Optimal Design and Model Predictive Control of an Over Actuated UAV for Internal Wind Turbine Blade Inspection

MCE4-1028

Department of Energy Technology, June 2021

Masters thesis

AALBORG UNIVERSITY
STUDENT REPORT

Fifth year of study w/
School of Engineering and Science (SES)
Department of Energy Technology
Pontoppidanstræde 111
9220 Aalborg
http://www.tnb.aau.dk

Title:

Optimal Design and Model Predictive Control of an Over Actuated UAV for Internal Wind Turbine Blade Inspection

Project:

Masters thesis

Project period:

February 2021 - June 2021

Project group:

MCE4-1028

Members:

Daniel Revsbech Pedersen

David Ársælsson

Jesper Friis Ankersen

Supervisor: Petar Durdevic Løhndorf

Synopsis:

Internal wind turbine blade inspection is a time consuming and costly process which can result in injuries of personnel performing the inspection manually. To accommodate for these complications, this project propose an over actuated UAV design to perform the inspection autonomously. The design is consisting of 8 rotors, where the individual rotor orientation are determined through an optimization algorithm. A control structure is presented for attitude and position tracking control containing an unscented Kalman filter for state estimation, a augmented Kalman filter for translational disturbance estimation and 2 model predictive controllers, one for attitude and one for position control. The model predictive controllers are tuned by the use of Bayesian optimization. For simulating a wind turbine blade inspection the drone is provided with a reference corresponding to maintaining a constant heading while flying at a constant velocity through an oscillating blade. From simulation results the developed MPC controllers were found sufficient for stabilization and semi-aggressive trajectory tracking.

Publications: Uploaded digitally.
Page number: 164
Appended document: 38 pages
Completed: 28. May 2021

Preface

This project is a 10. Semester Master thesis written by MCE-1028 at the Department of Energy Technology with specialization in Mechatronic and Control Engineering at Aalborg University in 2021. This project was not a predefined project proposal supplied by the department at the University, but was proposed by the authors. Furthermore, the project was conducted in collaboration with our supervisor Petar Durdevic Løhndorf ph.d., which is based at the Esbjerg branch of Aalborg University. The authors wish to thank Morten Veng, a 10. semester student at Aalborg University, for his inputs and sparring throughout this project.

Reading Guide

In this report the IEEE citation standard is utilized. In the PDF version each citation is a clickable hyperlink which directs to the bibliography which is provide in the report. Furthermore, internal references are clickable which redirect to the related section, equation, etc. A list of abbreviations is given at the beginning of each chapter. A table of of content is provided in the beginning of the report which has clickable hyperlinks which direct to the appropriate chapter or section. All modeling and all simulations were conducted in MATLAB and Simulink, and all figures were created with Inkscape. The project was written in LATEX at Overleaf.com. Matrices and vectors are denoted in bold in equations and inline text. Within Appendix A a list of the assumptions is stated, which are referred to throughout the project. Due to the pandemic "corona virus" this project is purely a simulation based project.

Abstract

Indvendig inspektion af vindmøllevinger kan være en tidskrævende process med nedetid op til 8 timer, og der kan forekomme ulykker og personskader under manuel inspektion. En metode, der bliver forsket i, er brugen af droner til at fortage inspektionen, hvilket er inspirationen til dette projekt. For flyvining med en drone indeni en vindmøllevinge, er der blevet udarbejdet et nyt dronedesign baseret på 8 rotorer, der tillader individuel aktuering for alle frihedsgrader. Orienteringen af rotorerne er udvalgt igennem en optimeringsprocess, der involvere kobling, inputbegrænsninger og effektivitet. Projektet er udført på simuleringsbasis, og derfor er der opsat en detaljeret simuleringsmodel af dronens og rotorernes dynamikker, målestøj samt rotationelle og translatoriske forstyrrelser. Det har været et mål med projektet, at dronen skulle kunne følge et traktorie inde i en vindmøllevinge, mens den modstår forstyrrelser og samtidig følger vindmøllevingens svingninger. Først er der designet et "unscented Kalman filter" for at reducere målestøj, hvor filteret er tunet med en designed "grid search" algoritme. For at modstå translatorisk forstyrrelse i form af jord- og lofteffekt er den globale z position benyttet i et augmenteret Kalman filter for at estimere forstyrrelserne. For at afværge den translatoriske forstyrrelse er estimatet heraf benyttet i positionsregulatoren. For reguleringsstrukturen er der benyttet 2 "model predictive controllers", hvor den ene er tilknyttet regulering af dronens rotationelle position, mens den anden er relateret til regulering af dronens translatoriske position. Begge regulatorer indeholder en række tunings parametre, som skal bestemmes og til denne process, er simuleringsmodellen approksimeret som en Gaussisk process regression, der bliver optimeret ved hjælp af Baysisk optimering. Det præsenterede design var observeret til stadig at have indflydelse på translatoriske bevægelser, når den blev udsat for rotation. Til gengæld var den i stand til at styre alle translatoriske frihedgrader individuelt uden brug af rotationer. Med den rotationelle regulator var dronen i stand til at opretholde den samme orientation. Der forekommer stadig rotationelle afvigelser på maksimalt 0.3 $[-]$ enheder i quaternioner tilsvarende udsving på 22° for rul, tilt og drej udsat for den værste omstændigheder og 0.12 $[-]$ tilsvarende 8.8° for normale omstændigheder. Den translatoriske regulator udviste flukture ringer på $\pm 4 - 6$ $[cm]$, men har vist sig at kunne følge frekvensen for vindmøllevingens svingninger. Dronen opnåede en længde på 24 $[m]$ udsat for de værste omstændigheder og 45 $[m]$ under normale omstændigheder.

Contents

Preface **ii**
Reading Guide . . . ii

Abstract **iii**

Contents **iv**

1 Introduction **1**
1.1 Objectives . . . 3
1.2 Description of Environment . . . 4
1.3 Relevant Designs . . . 8
1.4 Problem Statement . . . 10

2 Design **11**
2.1 Design Objectives . . . 11
2.2 Trajectory . . . 12
2.3 Orientation and Displacement of the Rotors . . . 12
2.3.1 Application Dependent Design . . . 12
2.3.2 Forces and Torques . . . 15
2.3.3 Choice of Motor/Propeller Configuration . . . 21
2.4 Size Estimation . . . 24
2.5 Component List and Weight Estimation . . . 25

3 Modelling of the UAV **26**
3.1 Modelling of Rotor Configuration . . . 26
3.1.1 Force and Torque Coefficients . . . 27
3.1.2 Dynamic Response . . . 27
3.1.3 Discrete Rotor Approximation . . . 29
3.2 Derivation of Translational Dynamics . . . 30
3.3 Derivation of Attitude Dynamics . . . 32
3.3.1 Estimation of Inertia . . . 35
3.4 Open Loop Test of the Mathematical Model . . . 37
3.5 Disturbance Model . . . 41
3.5.1 Translational Disturbances . . . 41
3.5.2 Rotational Disturbances . . . 44
3.6 Measurement Noise . . . 46
3.7 Model Summary . . . 47

4 Control Considerations **51**
4.1 Control Criteria . . . 51

4.2 Literature Review on Relevant Control Methods . . . 53
4.2.1 Under Actuated UAVs . . . 53
4.2.2 Fully and Over Actuated UAVs . . . 54
4.3 Availability of Measurements and Trajectory Generation . . . 56
4.4 Control Allocation . . . 58
4.5 Closed Loop Bandwidth Assessment . . . 59
4.6 Proposed Control Structure . . . 61

5 Sensor Measurement and State Estimation 63
5.1 State and Disturbance Estimators . . . 63
5.2 UKF Algorithm . . . 65
5.3 Implementation . . . 68
5.4 Augmented KF for Disturbance Estimation . . . 73
5.4.1 General Kalman Filter Structure . . . 73
5.4.2 Augmented Kalman Filter . . . 74

6 MPC and Trajectory Tracking 79
6.1 Introduction to MPC . . . 79
6.2 Attitude MPC . . . 82
6.2.1 Prediction Model . . . 82
6.2.2 Cost Function . . . 83
6.2.3 Optimization Process . . . 83
6.2.4 Optimization Method . . . 86
6.3 Position Tracking MPC . . . 87
6.3.1 Prediction Model . . . 87
6.3.2 Cost Function . . . 87
6.3.3 Optimization Process . . . 87

7 Bayesian Optimization and Controller Tuning 89
7.1 Bayesian Optimization . . . 90
7.2 Attitude MPC Tuning . . . 94
7.3 Position Tracking MPC Tuning . . . 97
7.4 Results . . . 98
7.4.1 Stabilization Problem with Disturbances . . . 99
7.4.2 Normal Conditions Simulation with disturbances . . . 103
7.4.3 Worst Case Conditions with Disturbances . . . 107

8 Discussion 111
8.1 On Assumptions Regarding Measurements and Trajectory Generation . . . 111
8.2 Design Considerations . . . 112
8.3 On The Disturbance Models . . . 113
8.4 On Results and Bayesian Optimization . . . 114

9 Conclusion 115

10 Future Work 117

Bibliography 119

A Assumptions **127**

B Design Tables **129**
B.1 WTB Dimensions . . . 129
B.2 Feasibility . . . 130
B.3 Rotor Candidates . . . 130

C Quaternions: Definition and Mathematics **134**
C.1 Rotational matrix . . . 136
C.2 Motor Mixing Algorithm . . . 136

D Simulation Results of the Mathematical Model **137**

E 2nd Order Accuracy of UKF **142**

F SVD Design **144**

G MATLAB Code **146**
G.1 Attitude MPC MATLAB Code . . . 146
G.1.1 C Code Generation of Attitude MPC . . . 149
G.2 Position Tracking MPC . . . 149
G.3 Design Optimization Algorithm . . . 151
G.4 Ground/Ceiling Effect Code . . . 154
G.5 UKF Design . . . 155
G.5.1 UKF Accelerations . . . 155
G.5.2 UKF Estimation . . . 157
G.5.3 UKF Grid Search Algorithm . . . 158
G.6 Bayesian Optimization . . . 160
G.6.1 Bayesian Optimization Test Code . . . 160
G.6.2 Bayesian Optimization Attitude MPC . . . 161
G.6.3 Bayesian Optimization Position Tracking MPC . . . 162

1 | Introduction

Abbreviation	Meaning
DOF	Degrees Of Freedom
ESC	Electrical Speed Controller
FAUAV	Fully Actuated Unmanned Aerial Vehicle
LQR	Linear Quadratic Regulator
MPC	Model Predictive Control
OC	OmniCopter
PID	Proportional Integral Derivative
RL	Reinforcement Learning
SLAM	Simultaneous Localization And Mapping
SMC	Sliding Mode Control
UAV	Unmanned Aerial Vehicle
UV	Unmanned Vehicle
WT	Wind Turbine
WTB	Wind Turbine Blade

Table 1.1: Abbreviations used in the following chapter.

The global wind turbine (WT) industry is expected to have an annual growth rate of around 5.3 % between 2019 and 2025 [1]. As the market is growing so is the need for optimization of certain issues regarding the WT industry, such as fault detection, inspection, maintenance etc.

The main issue addressed in this report is internal wind turbine blade (WTB) inspection using unmanned aerial vehicles (UAV).

The WTBs make up for $15 - 20$ % of the total cost of WTs and further have one of the highest failure rates. Failures affect the efficiency of the WT as well as the electricity production and due to the failure rate and the component cost, failure is one of the reasons for the high maintenance cost of the WT [2–4]. Multiple factors cause the WTB to fail such as cycling loads, weather conditions as strong winds, rain/hail and lightning strikes or from bird collision [5]. The typical failures of the WTB is surface damages as leading edge erosion or cracks and internal damages as delamination of plies, debonding of parts or cracks [3]. Minor failures could lead to secondary failures that could damage the WT and evidently the WT could fail completely making inspection necessary for optimal operation [6].
Typically three methods are used for inspection: Rope inspection, ground based inspection using high resolution cameras and UAV inspections [7]. Exterior and interior WTB inspection is usually carried out manually [8], but these are comprehensive, requires safety equipment and well coordinated execution. To accommodate for the comprehensive work and safety risks,

companies are emerging bringing new UV technologies designed for WTB inspection [9, 10]. The focus in this work is based on interior WTB inspection. Interior inspection implies that the inspection is conducted in one or more coherent constrained environments, such as blade, nacelle and hub. A constrained environment is throughout this report defined as a confined space with finite boundaries for all directions. The inside of a WTB is considered as a constrained environment.

Manual inspection of the interior of a WTB is a time consuming and a costly process. According to [11] it can take between 1-2 hours to inspect each blade plus the downtime for shutting down and starting up the WT. Furthermore, the maintenance crew are facing risk of injury while conducting the inspection. Those include e.g. fires, falling down, lightning strike or static electricity build-up, and due to the confined space, lack of oxygen can occur [11, 12]. To reduce the expenses consisting with downtime and the risk of injuries, the usage of UVs for inspection purposes (i.e wheeled vehicles) have been researched and tested [13–15], where [13, 14] focuses on interior blade inspection. In [16] a comparison is conducted on the efficiency between UV and manned inspection, in which it was concluded equal efficiency regarding identification of errors. However expenses and time consumption were found to be reduced with UVs. A research topic of interest is the usage of UAV for the inspection as a UAV is more flexible regarding the inner structure and the position of the WTB.

Utilizing UAVs, the design becomes a trade-off between flight time, equipment, size and weight. Small sized UAVs are capable of inspecting smaller confined spaces but the smaller size results in lesser lift force to carry sensors, cameras and batteries. By increasing the size of the UAV enable more room to attach batteries extending the flight time but limit accessibility of the WTBs narrow points e.g. the tip of the blade. Different complications are present for manual flight with a UAV within a confined space. First of is the extensive camera solution and illumination of the surroundings needed for the operator to navigate the UAV and avoid obstacles. Additionally is the constant radio communication between the operator and the UAV required for the steering of the UAV. To overcome some of the complications from manual flight, a solution is for the UAV to fly autonomously. It requires additional hardware for the UAV to fly autonomously as sensor or camera units must be utilized for localization and obstacle avoidance [11, 17, 18].

As the UAV is flying within a constrained environment it is affected by disturbances in form of ground, ceiling and Wall effect, which need to be accounted for [19]. Different methods of control have been researched and utilized to accommodate for these disturbances. Simple linear PID controller designs are commonly used for the altitude and attitude control of a quadcopter but due to non-linearities and external disturbances a more robust control methods have been investigated. To improve the robustness of the controller design, fuzzy-logic and self-tuning algorithms have been utilized to correct the controller gains during flight while others have researched multivariable control methods as the linear quadratic regulator (LQR) [20–23]. Another approach for robust control is to utilize non-linear control methods as sliding mode controllers (SMC) as seen in [24–26]. Aside the disturbances, a UAV flying within a WTB has to accommodate for the infrastructure of the WTB and the constant changing boundaries as the WTB oscillates when exposed to wind. Different methods for path planning and obstacle avoidance have been researched for indoor flight of quadcopters. In [27], different bug algorithms are investigated for flying a quadcopter in an indoor environment with different obstacles where the quadcopter follows a general direction until it encounters an object. Hereafter, it follows the edge of the object to avoid collision. [28, 29] are utilizing image processing or point clouds for simultaneous localization and mapping (SLAM) algorithms to determine and navigate around obstacles. Different variations of model prediction control (MPC) such as simple linear MPC

or robust nonlinear MPC have been investigated to follow predicted trajectories for object avoidance as seen in [30]. Another researched approach is reinforcement learning (RL) to determine the flight direction for the shortest distance around obstacles using Q-learning [31, 32]. A common problem for the controller designs and object avoidance algorithms for UAVs utilized for indoor flight is to get reliable measurements from sensor units and estimation of the UAVs position. The accuracy of the sensor units is not investigated further as it is out of the scope of this project.
Research has been made regarding UAV inspection in constrained spaces such as pipes [33, 34]. Companies as [35, 36] even specializes in UAV tasks performed in confined areas, however most of these are underactuated which impacts the agility of the vehicle. An open issue that remains in this context is the use of fully actuated UAVs for confined space inspection. In this work a UAV with higher degrees of freedom actuation than a traditional quadcopter is investigated. The fully or overactuated UAV is referred to as a FAUAV within this work.
In [37–39] a five blade FAUAV is designed. It consists of two counter-rotating propellers in the middle to generate thrust and control the yaw orientation whereas three ducted fans with variable angles enable lateral flight and the control of roll and pitch. In [40] an overactuated quadcopter is presented. Each propeller can tilt around its own arm due to a mounted servo motors. In [41] a classic hexrotor is designed, but with all the propellers tilted at a fixed angle. In [42] six propellers are each mounted on tetrahedrons which are joined together at a single point in the middle. Lastly, in [43, 44] examples of an eight rotor FAUAV is presented. Each rotor is oriented in a fixed position, which gives the design 6 DOF. Those designs are examples of existing FAUAV designs. In the survey [45], existing designs of FAUAVs are analysed in terms of the respective design optimization method used. In section 1.3 some of these designs are further discussed.

1.1 Objectives

In the introduction the advantages of more efficient WTB inspection solutions are described as well as an analysis of the most recent research within the field of UAV design, robust control and trajectory tracking. The main objectives of this work are stated below.

- 1. Reference tracking
- 2. Disturbance rejection
- 3. Application dependent design

Reference tracking is one of the primary issues since the whole WT structure is affected by external forces such as wind. The oscillations could cause a UAV inside the wing to crash into the walls of the WTB. Therefore the control solution must be able to track or react to a fast changing reference. In Section 1.2 further analysis is made on the WTB including determination of the nature of the oscillations, that is expected to occur.

Another aspect of importance is the aerodynamic effects that are caused by the rotors air streams. Such disturbance could cause system instability. Therefore robust control are necessary for UAV inspection in a confined space to reject the aerodynamic disturbances. For this to be investigated a disturbance model is presented in Section 3.5 based on ceiling, ground and interfering airflow effects.

In order to ease the control criteria, which are described in detail in Section 4.1, a relevant design is desired. A relevant design in this context is a design which allows to decouple translational motion from rotational motion. This is desired to independently control all 6DOF and as discussed earlier such designs falls into the category of FAUAVs. In Section 1.3 a set of candidate designs are investigated and discussed in order to a suitable design for WTB inspection. In order to evaluate a suitable solution of the UAV design for performing internal WTB inspection, the complications regarding the environment must be clarified to establish desired design criteria and requirements. The inspection environment is analyzed in Section 1.2.

1.2 Description of Environment

The UAV is desired to fly within a WTB from a 5MW offshore WT with blade type LMH64-5 [46]. The WTB is consisting of spar caps, shear webs and the outer shell. The spar caps and shear webs are extending from the root and close to the tip of the WTB as illustrated in Figure 1.1. [47]

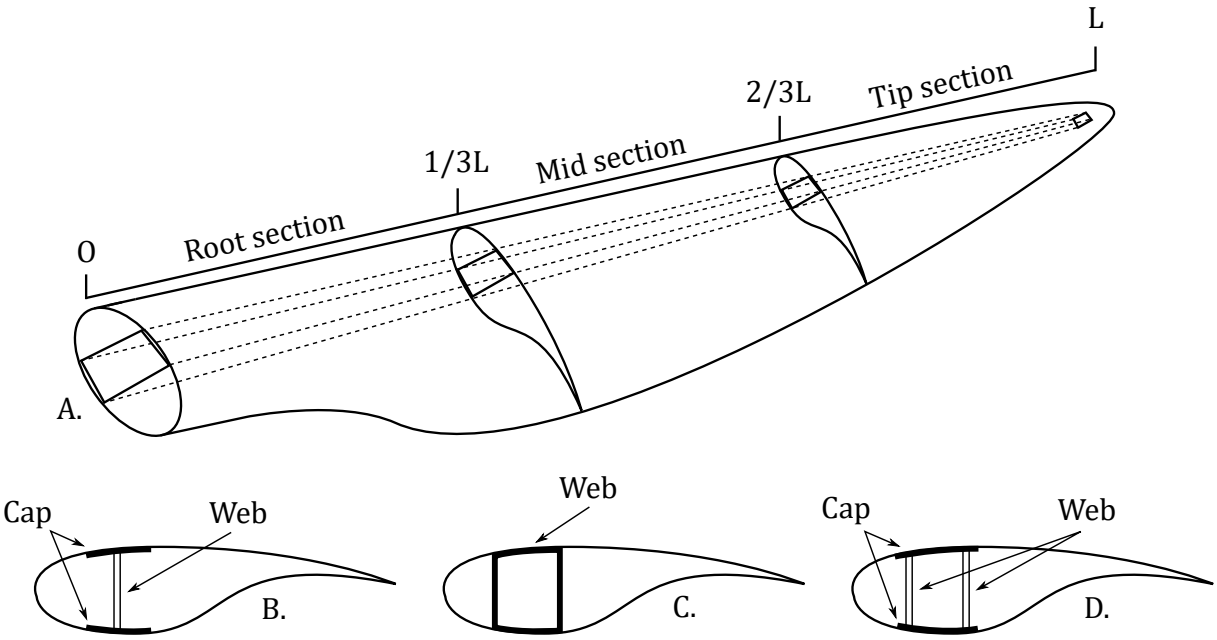


Figure 1.1: Sketch of the internal structure of a WTB. A) The WTB divided in sections. B) Closed shell. C) Box spar. D) Load carrying shell. [11, 48]

where O is considered the zero length and L the total length of the blade. Three different caps-webs configurations are shown in Figure 1.1; the closed shell utilizing 1 web, the box spar with the web in the shape of a box and the load carrying shell utilizing 2 webs. The caps-webs are used to increase the blades strength, relieve the outer shell of shear stresses and keep the aerodynamic shape. The outer shell is consisting of a downwind suction side and an upwind pressure side both mounted to the caps-webs configuration by adhesive layers as illustrated in Figure 1.2.

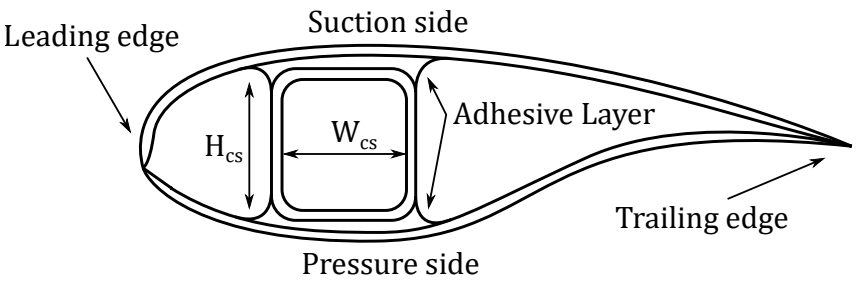
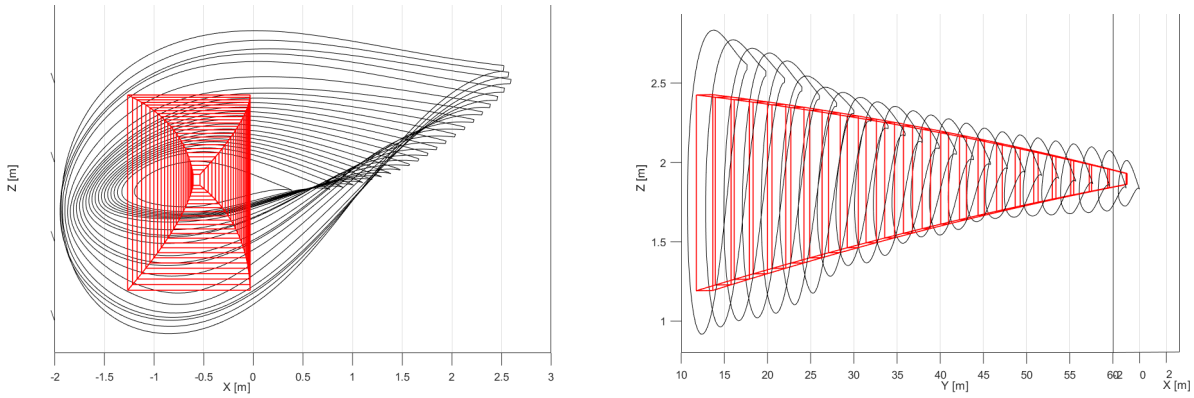


Figure 1.2: Cross section of WTB with box spar. [47]

Figure 1.2 show the cross section of a WTB with box spar. Here W_{cs} is the inner length and H_{cs} is the inner height of the box spar. By this configuration and the load carrying shell 3 confined spaces occur, one at the leading edge, one at the trailing edge and the inner of the caps-webs configuration.

For the scope of this project the constrained environment to be inspected is the construction of a box spar approximated as a tube with width W_{cs} and height H_{cs} (**Assumption 1**). The tube is chosen as a quadratic cross section thus $W_{cs} = H_{cs}$. Within the quadratic tube the UAV has to compensate for aerodynamic disturbances caused by the propellers and the oscillating motion of the WTB as it is affected by wind. The cross sectional area of the tube is largest at the root and narrows as the UAV moves along the length of the blade as illustrated in Figure 1.3. The calculations of the outer constraints of the quadratic tube is described in Section 2.2.



(a) Confined space in the x/z plane.

(b) Confined space in the y/z plane.

Figure 1.3: Quadratic tube within the WTB.

In the following three subsections further considerations of the WTB are described. Firstly, the frequency and amplitude of the oscillations which the WTB can experience are discussed. Secondly, the aerodynamic disturbances which the UAV can experience inside the WTB are listed. Lastly, size considerations of the UAV are discussed and which limitations there are.

WTB Natural Frequency and Amplitude of Deflection

Due to wind conditions the WTB is oscillating thus bending the confined space, the UAV is flying within as illustrated in Figure 1.4.

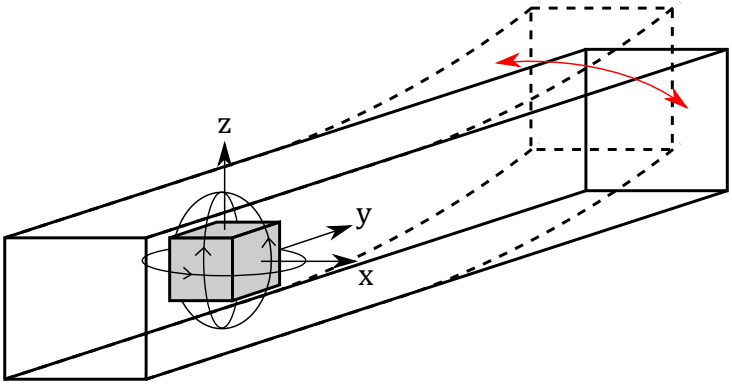


Figure 1.4: Illustration of the UAV within a WTB.

Due to the moving environment the expected deflection and frequency of the dominating WTB oscillations are analyzed for the blade. The oscillations are utilized as a reference for the UAV to track to avoid colliding with the inner wall of the WTB. Figure 1.5 illustrates a WT where 2 different deflections are defined, firstly the out of plane deflection and secondly the deflection in the rotor plane direction, these are commonly referred to as flapwise and edgewise deflection, respectively.

