8.2 Hovering 92
8.3 NED x velocity 97
8.4 NED y velocity 97
8.5 NED z velocity 98
8.6 Force commands 98
8.7 Tilt and twist commands 99
8.8 Velocity integrals 99
8.9 Roll and pitch angles 100
8.10 Roll and pitch rates 100
8.11 Yaw rate 101
8.12 Moment commands 101
8.13 Angular velocity integrals 102
8.14 Commanded propeller speed 102
8.15 PWM outputs 103
8.16 Voltage 103
8.17 Thrust boost, reduction and moment scaling 104
8.18 Control allocation sample time 104
8.19 Roll response 105
8.20 Roll rate and moment command 106
8.21 Commanded twist angle 106

8.22 PWM outputs 107

8.23 Yaw rate response 108

8.24 PWM outputs 109

8.25 Yaw moment command and yaw rate integral 109

B.1 Body x velocity, SITL A9

B.2 Body y velocity, SITL A10

B.3 Body z velocity, SITL A10

B.4 Force commands, SITL A11

B.5 Tilt and twist commands, SITL A11

B.6 Velocity integrals, SITL A12

B.7 Roll and pitch angles, SITL A12

B.8 Roll and pitch rates, SITL A13

B.9 Yaw rate, SITL A13

B.10 Moment commands, SITL A14

B.11 Angular velocity integrals, SITL A14

B.12 Commanded propeller speed, SITL A15

B.13 PWM outputs, SITL A15

B.14 Voltage, SITL A16

B.15 Control allocation stats, SITL A16

List of Tables

3.1 Mass of each rigid body 44

3.2 Center of mass for each rigid body 44

3.3 Inertia tensor for each rigid body $[\mathrm{kg} \cdot \mathrm{m}^2]$ 45

3.4 Components used in tests 46

3.5 Thrust parameters 49

3.6 Supplier test data 49

A.1 Attitude controller parameters used in SITL A2

A.2 Velocity controller parameters used in SITL A3

A.3 Control allocation parameters used in SITL A4

A.4 Attitude controller parameters used in flight tests A5

A.5 Velocity controller parameters used in flight tests A6

A.6 Control allocation parameters used in flight tests A7

List of Code Listings

7.1 SDF excerpt 86

A

Parameter List

Parameter	*Symbol*	*Value*
MC_PITCHRATE_D		0.03
MC_PITCHRATE_I		0.5
MC_PITCHRATE_P		2.0
MC_PITCH_P		0.5
MC_PR_INT_LIM		0.3
SP_PITCH_MAX		45
MC_ROLLRATE_D		0.03
MC_ROLLRATE_I		0.7
MC_ROLLRATE_P		3.0
MC_ROLL_P		1.0
MC_RR_INT_LIM		0.3
SP_ROLL_MAX		30
MC_YAWRATE_D		0.0
MC_YAWRATE_I		0.2
MC_YAWRATE_P		0.4
MC_YAW_FF		1.0
MC_YAW_P		0.0
MC_YR_INT_LIM		0.3
SP_R_MAX		1.0

Table A.1: Attitude controller parameters used in SITL

Parameter	*Symbol*	*Value*
SP_X_VEL_I		0.0
SP_X_VEL_P		4.0
SP_X_VEL_I_MAX		50.0
SP_X_VEL_MAX		4.0
SP_Y_VEL_I		0.0
SP_Y_VEL_P		3.0
SP_Y_VEL_I_MAX		50.0
SP_Y_VEL_MAX		2.0
SP_Z_VEL_I		10.0
SP_Z_VEL_P		20.0
SP_Z_VEL_I_MAX		50.0
SP_Z_VEL_MAX_UP		1.0
SP_Z_VEL_MAX_DWN		0.5
SP_CTRL_MODE		0 (body)
SP_THR_HOVER	mg	49.05