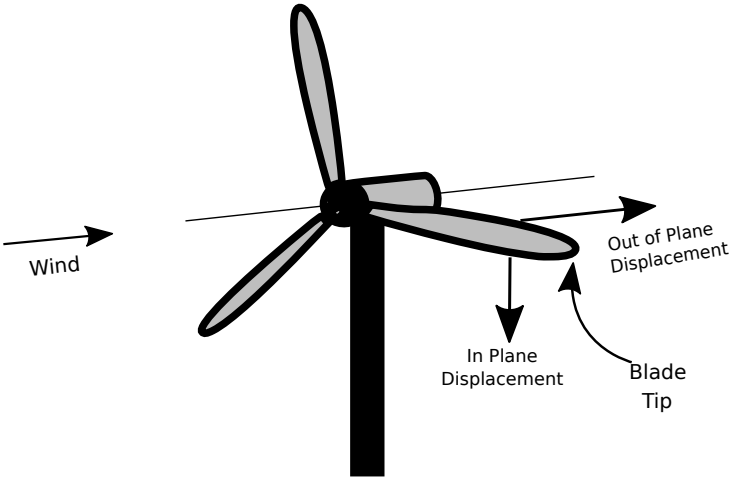
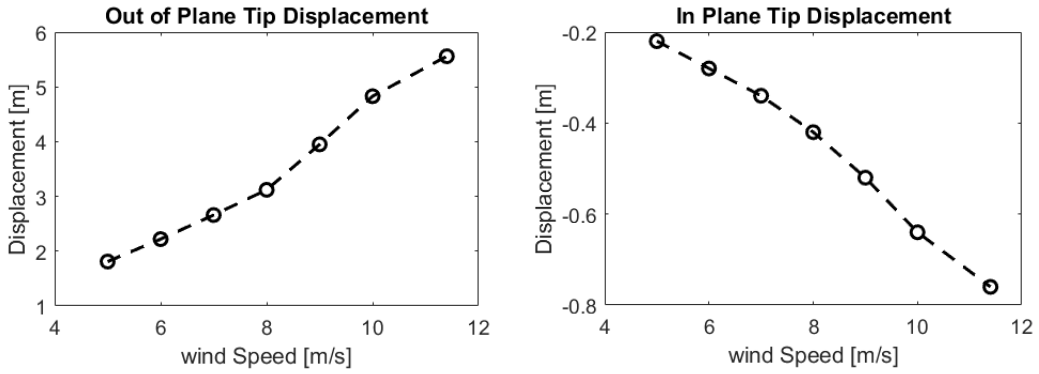


Figure 1.5: Illustration of deflection directions of a WTB.

To establish the amplitude of the WTBs oscillations for both flapwise and edgewise deflection a study of the NREL 5 $[MW]$ [49] WT is utilized. The data for the WTB deflections are seen in Figure 1.6.



(a) Out of plane displacement for NREL 5 $[MW]$. [49]

(b) In rotor plane displacement for NREL 5 $[MW]$. [49]

Figure 1.6: Displacement of WTB caused by wind disturbances.

From another study on the NREL 5 $[MW]$, a calculation of WTB eigenfrequency (ω_{WTB}) is performed. The collective flapwise eigenfrequency is determined to be 0.699 Hz [50]. From Figure 1.6a it is shown that at a wind speed of 12 $[^m/_s]$ the displacement in the out of plane direction is 6 $[m]$ and for the rotor plane direction the displacement is -0.8 $[m]$ at 12 $[^m/_s]$ wind speed which is illustrated on Figure 1.6b. The usage of the deflections and the natural frequency for trajectory design is further described in Section 2.2.

Aerodynamic Disturbances

Within a confined space the UAV will encounter aerodynamic disturbances. Among those are air resistance which causes drag and airflow interference which can be a result of the UAV being in close vicinity of walls or if the rotors are tilted and the air streams crosses. As for the drag the design of the UAV can be minimized by streamlining the design, thus reducing the flow separation caused by the frontal area of the UAV. For the airflow disturbance the diameter of the propellers is the dominating factor where the size determines the distance from e.g. wall where the disturbances affect the UAV. Furthermore, by tilting the rotors the wake of two rotors might collide which disrupts the airflow. This can be reduced by considering the angle of the tilting and the distance between the rotors. The disturbances presented above will be explained further in Section 3.5.

Size

The size of the UAV is limited as it do not only need to fit inside the WTB but further needs space to maneuver to follow a trajectory and compensate for the disturbances. Contrary the UAV has to carry sensor equipment such as camera solutions for trajectory control and the inspection process increasing the weight and the required size of the UAV. The number of rotors and the length of the propellers is a trade off between size and generated thrust whereas a quadcopter can be made compact compared to a hexacopter but cant carry the same load. Another aspect is the UAVs capability for full actuation. For a UAV utilizing tilting rotors the maneuverability is enhanced but these UAVs require additional actuators for the tilting motion of the rotors thus increasing the size and weight.

Optics and Positioning

The closed space within the WTB is dark and thereby a light source is required to illuminate for the cameras. Additionally if the cameras for either trajectory tracking or inspection are fixed at the UAVs body the UAV has to adjust the line of sight to the WTB. As the WTB can attain different positions at downtime the UAV has to fly horizontally, vertically or at an angle. For configurations such as the quadcopter, multiple fixed cameras or cameras with the capability of rotating independently are required for inspection at all blade positions. Another solution for the camera heading is to utilize an omnicopter (OC). By this the heading of the UAV can be chosen for all orientation thus aligning the camera with the WTB. The OCs are larger than the quadcopter configuration due to the amount of rotors needed and the complexity is increased with the different structure but it allows the omnidirectional flight.

1.3 Relevant Designs

The UAV is desired to control all 6DOF independently to avoid roll or pitch for moving in the x or y directions. Further, it is desired to align the UAVs heading with the line of flight. For this purpose it is of interest to investigate design possibilities for an OC. In this section a list of existing designs is briefly described from which inspiration for a new design is found.

ArduCopter: Fixed bidirectional rotors:
The ArduCopter has an unique form, which consists of 6 tetrahedrons that are joined together in the center of gravity. It consists of 6 bidirectional rotors with one rotor attached to each of the tetrahedrons. The UAV utilizes ArduPilot software for control. Otherwise there exists scarce amount of information about design and control as the UAV was designed by a hobbyist. The ArduCopter has the advantage of omnidirectionality which in principal could point a single inspection sensor at any surface inside the WTB. However, the limited description of this system as well as the use of bidirectional propellers makes the design less desirable. [42]

HexaCopter: Beam fixed bidirectional rotors:
This OC is consisting of 6 opportunistically aligned bidirectional rotors fixed on a beam shaped body, 3 rotors at each end of the beam. To enable the omnidirectional flight an optimization algorithm is designed to determine the orientation and displacement of the rotors from the center of mass. The optimization algorithms purpose is to maximizes the control wrench. For control, a simple PID control structure is adopted. An advantage is the high wrench but the beam structure is designed for carrying tools and induce increased size compared to the traditional hexacopter structure. [51]

HexaCopter: Tilting unidirectional rotors:
The first tilting rotor hexacopter to be experimentally tested is given in [52]. The mechanical structure is compact and allows to decouple rotational motion from translational. The tilting property allows the hexacopter to be more efficient when hovering compared to other fixed tilt omni-vehicles, as the rotor thrust vectors can be parallel with the gravity vector. The control structure is employing a position PID control law and a cascaded attitude control law. An advantage for this design is the high efficiency while hovering since the plane unidirectional rotors can be aligned perpendicular to the gravitational vector. However, the tilting property of the rotors require 6 additional actuators which yields a total of 12 actuators.

HeptaCopter: Fixed unidirectional rotors:

The heptacopter is an OC consisting of 7 rotors and has an unique asymmetric design. By adding a 7th rotor, the UAV gets full actuation with unidirectional rotors. Furthermore, the UAV can handle actuator failure. The unique form which the OC has is a result of an optimization algorithm to minimize the volume. The attitude loop is cascaded and highly nonlinear where an adaptive backstepping control strategy is utilized. Lastly the stability of the system is proofed utilizing Lyapunov theory. An advantage is that the rotors are unidirectional. However, the performance of the UAV has never been experimentally tested. [53]

OctoCopter: Box fixed bidirectional rotors:
Articles [43, 54] both concerns the same work and OC where [54] contains the most detailed description. The design is optimized based on wrench, which implies that the OC outputs concerning translational and rotational movement can be reached equally efficient with the actuator inputs. Conclusively thrust and torque generation of the OC in [43, 54] was proven feasible, however it was not possible to fully decouple translational from rotational motion with the proposed control structure. Whereas the OC presented in [44] are optimized based on minimization of power usage. This article also provides a more comprehensive actuator model taking into account interacting air flows from other nearby rotors. Article [44] analyzed the energy consumption of the design at different trajectories. However, it was not concluded whether the airflow interaction model was able to decouple translational and rotational motion through control allocation. An advantage of the structural design of the OCs discussed here is that their inertia tensor contains equal diagonal elements, and due to the propeller orientations and the bidirectional property all angle orientations are an operating point. The bidirectional rotors will have increased response time when a change of direction is required which approximately doubles the settling time of the actuator (**Assumption 2**). Another disadvantage is that the propellers are angled in different and opposing directions making hovering inefficient and therefore decreases flight time. [43, 44, 54]

OctoCopter: Beam fixed bidirectional rotors:
An OC designed and optimized with the purpose of mounting tools for physical interaction while in midair. One example is to utilize the UAV for the steering of a mechanical arm as seen in [55]. It is consisting of 8 bidirectional rotors each with 4 propeller blades. The OC has a beam shaped body with 4 rotors fixed at both ends. The orientation and displacement from the center of mass are determined through an optimization algorithm designed to maximize the minimum applicable thrust and torque possible. The design ensure an omnidirectional flight with the possibility of carrying additional weight but due to the 8 rotors with 4 propeller blades each the design required a large amount of battery capacity. The authors did not find any commercial rotor-battery combination for the UAV to build and test a functional prototype. Further they had to design a selective mapping algorithm to partially subdue the destabilizing effects of the ESC-induced singularities caused when the rotors change spin direction and the thrust is temporarily lost. [56]

For internal inspection of WTBs the following properties are seen as advantages:
Bidirectional rotors introduce singularities due to the rotor switching direction and also have a slower response than unidirectional rotors. But, bidirectional rotors have increased actuation as all rotors can contribute to thrust in all DOF which is found more desirable. A possible solution to overcome the disadvantages of the bidirectional rotors is to incorporate a strategy to reduce the frequent changing of direction of the rotors.
Compact design are of high importance, since the larger the UAV is the less of the WTB can be

inspected. For this a more symmetrical design is preferred where the width, length and height of the UAV is of similar size. This would make rotational motions easier as the cross sectional area of the tube gets smaller.
The flight time is important as the UAV do not only have to fly a far distance inside the WTB but further fly at a speed which allows the inspection sensors to detect damages. The lesser rotors and a light design increase the flight time.
A symmetric structure as the construction used in articles [43, 54] are utilized as a starting point for the design process. Due to the 8 rotors the flight time is of concern, firstly because of the redundancy for utilizing 2 additional inputs than outputs. Secondly because of the increased energy consumption gained from the relation in Equation 2.25.

With the initial structure of the design the rest of the work presented throughout this project will revolve around the design and control objectives stated in Section 1.4

1.4 Problem Statement

The main objectives was briefly discussed in Section 1.1. In this section the objectives is specified and decomposed into subproblems necessary to be investigated in order to obtain our main objectives. This work is divided into 2 main parts. The first part concerns the design procedure and modelling of an overactuated UAV for the application of conducting internal WTB inspection. The second part involves the choice and design of a relevant control structure. The objectives regarding design is stated separately from the objectives from the objectives regarding control, as seen below.

Design Objectives:

1. Capability of producing thrust and torque for all DOF.
2. Determine optimal rotor locations and orientations based on the application.
3. Realistic proposal of UAV build.
4. Derive a realistic simulation model capturing dominant dynamics for the purpose of control design.

Control Objectives:

1. Follow aggressive and smooth trajectory of sinusoidal tendency with a frequency corresponding to ω_{WTB}.
2. Keep the orientation aligned with the inertial frame.
3. Robustness towards unknown and unmodelled disturbances.

The design and control objectives are further elaborated in Sections 2.1 and 4.1, respectively.

2 | Design

The following chapter elaborates the application and the chosen trajectory for the UAV. Next, the design is presented where the orientation of the rotors are determined from an application dependent optimization algorithm. Lastly, the components for the UAV build is described with a weight and size estimate of the UAV leading to the choice of a motor/propeller combination.

Abbreviation	Meaning
BLDC	BrushLess Direct Current
CAM	Control Allocation Matrix
DOF	Degrees Of Freedom
ESC	Electrical Speed Controller
LED	Light Emitting Diode
NRDM	Normalized Rotor Displacement Matrix
RGB	Red Green Blue
UAV	Unmanned Aerial Vehicle
WTB	Wind Turbine Blade

Table 2.1: Abbreviations used in this chapter.

2.1 Design Objectives

In this section the chosen design objectives are elaborated. The design objectives are the guidelines throughout Chapter 2 and 3 and lay the ground work for the later presented control structure. The design objectives are stated in the list below.

1. Capability of producing thrust and torque for all DOF.
2. Determine optimal rotor locations and orientations based on the application.
3. Realistic proposal of the UAV build.
4. Derive a realistic simulation model capturing dominant dynamics for the purpose of control design.

The first design objective revolve the initial UAV design and the consideration for attaining a fully actuated UAV. This involves number of rotors and general shape of the structure.

The second design objective involves the optimal position of the rotors depending on the stated application. For this an optimization algorithm must be defined which is subject to

physical concerns to ensure efficient flight such as decoupling of outputs, rotor efficiencies and input constraints.

The third design objective relates to the realization of the UAV design. Even though this project is purely simulation based and a prototype of the final design is not realized it is still desired to base the simulations on realistic measures. To obtain the proposal of the UAV build, components have to be evaluated for both flying and inspection together with weight and size estimations.

The fourth design objective concerns the establishment of a realistic simulation model to design a sufficient control structure for the application. For this, the descriptive equations of the UAV dynamics are required together with approximations of the rotor model, disturbances affecting the UAV both rotational and translational and measurement frequencies and uncertainties.

2.2 Trajectory

The trajectory planning of the UAV for the flight within the WTB is designed to fly at the center of the quadratic tube along the y axis as seen in Figure 1.4. The trajectory is based on the deflection of the WTB as described in Section 1.2. The trajectory has to change with the deflection for both the x and z direction to prevent the UAV from colliding with the walls of the WTB. To mimic the wind disturbances the trajectory is consisting of 2 sinusoidal waveforms, one for the x direction and one for the z direction. Both have a frequency of the WTBs natural frequency at $0.699\,[Hz]$ with the amplitude as a function of the UAVs y position and the maximum tip deflection. The trajectory is expressed as in Equation 2.1 and 2.2 and illustrated at Figure 2.1. (**Assumption 3**)

$$x(y,t) = x_{def}\left(\frac{y}{L}\right)^2 sin(\omega_{WTB}t) \tag{2.1}$$

$$z(y,t) = z_{def}\left(\frac{y}{L}\right)^2 sin\left(\omega_{WTB}t + \frac{\pi}{2}\right) \tag{2.2}$$

where x_{def} and z_{def} are the maximum tip deflections in the x and z directions respectively, ω_{WTB} is the natural frequency in radians, t is time and L is the total length of the WTB. The y position divided by the total length is representing the change in amplitude as the UAV moves further into the WTB.

Figure 2.1 is an example of how the trajectory change as the UAV is moving at a constant velocity through the WTB. Combining the 2 parts it is evident that the x and z reference is constantly moving with the amplitude as a function of the UAVs y position. To perform the inspection the y reference is chosen as a ramp input as the UAV is desired to fly slow and continuously through the WTB. Set points of the y position are stated in Table B.1.

2.3 Orientation and Displacement of the Rotors

2.3.1 Application Dependent Design

As mentioned in Section 1.3 the initial design of the UAV, presented in this project, is based on the symmetric design seen in [43, 54], where bidirectional electronic speed controllers (ESCs),

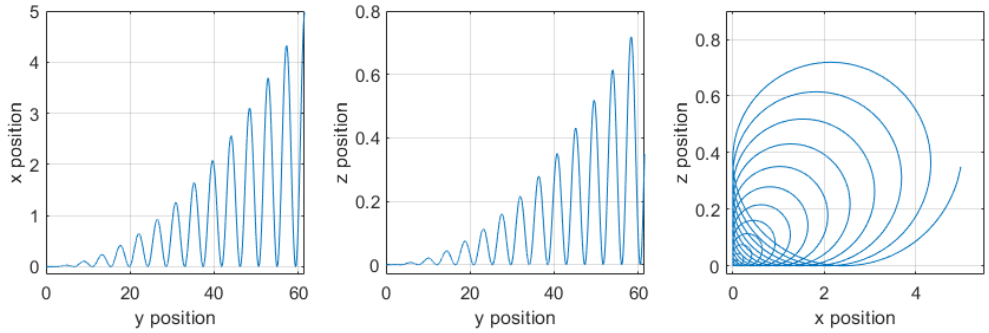


Figure 2.1: Illustration of the trajectory caused by wind for the UAV moving through the WTB at a constant velocity.

brushless dc motors (BLDCs) and propeller configuration is used. This configuration is too used in this work. For the inspection it is desired to align the body frame y axis with either the positive or negative WTB y axis. The alignment of the y axis is convenient for the orientation of the camera solutions but requires some extend of omnidirectional flight for the UAV to inspect the WTB at any given position. Inspection at all WTB positions is possible if the UAV is fully actuated at an angular span of 90° rotation revolving the x axis. This can be visualized in Figure 2.2. If a rotation around x_b occurs in the interval $[0°; 90°]$ all directions contained in the blue areas can be reached. If rotation revolving z_b has a full span (i.e 360°) a rotation of 180° around z_b will render the directions marked by green reachable.

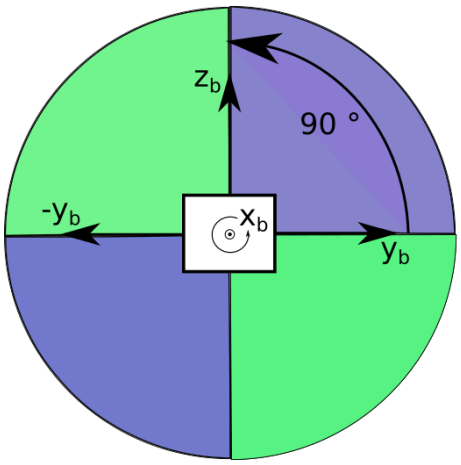


Figure 2.2: Illustration of how the UAV can translate in the y_b direction when a 90° rotation around x_b can be applied.

The design presented throughout this project is optimized for the case where the WTB is at a horizontal position and thereafter analyzed to determine actuation at other angular positions. The proposed design is illustrated in Figure 2.3.

(a) Illustration of the rotor design.

(b) Illustration of the UAV design.

Figure 2.3

In Figure 2.3a the rotor fastening is illustrated whereas the motor is fixed to a base of carbon fiber. From the base, 2 bent bars are extruded at each side with a protective ring attached, surrounding the rotor to prevent the propeller from colliding with the ground, ceiling and walls. As seen from Figure 2.3b the bars are further acting as the fixing point for the rotors and the arms. The orientation of the rotors are for illustration whereas the exact orientations are presented through the later optimization algorithm. The hollow cube at the center of 2.3b is representing the case for the hardware, batteries and sensors utilized for the UAV. The illustrations in Figure 2.3 are conceptual.

The UAV is desired to be fully actuated to control all 6DOF independently and minimize necessary motions. Additionally it is desired to follow a moving trajectory meanwhile compensate for aerodynamic disturbances. To achieve these desired specifications the orientation of the rotors are of great importance. Within this section the orientation of the rotors are chosen through an optimization algorithm where the optimization criteria are determined for flight at a horizontal WTB position as illustrated in Figure 2.4.

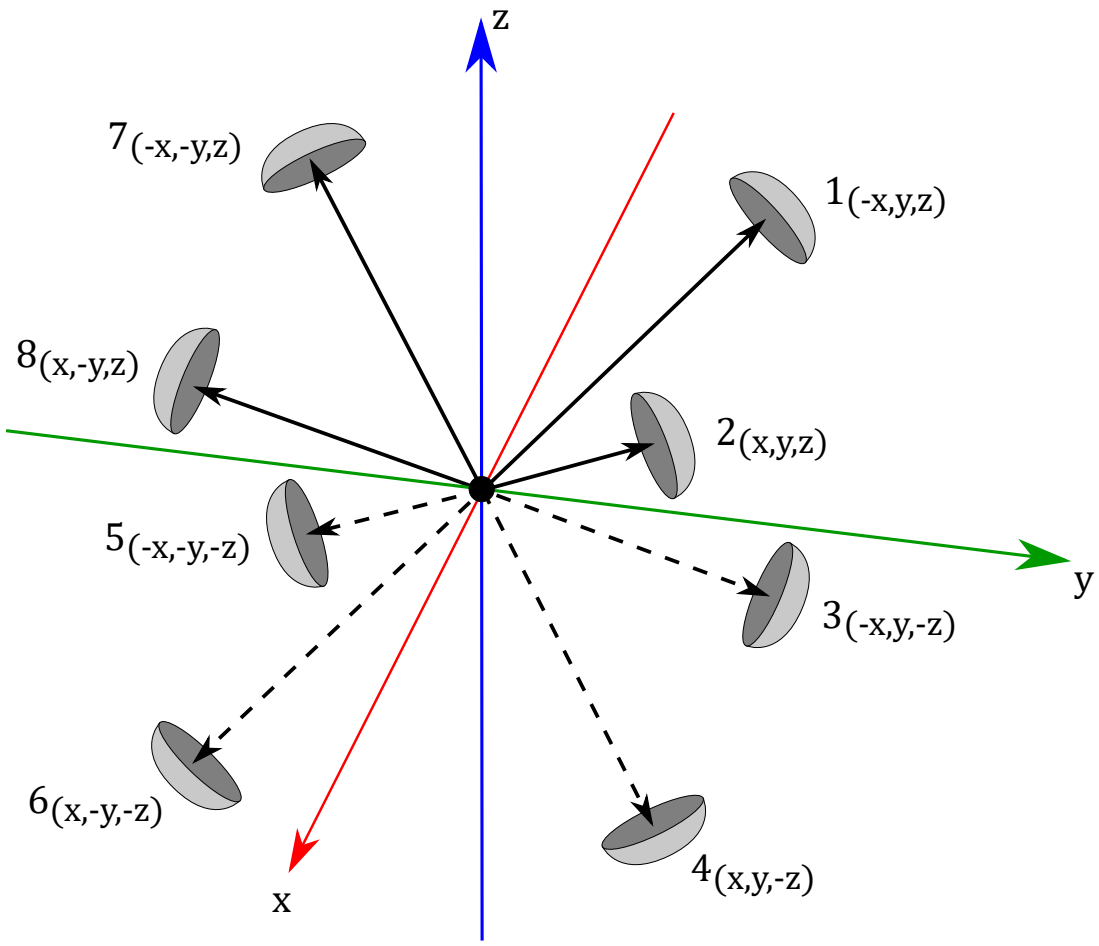


Figure 2.4: Illustration of possible thrust directions for each rotor.

In Figure 2.4 the UAV is seen with the body frame y axis aligned with the inertial frame y axis with the x axis pointing out of the plane. The circumference of the grey semi-spheres are seen as the possible positions attainable for the individual rotor direction where each point on the sphere denotes a direction which the rotor thrust vector can point towards. The subscript for each rotor number express which direction in the x,y,z plane the rotor is placed. This will allow any of the 8 rotors to affect in the 3 translational directions. The span of the orientations are chosen due to the application of WTB inspection. For inspection, the UAV has to move back and forth inside the WTB thus must be allowed to generate thrust in the y direction. Further, due to the deflection appearing in the inertial frame x and z axis the UAV must have significant actuation in these directions as well. The span of orientations each rotor can attain are utilized for the optimization algorithm. The optimization method is presented in [54]. The method is adopted in this work as it allows to adjust the design constraints as desired. The placement of the rotors and their orientations are described by a displacement matrix $\boldsymbol{P}$ and a orientation matrix $\boldsymbol{X}$ respectively. These matrices are defined from the relationship between the individual propeller thrust and the body frame forces and torques.

2.3.2 Forces and Torques

The 8 rotors is generating a thrust and torque when rotating which affect the motion of the UAV. If all the rotors thrust vectors are pointing in the body frame z-axis as for conventional quad- or hexacopters, motions in the x and y axis are only obtainable by coupling these motions

of roll and pitch. The design of the UAV is desired to achieve full controllability for traveling in all translational directions while keeping the angular positions steady. Therefore the rotors have to be configured to generate thrust in all 3 translational directions at all possible orientations. The force vector containing the forces acting in the x, y and z body frame directions is calculated as in Equation 2.3.

$$\boldsymbol{F} = \sum_{i=1}^{N} F_{prop,i}\boldsymbol{x}_i \tag{2.3}$$

where N is number of rotors, $F_{prop,i}$ is the force generated for each rotor and $\boldsymbol{x}_i$ is the respective orientation of the rotor given as the rotor disk normal.
The torque affecting for roll, pitch and yaw is calculated as Equation 2.4.

$$\boldsymbol{\tau} = \sum_{i=1}^{N} F_{prop,i}(\boldsymbol{p}_i \times \boldsymbol{x}_i) + \kappa F_{prop,i}\boldsymbol{x}_i \tag{2.4}$$

here $\boldsymbol{p}_i$ is the respective displacement of the rotor from the center of mass of the UAV and κ is the thrust-to-drag coefficient of the propeller given as the relation between the torque and thrust coefficients as seen in Equation 2.5. [43, 54]

$$\kappa = \frac{K_\tau}{K_F} \tag{2.5}$$

where K_τ is the torque coefficient and K_F is the force coefficient. These coefficients are determined in Section 3.1.1 from experimental data of the rotor configuration chosen in Section 2.3.3.

From Equation 2.3 and 2.4 the forces $\boldsymbol{F}$ and torques $\boldsymbol{\tau}$ can be described as in Equation 2.6.

$$\boldsymbol{F}_{gen} = \begin{bmatrix} \boldsymbol{F} \\ \boldsymbol{\tau} \end{bmatrix} = \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{P} \times \boldsymbol{X} + \kappa \boldsymbol{X} \end{bmatrix} \boldsymbol{F}_{prop} \tag{2.6}$$

where $\boldsymbol{X}$ is the orientation matrix containing the normal of each propeller disk and $\boldsymbol{P}$ is the propeller displacement matrix. For the displacement matrix the structure of the UAV is seen symmetric in which all rotors are placed with the same displacement from the UAVs center of mass. The displacement matrix is seen in Equation 2.7.

$$\boldsymbol{P}_1 = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \end{bmatrix} \tag{2.7}$$

The gain of the displacement matrix comes from the normalization as the length of each vector is defined as the length 1. The gain is calculated from Equation 2.8 and 2.9.

$$|\boldsymbol{p}_i| = \sqrt{p_x^2 + p_y^2 + p_z^2} = 1 \tag{2.8}$$

$$p_x^2 = p_y^2 = p_z^2 \rightarrow \sqrt{3p_x^2} = 1 \rightarrow p_x^2 = \frac{1^2}{3} \rightarrow p_x = \frac{1}{\sqrt{3}} \tag{2.9}$$

The orientation matrix is seen in Equation 2.10.

$$\boldsymbol{X}_1 = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_8 \\ b_1 & b_2 & b_3 & b_4 & b_5 & b_6 & b_7 & b_8 \\ c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 \end{bmatrix} \tag{2.10}$$

where the coefficients a_i, b_i and c_i determine the effect of the inputs in the x, y and z direction and these are used as the variables for the optimization. The length of each column vector in the orientation matrix have to be 1 and the 3 coefficients thus have to satisfy the relation stated in Equation 2.11.

$$\sqrt{a_i^2 + b_i^2 + c_i^2} = 1 \quad , \quad i = 1, 2, \ldots, 8 \tag{2.11}$$

The values the a_i, b_i and c_i coefficients can attain are determined from the area span of each rotor defined at Figure 2.4. The rotors are paired into groups of 2 to reduce the complexity of the optimization algorithm whereas 4 different coefficient spans are defined as seen in Table 2.2.

Rotor 1 and 2	**Rotor 3 and 4**	**Rotor 5 and 6**	**Rotor 7 and 8**
$0 \leq c_1 \leq 1$	$-1 \leq c_3 \leq 0$	$-1 \leq c_5 \leq 0$	$0 \leq c_7 \leq 1$
$b_{1,max} = \sqrt{1 - c_1^2}$	$b_{3,max} = \sqrt{1 - c_3^2}$	$b_{5,min} = -\sqrt{1 - c_5^2}$	$b_{7,min} = -\sqrt{1 - c_7^2}$
$0 \leq b_1 \leq b_{1,max}$	$0 \leq b_3 \leq b_{3,max}$	$b_{5,min} \leq b_5 \leq 0$	$b_{7,min} \leq b_7 \leq 0$
$a_1 = \sqrt{1 - b_1^2 - c_1^2}$	$a_3 = \sqrt{1 - b_3^2 - c_3^2}$	$a_5 = \sqrt{1 - b_5^2 - c_5^2}$	$a_7 = \sqrt{1 - b_7^2 - c_7^2}$
$c_2 = c_1$	$c_4 = c_3$	$c_6 = c_5$	$c_8 = c_7$
$b_2 = b_1$	$b_4 = b_3$	$b_6 = b_5$	$b_8 = b_7$
$a_2 = -a_1$	$a_4 = -a_3$	$a_6 = -a_5$	$a_8 = -a_7$

Table 2.2: Attainable values of the orientation matrix coefficients.

The definitions of the coefficients spans are utilized for the optimization algorithm whereas different orientation matrices are calculated through a total of 8 loops.

As for now the 8 loops are only utilized to create an array of possible orientations but these are further substituted into Equation 2.6 to determine the relation between the individual propeller thrust and the body frame forces and torques. The relationship matrix is called the control allocation matrix (CAM) and is represented in Equation 2.12.

$$\boldsymbol{V} = \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{P} \times \boldsymbol{X} + \kappa \boldsymbol{X} \end{bmatrix} \tag{2.12}$$

The objective of the optimization is to maximize the CAM from which the propeller thrusts are used efficient for most possible motion. For each CAM the rows are evaluated by summation of the absolute coefficients thus a measure is given for each motion. Figure 2.5 illustrates a flow chart of the optimization procedure. Firstly, the code is initialized with the desired number of

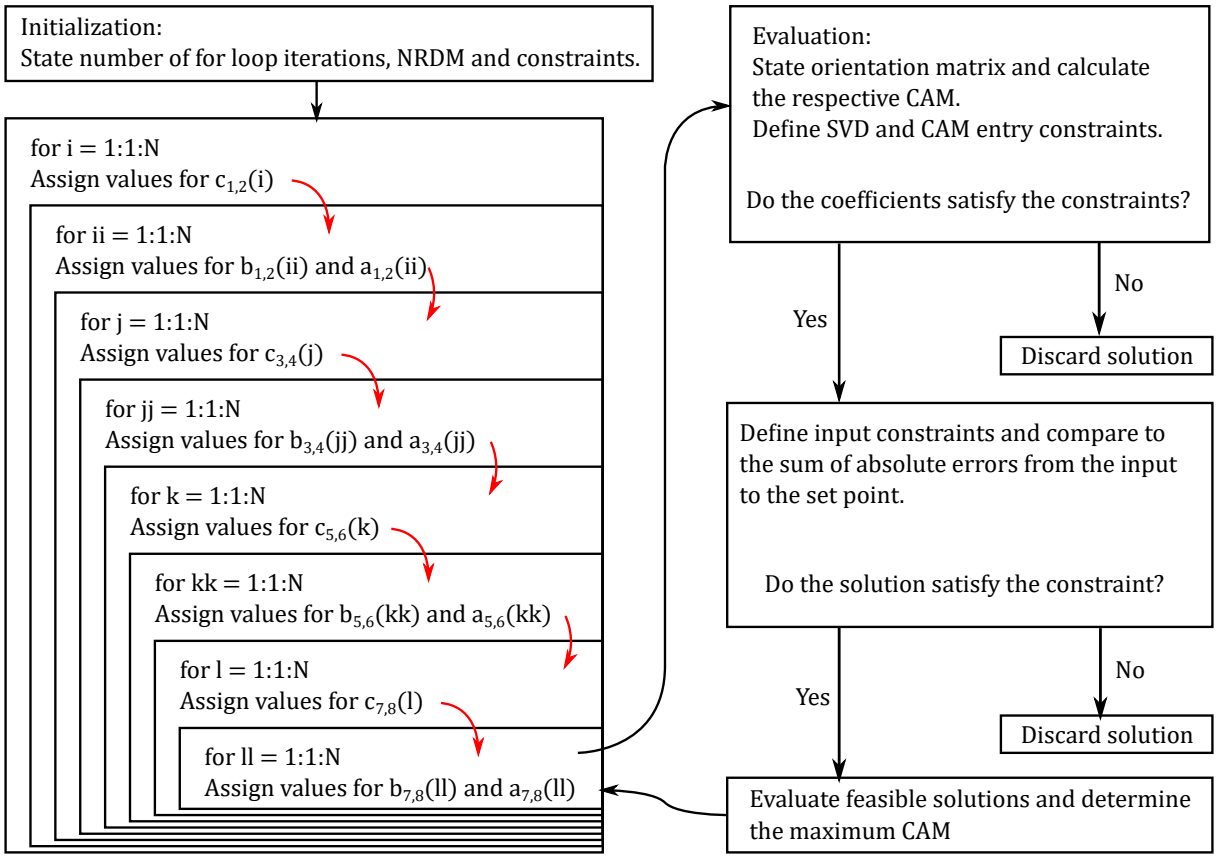


Figure 2.5: Visualization of design procedure.

for loop iterations, the normalized rotor displacement matrix (NRDM) and constraints. Then 8 for loops assign values for the a_i, b_i and c_i coefficients based on the definitions in Table 2.2. Thereafter the orientation matrix is stated and is used to calculate the CAM. The constraints listed in Equation 2.14 - 2.17 are then defined. The first constrain is the condition number based on the ratio between the highest and the lowest singular value and is defined not to be larger than 7.5. The next three constrains relate to x, y, z direction values and are set to be not equal 0 (**Assumption 4**). These constraints are defined to reduce the coupling of outputs, to reduce effort of the inputs and to utilize all rotors for every translational motion. The concept behind SVD and the reasoning for using the condition number as a constraint are further elaborated in Appendix F. Thereafter, the coefficients are evaluated with respect to the constrains and if they satisfy the given constrains then the optimization algorithm continues, otherwise the solution is discarded. Figure 2.6 illustrates the inputs required for the UAV design to hover. To the left the inputs of the maximum CAM without constraints are illustrated. The design might state the maximum values of the CAM but some of the inputs required for hover exceed the saturation limit while other are close to zero. By introducing the constrains listed in Equations 2.14 - 2.17 it can be seen in Figure 2.6 (right) that all the inputs are within the saturation limits.
The next step in Figure 2.5 is to define an input constrain which is seen in Equation 2.18 and then compare each input to a chosen set point. If any of the input errors violates the constraint, the solution is discarded. On Figure 2.7 the orange lines are the set points and the error for each input is the distance from input to set point. This constraint should draw the inputs away from zero crossing and the saturation limit. If the all the errors fulfills the given constrain then the CAM is stored. At the the end of the algorithm all suitable solutions are evaluated and the maximum CAM is chosen as the final design.

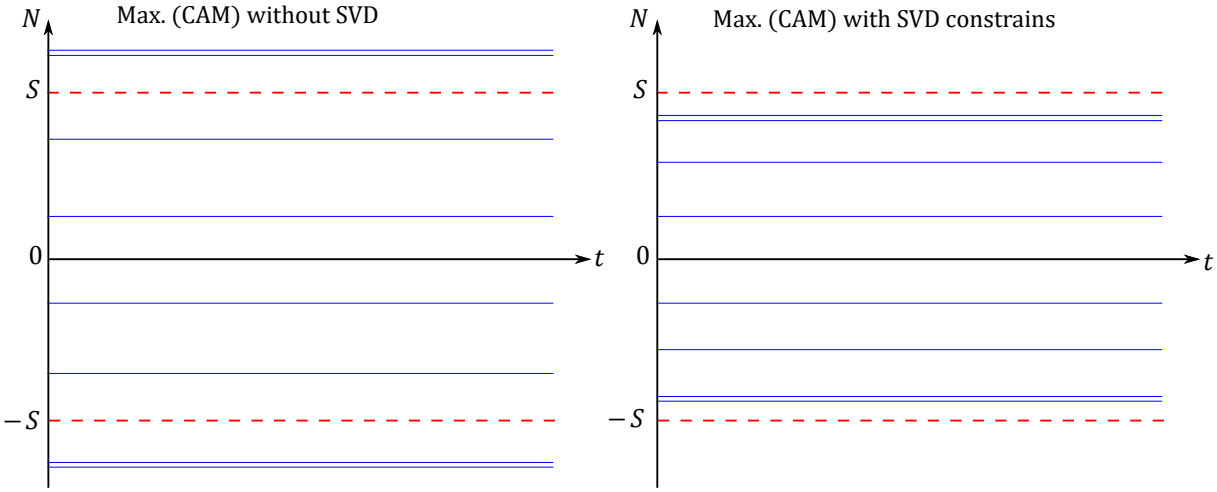


Figure 2.6: Illustration of arbitrary inputs with and without SVD constrains.

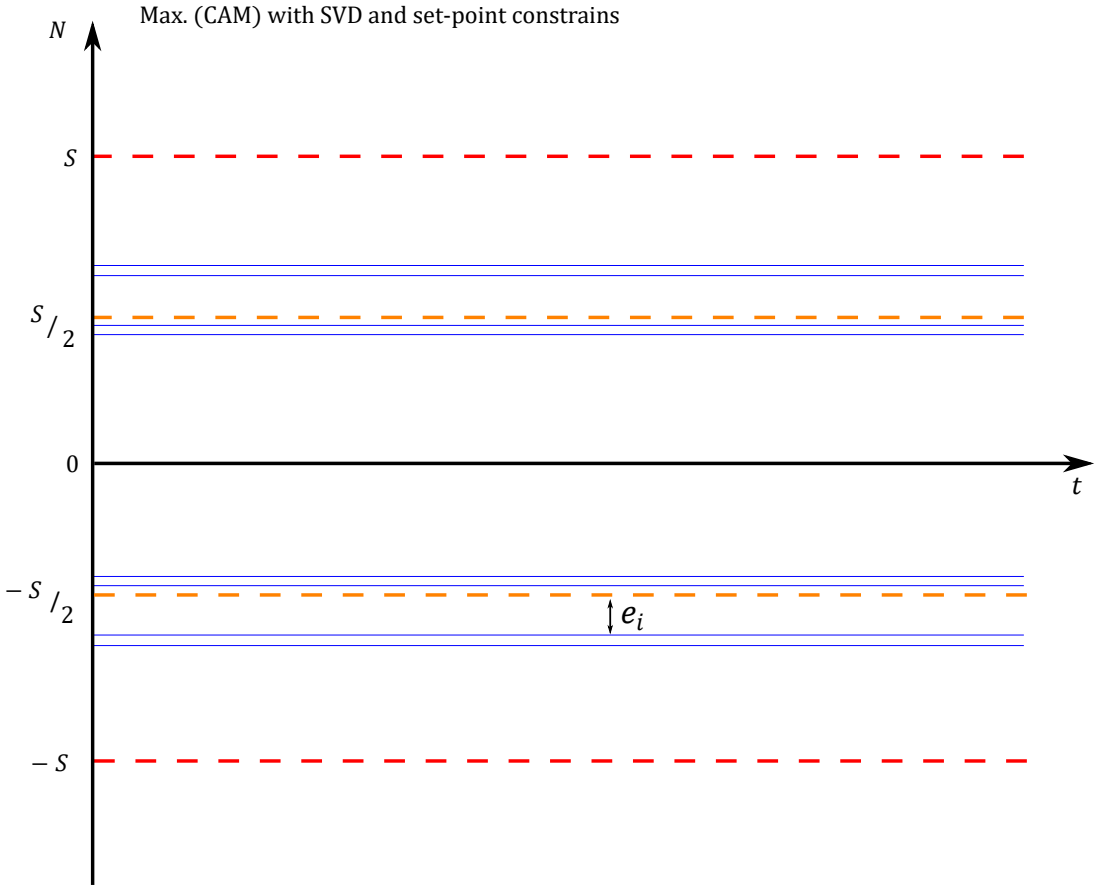


Figure 2.7: Illustration of arbitrary inputs with a set point constrain.

With the given constraints the optimization problem is stated as a maximization as seen in Equation 2.13 subject to Equations 2.14-2.18.

$$\max_{x} \quad \sum_{j=1}^{8}\sum_{i=1}^{6}|\boldsymbol{V}_{ij}| \tag{2.13}$$

$$\text{subject to:} \quad \gamma(\mathbf{V}) < 7.5 \tag{2.14}$$

$$v_x \neq 0 \tag{2.15}$$

$$v_y \neq 0 \tag{2.16}$$

$$v_z \neq 0 \tag{2.17}$$

$$\frac{S}{2} - 2.5 < |e_i| < \frac{S}{2} + 2.5 \tag{2.18}$$

From the stated maximization the final CAM is given in Equation 2.19 and the corresponding thrust vectors are seen in Figure 2.8 and Figure 2.9.

$$\boldsymbol{V} = \begin{bmatrix} 0.839 & -0.839 & 0.839 & -0.839 & 0.839 & -0.839 & 0.839 & -0.839 \\ 0.484 & 0.484 & 0.484 & 0.484 & -0.484 & -0.484 & -0.484 & -0.484 \\ 0.250 & 0.250 & -0.250 & -0.250 & -0.250 & -0.250 & 0.250 & 0.250 \\ -0.126 & -0.144 & 0.144 & 0.126 & -0.126 & -0.144 & 0.144 & 0.126 \\ 0.345 & -0.334 & -0.334 & 0.345 & -0.345 & 0.334 & 0.334 & -0.345 \\ -0.202 & 0.207 & -0.207 & 0.202 & 0.202 & -0.207 & 0.207 & -0.202 \end{bmatrix} \tag{2.19}$$

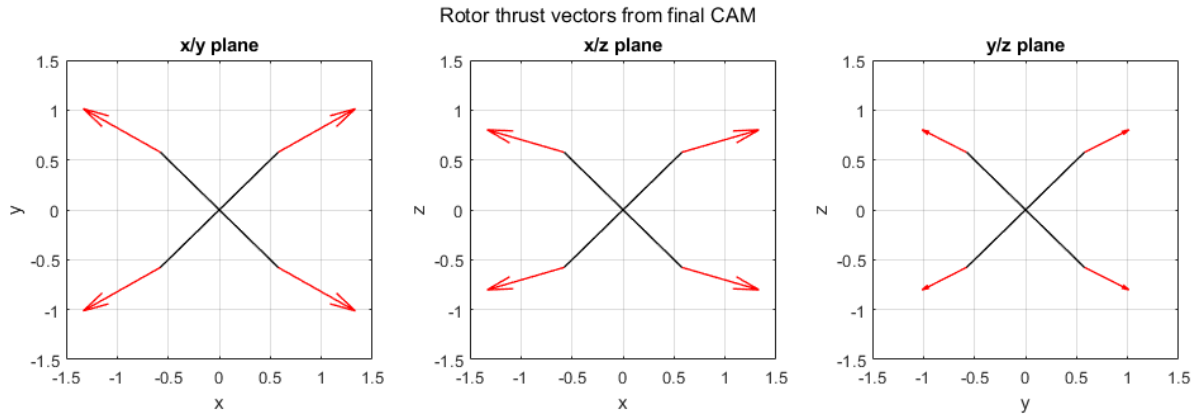


Figure 2.8: Rotor thrust vectors.

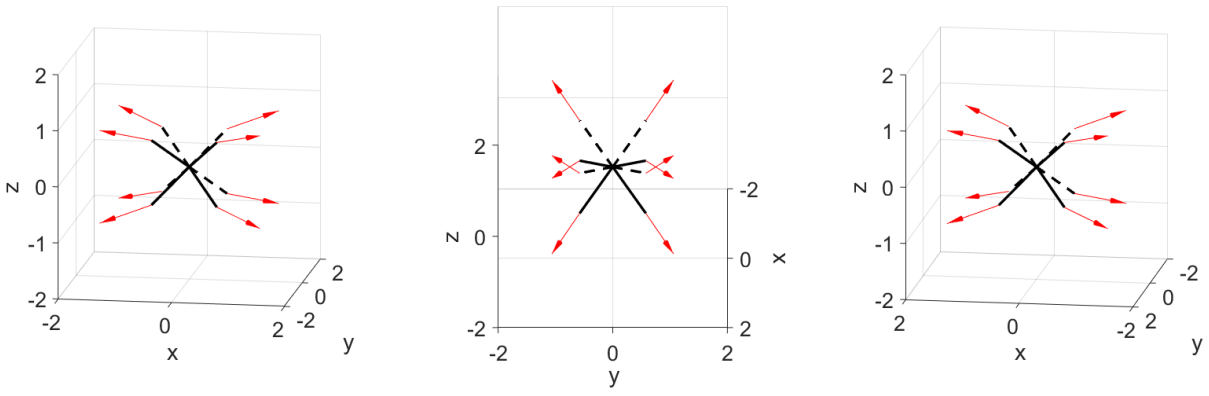


Figure 2.9: 3D plot of the final design. The solid black lines are pointing outside the plane while the dashed black lines are pointing into the plane.

The final design is optimized from the WTB at a horizontal position but if the WTB position change and the body frame y axis is still desired to be aligned with the WTB length the effects of the CAM would change in the inertial frame. To analyze the CAM in the inertial frame at different positions the orientation and displacement matrices are rotated around the x axis with an increment of 10° from a span of $-90°$ to 90° seen from the original position. At first the rotation matrix referring to rotation revolving the x axis is defined in Equation 2.20

$$\boldsymbol{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{bmatrix} \tag{2.20}$$

The orientation and displacement matrices can then be calculated as in Equation 2.21 and 2.22 whereas the CAM is seen in Equation 2.23.

$$\boldsymbol{P}_i(\phi) = \boldsymbol{R}_x(\phi)\boldsymbol{P} \quad , \qquad \phi = -90° + i \cdot 10° \quad , \qquad i = 0, 1, \ldots, 18 \tag{2.21}$$

$$\boldsymbol{X}_i(\phi) = \boldsymbol{R}_x(\phi)\boldsymbol{X} \tag{2.22}$$

$$\boldsymbol{V}_i(\phi) = \begin{bmatrix} \boldsymbol{X}_i(\phi) \\ \boldsymbol{P}_i(\phi) \times \boldsymbol{X}_i(\phi) + \kappa \boldsymbol{X}_i(\phi) \end{bmatrix} \tag{2.23}$$

Within Table B.3 the summation of absolute CAM values for all translational and rotational motions are stated. Here the coefficients for thrust in the x direction and torque for performing roll are constant at all rotational positions. The thrust for the z direction is seen to be reduced for rotations ±30° but then starts to increase for the additional rotational positions. The same tendency appear for torque in the pitch direction. The thrust for the y direction is though seen to decrease the further the UAV is rotated from the original orientation. This is also the case for the performance of yaw motion. Though the coefficients change with rotation, all DOF are obtainable for the chosen design at an angular span of 180°. The final CAM is not only used to determine the individual propeller thrusts to generate force and torque in the desired directions but further utilized as a measure for the selection of the motor/propeller configuration for the UAV as described in Section 2.3.3.

2.3.3 Choice of Motor/Propeller Configuration

For the calculation of the CAM the thrust-to-drag coefficient is present. As κ is based on the rotor thrust and torque coefficients it will vary with different rotor configurations. The thrust and torque coefficients are experimentally determined but as no experimental work has been conducted throughout this project experimental data for different motor/propeller configurations have been gathered from RCbenchmarks database [57]. Together with the motor and propeller the ESC is presented for each data set. For the rotors the propeller size is chosen as 5 $[in]$ in diameter based on prior knowledge of thrust generated and the relatively small size to keep the UAV compact. From RCbenchmark 5 different rotor configurations have been selected for comparison:

1. The Carbon Fiber 5030 propeller with a TBS Ethix Mr Steele 2306 2345kv Silk V2 motor and a EMAX BLHeli 80A UBEC 5A 5V ESC.

2. The Gemfan 5040 R propeller with a Hypetrain Blaster 2207 2450Kv motor and a EMAX BLHeli 30A BEC 2A 5V ESC.

3. The HQ Prop 5045 propeller with a DYS Samguk Shu 2306 2500kv motor and a Flycolor XCross Blheli32 50A ESC.

4. The HQ Prop 5050 propeller with a DYS Samguk Shu 2306 2500kv motor and a Flycolor XCross Blheli32 50A ESC.

5. The Gemfan 5050 3-Blades BN propeller with a Hypetrain Blaster 2207 2450Kv motor and a EMAX BLHeli 80A UBEC 5A 5V ESC.

To choose the rotor configuration for the UAV each of the 5 candidates are evaluated at 4 characteristics: Motor efficiency and total efficiency at hover position and maximum thrust and torque. The motor efficiency is related to the power consumption as operating at a higher efficiency increases the flight time while the total efficiency is evaluated to utilize more of the configurations potential. As the UAV has to follow a trajectory meanwhile compensate for WTB deflections and aerodynamic disturbances the maximum thrust and torque are desired to be large to allow faster movements of the UAV and prevent it from colliding with the walls.

To determine the efficiencies at hover position the capable thrust capacity in the positive z axis is utilized. For this the coefficient v_z from the feasibility analysis seen in Table B.3 is seen as a measure of propeller forces acting against the gravitational force. The mean of the propeller force needed to hover is then calculated as in Equation 2.24.

$$f_{p,hover} = \frac{mg}{v(0°)_z} = \frac{0.8422kg \cdot 9.8\frac{m}{s^2}}{2} = 4.13N \tag{2.24}$$

The gravitational force is calculated by the gravitational acceleration g and the mass m whereas the mass is found in Section 2.5. With the mean of propeller force the rotational speed for hovering can be determined from the data set and further utilized to determine the efficiencies at the given speed. At first, each candidates force measurements are plotted with the corresponding rotational speed as seen in Figure 2.10 with the corresponding data sets seen in Appendix B.3.

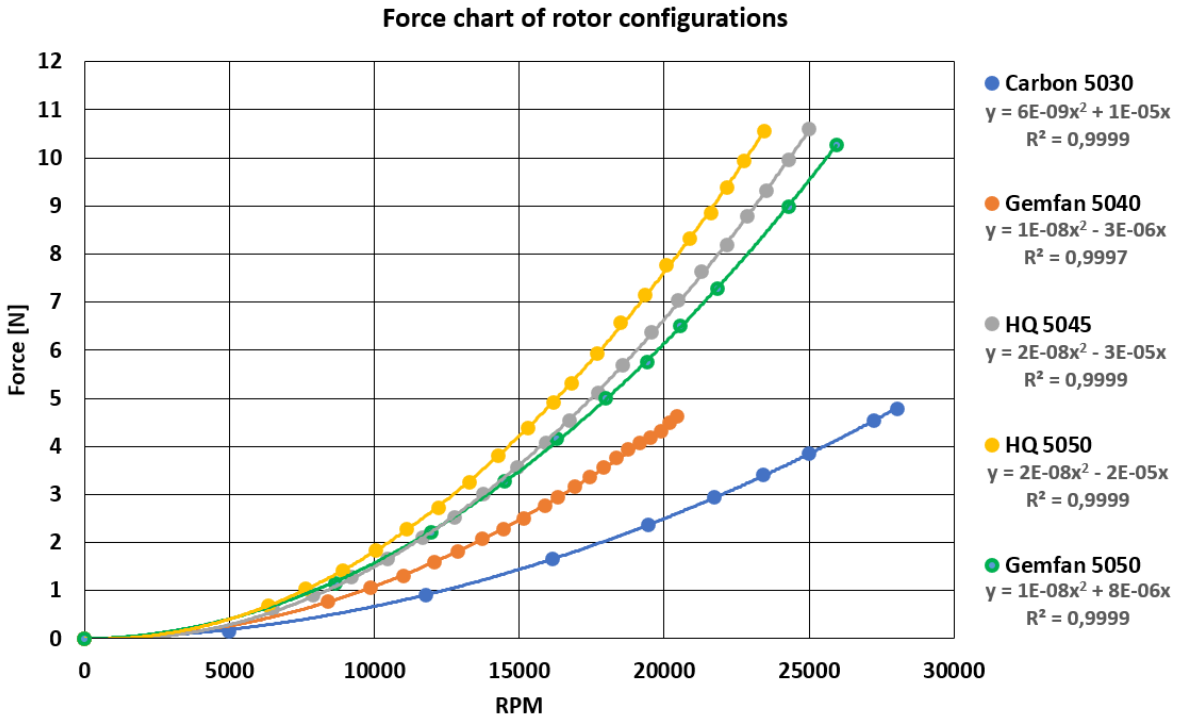


Figure 2.10: Force plot of the rotor candidates with corresponding 2nd order regression line, equation and R^2 fit.

From the data points a 2nd order regression forced through point (0,0) is approximating the relation of the rotational speed and the rotor force. The 2nd order regression is due to the assumption that force and torque are proportional to the rotational speed squared as seen in Equation 2.25 [58] (**Assumption 5**).

$$f_p \propto \omega^2 \,, \quad \tau \propto \omega^2 \tag{2.25}$$

As an example the 2nd order regression of the HQ Prop 5045 is given in Equation 2.26.

$$f_p = 0.182 \cdot 10^{-7}\, RPM^2 - 0.323 \cdot 10^{-4}\, RPM \tag{2.26}$$

By substituting the mean propeller force at hover position into Equation 2.26 it can be solved for the approximated rotational speed needed for hover. From the calculated rotational speed the data sets are used to determine the corresponding efficiencies. The efficiencies and maximum force and torque are stated in Table 2.3.

Candidate	**1.**	**2.**	**3.**	**4.**	**5.**
Motor Eff. [%]	62	72	71	63	62
Total Eff. [%]	59	76	77	75	68
Max. Force $[N]$	4.77	4.62	10.59	10.55	10.27
Max. Torque $[Nm]$	$44.64 \cdot 10^{-3}$	$43.81 \cdot 10^{-3}$	$110.7 \cdot 10^{-3}$	$115.12 \cdot 10^{-3}$	$168.18 \cdot 10^{-3}$

Table 2.3: Characteristics of the 5 rotor configurations at hover position.