Table A.2: Velocity controller parameters used in SITL

Parameter	*Symbol*	*Value*
SP_THRUST_COEFF	k	5.2389e-05
SP_MOMENT_CONST	c	0.0204
SP_L_T	l_t	0.585
SP_L_TW	l_{tw}	0.343
SP_F_XZ_MIN		10.0
SP_FX_MAX		70
SP_FY_MAX		70
SP_FZ_MAX		100
SP_MX_MAX		20
SP_MY_MAX		20
SP_MZ_MAX		2
SP_TILT_MIN		-60
SP_TILT_MAX		60
SP_TILT_DELTA		0 (disabled)
SP_TWIST_MAX		30
SP_TWIST_DELTA		0 (disabled)
SP_SPEED_MAX		700
SP_ESC_DELTA		0 (disabled)
PWM_MIN		1000
PWM_MAX		2000
PWM_RATE		400

Table A.3: Control allocation parameters used in SITL

Parameter	*Symbol*	*Value*
MC_PITCHRATE_D		0.05
MC_PITCHRATE_I		0.2
MC_PITCHRATE_P		1.5
MC_PITCH_P		3.5
MC_PR_INT_LIM		4.0
SP_PITCH_MAX		45
MC_ROLLRATE_D		0.05
MC_ROLLRATE_I		0.4
MC_ROLLRATE_P		3.0
MC_ROLL_P		5.0
MC_RR_INT_LIM		8.0
SP_ROLL_MAX		30
MC_YAWRATE_D		0.005
MC_YAWRATE_I		1.0
MC_YAWRATE_P		2.0
MC_YAW_FF		1.0
MC_YAW_P		0.0
MC_YR_INT_LIM		2.0
SP_R_MAX		1.0

Table A.4: Attitude controller parameters used in flight tests

Parameter	*Symbol*	*Value*
SP_X_VEL_I		0.2
SP_X_VEL_P		3.0
SP_X_VEL_I_MAX		50.0
SP_X_VEL_MAX		3.0
SP_Y_VEL_I		0.2
SP_Y_VEL_P		3.0
SP_Y_VEL_I_MAX		50.0
SP_Y_VEL_MAX		2.5
SP_Z_VEL_I		1.5
SP_Z_VEL_P		10.0
SP_Z_VEL_I_MAX		50.0
SP_Z_VEL_MAX_UP		1.0
SP_Z_VEL_MAX_DWN		5.0
SP_CTRL_MODE		1 (NED)
SP_THR_HOVER	mg	20.0

Table A.5: Velocity controller parameters used in flight tests

Parameter	*Symbol*	*Value*
SP_THRUST_COEFF	k	5.2389e-05
SP_MOMENT_CONST	c	0.0204
SP_L_T	l_t	0.585
SP_L_TW	l_{tw}	0.343
SP_F_XZ_MIN		30.0
SP_FX_MAX		70
SP_FY_MAX		70
SP_FZ_MAX		100
SP_MX_MAX		20
SP_MY_MAX		20
SP_MZ_MAX		2
SP_TILT_MIN		-70
SP_TILT_MAX		70
SP_TILT_DELTA		30
SP_TWIST_MAX		70
SP_TWIST_DELTA		40
SP_SPEED_MAX		750
SP_ESC_DELTA		0 (disabled)
PWM_MIN		1075
PWM_MAX		1950
PWM_RATE		400