From Table 2.3 the 3rd candidate HQ Prop 5045 has the second highest efficiency for the motor, the highest for total efficiency, is capable of generating most force and is within top 3 in maximum torque. This rotor configuration is thus chosen for the UAV design. In Section B.3 all relevant data for the five candidates is presented.

2.4 Size Estimation

In this section the size of the FAUAV is estimated. The estimation is based on the space available inside the WTB at a depth of $^2/_3$, this depth corresponds to approximately 40 $[m]$ inside the WTB.

From Table B.1 in Appendix B.1 it can be seen that at depth 40.29 $[m]$ the box-spar has length and width of 0.53 $[m]$. It is therefore argued that the maximum diameter of the UAV must be less than the length and width at this given depth, that is $D_{max} < 0.53\,[m]$. Figure 2.11 a rough sketch of the FAUAV is presented, in the sketch the relevant parameters for sizing the FAUAV is given.

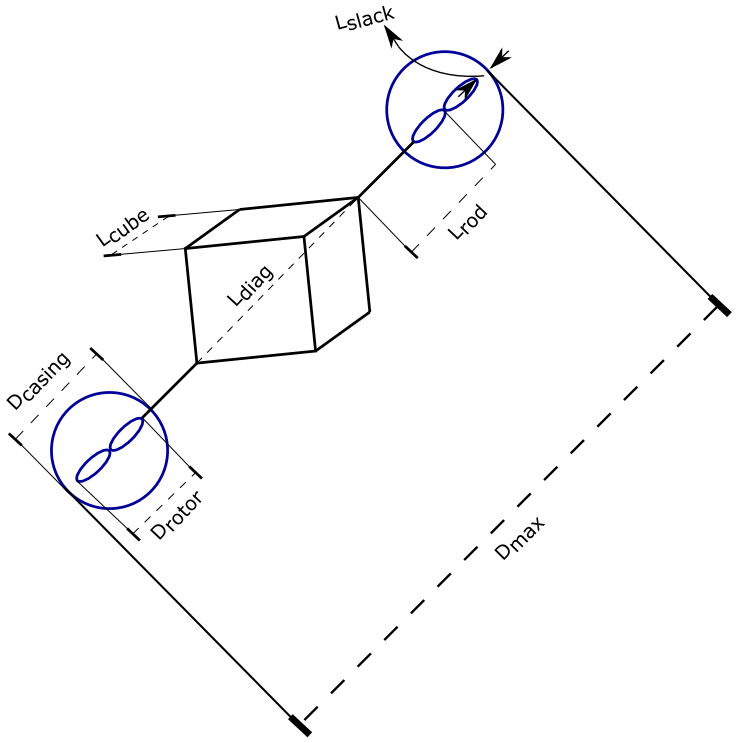


Figure 2.11: Sketch of the designed UAV displaying length and diameter parameters including the maximum diameter of the vehicle.

The parameters L_{diag}, D_{rotor}, D_{casing}, L_{slack}, L_{cube} and D_{max} are known and from these the unknown parameter L_{rod} is calculated as seen in Equation 2.27.

$$L_{rod} = \frac{D_{max} - (D_{casing} + L_{diag})}{2} \tag{2.27}$$

Choosing $D_{max} = 500\,[mm]$, L_{rod} attains the value seen in Table 2.4.

Parameter	**Value** $[mm]$
D_{casing}	147
D_{rotor}	127
L_{cube}	60
L_{diag}	103.92
L_{slack}	10
L_{rod}	113

Table 2.4: Lengths and diameters of the UAV structure components.

2.5 Component List and Weight Estimation

In order to provide a realistic weight estimate, a hypothetical component list is constructed, consisting of materials, actuators and sensors. Due to the geometric similarity between the proposed design and the design in [54] certain components are incorporated in the weight estimation in Table 2.5. The components that are incorporated from [54] are referred to as basic components.

Basic Components			
Component	Units	Unit Weight $[kg]$	Summed Weight $[kg]$
Motor	8	0.0455	0.36
Propeller	8	0.0105	0.084
ESC	8	0.002	0.016
Battery	1	0.196	0.196
Frame	1	0.152	0.152
Wiring	1	0.02	0.02
Others	1	0.02	0.02
Sensors for Control and V-SLAM			
Depth Camera	1	0.0756	0.0756
Tracking Camera	1	0.055	0.055
Sensors for Inspection			
RGB Camera & LED Light Source	1	0.04	0.04
Total Weight	-	-	0.8422

Table 2.5: Weight estimation of required components.

The reason for taking depth and tracking cameras into account are further elaborated in Chapter 4 Section 4.3. The RGB camera and LED light source are required to perform non-destructive testing inside the WTB. An advantage by using RGB camera is low price and ease of use while disadvantages include the requirement of illumination and it can only detect surface damages. [11]

3 | Modelling of the UAV

Throughout this chapter the force and torque coefficients are determined and a mathematical model is made to approximate the dynamics of the ESC, motor and propeller combination. Additionally a mathematical model of the UAV is constructed by rigid body dynamics. After the mathematical model, a noise model is presented for inputs and measurements together with a disturbance model affecting translational and rotational motions. Finally the equations utilized for the simulation model are summarized in the last section.

Abbreviation	**Meaning**
AI	Aerodynamic Disturbance
CAM	Control Allocation Matrix
CE	Ceiling Effect
DOF	Degrees Of Freedom
ESC	Electrical Speed Controller
FOPDT	First Order Plus Dead Time
GE	Ground Effect
ICE	In Ceiling Effect
IGE	In Ground Effect
IMU	Inertial Measurement Unit
IWE	In Wind Effect
OCE	Out of Ceiling Effect
OGE	Out of Ground Effect
UAV	Unmanned Aerial Vehicle
WE	Wind Effect
WTB	Wind Turbine Blade

Table 3.1: Abbreviations used in this chapter.

3.1 Modelling of Rotor Configuration

The modelling of the rotor configuration chosen in Section 2.3.3 is utilized for different purposes presented throughout the following 3 subsections. At first a static gain is calculated from the relation between angular velocity squared and the thrust and torque respectively. These gains are utilized to determine the thrust-to-drag coefficient used in the CAM and to determine the thrust saturation of the rotors. Secondly a dynamic model is derived to mimic the delay as a input is commanded until it affects the UAV. This dynamic response is afterwards formulated as a discrete model in the third part for estimation and control purposes.

3.1.1 Force and Torque Coefficients

From Section 2.3.2 the forces and moments affecting the UAV can be described through the CAM and the propeller thrusts. From Equation 2.25 force and torque are assumed proportional to the rotational speed squared and the relation is confirmed at Figure 2.10 where the rotational speed to force can be approximated as a 2nd order regression with the lowest R^2 value at 0.9997. The data sets utilized for Figure 2.10 are obtained by static experiments thus all dynamics are settled when rotational speed is measured. Further at settled condition the input voltage is assumed proportional to the rotational speed making the force and torque proportional to the input voltage squared (**Assumption 6**). By normalizing the rotational speed squared the force and torque coefficients can be found as the gains between the normalized values and the force and torque measurements. The gains are obtained as the slope of the linear regressions seen in Figure 3.1.

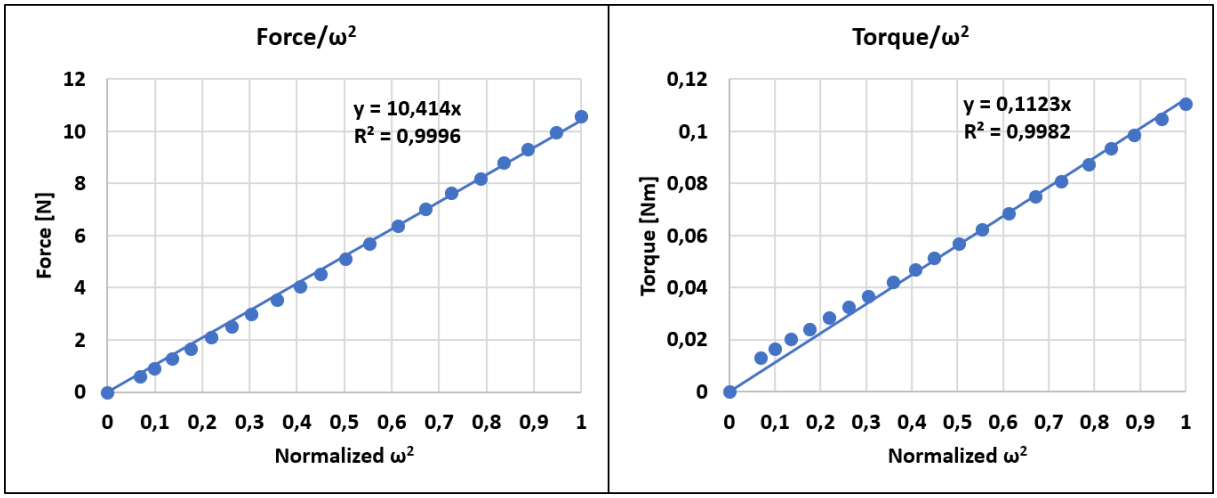


Figure 3.1: Linear regression with the slope assigning the force and torque coefficients.

The slope of the linear regressions are determined as the force and torque coefficients seen in Equation 3.1 and 3.2 whereas the thrust-to-drag coefficient is calculated in Equation 3.3.

$$K_F = 10.414\,[N] \tag{3.1}$$

$$K_\tau = 0.1123\,[Nm] \tag{3.2}$$

$$\kappa = \frac{K_\tau}{K_F} = 10.78 \cdot 10^{-3}\,[m] \tag{3.3}$$

3.1.2 Dynamic Response

At the static experiments the data is measured as the dynamic response of the rotor has settled and thus do not capture the dead time and time delays occurring in the transition from voltage to rotational speed. To incorporate a dynamic model of the rotor configuration a different data set is utilized with faster sampling time to capture the dynamic response. No such data is obtainable for the chosen rotor configuration at RCbenchmarks database and it was chosen to approximate the dynamics from data utilized in [43]. The results in [43] are of 2 different step applied for the rotor configuration to determine the dynamic response. For the first step, the rotor is initially providing $1.6\,[N]$ of thrust whereas a step of $0.4\,[N]$ is additionally commanded. The rotor is thus commanded to increase rotational speed for the same spinning direction and the settling time is

approximated to 0.2 $[s]$. The second response is also as the rotor is commanded a step of 0.4 $[N]$ but with an initial thrust generation of -0.2 $[N]$. The step forces the rotor to change spinning direction and the settling time for this response is approximated to 0.45 $[s]$ thus more than double of the step maintaining the same spinning direction. 2 different approximations of the rotors dynamic response are made as seen in Figure 3.2. The first approximation is a continuous model consisting of 2 first order transfer functions, one with time constants at 0.085 $[s]$ and one with a time constant at 0.3 $[s]$. The reason for the 2 different time constants are due to the dynamics occurring as the rotors change direction. If a rotors input corresponds to the rotor is always spinning the same direction, the time constant is kept at 0.085 $[s]$ as seen from the step at the left of Figure 3.2. If the input forces the rotor to change direction the dynamics change to the slow time constant when close to zero velocity to simulate the deceleration and acceleration of the rotor as seen at the right of Figure 3.2. In addition a delay occur in the data acquired in [43] whereas the steps are applied at time 0.25 $[s]$ but the response starts to act at time 0.27 $[s]$ thus the rotor model is extended to a first order plus dead time (FOPDT) with a dead time of $t_d = 0.02$ $[s]$. The FOPDT model is then seen in Equation 3.4.

$$G_{FOPDT}(s) = \frac{1}{\tau_s s + 1} e^{-t_d s} \tag{3.4}$$

with τ_s being 0.085 $[s]$ under normal conditions or 0.3 $[s]$ if the rotor change direction.
Increasing the time constant as the rotor is spinning with an angular velocity close to zero is an impractical implementation in the simulation model as the slow response is only desired when the rotor actually change spinning direction. Therefore the first approximation is only utilized as a reference model for the second approximation seen as the red response at Figure 3.2.

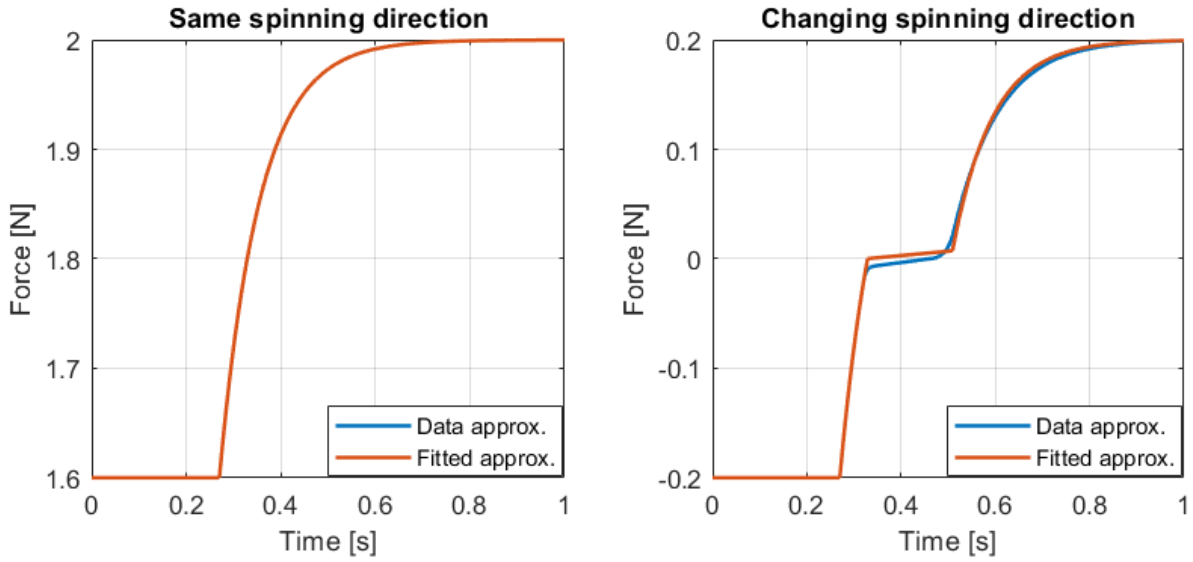


Figure 3.2: Dynamic response of the rotor model.

Here the rotor response is still the FOPDT with a time constant of 0.085 $[s]$ though if the rotors angular velocity change sign the time constant is increased to 5 $[s]$ for a time period of 0.18 $[s]$ to fit the blue rotor approximation. The red response is then implemented in the simulation model for each rotor.

3.1.3 Discrete Rotor Approximation

The rotor model presented in Section 3.1 is of continuous form but to incorporate the dynamics in the observers and controllers some assumptions are made and a difference equation is derived. At first the rotor model is a FOPDT but to compensate for the dead time the rotor model utilized for the observers and controllers is chosen as a second order transfer function to gain an acceleration phase. The second order transfer function is presented in Equation 3.5.

$$\begin{aligned}\frac{Y(s)}{U(s)} &= \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \\ &\rightarrow (s^2 + 2\zeta\omega_n s + \omega_n^2)Y(s) = \omega_n^2 U(s)\end{aligned} \tag{3.5}$$

here ζ is the damping coefficient and ω_n is the eigenfrequency. The Laplace transform can be described as the differentiation of the given variable. The time derivatives of $Y(s)$ is then given as in Equation 3.6 and 3.7.

$$Y(s)s = \frac{dy}{dt} = \dot{y} \approx \frac{y(n) - y(n-1)}{T} \tag{3.6}$$

$$Y(s)s^2 = \frac{d\dot{y}}{dt} = \ddot{y} \approx \frac{\dot{y}(n) - \dot{y}(n-1)}{T} = \frac{y(n) - 2y(n-1) + y(n-2)}{T^2} \tag{3.7}$$

here T is the sampling time while n is the discrete time indicator. By substituting Equation 3.6 and 3.7 into Equation 3.5 the output can be defined from the difference equation given in Equation 3.8.

$$\begin{aligned}&\frac{1}{T^2}(y(n) - 2y(n-1) + y(n-2)) + \frac{2\zeta\omega_n}{T}(y(n) - y(n-1) + \omega_n^2 y(n) = \omega_n^2 u(n) \\ &\rightarrow \left(\frac{1}{T^2} + \frac{2\zeta\omega_n}{T} + \omega_n^2\right) y(n) = \omega_n^2 u(n) + 2\left(\frac{1}{T^2} + \frac{\zeta\omega_n}{T}\right) y(n-1) - \frac{1}{T^2} y(n-2) \\ &\rightarrow y(n) = \frac{\omega_n^2 u(n) + \frac{2}{T}\left(\frac{1}{T} + \zeta\omega_n\right) y(n-1) - \frac{1}{T^2} y(n-2)}{\frac{1}{T^2} + \frac{2\zeta\omega_n}{T} + \omega_n^2}\end{aligned} \tag{3.8}$$

The output becomes an expression of the current input $u(n)$ together with the 2 former outputs $y(n-1)$ and $y(n-2)$. The damping coefficient is chosen as $\zeta = 1$ as no overshoot is occurring for the FOPDT and the eigenfrequency is chosen as $\omega_n = 19[rad/s]$ to attain similar rise and settling time as the rotor model presented in Section 3.1.2. The response of the second order transfer function is seen in comparison with the FOPDT at Figure 3.3.

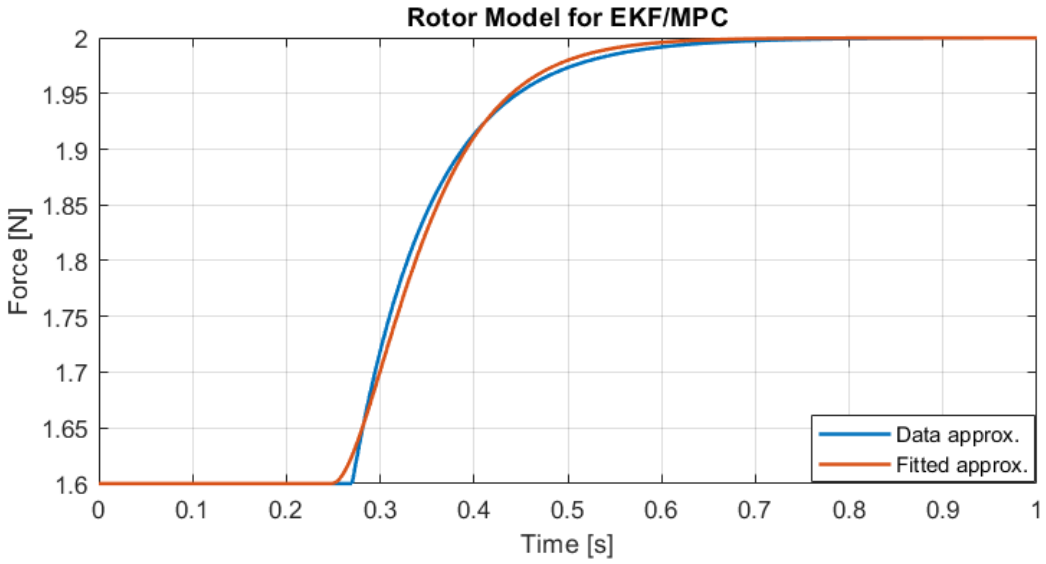


Figure 3.3: Approximation of the rotor model for the EKF/MPC.

With the second order response the dead time is not fully captured, but still forces to accelerate slower. Additionally as both the rise time and settling time is closely related for the 2 models, the rotor approximation is deemed proper for the observer and controller designs.

With the different rotor model approximations the input dynamics have been stated. The following sections derive the descriptive equations for the UAVs motions where Section 3.2 describes the translational dynamics and Section 3.3 describes the rotational dynamics.

3.2 Derivation of Translational Dynamics

The thrust of the rotors are translated into general translational forces by the relation seen in Equation 2.6. The translational dynamics of the designed UAV covers the relation between the general forces and the UAVs motion in the x, y and z directions. The general forces can be transform to a fixed global frame to determine the UAVs position in global coordinates (**Assumption 7**).

To describe the translational accelerations Newton's 2nd law is utilized as stated in Equation 3.9.

$$\boldsymbol{F}_b = m\ddot{\boldsymbol{p}} \tag{3.9}$$

where $\boldsymbol{F}_b$ is a vector containing the translational body forces and $\ddot{\boldsymbol{p}}$ is a vector containing the linear accelerations both correlated to the x, y and z axis. The forces acting on the UAV is the sum of forces generated from the 8 rotors and the gravitational force. To describe the effects of the rotor forces and the gravitational force an inertial and a body frame is presented. The inertial frame correlates to the motions of the UAV from a globally fixed coordinate system while the body frame is fixed at the center of mass (CM) for the UAV and is correlating to the orientation of the UAV. The 2 coordinate frames in relation to the UAV can be seen in Figure 3.4.

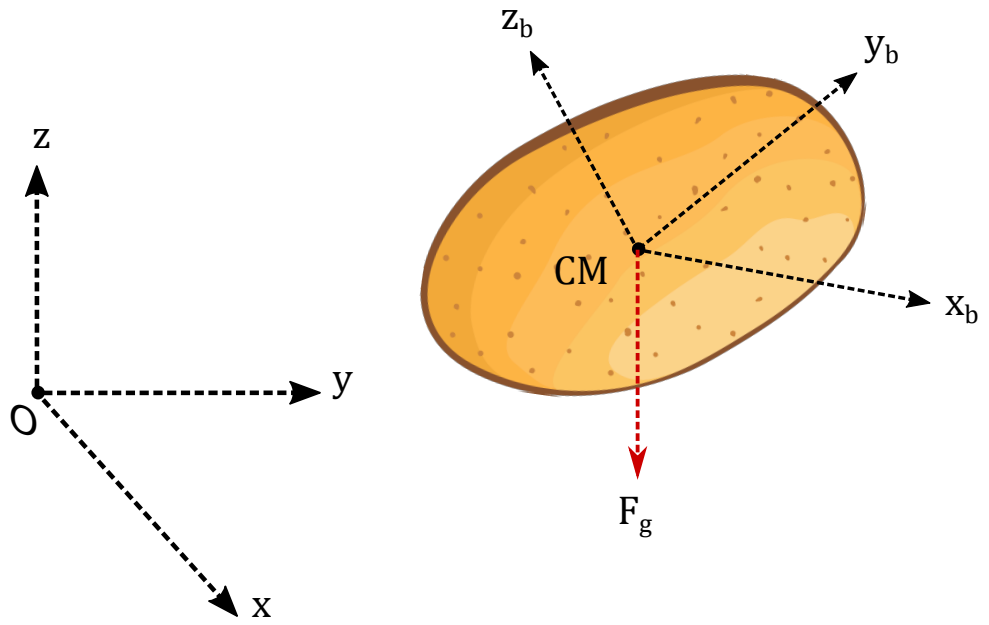


Figure 3.4: Free body diagram contained in an inertial frame. When describing rigid body dynamics a potato proves sufficient for the purposes of illustration.

Figure 3.4 shows the inertial frame extending from origo $\mathcal{O}$ and the body frame extending from the CM of the UAV. The body frame is seen rotated around the CM whereas the gravitational force $\boldsymbol{F}_g$ is affecting at the CM and directed in the global negative z axis. The forces acting on the UAV can then be defined as in Equation 3.10.

$$\sum \boldsymbol{F} = \boldsymbol{F}_i - \boldsymbol{F}_g \,, \quad \boldsymbol{F}_g = mg\boldsymbol{e} \tag{3.10}$$

where $\boldsymbol{F}_i$ is the rotor force acting on the rigid body in the inertial frame and $\boldsymbol{F}_g$ is the gravitational force with $\boldsymbol{e}$ as the orientation vector. As the gravitational force is only acting in the z axis of the inertial frame the orientation vector is defined as $\boldsymbol{e} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$. The forces from the rotors both depend on the orientation of the rotors with respect to the body frame and the body frames orientation in relation to the inertial frame. The orientation of the rotors with respect to the body frame is conveyed in Section 2.3.2, but the relation of the forces between body and inertial frame is described through the rotation matrix $\boldsymbol{Q}$ as seen in Equation 3.11.

$$\boldsymbol{F}_i = \boldsymbol{Q}\boldsymbol{F}_b \tag{3.11}$$

$$\boldsymbol{Q} = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_0q_2 + q_1q_3) \\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_0q_1 + q_2q_3) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{3.12}$$

The rotational matrix seen in Equation 3.12 is described in quaternion units. A quaternion is a 4 dimensional vector consisting of a real part q_0 (scalar part) and 3 imaginary parts q_1, q_2 and q_3 (rotational part) as seen in Equation 3.13.

$$\boldsymbol{q} = q_0 + iq_1 + jq_2 + kq_3 \tag{3.13}$$

The usage of quaternions for the rotational matrix require the 4 components to be normalized thus they have to satisfy the second-norm being equal to 1 as seen in Equation 3.14.

$$|\boldsymbol{q}|_2 = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} = 1 \tag{3.14}$$

Using quaternions to describe rotation is less intuitive than applying Euler angles but has it advantages as the sine and cosine terms from Euler angles are removed. Further, quaternions can handle omnidirectional rotation without resulting in loss of DOF as Gimbal lock does not occur. Further explanation of quaternion theory and derivation of the rotational matrix are seen in Appendix C. [59–61]. By substituting Equations 3.9 and 3.11 into Equation 3.10 the translational acceleration can be seen in Equation 3.15 where m denotes the mass and g is the gravitational acceleration. [62]

$$\ddot{\boldsymbol{p}} = \frac{1}{m}\boldsymbol{Q}\boldsymbol{F}_b - g\boldsymbol{e} \tag{3.15}$$

The global acceleration of the UAV then becomes a function of the body forces rotated with the orientation described in quaternions and the gravitational acceleration. As the UAV is further capable of rotating the next section covers the rotational dynamics.

3.3 Derivation of Attitude Dynamics

The attitudinal dynamics are described in 2 part. The first part is the derivation of the quaternion rates and the second part is the derivation of the angular accelerations.

The quaternion dynamics can be established by the quaternion rates calculated by the cross and dot product of the quaternion $\boldsymbol{q}$ itself and a quaternion for the angular velocity in the body frame. The angular velocity quaternion is defined as $\boldsymbol{\omega} = 0 + ip + jq + kr = \begin{bmatrix} 0 & p & q & r \end{bmatrix}^T$ thus only pure rotation. For simplicity the two quaternions are defined as a scalar and a vector part given as $\boldsymbol{q} = [q_0, \overline{\boldsymbol{q}}]$ and $\boldsymbol{\omega} = [\omega_0, \overline{\boldsymbol{\omega}}]$. The quaternion derivative is then calculated as in Equation 3.16. [61]

$$\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{q} \otimes \boldsymbol{\omega} = \frac{1}{2}\begin{bmatrix} q_0\omega_0 - \overline{\boldsymbol{q}} \cdot \overline{\boldsymbol{\omega}} \\ q_0\overline{\boldsymbol{\omega}} + \omega_0\overline{\boldsymbol{q}} + \overline{\boldsymbol{q}} \times \overline{\boldsymbol{\omega}} \end{bmatrix} \tag{3.16}$$

The two expressions from Equation 3.16 are calculated in Equation 3.17 and Equation 3.18. [63]

$$q_0\omega_0 - \overline{\boldsymbol{q}} \cdot \overline{\boldsymbol{\omega}} = \begin{bmatrix} 0 & -q_1 p & -q_2 q & -q_3 r \end{bmatrix} \tag{3.17}$$

$$q_0\overline{\boldsymbol{\omega}} + \omega_0\overline{\boldsymbol{q}} + \overline{\boldsymbol{q}} \times \overline{\boldsymbol{\omega}} = q_0 \begin{bmatrix} p \\ q \\ r \end{bmatrix} + \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} \times \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} q_0 p + q_2 r - q_3 q \\ q_0 q - q_1 r + q_3 p \\ q_0 r + q_1 q - q_2 q \end{bmatrix} \tag{3.18}$$

By substitute Equation 3.17 and 3.18 into Equation 3.16 the quaternion derivative can be expressed by the transformation matrix containing the angular velocities and quaternion vector q as seen in Equation 3.19. [64]

$$\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{T}\boldsymbol{q} = \frac{1}{2}\begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix}\begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{3.19}$$

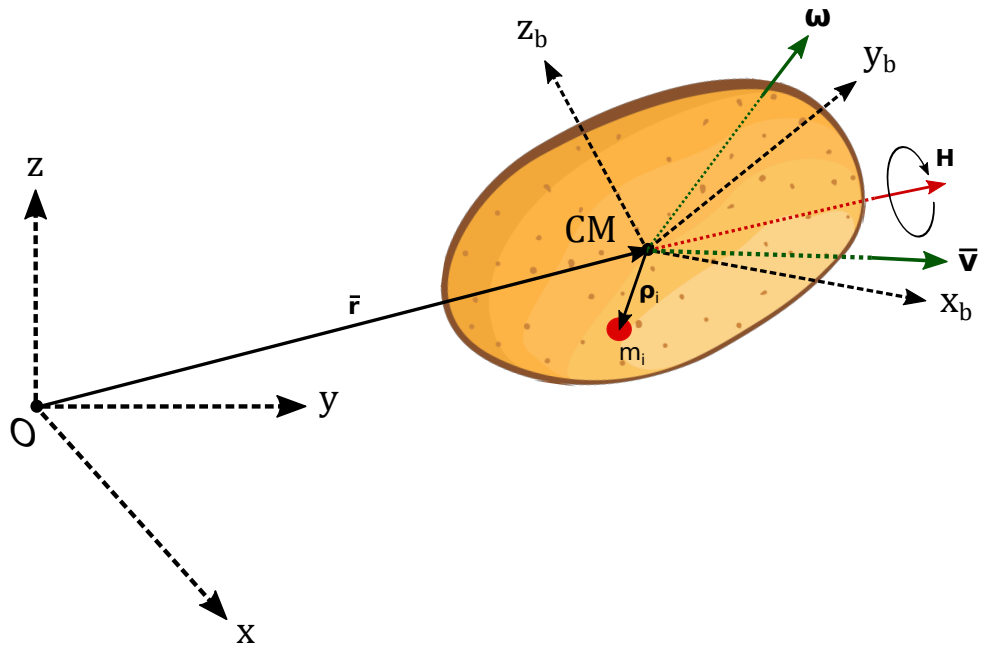


Figure 3.5: Free potato-body diagram, containing the necessary parameters for the derivation of attitude dynamics of a rigid body.

The angular acceleration is derived from the absolute angular momentum as seen in Equation 3.20.

$$\boldsymbol{H} = \sum \boldsymbol{\rho}_i \times m_i \boldsymbol{v}_i \tag{3.20}$$

The absolute angular momentum $\boldsymbol{H}$ is the sum of moments for all given point masses illustrated in an arbitrary position in Figure 3.5. From Equation 3.20, $\boldsymbol{\rho}_i$ is the length from origin to the ith point mass, m_i is the mass and $\boldsymbol{v}_i$ is the velocity of the ith point mass. The velocity is defined as in Equation 3.21.

$$\boldsymbol{v}_i = \overline{\boldsymbol{v}} + \boldsymbol{\omega} \times \boldsymbol{\rho}_i \tag{3.21}$$

here $\overline{\boldsymbol{v}}$ is the velocity of the center of mass and $\boldsymbol{\omega} \times \boldsymbol{\rho}_i$ is the relative velocity of the ith point mass, both illustrated in Figure 3.5. Substituting Equation 3.21 into Equation 3.20 the absolute angular momentum is seen in Equation 3.22.

$$\boldsymbol{H} = \sum (\boldsymbol{\rho}_i \times m_i(\overline{\boldsymbol{v}} + \boldsymbol{\omega} \times \boldsymbol{\rho}_i)) = \sum \boldsymbol{\rho}_i \times m_i \overline{\boldsymbol{v}} + \sum \boldsymbol{\rho}_i \times m_i(\boldsymbol{\omega} \times \boldsymbol{\rho}_i) \tag{3.22}$$

From Equation 3.22 the momentum can be divided into 2 terms whereas the first term can be rewritten as Equation 3.23.

$$\sum \boldsymbol{\rho}_i \times m_i \overline{\boldsymbol{v}} = -\overline{\boldsymbol{v}} \times \sum m_i \boldsymbol{\rho}_i \tag{3.23}$$

As the center of mass is considered the origin the sum of the product between the mass and length to the ith mass points are zero. Thus, the absolute angular momentum is reduced to the expression in Equation 3.24.

$$\boldsymbol{H} = \sum \boldsymbol{\rho}_i \times m_i(\boldsymbol{\omega} \times \boldsymbol{\rho}_i) \tag{3.24}$$

From Equation 3.24 $\boldsymbol{\rho}_i$ can be described as a vector $\boldsymbol{\rho}_i = ix + jy + kz$. Further by substituting dm for the mass, the summation is turned to an integral action defining the absolute angular momentum as Equation 3.25.

$$\begin{aligned}\boldsymbol{H} &= \int \begin{bmatrix} x \\ y \\ z \end{bmatrix} \times \left(\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \end{bmatrix} \right) dm = \int \begin{bmatrix} x \\ y \\ z \end{bmatrix} \times \begin{bmatrix} \omega_y z - \omega_z y \\ \omega_z x - \omega_x z \\ \omega_x y - \omega_y x \end{bmatrix} dm \\ &= \int \begin{bmatrix} (y^2+z^2)\omega_x - xy\omega_y - xz\omega_z \\ -xy\omega_x + (x^2+z^2)\omega_y - yz\omega_z \\ -xz\omega_x - yz\omega_y + (x^2+y^2)\omega_z \end{bmatrix} dm \end{aligned} \tag{3.25}$$

The integral can be described in matrix form as seen in Equation 3.26.

$$\begin{aligned}\boldsymbol{H} &= \int \begin{bmatrix} y^2+z^2 & -xy & -xz \\ -xy & x^2+z^2 & -yz \\ -xz & -yz & x^2+y^2 \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} dm \\ &= \begin{bmatrix} \int (y^2+z^2)dm & -\int xydm & -\int xzdm \\ -\int xydm & \int (x^2+z^2)dm & -\int yzdm \\ -\int xzdm & -\int yzdm & \int (x^2+y^2)dm \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \end{aligned} \tag{3.26}$$

The integration of the length units squared related to mass results in the inertia for the x, y and z axis with respect to the angular velocities. Equation 3.26 can then be rewritten as the inertia matrix and the vector of angular velocities as seen in Equation 3.27.

$$\boldsymbol{H} = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{zy} \\ -I_{zx} & -I_{yz} & I_{zz} \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \boldsymbol{J}\boldsymbol{\omega} \tag{3.27}$$

Due to the symmetry of the UAVs structure all of the off-diagonal terms in the inertia matrix are zero and the absolute angular momentum is expressed as in Equation 3.28.

$$\boldsymbol{H} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = iI_{xx}\omega_x + jI_{yy}\omega_y + kI_{zz}\omega_z \tag{3.28}$$

To relate the angular acceleration to the moments the sum of moments can be described as the derivative of the absolute angular momentum as seen in Equation 3.29.

$$\sum \boldsymbol{M} = \dot{\boldsymbol{H}} = \left(\frac{di}{dt} I_{xx}\omega_x + \frac{dj}{dt} I_{yy}\omega_y + \frac{dk}{dt} I_{zz}\omega_z\right) + (iI_{xx}\dot{\omega}_x + jI_{yy}\dot{\omega}_y + kI_{zz}\dot{\omega}_z) \tag{3.29}$$

The derivatives of the i, j and k can be combined and described by the angular velocity vector cross the orientation vector as seen in Equation 3.31.

$$\frac{di}{dt} = \boldsymbol{\omega} \times i\,, \quad \frac{dj}{dt} = \boldsymbol{\omega} \times j\,, \quad \frac{dk}{dt} = \boldsymbol{\omega} \times k \tag{3.30}$$

$$\frac{di}{dt} + \frac{dj}{dt} + \frac{dk}{dt} = \boldsymbol{\omega} \times \boldsymbol{r}\,, \quad \boldsymbol{r} = \begin{bmatrix} i \\ j \\ k \end{bmatrix} \tag{3.31}$$

By this notation the sum of moments is given as in Equation 3.32.

$$\sum \boldsymbol{M} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \times \left(\begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \right) + \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} = \boldsymbol{\omega} \times \boldsymbol{J}\boldsymbol{\omega} + \boldsymbol{J}\dot{\boldsymbol{\omega}} \tag{3.32}$$

The sum of moments are equal to the torque generated in the body frame. By substituting the torque into Equation 3.32 the angular acceleration can be isolated as seen in Equation 3.33 [62]. Where $\sum \boldsymbol{M} = \tau_b$.

$$\dot{\boldsymbol{\omega}} = \boldsymbol{J}^{-1}(\boldsymbol{\tau}_b - \boldsymbol{\omega} \times \boldsymbol{J}\boldsymbol{\omega}) \tag{3.33}$$

where $\boldsymbol{\omega} = [p\ q\ r]^T$. It is evident from Equation 3.33 that the inertia is of great importance. Therefore an accurate estimation of the moment of inertia is desired and derived in Subsection 3.3.1.

3.3.1 Estimation of Inertia

In this section an expression of the moment of inertia of the UAV is derived, assuming the UAV to be a cuboid with uniformly distributed mass as illustrated in Figure 3.6 (**Assumption 8**).

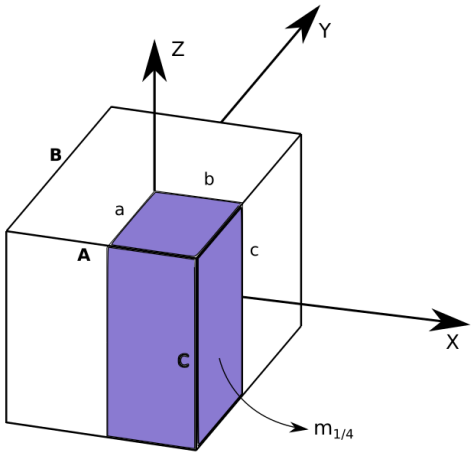


Figure 3.6: A sketch of a cuboid illustrating how mass moment of inertia is estimated, notice the marked area constitute a quarter of the volume and thereby quarter of the mass of the cuboid.

The quarter of the mass can be defined as in Equation 3.34.

$$m_{1/4} = \rho abc \tag{3.34}$$

where a, b, c are half the lengths of each side of the cuboid and ρ is the density. By inserting this expression into the diagonal elements in Equation 3.26 the inertia of the quarter of the cuboid can be derived seen in Equation 3.35. The derivation in Equation 3.35 is for the inertia around the z axis, but the procedure is the same for the x and y axis.

$$\begin{aligned} I_{zz,1/4} &= \int \left(x^2 + y^2\right) dm = \rho c \int_{-a}^{a} \int_{-b}^{b} \left(x^2 + y^2\right)\, dy\, dx \\ &= \rho c \int_{-a}^{a} \left(x^2 b + \frac{1}{3} b^3\right) - \left(-x^2 b - \frac{1}{3}\right)\, dx = \rho c \int_{-a}^{a} 2\left(x^2 + \frac{1}{3} b^3\right)\, dx \\ &= \rho c \left(2\left(\frac{1}{3} a^3 b + \frac{1}{3} a b^3\right) - 2\left(-\frac{1}{3} a^3 b - \frac{1}{3} a b^3\right)\right) = \frac{4\rho c}{3}\left(a^3 b + a b^3\right) \\ &= \frac{4}{3} m_{1/4}\left(a^2 + b^2\right) \end{aligned} \tag{3.35}$$

By further defining the sides as presented in Equation 3.36 and inserting into 3.35 the inertia can be expressed as seen in Equation 3.37.

$$a = \frac{1}{2} A \qquad\qquad b = \frac{1}{2} B \tag{3.36}$$

$$I_{zz,1/4} = \frac{4}{3} m_{1/4}\left(\left(\frac{1}{2} A\right)^2 + \left(\frac{1}{2} B\right)^2\right) = \frac{4}{3} m_{1/4}\left(\frac{A^2}{4} + \frac{B^2}{4}\right) = \frac{1}{3} m_{1/4}\left(A^2 + B^2\right) \tag{3.37}$$

Furthermore, by inserting Equation 3.36 into Equation 3.34 the mass can be stated as in Equation 3.38.

$$m_{1/4} = \rho abc = \rho \frac{1}{2} A \frac{1}{2} B c = \frac{\rho}{4} ABC \tag{3.38}$$

Then by multiplying Equation 3.38 by 4 the total mass of the UAV is expressed in Equation 3.39.

$$m_{total} = \rho ABC \tag{3.39}$$

The total mass is then inserted into Equation 3.37 which yields the total inertia around the z axis.

$$I_{zz} = \frac{1}{3} \rho \frac{1}{4} ABC\left(A^2 + B^2\right) = \frac{1}{12} \rho ABC\left(A^2 + B^2\right) = \frac{1}{12} m_{total}\left(A^2 + B^2\right) \tag{3.40}$$

With the derivation of the moment of inertia, the foundation of the mathematical model is now defined and the following section shows open loop testing of the simulation model based on the presented design.

3.4 Open Loop Test of the Mathematical Model

This section demonstrates how the designed UAV acts in the simulation model as different open loop inputs are commanded. The first simulation involve the UAVs motions as only hover thrust is applied as seen in Figures 3.7 and 3.8. Then the UAV is exerting thrust for individual translational motions such as the x direction seen in the Figures 3.9 and 3.10. These are followed by simulations of the UAV as torques are applied for either a roll, pitch or yaw motion with the simulations of roll presented in Figures 3.11 and 3.12.

While only applying hover thrust the UAV is kept at the same global x, y and z coordinates without performing any rotations. For Figures 3.9, 3.10, D.1, D.2, D.3 and D.4 the UAV is applied with hover thrust plus thrust for either the body x, y or z direction. The UAV is only seen to move in the direction of the corresponding force without rotating, thus proving the design capable of controlling each translational motion independently without the use of rotation. At last the Figures 3.11, 3.12, D.5, D.6, D.7 and D.8 presents the UAVs motions as hover thrust is provided together with torques generating different rotations. The hover thrust is applied for the body frame and not for the global which results in the translation motion when performing roll or pitch. This is due to the forces applied for the body z axis is rotated in the global frame thus redirecting the thrust in the y direction with a roll motion and in the x direction with a pitch motion.

The results presented above is motions of the UAV only with the descriptive dynamic equation utilizing ideal inputs. The rotor model derived in Section 3.1.2 is then incorporated and the effects are shown in Figure 3.13.

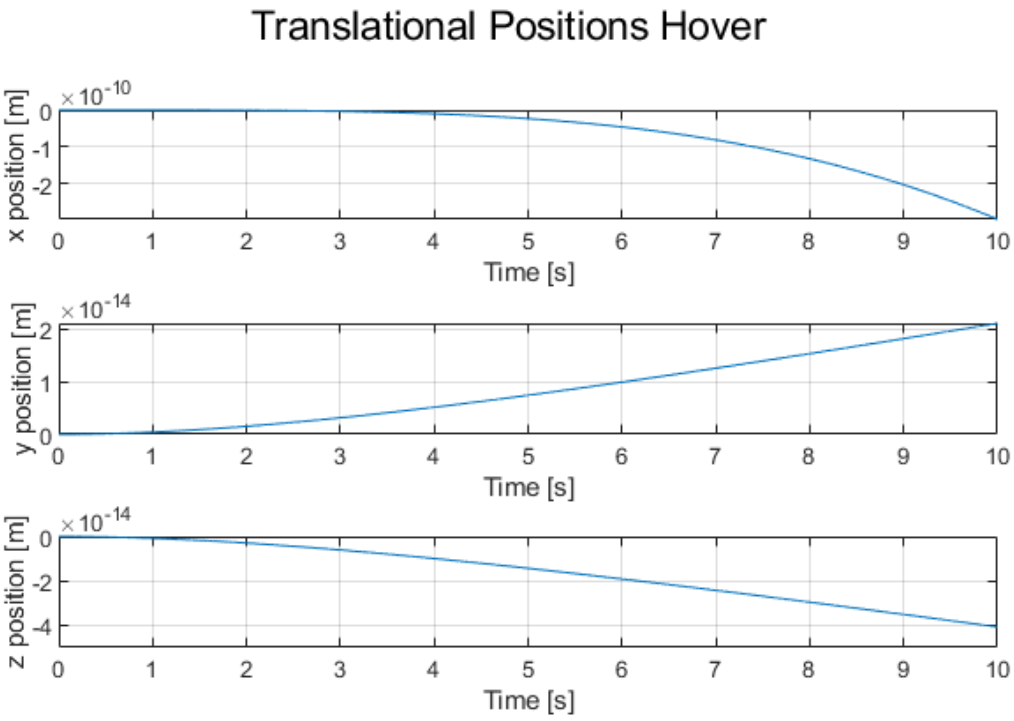


Figure 3.7: Translational positions applying hovering thrust.

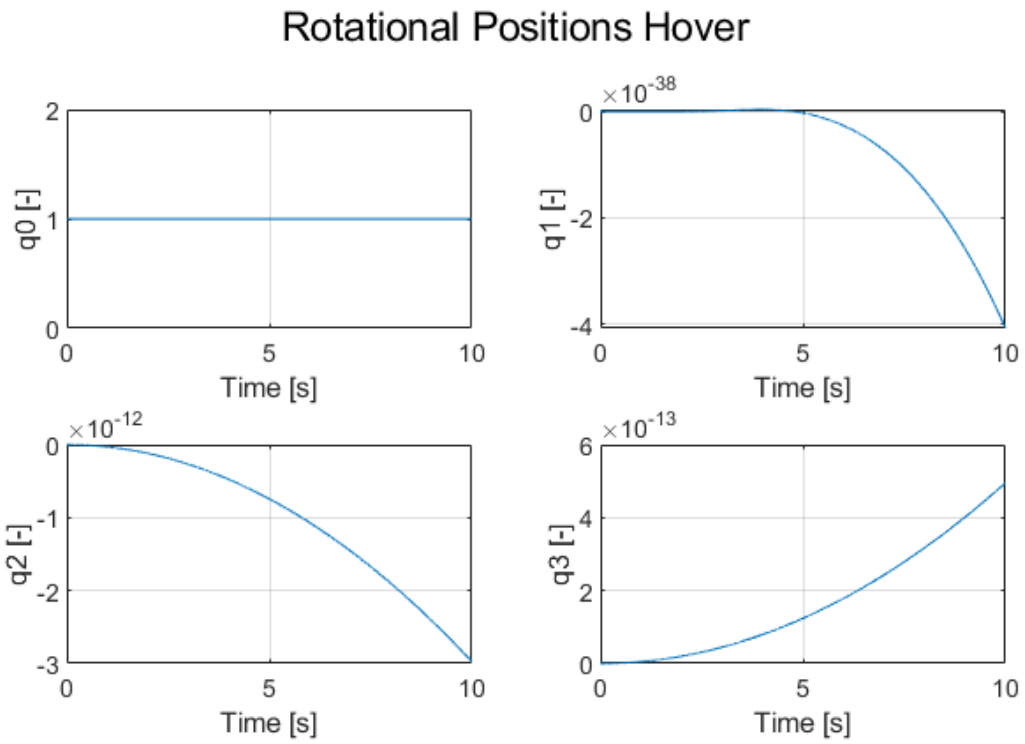


Figure 3.8: Rotational positions applying hovering thrust.

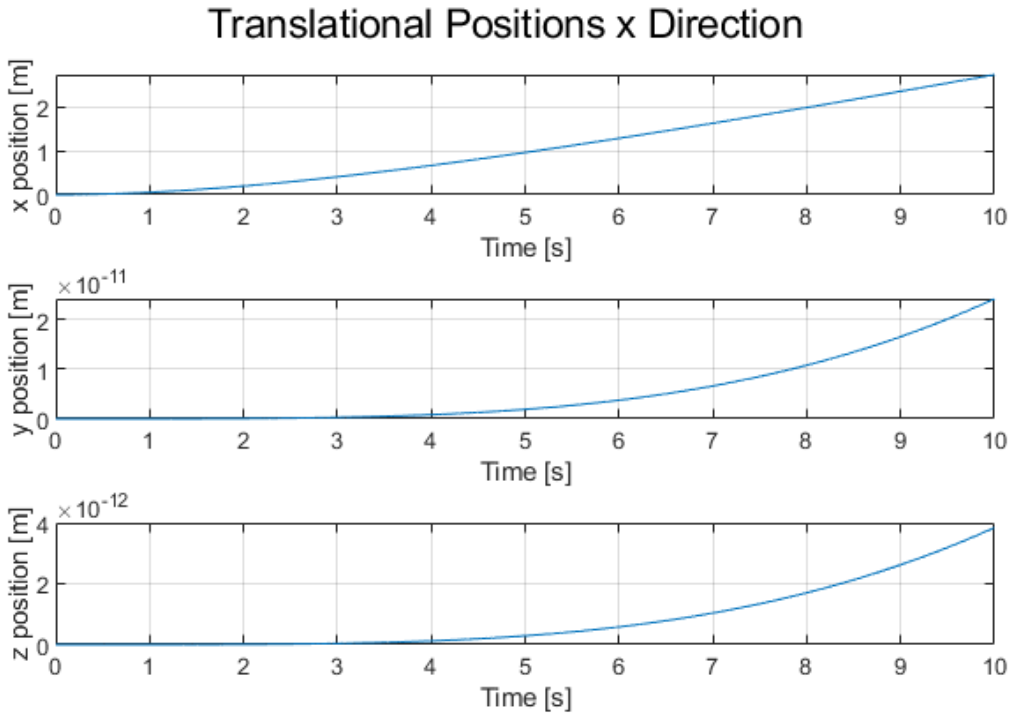


Figure 3.9: Translational positions applying hovering thrust and 0.1 $[N]$ for positive x direction.

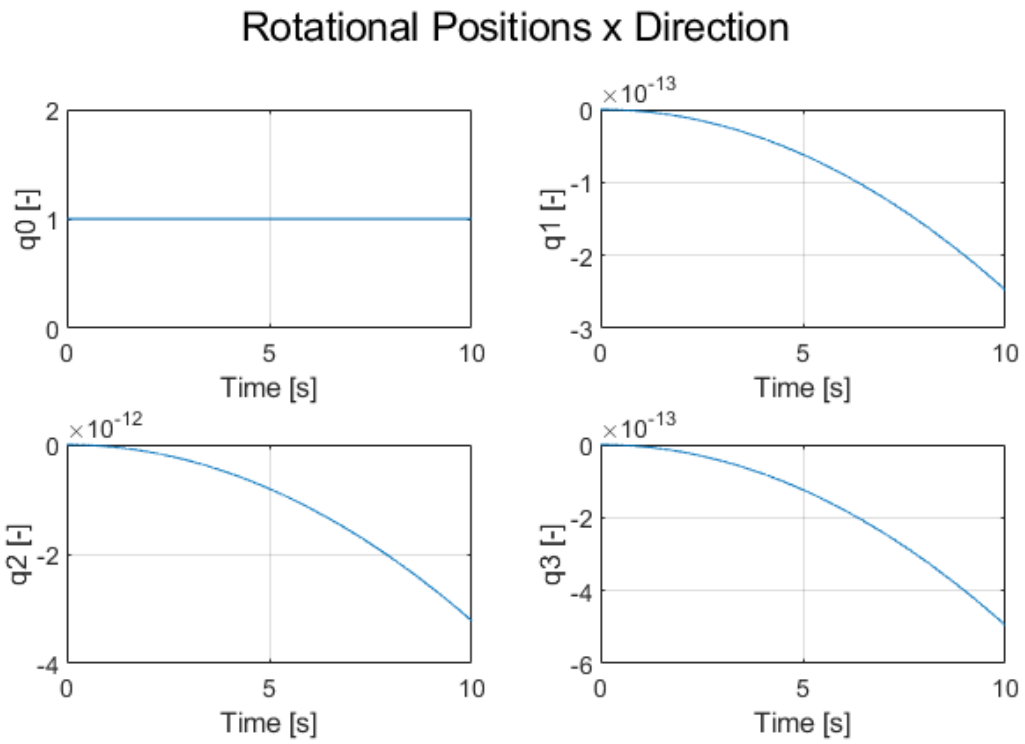


Figure 3.10: Rotational positions applying hovering thrust and 0.1 $[N]$ for positive x direction.

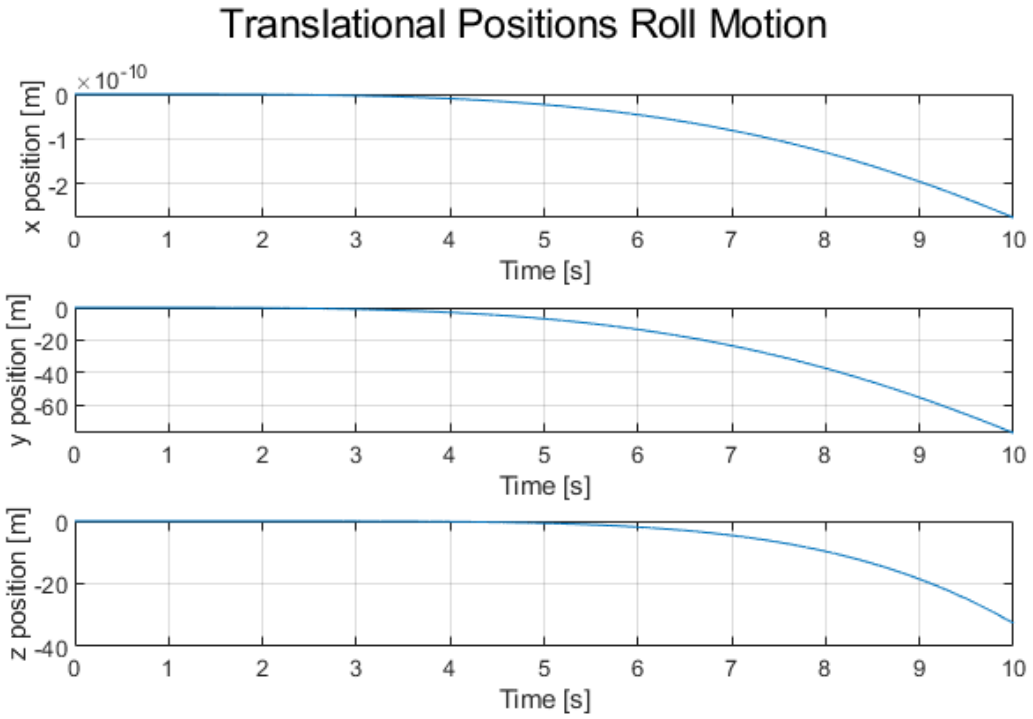


Figure 3.11: Translational positions applying hovering thrust and 0.0001 $[Nm]$ for positive roll.