Table A.6: Control allocation parameters used in flight tests

B

SITL Plots

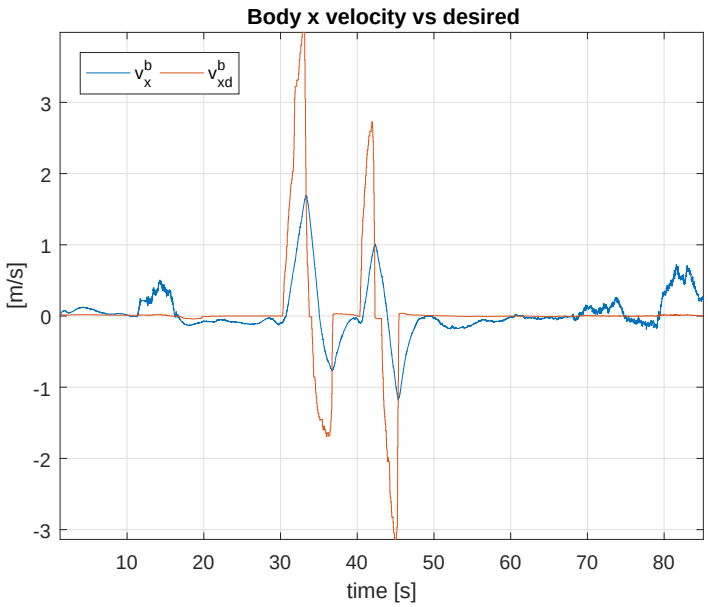


Figure B.1: Body x velocity, SITL

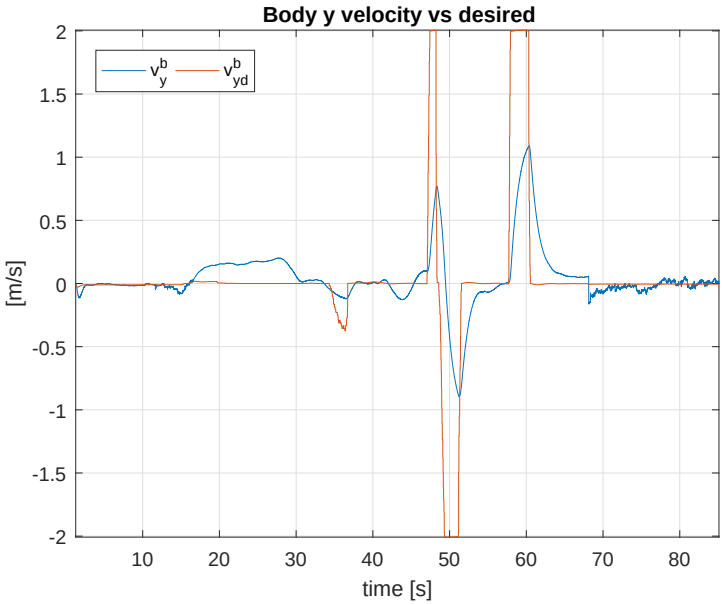


Figure B.2: Body y velocity, SITL

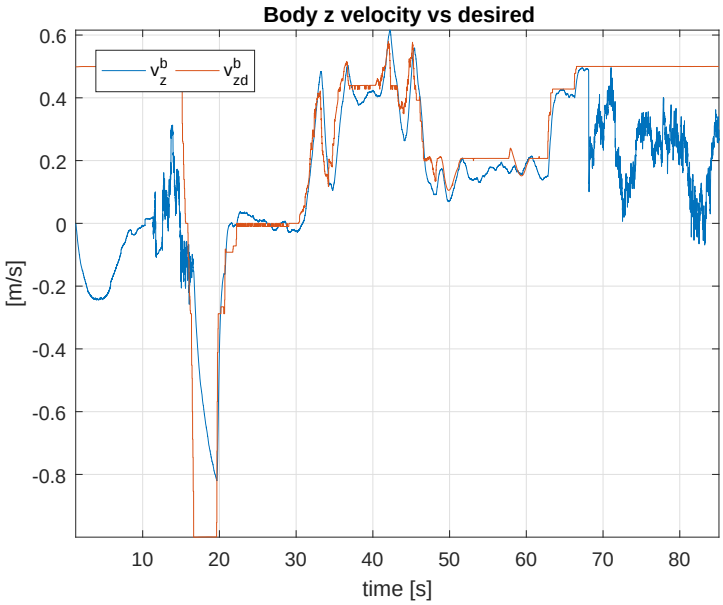


Figure B.3: Body z velocity, SITL

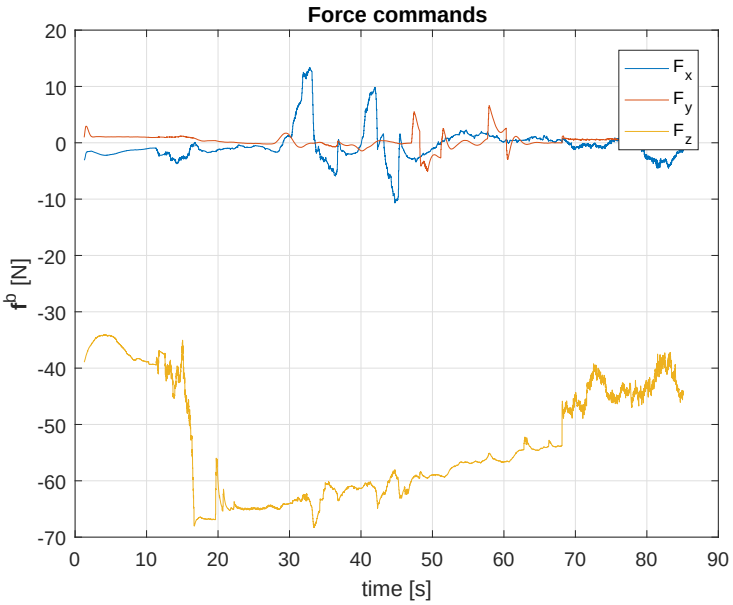


Figure B.4: Force commands, SITL

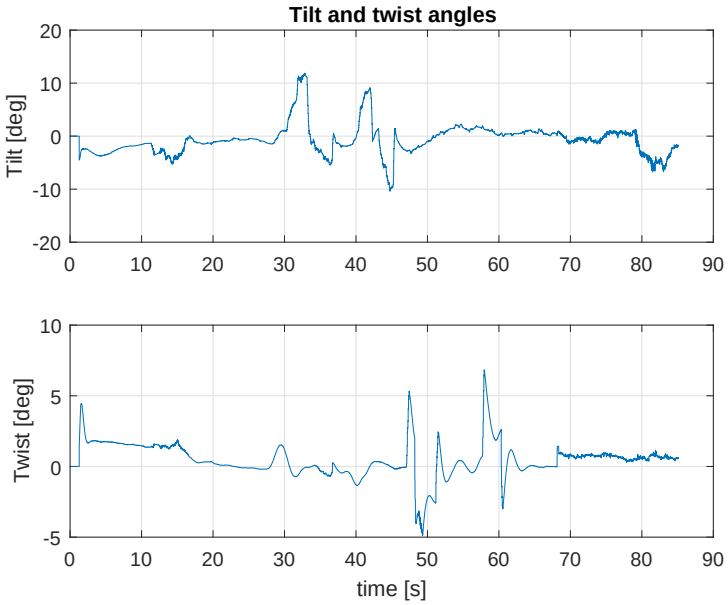


Figure B.5: Tilt and twist commands, SITL

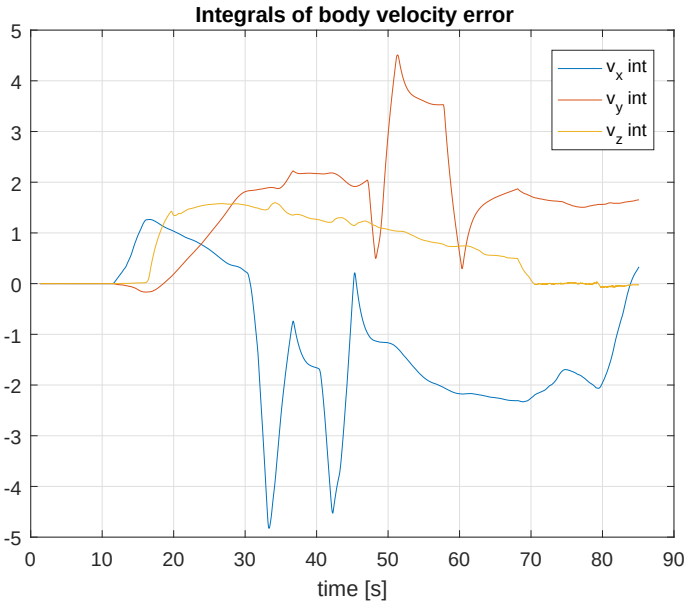