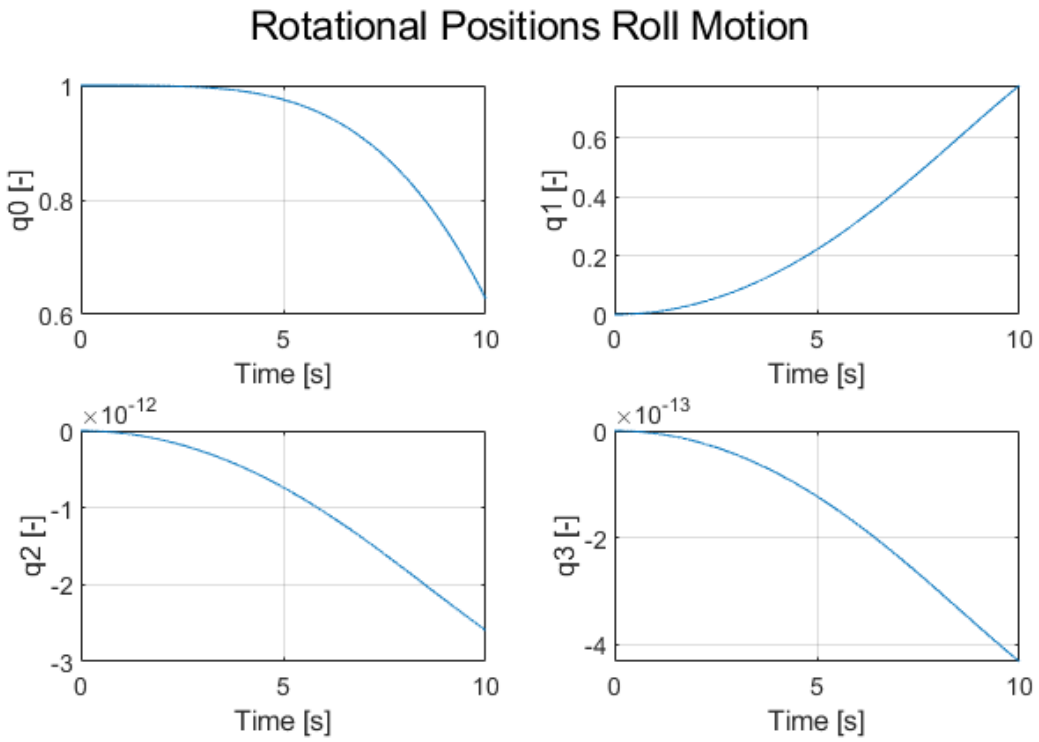


Figure 3.12: Rotational positions applying hovering thrust and 0.0001 $[Nm]$ for positive roll.

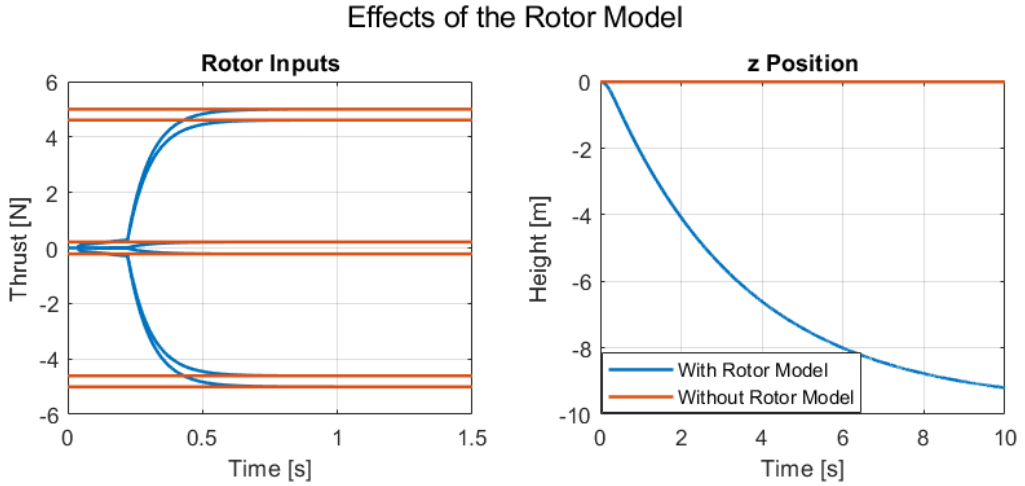


Figure 3.13: Comparison of dynamic response with and without the rotor model.

The simulation shown in Figure 3.13 are seen as the UAV is initialized at z position 0 $[m]$ and simply dropped. The rotors are then commanded to provide hover thrust to illustrate how the rotor model corrects for the drop. It is seen from the response without the rotor model, that the inputs act instantaneously and the UAV is kept at the initial position. For the response with the rotor model, the inputs have a slow phase at the first 0.25 $[s]$ corresponding to the acceleration of the rotors and afterwards change to the faster time constant before settling at the commanded rotor inputs. During this period the UAV is seen to fall and slightly correct for the drop as the rotors increase in velocity. Due to these results the rotor model is concluded important to incorporate in the design of estimators and controllers in future aspects.

The results presented in this section is only for an ideal flight thus not affected by either translational or rotational disturbances. When flying with a UAV, and especially within a confined space as a WTB, different disturbances occur. To make the simulation model more realistic and to create a robust control structure to accommodate for these disturbances they have to be modelled as seen in Section 3.5.

3.5 Disturbance Model

This section introduce the simulation model approximations of the disturbances affecting the UAV within a confined space. Firstly, translational disturbances are introduced which include ground effect (GE) and ceiling effect (CE) presented in Section 3.5.1. Secondly, rotational disturbances are described in Section 3.5.2 and include wall effect (WE) and airflow interference (AI) caused by the propellers.

3.5.1 Translational Disturbances

GE and CE are phenomenons which occur when the UAV is in near vicinity of the ground or ceiling respectively. When the UAV is in ground effect (IGE) the thrust required to sustain the same altitude or fly upwards is lower than the required thrust when out of ground effect (OGE). This is due to the increased pressure under the UAV compared to above which is a result of the airflow bouncing of the ground. Figure 3.14 (top right) illustrates how the airflow behaves IGE compared to Figure 3.14 (top left) when the UAV is OGE. Further, when the UAV is in ceiling effect (ICE) the air pressure under the UAV is larger than above which causes suction towards the ceiling. This results in the same tendency for ICE as for IGE where lesser thrust is required to maintain or increase altitude as when the UAV is out of ceiling effect (OCE). On Figure 3.14 (bottom left) ceiling effect is illustrated.

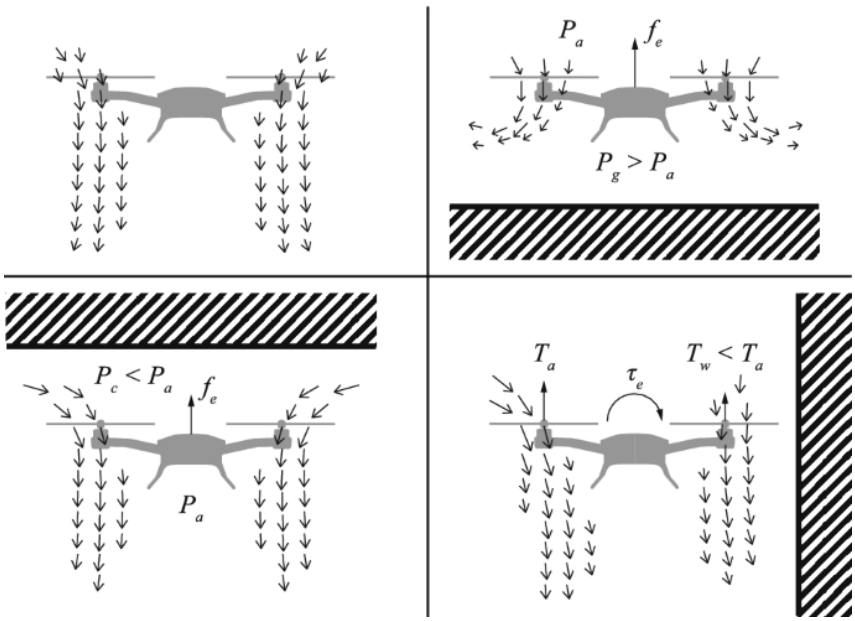


Figure 3.14: Illustration of the airflow for: (top left) no effect, (top right) ground effect, (bottom left) ceiling effect and (bottom right) wall effect. [65]

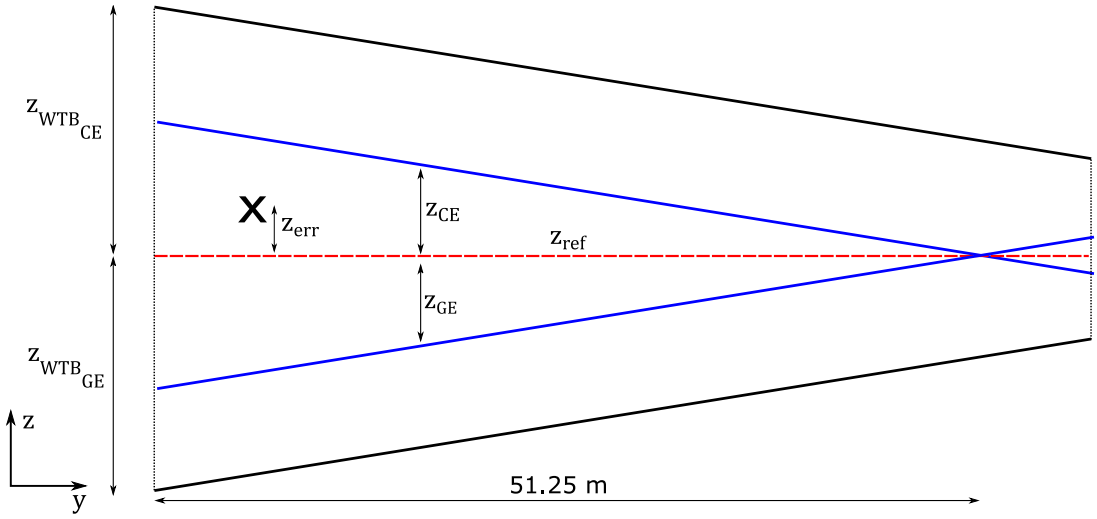


Figure 3.15: Simplified sketch of the WTB along with z axis definitions. The black X marks an arbitrary position of the UAV.

A simplified model of the WTB is illustrated in Figure 3.15 (**Assumption 9**). The black lines represent the outer walls of the WTB and the blue lines are the height when the UAV enters GE or CE. Lastly, the red line is the reference trajectory which was set as $z_{ref} = 0$. A linear function of the WTB walls was derived based on data from Table B.2 in Appendix B.1. Equation 3.41 describes the slope from the lower WTB wall to the reference trajectory. Further, Equation 3.42 represents the slope for the upper wall of the WTB down to the trajectory.

$$z_{WTB_{GE}}(y) = 0.012\, y - 0.615 \tag{3.41}$$

$$z_{WTB_{CE}}(y) = -0.012\, y + 0.615 \tag{3.42}$$

To determine the actual position of the UAV inside the WTB an error function is defined and is presented in Equation 3.43 were z_{err} is the distance between the trajectory and the UAVs z position where z denotes the UAVs z axis position.

$$z_{err} = -z_{ref} + z \tag{3.43}$$

The distance from the WTB wall where the UAV enters GE or CE is defined as $5R$ were R is the radius of the propellers. To determine when the UAV is IGE or ICE, the relative position of the UAV compared to the ground and ceiling are determined in Equations 3.44 and 3.45 respectively. When z_{GE} *or* $z_{CE} < 5R$, the UAV is either IGE or ICE.

$$z_{GE} = -z_{WTB_{GE}} + z_{err} \tag{3.44}$$

$$z_{CE} = z_{WTB_{CE}} - z_{err} \tag{3.45}$$

The effect of GE and CE on the thrust is seen in Equations 3.46 and 3.47 respectively [66] (**Assumption 10**). Here T_{IN} is the commanded input thrust and T_{OUT} is the actual thrust

seen from the UAV. When z_{GE} *or* $z_{CE} > 5R$ then $T_{IN} \approx T_{OUT}$ and GE or CE are negligible. Furthermore, when z_{GE} *and* $z_{CE} < 5R$ then the UAV is affected by GE and CE simultaneously which is expressed in Equation 3.48. The constants $\rho,\ a_1,\ a_2$ are tuning parameters and the values are seen in Table 3.2.

$$\frac{T_{OUT}}{T_{IN}} = \frac{1}{1 - \rho\left(\frac{R}{4z_{GE}}\right)^2} \tag{3.46}$$

$$\frac{T_{OUT}}{T_{IN}} = \frac{1}{1 - \rho\left(\frac{1}{a_1}\right)\left(\frac{R}{a_2 + z_{CE}}\right)^2} \tag{3.47}$$

$$\frac{T_{OUT}}{T_{IN}} = \frac{1}{\left(1 - \rho\left(\frac{1}{a_1}\right)\left(\frac{R}{a_2 + z_{CE}}\right)^2\right)\left(1 - \rho\left(\frac{R}{4z_{GE}}\right)^2\right)} \tag{3.48}$$

Parameter	**Value**
a_1	0.9
a_2	0.5
R	0.0635 $[m]$
ρ	1

Table 3.2: Parameters utilized in the GE and CE model.

The effects of the GE and CE can be seen in Figure 3.16 by a simple simulation. The UAV is supplied with hover thrust and given a constant thrust input for the y direction causing the UAV to fly into the WTB. The y and z position in the global frame can be seen at the left where the UAV is initialized at a length of $24.2[m]$ inside the WTB. When the UAV reach $24.8[m]$ in the y direction the z position are seen to change as the UAV enters GE and CE and the thrust required to hover is reduced. As the UAV is constantly supplied with the initial hover thrust it keeps flying upwards until it crashes into the ceiling where the rotors are cut off and it falls to the floor of the WTB. At the right the thrust seen from the UAV is shown. Before entering the translational disturbances the thrust is constant but when the UAV reaches GE/CE at 18 $[s]$ 2 sudden jumps occur for the thrust before it continuously increases until the UAV collides with the ceiling of the WTB. The sudden spike appearing when the disturbances affects is due to the ground and ceiling models do not have a smooth transition. This reaction might not occur in real life but deemed suitable for testing robustness for the later control structure. The slight drop right after entering the disturbances is caused as the UAV begins to fly upward and quickly leaves the ground effect again. The UAV is though only affected by ceiling effect for the rest of the simulation until crash. In Appendix G.4 the code which was implemented in MATLAB is presented.

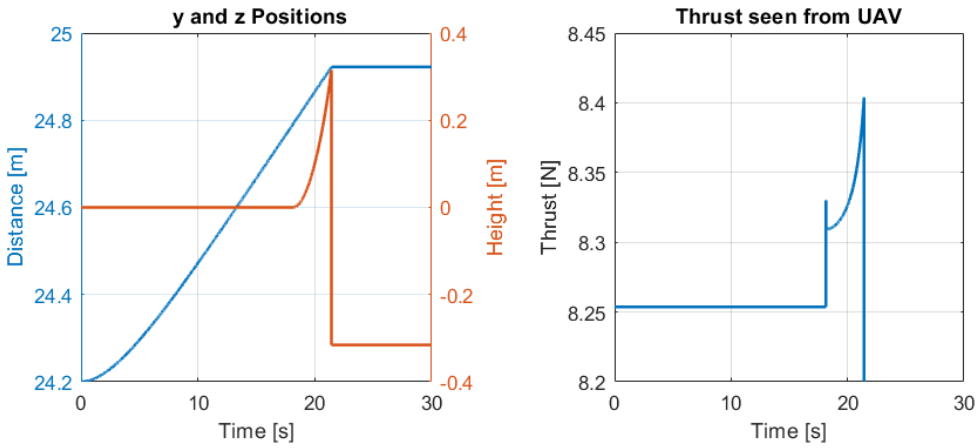


Figure 3.16: The effects on the UAV when entering GE/CE.

3.5.2 Rotational Disturbances

The rotational disturbances described in this section include WE and AI. WE occurs when the propellers are in a perpendicular position and in close vicinity of a wall. When the UAV is in wall effect (IWE) the airflow to the propellers closest to the wall reduces compared to the propellers further away. This causes the thrust from the propellers closer to the wall to be lower than the thrust from propellers further away, which then induces a torque around the center and the UAV flips towards the wall. WE is illustrated in Figure 3.14 (bottom right). AI is an effect which occurs as the airflow from the propellers collide with each other. Due to the angles of the rotors this disturbance has larger effect than with a conventional quadcopter. The established rotational disturbance model for this work will apply small but fast changing torques to the body. This is done to mimic effects of turbulent airflow. The model is defined as a statistic function following a normal distribution that randomly applies torques about all axes in both positive and negative direction. To determine the variances of the torques, row 4 for roll, row 5 for pitch and row 6 for yaw are extracted from the CAM. The CAM seen as the output of the design optimization are though based on the normalized displacement matrix. The CAM utilized for calculating the variances of the rotational disturbances are based on the CAM with the real displacements of the rotors and can be seen in Equation 3.49

$$\boldsymbol{V} = \begin{bmatrix} 0.839 & -0.839 & 0.839 & -0.839 & 0.839 & -0.839 & 0.839 & -0.839 \\ 0.484 & 0.484 & 0.484 & 0.484 & -0.484 & -0.484 & -0.484 & -0.484 \\ 0.250 & 0.250 & -0.250 & -0.250 & -0.250 & -0.250 & 0.250 & 0.250 \\ -0.0054 & -0.0252 & 0.0252 & 0.0054 & -0.0054 & -0.0252 & 0.0252 & 0.0054 \\ 0.0441 & -0.0327 & -0.0327 & 0.0441 & -0.0441 & 0.0327 & 0.0327 & -0.0441 \\ -0.0202 & 0.0261 & -0.0261 & 0.0202 & 0.0202 & -0.0261 & 0.0261 & -0.0202 \end{bmatrix} \tag{3.49}$$

The absolute values of each row concerning the torques are summarized where the row for roll is utilized as an example in Equation 3.50.

$$V_{\phi,total} = \sum_{i=1}^{4} V_{\phi,i} \qquad i = 1, 2, \cdots, 8 \tag{3.50}$$

To find the maximum variance the total value is multiplied by the maximum thrust the UAV can yield which is seen in Equation 3.51.

$$M_{\phi,max} = V_{\phi,total}\, f_{p,max} \tag{3.51}$$

Where $f_{p.max} = 10.414[N]$. With the maximum value of the torque found, the variance is chosen to be 5 % of that value which is seen in Equation 3.52 (**Assumption 11**).

$$M_{\phi,var} = (M_{\phi,max} \cdot 0.05)^2 \tag{3.52}$$

The same procedure is used to find $M_{\theta,var}$ and $M_{\psi,var}$. Table 3.3 presents the values of the variance for roll, pitch and yaw.

Parameter	Value
$M_{\phi.var}$	$1.87 \cdot 10^{-5}$
$M_{\theta.var}$	$1.18 \cdot 10^{-4}$
$M_{\psi.var}$	$4.28 \cdot 10^{-5}$

Table 3.3: Parameters utilized in the rotational disturbance model.

The values within Table 3.3 are chosen with a small variance at 5% of the torque that the given UAV design is capable of producing. The disturbance values are mostly chosen to investigate how sensitive the UAVs motions are and if the presented control structure can compensate for small disturbances. The variances are seen as the minimum peak values for the rotational disturbances and are later increased to test robustness of the control structure. With the stated variances the effects can be seen in Figure 3.17 where the UAV is applied with hover thrust in the body frame and affected by the rotational disturbances.

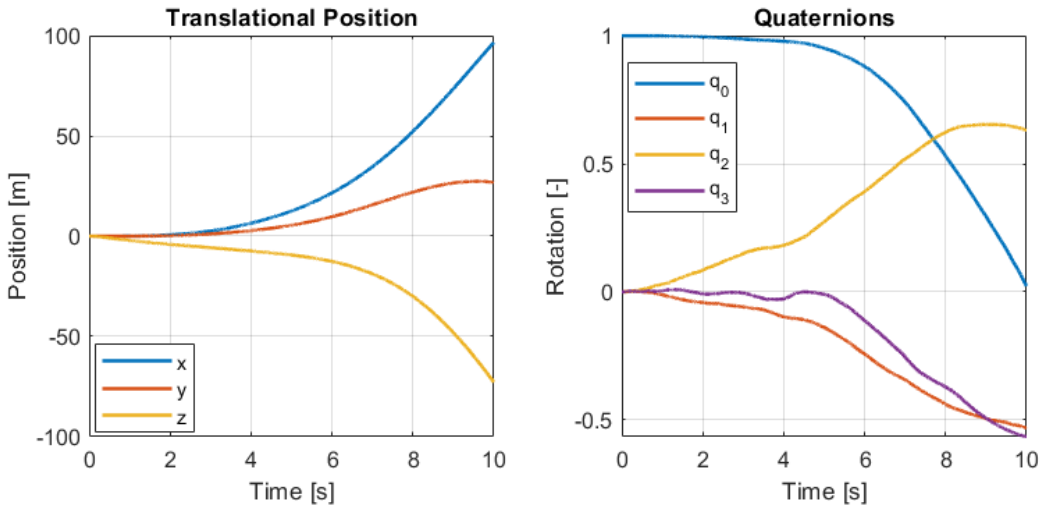


Figure 3.17: UAV supplied by hover thrust and affected by rotational disturbances.

The rotational disturbances affects the UAV instantaneously causing rotation seen from the quaternions to the right at Figure 3.17 as they deviate from the set point. This causes the hover thrust in the body frame to be redirected in the x and y direction in the global frame and the

UAV starts moving in all translational directions. This proves that even small disturbances have a significant influence on the motions of the UAV and must be considered in the later control structure.

Another aspect of the simulation model is to apply measurement noise for the states which can be monitored. The measurement noise is depending on the individual sensors utilized to measure the different states. In Section 3.6, the different measurements available are presented with the corresponding measurement noise models.

3.6 Measurement Noise

Different sensors are to be attached to the UAV to determine its position and orientation in relation to the WTB and the desired flight trajectory. The measurements of the sensors are affected by noise which is modelled in this section for simulation purposes. To obtain attitude and a general direction for yaw (magnetic north) a 9DOF inertial measurement unit (IMU) is used. The IMU considered is the BNO055 [67] consisting of an accelerometer measuring the translational accelerations, a gyroscope to measure rotational position and a magnetometer determining the yaw orientation from the earths magnetic field. The IMU already contains fusion filters to sort out measurement noise but due to the condition of the UAV flying within a confined space with electrical and magnetic disturbances, measurement noise is modelled for the quaternion measurements. The measurement noise for the IMU is modelled as white noise with a mean of zero whereas the frequency is determined from the fused output data stated as 100 $[Hz]$ in the data sheet. The variance is based on a deviation of 2 % of the quaternion measurement span of -1 to 1. An example of the quaternion measurement noise is seen in Figure. 3.18.

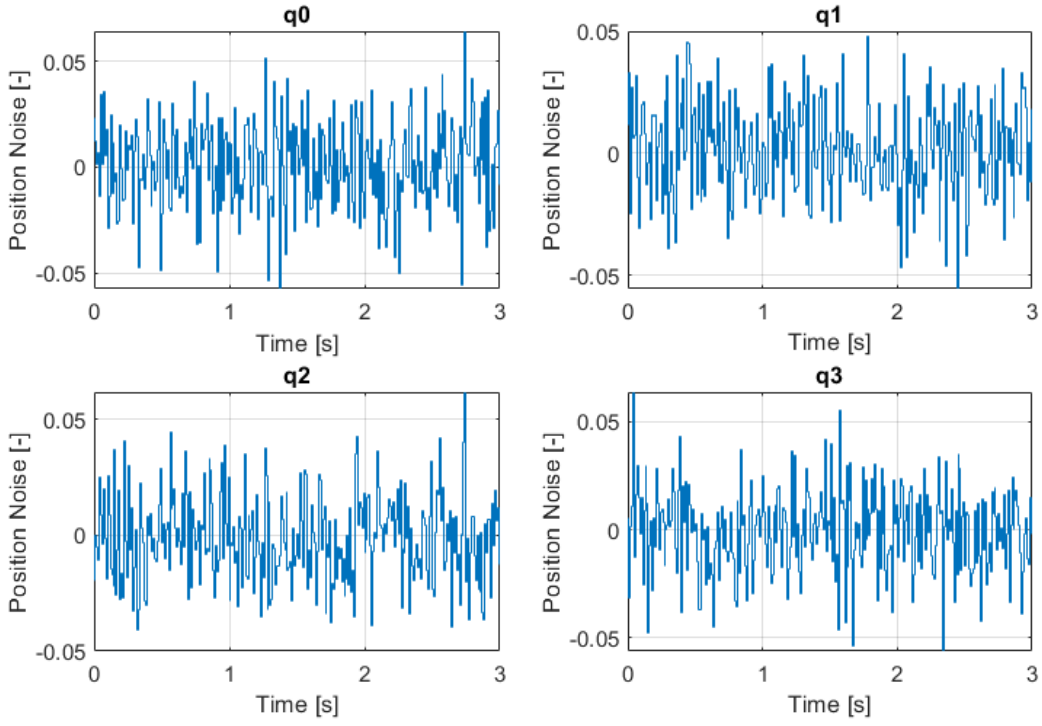


Figure 3.18: Quaternion measurement noise.

The noise models applied for the additional measurements are of same nature as for the quaternions though with different variances. For the translational position within the WTB two cameras are utilized. The cameras are the Intel RealSense Tracking Camera T265 [68] and the Intel RealSense Depth Camera D455 [69]. Both cameras contain a 6DOF IMU consisting of a accelerometer and a gyroscope. In this work it is assumed that the depth camera can be used to increase accuracy of the tracking camera without influencing the sampling frequency of the tracking camera which is maximum 200 $[Hz]$ as stated in Table 3.4. Elements in Table 3.4 are intentionally left blank to indicate knowledge we do not have or can't obtain from the sensors.

Measurement	**Tracking Camera T265**		**IMU BNO055**	
	$\boldsymbol{f_s}$ $[Hz]$	$\boldsymbol{\sigma^2}$	$\boldsymbol{f_s}$ $[Hz]$	$\boldsymbol{\sigma^2}$
$\boldsymbol{p}$	200	0.0004*	-	-
$\boldsymbol{\dot{p}}$	200	0.001*	-	-
$\boldsymbol{\ddot{p}}$	200	*	100	0.013
$\boldsymbol{\omega}$	200	*	100	*
$\boldsymbol{q}$	200	*	100	0.0004

Table 3.4

The stars (*) refers to values that must be assumed, since they were not mentioned in the data sheets [67–69] (**Assumption 13**).

Since 200 $[Hz]$ is the maximum attainable frequency it is found reasonable to assume that a realistically obtainable sampling frequency is 100 $[Hz]$ which is the same sampling frequency as the BNO055 (**Assumption 14**). Lastly it is assumed that sensor fusion is possible between the camera combination and the BNO055 for more reliable measurements (**Assumption 15**). Table 3.5 summarizes variances and sampling frequencies used for the white noise sensor models contained in the overall simulation model. The values are either assumed or directly found in data sheets of the candidate sensors.