Figure B.6: Velocity integrals, SITL

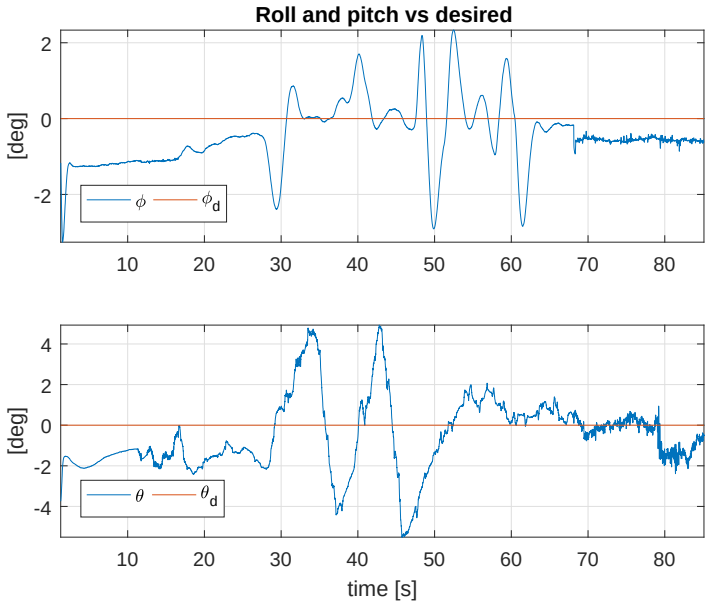


Figure B.7: Roll and pitch angles, SITL

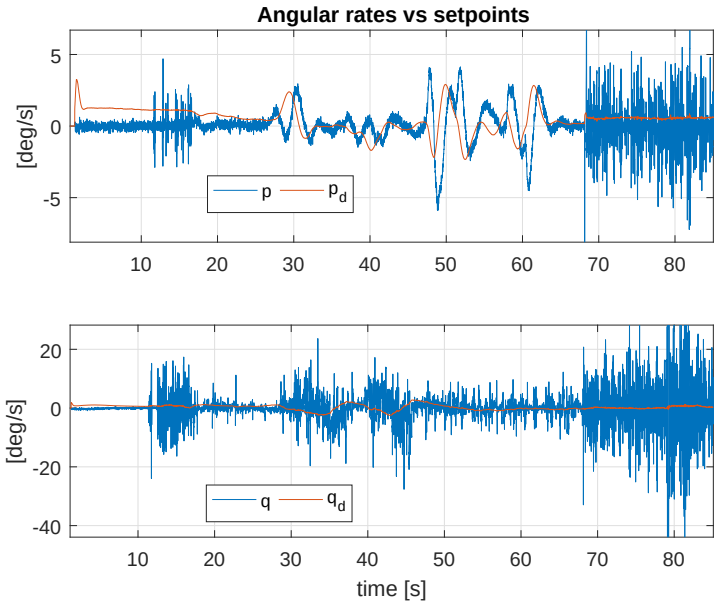


Figure B.8: Roll and pitch rates, SITL

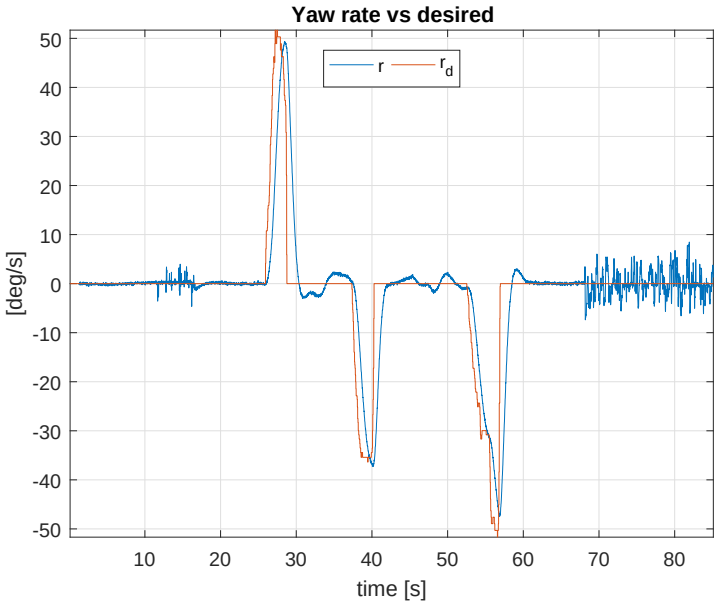


Figure B.9: Yaw rate, SITL

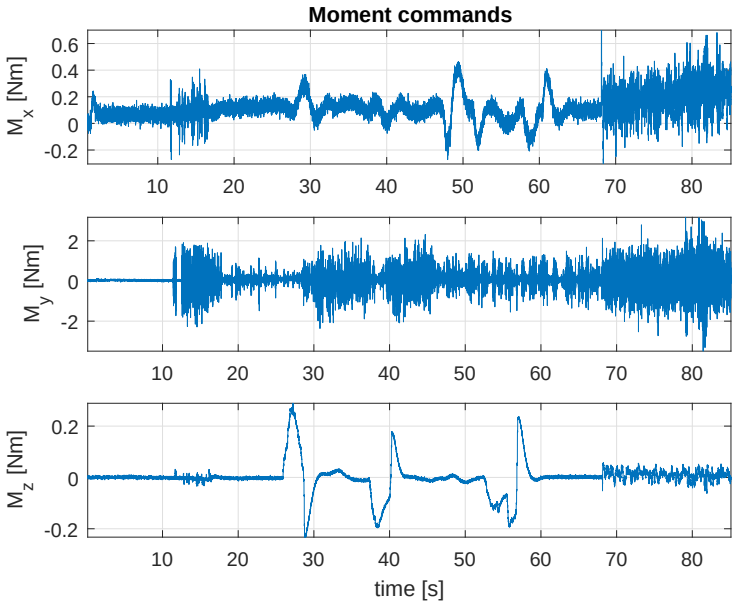


Figure B.10: Moment commands, SITL

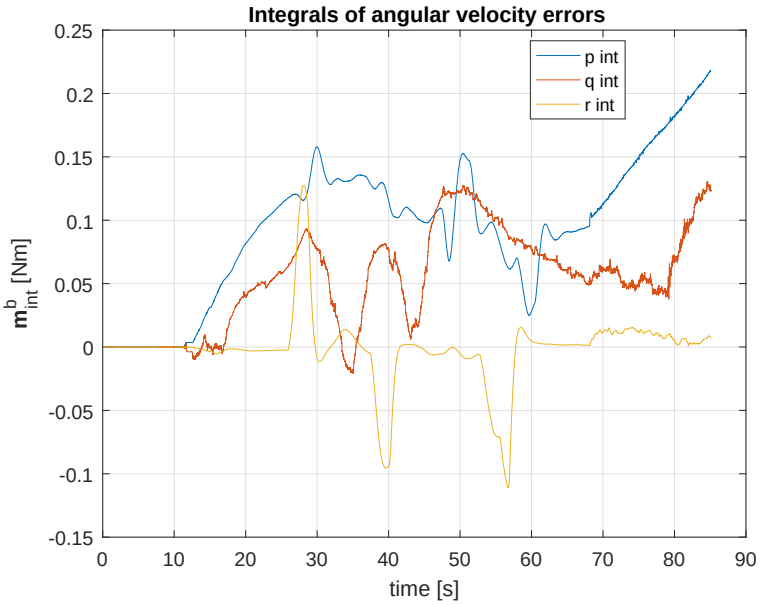


Figure B.11: Angular velocity integrals, SITL

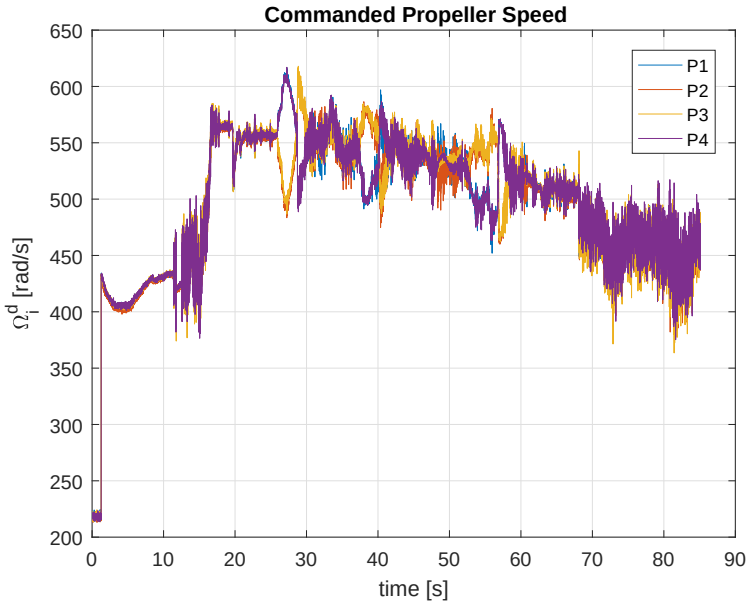