Measurement	**Measurement Model**	
	$\boldsymbol{f_s}$ $[Hz]$	$\boldsymbol{\sigma^2}$
$\boldsymbol{p}$	100	0.0004
$\boldsymbol{\dot{p}}$	100	0.001
$\boldsymbol{\ddot{p}}$	100	0.01
$\boldsymbol{\omega}$	100	0.0004
$\boldsymbol{q}$	100	0.0004

Table 3.5

3.7 Model Summary

In this section the governing equations of the UAV model are summarized. In Section 3.1 a FOPDT model of the motors is derived as seen in Equation 3.53 where τ_s is the time constant and t_d is the time delay constant.

$$G_{FOPDT} = \frac{1}{\tau_s s + 1} e^{-t_d s} \tag{3.53}$$

In Section 3.2 the translational dynamics are derived, where the states are viewed in Equation 3.54 with $\mathbf{p}$ being the global position states, $\mathbf{q}$ being the global orientation states and $\boldsymbol{\omega}$ being the body frame angular velocity.

$$\mathbf{p} = \begin{bmatrix} x & y & z \end{bmatrix}^T \qquad \mathbf{q} = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^T \qquad \boldsymbol{\omega} = \begin{bmatrix} p & q & r \end{bmatrix}^T \tag{3.54}$$

The final model describing the translational acceleration is presented in Equation 3.55. Here $\mathbf{Q}$ is the rotation matrix shown in Equation 3.56. Furthermore, $\mathbf{F_B}$ are the forces acting on the UAV in body frame, m is the mass, g is the gravitational acceleration and e is an orientation vector.

$$\ddot{\mathbf{p}}(\mathbf{q}, \dot{\mathbf{p}}, \mathbf{F_b}) = \frac{1}{m}\mathbf{Q}\,\mathbf{F_B} - g\,\mathbf{e} - \frac{1}{m}C_d\dot{\mathbf{p}} \tag{3.55}$$

$$\mathbf{Q} = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_0q_2 + q_1q_3) \\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_0q_1 + q_2q_3) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{3.56}$$

Section 3.3 describes the derivation of the attitude dynamics, where the final model of the rotational acceleration is stated in Equation 3.57. $\mathbf{J}$ is the inertia matrix, $\boldsymbol{\tau_B}$ are the torques acting in the body frame, $\boldsymbol{\omega}$ is the angular velocity and C_d is the aerodynamic damping.

$$\dot{\boldsymbol{\omega}}(\boldsymbol{\omega}, \boldsymbol{\tau_b}) = \boldsymbol{J}^{-1}(\boldsymbol{\tau_B} - \boldsymbol{\omega} \times \boldsymbol{J}\boldsymbol{\omega}) \tag{3.57}$$

In Appendix C the derivative of the quaternion vector is calculated and can be seen in Equation 3.58. Here $\mathbf{W}$ is the transformation matrix containing the angular velocities and $\mathbf{q}$ is the quaternion vector.

$$\dot{\mathbf{q}}(\mathbf{q}, \boldsymbol{\omega}) = \frac{1}{2}\mathbf{W}\mathbf{q} = \frac{1}{2}\begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix}\begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{3.58}$$

The system model is summarized in Equation 3.59.

$$f_{sys} = \begin{bmatrix} \ddot{\boldsymbol{p}}(\boldsymbol{q}, \dot{\boldsymbol{p}}, \boldsymbol{F_b}) \\ \dot{\boldsymbol{\omega}}(\boldsymbol{\omega}, \boldsymbol{\tau_b}) \\ \dot{\mathbf{q}}(\boldsymbol{q}, \boldsymbol{\omega}) \end{bmatrix} \tag{3.59}$$

and the state vector is seen in 3.60

$$\boldsymbol{\Theta} = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} & x & y & z & p & q & r & q_0 & q_1 & q_2 & q_3 \end{bmatrix}^T \tag{3.60}$$

A disturbance model was derived in Section 3.5 describing both translational and rotational disturbances. The translational disturbance models for GE, CE and combined GE/CE are presented in Equations 3.61, 3.62 and 3.63 respectively.

$$\frac{T_{OUT}}{T_{IN}} = \frac{1}{1 - \rho\left(\frac{R}{4z_{GE}}\right)^2} \tag{3.61}$$

$$\frac{T_{OUT}}{T_{IN}} = \frac{1}{1 - \rho\left(\frac{1}{a_1}\right)\left(\frac{R}{a_2+z_{CE}}\right)^2} \tag{3.62}$$

$$\frac{T_{OUT}}{T_{IN}} = \frac{1}{\left(1 - \rho\left(\frac{1}{a_1}\right)\left(\frac{R}{a_2+z_{CE}}\right)^2\right)\left(1 - \rho\left(\frac{R}{4z_{GE}}\right)^2\right)} \tag{3.63}$$

As for the rotational disturbance model, it was modeled as a statistic function described in Equation 3.64 where μ is the mean, σ^2 is the variance and N denotes that the function follows a normal distribution. Likewise, in Section 3.6 a measurement noise models are stated as statistic functions seen in Equations 3.65

$$M_R = N(\mu, \sigma_R^2) \tag{3.64}$$

$$M_{meas} = N(\mu, \sigma_{meas}^2) \tag{3.65}$$

Figure 3.19 illustrates a block diagram which summarizes all the equation which were utilized in the open loop model.

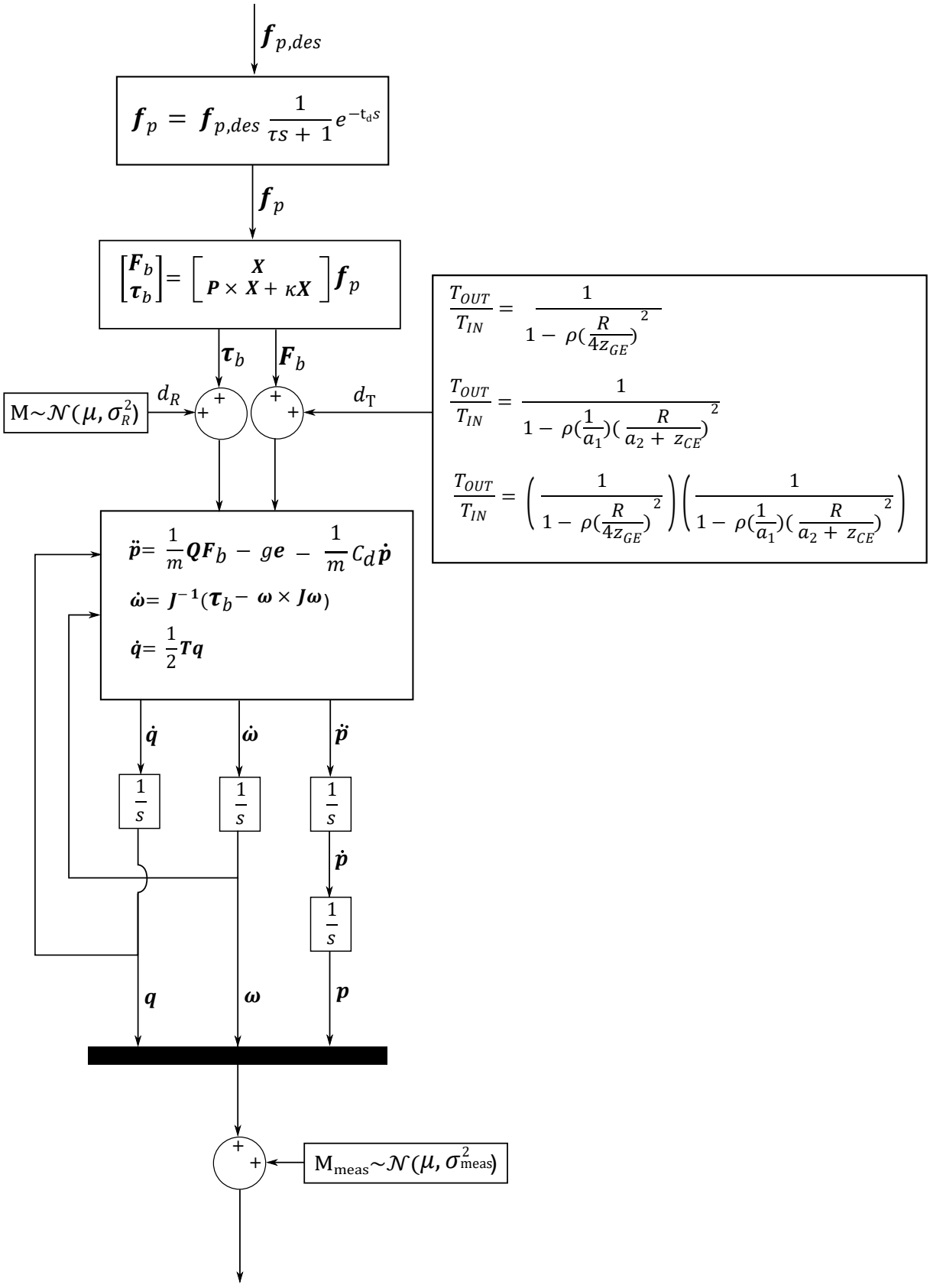


Figure 3.19: A block diagram of the full model.

4 | Control Considerations

This chapter establish a candidate control structure for trajectory tracking and disturbance rejection of the designed UAV. At first the desired control criteria are presented followed by a literature review of existing relevant control structures. The review is followed by considerations on availability of measurements, choice of control allocation method, assessment of significant closed loop bandwidths and lastly the proposed control structure is presented.

Abbreviation	**Meaning**
BLDC	BrushLess Direct Current
BS	BackStepping
DOF	Degrees Of Freedom
EKF	Extended Kalman Filter
FAUAV	Fully Actuated Unmanned Aerial Vehicle
IMU	Inertial Measurement Unit
LMPC	Linear Model Predictive Control
LQR	Linear Quadratic Regulator
MCU	MicroController Unit
MPC	Model Predictive Control
NMPC	Nonlinear Model Predictive Control
PID	Proportional Integral Derivative
SMC	Sliding Mode Control
UAV	Unmanned Aerial Vehicle
V-SLAM	Visionbased-Simultaneous Localization And Mapping
WTB	Wind Turbine Blade

Table 4.1: Abbreviations used in this chapter.

4.1 Control Criteria

In this section the chosen control criteria are outlined. They are chosen based on the environment in which the UAV is supposed to carry out the inspection tasks. These criteria are summarized in the list below.

1. Follow aggressive and smooth trajectory of sinusoidal tendency with a frequency corresponding to ω_{WTB}.
2. Keep the orientation aligned with the inertial frame.
3. Robustness towards unknown and unmodelled disturbances.

The first control criterion on the list concerns translational movement since the UAV can be commanded to move in global x, y and z directions independently from angular movement. It is required that the magnitude of translational disturbances in any direction are less than the forces that can be asserted by the actuators on the body of the UAV. This can be justified by rewriting equation 3.15 to contain a bounded disturbance term as seen in equation 4.1.

$$\ddot{\boldsymbol{p}} = \frac{1}{m}\boldsymbol{Q}(\boldsymbol{F}_{cmd} - \boldsymbol{F}_{ext}) \tag{4.1}$$

It is implied that the gravitational force is contained in $\boldsymbol{F}_{ext}$. If the acceleration is to be controlled in the presence of disturbances the inequality in Equation 4.2 must be fulfilled.

$$|\mathbf{F}_{cmd}| > |\mathbf{F}_{ext}| \tag{4.2}$$

In addition the closed loop bandwidth for translational motion must be much larger than the frequency of the disturbances that affects the systems translational movements as stated in 4.3.

$$|\omega_{cl,trans}| >> |\omega_{WTB}| \tag{4.3}$$

If this can be achieved with a candidate controller, the closed loop system can react to a fast changing reference and additional disturbances without colliding with the inner walls of the WTB.

Likewise, the magnitude of the attitudinal disturbances and gyroscopic effects must be less than the maximum absolute moments applied by the actuators. The gyroscopic effects are contained in the term $(-\boldsymbol{\omega} \times \boldsymbol{J\omega})$. This can be seen by rewriting 3.33 into 4.4 and including a bounded disturbance term $(\boldsymbol{\tau_{ext}})$.

$$\dot{\boldsymbol{\omega}} = \boldsymbol{J}^{-1}(\boldsymbol{\tau}_b - \boldsymbol{\tau_{ext}} - \boldsymbol{\omega} \times \boldsymbol{J\omega}) \tag{4.4}$$

The closed loop system bandwidth related to attitudinal motions must also be much larger than the frequency of the attitudinal disturbances as seen in Equation 4.5.

$$|\omega_{cl,att}| >> |\omega_{att,dist}| \tag{4.5}$$

If this condition can be achieved by means of a candidate control structure the FAUAV is in theory stabilizable since the closed loop response can react to an error signal faster than the dominant disturbance frequencies affecting the systems attitudinal dynamics.

The second control criterion states that a constant attitude reference must be kept during flight. The constant reference should be aligned with the inertial frame. In Euler angles this can be expressed as $\boldsymbol{\eta} = [0\ 0\ 0]^T$ and in quaternions this reference is $\mathbf{q} = [1\ 0\ 0\ 0]^T$.

In order to account for ***the third control criterion*** disturbance estimation is desired. By acquiring an estimate of the magnitude and direction of the disturbances at a specific moment the candidate control structure can bring this information into account when calculating the control input to the FAUAV and in theory achieve better robustness.

With the control criteria, the foundation for the selection of a suitable control structure is defined. On the other hand the control criteria only point in a general direction and do not considerably limit the field of possible control structures. To get a better understanding of applicable structures with their respective strengths and weaknesses a literature review is constructed in the following section.

4.2 Literature Review on Relevant Control Methods

In this section recent articles are considered from a control perspective in order to gain an understanding of efficient and modern control techniques for UAV trajectory tracking.

The reviewed articles are based on UAVs with a control architecture designed for aggressive flight of under actuated UAVs [30, 70–76] and on the control architecture of FAUAV similar to the proposed UAV design [37, 43, 53, 77–81].

4.2.1 Under Actuated UAVs

This section reviews state of the art control of under actuated UAVs. Trajectory tracking of UAVs is a well documented and researched issue in which numerous control structure solutions exist, many of which are variations of one another. Popular controller types include PID, LQR, backstepping control (BS), SMC, and MPC. Even though other control methods exist this review is limited to these controller types as the aforementioned controller types primarily has been researched for applicability to under actuated UAVs. This renders the possibility of researching the extension of these controller types to be applicable to fully or over actuated vehicles. Before presenting literature on the control issue of FAUAVs, recent state of the art control structures for under actuated UAVs are investigated. The simplest trajectory tracking control laws, in our opinion, are based on PID controllers in which x and y positions are indirectly controlled by pitch and roll movements respectively in a cascaded manner. Yaw and height are likewise controlled by respective PID controllers. Examples of such structures are found in [70, 71]. Further, in [71] a comparative study is conducted on PID and LQR approaches where the LQR controller enable position convergence closer to the defined reference than the PID controller. From this study LQR is considered favorable compared to PID. Further in [72] LQR was used for accurate attitude and elevation control in an indoor environment. Since multi rotor system dynamics is described by a set of nonlinear differential equations, nonlinear control methods such as SMC, BS and MPC are suitable. In [73] a pure SMC control structure is derived to control both attitude and elevation. The method was shown to be satisfactory for set point tracking, however no results were shown for tracking of a time varying trajectory. In [74] a BS control structure was derived and tested. 4 BS controllers were used to control attitude and elevation based on 4 subsystem that combined constitutes the full quadcopter system. The BS control structure achieved robust trajectory tracking in the presence of disturbances and model uncertainties. MPC has in literature been researched and used for both attitude trajectory tracking and for position trajectory tracking. It is challenging to use a costly procedure for attitude control as the update rate of an attitude controller must be fast. In fact the control iteration frequency should be significantly greater than the closed loop attitudinal system bandwidth to achieve stable flight. This is further elaborated in Section 4.5. In [75] an MPC attitude controller was designed where the control structure is cascaded with the inner loop being the attitude tracking MPC controller and the outer loop as an LQR position tracking controller. The controller structure showed highly agile for maneuvering and attained accurate tracking capability. Another MPC

trajectory tracking control structure is investigated in [30]. The article revolve a cascaded control structure with attitude control as the inner loop and translational control for the outer loop. The given UAV do already have a controller implemented for fast attitudinal control whereas a thorough comparison of trajectory tracking utilizing LMPC and NMPC is conducted. For the outer loop controller designs it is deemed necessary to include the inner loop dynamics. One conclusion reached by the authors is, that the NMPC is better for aggressive trajectory tracking presumably due to the more accurate prediction model. A final piece of work contained in the review of modern control methods for trajectory tracking of under actuated UAVs is a comparative study based on the following controller types; BS, SMC and MPC. In the study [76] the three controllers are ranked in terms of robustness, input optimality, intelligence, ability to adapt, tracking ability and disturbance rejection. In terms of robustness the SMC has highest ranking followed by MPC. The MPC ranks highest in input optimality which is a result of the controllers build-in optimization process. According to the study the tracking ability ranks highest for the BS and SMC. All three controller types yield the same rank regarding disturbance rejection.

In summary MPC, SMC and BS are found interesting due to their robustness and disturbance rejection properties. LQR is also considered a candidate since it improves performance when compared to conventional PID control. LQR can also be extended as a nonlinear control method by online linearization and repeatedly solving the algebraic Ricatti equation.

4.2.2 Fully and Over Actuated UAVs

The majority of articles on FAUAVs is, to our knowledge, published between 2015 and now and remains an active field of research both in terms of efficient application dependent designs and specialized control strategies. To simplify discussion FAUAV referrers to both fully and over actuated UAVs. In this section advances made in regard to control of such UAVs are presented. Fixed rotor FAUAVs differ from tilting rotor FAUAVs as the tilted rotor FAUAVs contain additional actuator dynamics (i.e. tilt angles) and thus additional control variables. In theory it is possible to implement the same control schemes to each type of FAUAV and assign tilt commands and motor force commands by means of a control allocation algorithm. However, one comparative study already has been made on control methods for trajectory tracking of tilting rotor FAUAVs [77]. Therefore in this review we specifically focus on fixed rotor FAUAVs.

Table 4.1 provides an overview and a block diagram representation of most of the control structures discussed here. It has been noted that the majority of control structures attempts to decouple translational dynamics from attitude dynamics as seen in block diagram 1 through 3. This implies that control design can be divided into two separate controllers, one for translational position and one for attitude based on individual subsystem dynamics. It should be noted that there exists several control structures similar to the structures 1 through 3 presented in Table 4.1, but is left out of the review. An example is the control structure presented in [37].

The first control structure in Table 4.1 consists of PID controllers for both Position and attitude. The position controller is regulating position by calculating desired body forces (F_{des}) based on the position error. The attitude controller functions in a similar manner by calculating the desired body moments (τ_{des}) based on orientation error. Control allocation is then utilized to map the vector $[F_{des}\ \tau_{des}]^T$ into rotor forces. The control method proves feasible as it keeps the position errors within $\pm 0.18\,[m]$ and orientation errors within $\pm 0.04\,[rad]$ as both controllers are running at $50\,[Hz]$. In 4.1 indoor flight is conducted in an ideal environment thus disturbances

are present. For the purpose of inspection of a WTB a tracking error of $\pm 0.18\,[m]$ is not acceptable however the attitude controller is deemed suitable.

The second control structure is slightly extended in complexity compared to the above structure. The position controller has a feed-forward term and uses additional information on measurements and reference trajectory including higher order derivatives of the position. The attitude control law is cascaded with an inner loop body rate controller running at $1000\,[Hz]$ in synchronicity with the control allocation. Both the position and outer loop attitude controllers runs at $50\,[Hz]$. The position error is kept within $\pm 0.18\,[m]$ and orientation error within $\pm 0.14\,[rad]$ when tracking the reference of a circle of $1.5\,[m]$ radius with a velocity of $1.95\,[^m/_s]$. Experiments are conducted indoors in an ideal environment.

The third control structure is decoupled like the above discussed structures. Like the first control structure the attitude control is not cascaded and always tracking the same set point equivalent to alignment with the inertial frame. The orientations are controlled individual with sliding mode controllers. The SMCs utilize chattering attenuation and are mentioned to be asymptotically stable. Only simulation studies are presented from which tracking errors are not deducible, however through simulation results stabilization is achieved. It is not mentioned at which frequency the SMCs are running, which is an important tuning parameter of the sliding mode controller in general. It is however mentioned that a PixHawk controller unit is intended to be used for experiments. This is restricting the controller maximum possible frequency to $433\,Hz$, as this is the fastest rate at which the PixHawk controller unit can read and process incoming signals. In their future work they mentioned that the prototype is yet to be realized and their control structure is to be implemented.

The fourth control structure is a control method unknown to the group and is considered out of scope for analysis and discussion, it is mentioned to make the reader aware of the limitations of this review.

The fifth control structure a BS controller is derived in a mathematically rigorous manner it is not directly mentioned whether it is a centralized or decoupled control structure. The control law contains a gain matrix $\in \mathbb{R}^{6\times 6}$ that is directly multiplied with the position-orientation vector. Such gain matrix could introduce coupling terms if it has nonzero values on off diagonal elements and therefore this control method could be centralized. The control gain matrices utilized are diagonal with all off diagonal elements zero and therefore their implementation and controller tuning results in a decoupled control structure. The developed control structure is validated by simulation. The tracking error can be observed to be within $\pm 0.1\,[m]$ and within $\pm 0.1\,[rad]$ approximately. The simulations are performed with introduction of bounded disturbances in the form of body forces and moments, the disturbances has maximum value of $\pm 0.5\,[N]$ and $\pm 0.5\,[Nm]$. It is not mentioned at which frequency the controller is running.

The sixth control structure is based on a nonlinear MPC controller. The MPC solves an online optimization problem in order to determine an optimal system input to bring the vehicle to the position reference as efficiently and smooth as possible. Efficient and smooth refers to the input and states being penalized in the optimization problem. The optimization is subject to the system dynamics where they utilize a rigid body model of the vehicle similar to the model derived in Chapter 3. The MPC controller was tuned to penalize pose and twist errors as well as rate of change of input. However, no specific tuning method was mentioned. Further an EKF is utilized to estimate disturbance forces ($\hat{F}_{dist}$) and moments ($\hat{\tau}_{dist}$) acting on the system. The

MPC takes the disturbance estimates into consideration when calculating optimal inputs. Lastly a state estimator is employed such that estimates of all states are available. The optimization executes at a rate of $100\,[Hz]$ where a new control input is calculated then passed to a zero order hold. The allocation is executed at a rate of $200\,[Hz]$ using the most recent control input. The prediction horizon for the optimization is set to $(N_p = 20)$ and utilizing a discretization time of $T_{s,MPC} = 0.05\,[s]$. Through experiments a maximum position and attitude tracking error was found to be $\pm 0.17\,[m]$ and $\pm 0.4\,[rad]$, respectively.

In summary all control methods discussed for under actuated UAVs have already been tested for FAUAVs to some extent with the exception being LQR control. BS control is a computationally efficient nonlinear control method, though as the fifth control structure is purely simulation based a performance comparison with the experimentally tested controllers would be inconclusive.

Even though MPC is a more computationally expensive method compared to both SMC, BS and LQR it is considered to be a suitable control method. This is partly because it is assumed that a trajectory generation algorithm to predict future reference points is available and these future reference points can be taken into account by the MPC. Another reason MPC is found suitable is the possibility to impose penalties on e.g. input, rate of change of input and state error as well as actuator constraints. In conclusion MPC is chosen as the candidate controller used further in this work.

To narrow the field of control structures some practical aspects have to be taken into consideration as seen in the following section. Section 4.3 presents the measurement units desired to be implemented on the presented UAV design together with an analysis of the available measurements which are assumed to be obtained. Section 4.4 covers different possibilities for applying the control allocation converting general forces and moments to the individual propeller thrusts and vice versa. Lastly. Section 4.5 presents a closed loop bandwidth assessment of UAV design to analyse the controller iteration frequency required for the final control structure.

4.3 Availability of Measurements and Trajectory Generation

As stated in Section 3.6 the tracking and depth camera attached to the UAV can be fused to establish the UAVs position and translational velocity in the body frame. Furthermore by fusing IMU and image processed data from the cameras with the measurements from the BNO055 full pose of the vehicle can be obtained. Full pose in this context means global accelerations $\ddot{\boldsymbol{p}}$, velocities $\dot{\boldsymbol{p}}$ and positions $\boldsymbol{p}$ together with quaternions $\boldsymbol{q}$, Euler angles $\boldsymbol{\eta}$ and rotational body rates $\boldsymbol{\omega}$. With the stated measurements all the control structures within Figure 4.1 are obtainable. Additionally these measurement can be utilized for designing full state observers for noise reduction or observers for disturbance estimation.