Figure B.12: Commanded propeller speed, SITL

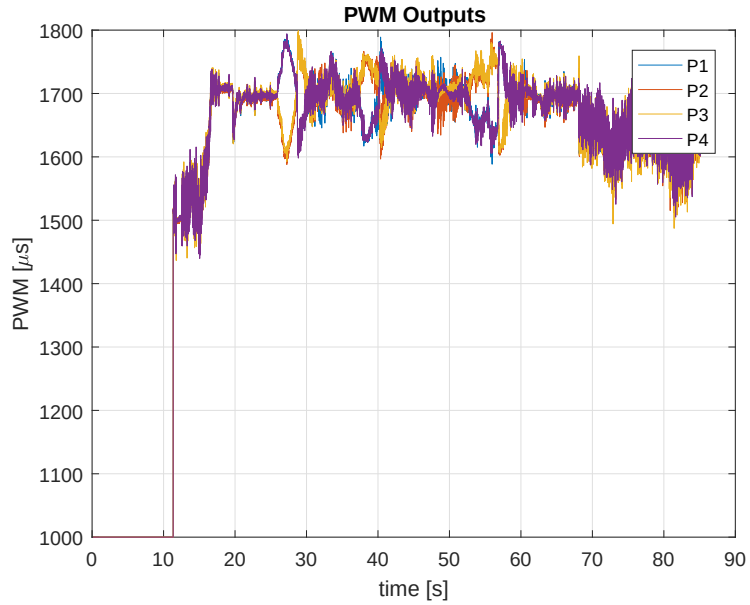


Figure B.13: PWM outputs, SITL

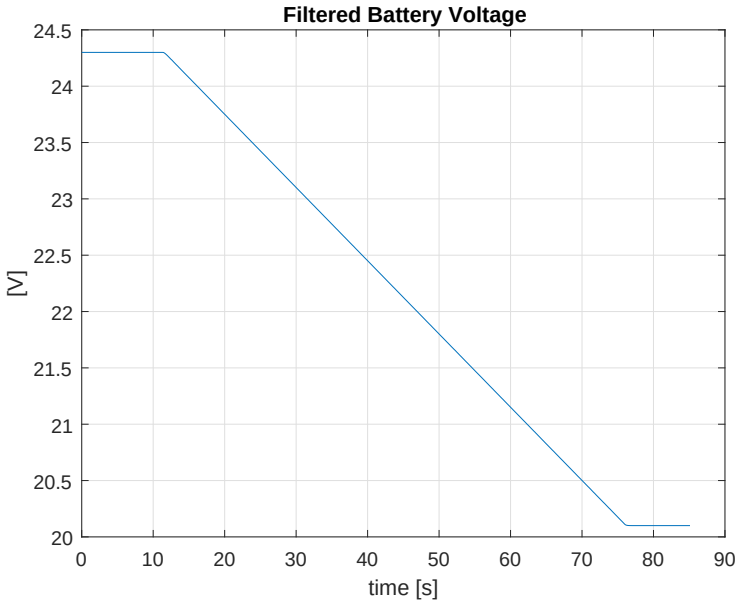


Figure B.14: Voltage, SITL

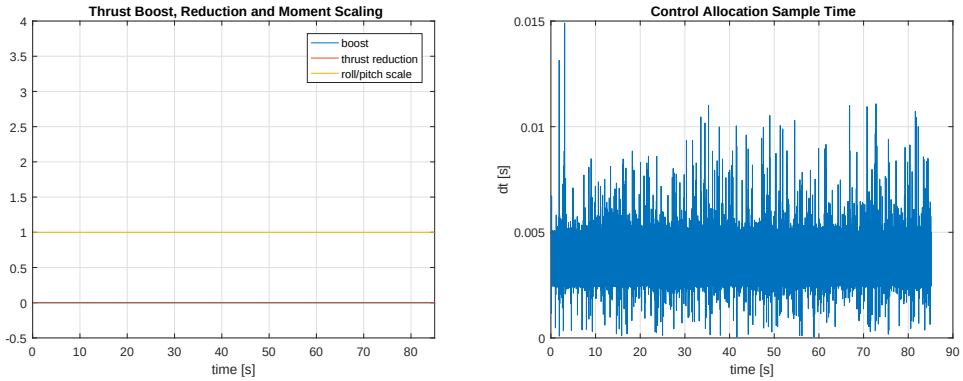


Figure B.15: Control allocation stats, SITL