Full pose feedback is not sufficient for online trajectory generation on its own, however the cameras can be utilized for this purpose. As an example in the recent article [82] a method of online trajectory generation and obstacle avoidance is described utilizing only a stereo camera similar to the aforementioned camera candidates. Furthermore a masters thesis is currently being written by Morten Veng, in which he develops a method of online trajectory generation using the depth and tracking camera. The title of the project is "Navigation and Predictive Control with Active Disturbance Compensation for Autonomous Quadrotor Flight inside Wind Turbine Blades" and can be found in the AAU project library after submissions. It should be

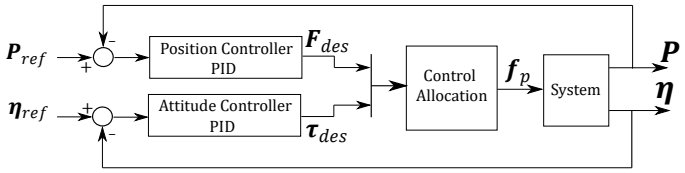
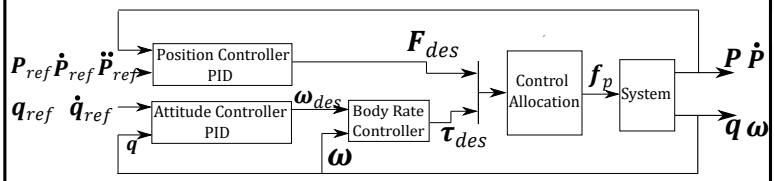
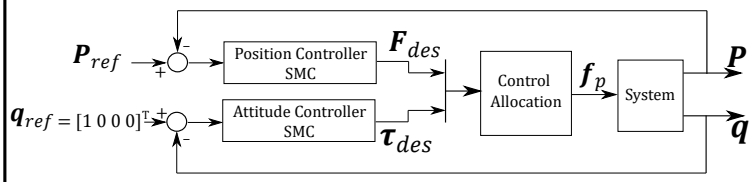
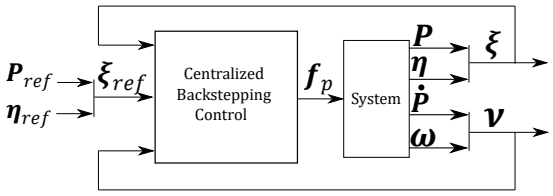
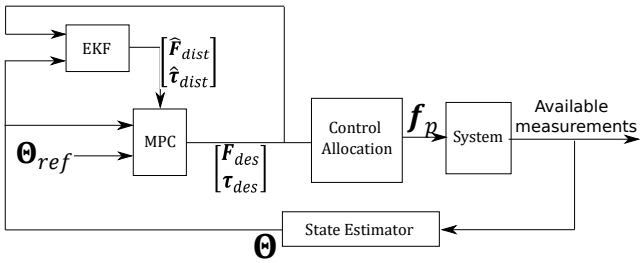
#	Control Block Diagram	Definitions
1	***Article:*** Design, Modeling and Control of Fully Actuated 2D Transformable Aerial Robot with 1 DoF Thrust Vectorable Link Module 	$\boldsymbol{P} = [x\ y\ z]^T$ $\boldsymbol{\eta} = [\phi\ \theta\ \psi]^T$ $\boldsymbol{q} = [q_0\ q_1\ q_2\ q_3]^T$ $\boldsymbol{F_{des}} = [F_{x,des}\ F_{y,des}\ F_{z,des}]^T$ $\boldsymbol{\tau_{des}} = [\tau_{x,des}\ \tau_{y,des}\ \tau_{z,des}]^T$ $f_p = [f_1\ f_2\ f_3\ f_4\ f_5\ f_6\ f_7\ f_8]^T$ $\boldsymbol{\omega} = [p\ q\ r]^T$
2	***Article:*** Design, Modeling and Control of an Omni-Directional Aerial Vehicle 	
3	***Article:*** Design of a Fully Actuated Passively Tilted Multirotor UAV with Decoupling Control System 	
4	***Article:*** Port-Hamiltonian Passivity-Based Control on SE(3) of a Fully Actuated UAV for Aerial Physical Interaction Near-Hovering Passivity Based Control (Out of Scope)	
5	***Article:*** Mechanical Design, Modelling and Control of a Novel Aerial Manipulator 	$\boldsymbol{\xi} = [\boldsymbol{P}\ \ \boldsymbol{\eta}]^T$ $\boldsymbol{\nu} = [\dot{\boldsymbol{P}}\ \ \boldsymbol{\omega}]^T$
6	***Article:*** Trajectory Tracking Nonlinear Model Predictive Control for an Overactuated MAV 	$\boldsymbol{\Theta} = [\dot{\boldsymbol{P}}\ \boldsymbol{P}\ \boldsymbol{\omega}\ \boldsymbol{q}]^T$

Figure 4.1: **1:** [78] **2:** [43] **3:** [79] **4:** [80] **5:** [53] **6:** [81]

noted that no such method is developed in this thesis and this discussion provides background for making the assumption that online trajectory generation is possible. Furthermore it is assumed possible to obtain a finite number of future trajectory reference points (**Assumption 16**).

4.4 Control Allocation

This section contains a brief discussion of the control allocation methods as it is necessary for mapping desired body forces and moments as utilized for some of the control structures in Figure 4.1. The allocation solves the system of linear equations as described in Section 2.3.2 Equation 2.6. The same Equation is solved for $\boldsymbol{F}_{prop}$, which in this context is referred to as $\boldsymbol{F}_{prop,des} \in \mathbb{R}^{8\times 1}$.

$$\boldsymbol{F}_{prop,des} = \boldsymbol{V}^{-1}\boldsymbol{F}_{gen,des} \tag{4.6}$$

Where $\boldsymbol{F}_{gen,des} \in \mathbb{R}^{6\times 1}$ refers to desired generalized forces and moments. The issue in this context is Equation 4.6 can not be obtained because $\boldsymbol{V} \in \mathbb{R}^{6\times 8}$ does not have an inverse as it is non-square. The purpose of control allocation is to approximate Equation 4.6 in the cases where such system is either underdetermined or overdetermined.

In a recently written thesis [83, p.14-32] a literature survey of existing control allocation methods can be found. The thesis further contributes with a new allocation algorithm. One of the methods proposed are the commonly used Moore-Penrose pseudo inverse [43, 81]. A property of this solution is that it provides a unique solution, however it is not possible to impose actuator constraints. Also this method can result in a rapidly changing rotor input command which in turn will decrease lifetime of the BLDCs as well as discharge the battery faster than a optimization algorithm subject to input rate constraints. The method proposed in [53] uses a search algorithm to iteratively find desired propeller forces. Such optimization procedure allows to impose constraints and to choose certain mapping solutions that ensure the force input of a certain rotor is not changing rapidly. This could be achieved by introducing penalty rate of change of propeller force inputs in the cost function. Such methods are computationally expensive and is thereby preferable for processes which do not require a fast calculation time or for systems with high computational power. A third option, also covered in [83], is the redistributed pseudo inverse method, which is a combination of aforementioned methods where an initial guess is obtained by means of the Moore-Penrose method. This initial guess is then used in an optimization problem. The result of this method is a less expensive optimization than the method proposed in [53] but still requires more than the Moore-Penrose pseudo inverse.

Since, the controllers developed in Chapter 6 performs online optimization and furthermore can handle constraints, this is of no concern to implement in the control allocation. Furthermore, the calculation of the pseudo-inverse is desired to be performed for each iteration to gain an optimal input and thus the Moore-Penrose method is chosen.

The Moore-Penrose is a simple method to cope with the inverse of a non-square matrix. The general problem of an overactuated system can be stated as in Equation 4.7.

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \quad (\boldsymbol{A} \in \mathbb{R}^{m\times n},\ m < n,\ \boldsymbol{x} \in \mathbb{R}^{n},\ \boldsymbol{b} \in \mathbb{R}^{m}) \tag{4.7}$$

$$\tag{4.8}$$

with $m = 6$ and $n = 8$ lesser equations are present than unknowns and taking the inverse of $\boldsymbol{A}$ to solve for $\boldsymbol{x}$ results in infinite solutions. The Moore-Penrose method introduce an elegant way to compute a unique solution. AT first the transpose of $\boldsymbol{A}$ is multiplied on both sides of the equal sign giving Equation 4.9.

$$\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{b} \tag{4.9}$$

As the product $\boldsymbol{A}^T\boldsymbol{A}$ becomes a square matrix this can be inverted resulting in Equation 4.10.

$$\boldsymbol{x} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b} \tag{4.10}$$

The pseudo inverse can then be defined as in Equation 4.11.

$$\boldsymbol{A}^\dagger = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T = \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{A}^T)^{-1} \tag{4.11}$$

Due to the properties of the Moore-Penrose method the right most term can be stated which is the same term yielded in [83]. [84, p.1-4]

As mention, the Moore-Penrose method is unconstrained and the inputs does thereby have to be considered in the controller design to restrict the solution generated by the pseudo inverse. On the other hand the Moore-Penrose offers a computationally cheap solution instead of implementing additional optimization algorithms.

4.5 Closed Loop Bandwidth Assessment

In this section a bandwidth estimate is made from a state space representation of the system described in Equation 3.59. Such estimate will help to determine at which iteration frequency or equivalently at what sample time the controller must update the next input command (**Assumption 17**).

The state space model formulation is given in Equation 4.12.

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}\,; \qquad \mathbf{y} = \mathbf{C}\mathbf{x} \tag{4.12}$$

Where $\mathbf{A}$ is the system matrix, $\mathbf{B}$ is the input matrix and $\mathbf{C}$ is the output matrix. The state vector is as previously stated in Equation 3.60. The system and input matrices are computed from the system Equations 3.59 by partial differentiation with respect to each state and input. More specifically they are computed as shown in Equation 4.13 and 4.14.

$$\mathbf{A} = \begin{bmatrix} \frac{\partial \ddot{\mathbf{p}}(\mathbf{q},\dot{\mathbf{p}},\mathbf{F_b})}{\partial \dot{x}} & \frac{\partial \ddot{\mathbf{p}}(\mathbf{q},\dot{\mathbf{p}},\mathbf{F_b})}{\partial \dot{y}} & \cdots & \frac{\partial \ddot{\mathbf{p}}(\mathbf{q},\dot{\mathbf{p}},\mathbf{F_b})}{\partial q_3} \\ \frac{\partial \dot{\mathbf{p}}(\dot{\mathbf{p}})}{\partial \dot{x}} & \cdots & \cdots & \cdots \\ \frac{\partial \dot{\boldsymbol{\omega}}(\boldsymbol{\omega},\boldsymbol{\tau_b})}{\partial \dot{x}} & \cdots & \cdots & \cdots \\ \frac{\partial \dot{\mathbf{q}}(\mathbf{q},\boldsymbol{\omega})}{\partial \dot{x}} & \cdots & \cdots & \cdots \end{bmatrix}_{13\times 13} \tag{4.13}$$

$$\mathbf{B} = \begin{bmatrix} \frac{\partial \ddot{\mathbf{p}}(\mathbf{q},\dot{\mathbf{p}},\mathbf{F_b})}{\partial F_x} & \frac{\partial \ddot{\mathbf{p}}(\mathbf{q},\dot{\mathbf{p}},\mathbf{F_b})}{\partial F_y} & \cdots & \frac{\partial \ddot{\mathbf{p}}(\mathbf{q},\dot{\mathbf{p}},\mathbf{F_b})}{\partial M_z} \\ \frac{\partial \dot{\mathbf{p}}()}{\partial F_x} & \cdots & \cdots & \cdots \\ \frac{\partial \dot{\boldsymbol{\omega}}(\boldsymbol{\omega},\boldsymbol{\tau_b})}{\partial F_x} & \cdots & \cdots & \cdots \\ \frac{\partial \dot{\mathbf{q}}(\mathbf{q},\boldsymbol{\omega})}{\partial F_x} & \cdots & \cdots & \cdots \end{bmatrix}_{13\times 6} \tag{4.14}$$

It is evident that $\mathbf{A}$ and $\mathbf{B}$ are computed as first order Jacobian matrices with number of columns corresponding to the length of the state and input vector respectively. The output matrix, $\mathbf{C}$, is given in Equation 4.15 and constructed to output $\mathbf{p}$ and $\mathbf{q}$.

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.15}$$

Further the open loop transfer function matrix is computed as in Equation 4.16.

$$\mathbf{G}_{7\times 6} = \mathbf{C}(s\mathbf{I}_{13\times 13} - \mathbf{A})^{-1}\mathbf{B} \tag{4.16}$$

The bandwidths to be analysed for control purposes are the closed loop bandwidths. To generate these, the MATLAB command seen in Equation 4.17 are utilized.

$$\omega_{i,j} = bandwidth(feedback(\boldsymbol{G_{i,j}}, 1)) \tag{4.17}$$

here i indicates the respective motion and j indicates the corresponding input. The function $feedback$ is the MATLAB command to close the loop and the 1 indicates unity feedback. A proportional controller with a gain of 1 is assumed to be present. Table 4.2 contains the largest bandwidth that relates a force input to a position output and likewise contains the largest bandwidth that relates a moment input to a rotation.

	F_x	M_x
x	$\omega_{trans} = 1.6\ [\frac{rad}{s}]$	-
q_1	-	$\omega_{rot} = 29.4\ [\frac{rad}{s}]$

Table 4.2

The bandwidth estimates of Table 4.2 provides valuable information in regards to how fast a controller must update the next input command, i.e. control iteration frequency. From [85] an iteration frequency between $10-30$ times that of the approximated bandwidth is recommended as a trade-off between accuracy and MCU workload. A minimum requirement is therefore, that controller iteration frequency be at least 10 times faster than the largest observed bandwidths (**Assumption 18**).

Referring back to the control criteria discussion, Equation 4.3 contains an important criteria. The criteria states that closed loop dynamics regarding translational motion must be faster than the the oscillations of the wing. However, by comparing the bandwidth relating x to F_x it is found to be less than ω_{WTB} and thus violating the criteria. Equation 4.18 describe what has been observed in this context.

$$|\omega_{trans}| = 1.6\,[\frac{rad}{s}] < |\omega_{WTB}| = 4.49\,[\frac{rad}{s}] \tag{4.18}$$

The bandwidths in Table 4.2 are very rough approximations and it was found the bandwidth relating x to F_x can be increased to a value around $\omega_{trans} = 10[\frac{rad}{s}]$, by replacing the trivial proportional controller with a gain of one with a PID controller. In conclusion it can be expected that Equation 4.19 holds true for sophisticated and fast reacting MPC controller.

$$|\omega_{trans}| > |\omega_{WTB}| \tag{4.19}$$

In summary, we can expect issues with tracking the trajectory described in Chapter 2 Section 2.2.

4.6 Proposed Control Structure

For this section, the proposed control structure is presented. The control structure is determined to fulfill the control criteria presented in 4.1 by utilizing the prior knowledge obtained from the literature review and the practical considerations. An illustration of the final control structure can be seen in Figure 4.2.

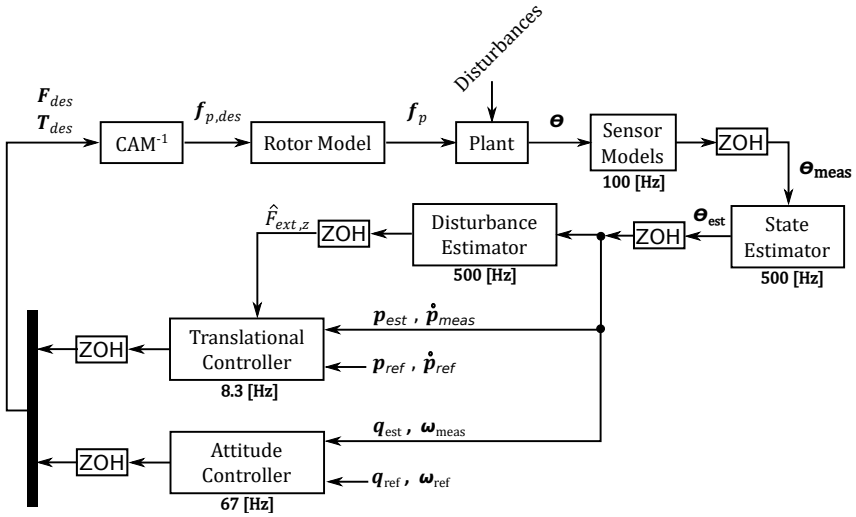


Figure 4.2: Illustration of the control structure outline.

It can be seen in Figure 4.2 that the control structure is consisting of 2 controllers, one for attitude and one for translational motion. Additionally the structure incorporates 2 observers, one for state estimation and one for disturbance estimation. At first the division of the controllers into one translational and one rotational is chosen due to the different tasks and disturbances related to each motion. The attitude of the UAV is desired to be kept constant during flight

which requires a robust and fast controller to compensate for the rotational disturbances. The translational motions on the other hand have constant changing references and the modelled disturbances are only affecting slowly (compared to rotational) in the global z direction. For following a smooth trajectory a predictive and lesser aggressive controller is desired. As the design allows to control each DOF individually it is chosen to apply the 2 different controllers. To contribute to a more smooth and accurate usage of the states for the controllers it is chosen to design a state observer for noise reduction. The estimated states are not just used for the controllers but also for a second observer to estimate the translational disturbance occurring in the z direction. This disturbance estimate is included in the translational controller to increase robustness against the disturbance. The design procedure and implementation of the observers are described in Chapter 5 while the controller designs are described in Chapter 6.

5 | Sensor Measurement and State Estimation

Throughout this chapter different methods for full state estimation and disturbance estimation are evaluated. The unscented Kalman filter (UKF) is chosen for state estimation and noise attenuation from which the unscented transform (UT) is elaborated together with the UKF algorithm. Further a augmented Kalman filter (AKF) is designed to estimate the translational disturbance present in the simulation model.

Abbreviation	**Meaning**
AKF	Augmented Kalman Filter
CE	Ceiling Effect
EKF	Extended Kalman Filter
GE	Ground Effect
GRV	Gaussian Random Variable
KF	Kalman Filter
PF	Particle Filter
SSE	Sum of Squared Errors
UAV	Unmanned Aerial Vehicle
UKF	Unscented Kalman Filter
UT	Unscented Transform
WTB	Wind Turbine Blade

Table 5.1: Abbreviations used in this chapter.

In Chapter 3 the mathematical model is presented together with disturbance and measurement noise models. Though all states are obtainable from the sensor measurements a state observer is desired for noise attenuation. Further as disturbances are introduced with GE/CE for translational motion it is desired to estimate these disturbances for the purpose of disturbance rejection. For these conditions to be met, different state and disturbance estimators are evaluated.

5.1 State and Disturbance Estimators

For state estimation numerous solutions have been developed all with different purposes and limitations. To confine the number of observer designs this evaluation focus on the estimators presented in [86]. The article distinguish between 3 categories of observer designs with the first one being the early observers as output and input based estimators and the Luenberger observer. The early observers lay ground for the two newer categories, the modern observers,

and the disturbance observers. The modern observers as the Kalman Filter (KF) and extensions hereof are useful tools for state estimation by the assumption that the plant model is an accurate representation of the real system. The KF utilizes a formulation of the disturbances to form an optimal solution and can be designed to handle nonlinear models with the extended Kalman filter (EKF) or UKF on the cost of computational power.
The disturbance observers adopt the idea of a combined state and disturbance estimation of lesser described plant models whereas the unknown system dynamics are ascribed as disturbances. One disturbance estimator is the unknown input observer. The unknown input observer is structured as a Luenberger observer but further augmented to transcribe the unmodelled disturbances as additional states. Another approach is the extended state observer to reject unmodeled dynamics and perturbations.[86]
It is assumed that the plant model is accurately reflecting the reality and thus modern observer design is chosen as the focus for state estimation (**Assumption 19**). Additionally, due to the non-linearities occurring in the plant model it is desired to investigate the extensions of the KF.

The 3 estimation methods investigated are the EKF, the UKF and the particle filter (PF). The EKF utilize local linearization to accommodate for soft non-linearities. It is an effective tool with high computational efficiency but requires the model to be fully differentiable at all time and for the noise to be Gaussian distributed. The UKF uses unscented transformation to convert the state vector to a set of weighted points. It is still a local approximation but do not rely on linearization thus the model does not have to be differentiable. This results in a stronger robustness toward non-linearities though the noise has to be Gaussian distributed as for the EKF. The third option is the PF that utilize the Monte Carlo method to generate a number of weighted points for the estimation. The PF can handle strong non-linearities and the noise can attain a different distribution than Gaussian. Some considerations have to be accounted for considering the number of points and their individual weighting but most dominant is the trade-off between accuracy and computational cost. In [87] the 3 different estimators have been evaluated and compared through the expected value test, the whiteness test and the weighted-sum-squared-residual test. The model used in [87] include strong non-linearities causing the EKF to deviate heavily compared to the other 2 candidates. Further as the noise do not follow a Gaussian distribution, the PF has an advantage yielding the best results though the authors conclude the UKF might be sufficient if this is not the case. [87].

The comparison of the 3 types of estimators is also conducted in [88] whereas the authors apply an additive Gaussian distributed noise. The authors conclude the EKF and UKF to be sufficient for real-life purposes as both filters have similar and satisfying results based on estimation error and computational cost. Contrary the PF has lesser estimation accuracy and further requires a higher computational cost. The same is the case in [89] where the work is extended to involve particle flow filtering and ensemble KF. The EKF and UKF are still among the most accurate filters and slight improvements from the other estimators results in greatly increased calculation time. In [90] a evaluation has been made for the EKF and UKF for 30 different real-life trajectories with the PF used for comparison. Here the EKF and UKF yield similar results whereas the EKF has the smallest computational cost but the UKF is more robust against changing tuning parameters. Other methods have been utilized to improve the estimation accuracy of the EKF and UKF as in [91]. This article compares the EKF and UKF with a PF of 5000 and 10000 samples but further presents 2 methods for merging UKFs, the pre-merging and the post-merging. The methods consists of 2 UKFs each estimating states by utilizing different output feedback. For the pre-merging method each UKF generate a feedback

and then combine the filters whereas the post-merging method utilize the UKF outputs for the merging and the feedback is then the merged output. The 2 merging methods yield better results than the EKF and the standard UKF whereas the post-merging UKF even surpasses the PF with 5000 samples in accuracy. Additionally the calculation time of the merging methods is greatly lesser than for the 2 PFs.

In conclusion, the PF has the advantage of handling strong non-linearities and is not restricted to the noise being Gaussian distributed. Though as noise is modelled as an additive Gaussian distributed noise for this project the disadvantages of the PF outweigh the advantages. At first former articles show only slightly better results for the PF with the noise being Gaussian distributed and, in some cases, the EKF and UKF yield more accuracy. Next, the trade-off between accuracy and computational cost makes the PF unpractical for real-life applications. The EKF and UKF have similar results with the UKF being slightly more accurate and reliable but slightly more costly as well. The UKF have the advantage of handling the strong non-linearities and seems to have a better robustness toward changing tuning parameters and the UKF is thereby further investigated for the purpose of state estimation.

5.2 UKF Algorithm

The UKF utilizes UT which is based on the principal idea that "it is easier to approximate a probability distribution than it is to approximate an arbitrary nonlinear function or transformation" [92, 93]. This means, that the UKF only needs the mean and covariance to the estimation while other nonlinear estimation methods either require assumptions of the model behavior or have a high computational cost impractical for real life applications. The UT is a linear transformation of states where a set of sigma points are chosen to represent possible state vectors, and these are utilized through the nonlinear equations. By this the statistics of the variables are captured by the transformation. The UKF uses a minimal of sigma points capturing the true mean and covariance of the Gaussian random variable (GRV) to reduce the calculation time and still approximate the true outcome to a 2nd order accuracy by Taylor series expansion. The estimation accuracy is further elaborated in Appendix E. In basic the UT is divided into 3 stages:

1. Defining sigma points and corresponding weights of the input space with known mean and covariance.
2. Nonlinear transformation of the sigma point.
3. Approximating the output variables mean and covariance from the transformed points and corresponding weights.

The concept of the UT is illustrated in Figure 5.1.

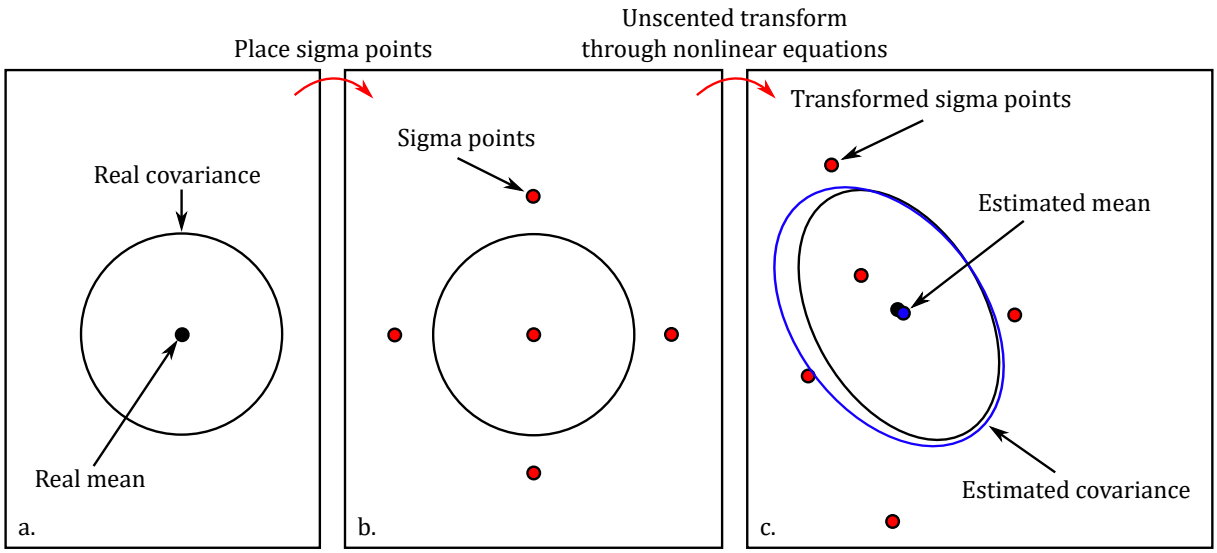


Figure 5.1: Illustration of the UT. [94]

At Figure 5.1a the real mean and covariance of the system are captured. Next, the sigma points are placed with one at the real mean and the others scattered symmetrically from the real mean as seen in Figure 5.1b. The sigma points in Figure 5.1b are then transformed through the nonlinear equations and placed as in Figure 5.1c. The transformed sigma points state an estimated mean and covariance, the blue dot and solid blue line respectively, approximating the real mean and covariance, the black dot and solid black line respectively, to a 2nd order accuracy. [92–95]

The UKF utilized in this project is based upon the original idea of the UKF giving the number of sigma points generated as $2L+1$ where L is the number of states. To initialize the algorithm a set of sigma points are generated from the initial state estimate as seen in Equations 5.1-5.3.

$$\boldsymbol{\chi}_0 = \overline{\boldsymbol{x}} \tag{5.1}$$

$$\boldsymbol{\chi}_i = \overline{\boldsymbol{x}} + (\sqrt{c\boldsymbol{P}_{x,0}})_i \quad , \qquad i = 1, 2, \ldots, 2L \tag{5.2}$$

$$\boldsymbol{\chi}_i = \overline{\boldsymbol{x}} - (\sqrt{c\boldsymbol{P}_{x,0}})_{i-L} \quad , \qquad i = 1, 2, \ldots, 2L \tag{5.3}$$

whereas $\boldsymbol{P}_{x,0}$ is the initial estimate of the state error covariance matrix and the subscripts i and $i-L$ correspond to the respective column, L is the number of states and c is a scalar determined in Equation 5.4.

$$c = L + \lambda \tag{5.4}$$

$$\lambda = \alpha^2(L + \kappa_\lambda) - L \tag{5.5}$$

Where α and κ_λ are tuning coefficients. Usually, κ_λ is chosen as zero while α is a real number close to zero. To calculate the sigma points it is needed to take the square root of the state error covariance matrix resulting in an optimization problem. To remove eventual complications in computation of the square root, such as infinitely small solutions, the square root is calculated from the eigenvalues. For this the MATLAB command $eig(\boldsymbol{A})$ is utilized as seen in Equation 5.6.

$$[\boldsymbol{V}, \boldsymbol{D}] = eig(c\boldsymbol{P}_{x,0}) \tag{5.6}$$

here $\boldsymbol{D}$ is a diagonal matrix of eigenvalues and $\boldsymbol{V}$ contains the right eigenvalues to satisfy the condition $c\boldsymbol{P}_{x,0}\boldsymbol{V} = \boldsymbol{V}\boldsymbol{D}$. The square root is then calculated in Equation 5.7.

$$\sqrt{c\boldsymbol{P}_{x,0}} = \boldsymbol{V}\boldsymbol{D}.^{1/2}\boldsymbol{V}^{-1} \tag{5.7}$$

With $\boldsymbol{D}.^{1/2}$ as the elementwise square root of $\boldsymbol{D}$. The set of sigma points are then utilized in the UT given in Equation 5.8.

$$\boldsymbol{\chi}_{i,UT} = f(\boldsymbol{\chi}_i, \boldsymbol{u}) \tag{5.8}$$

The mean of the transformed sigma points is calculated in Equation 5.9.

$$\overline{\boldsymbol{x}}_{UT} = \sum_{i=0}^{2L+1} W_{i,x}\boldsymbol{\chi}_{i,UT} \tag{5.9}$$

where $W_{i,x}$ is the respective weightings of the sigma points defined as in Equation 5.10 and Equation 5.11.

$$W_{0,x} = \frac{\lambda}{L+\lambda} \tag{5.10}$$

$$W_{i,x} = \frac{1}{2(L+\lambda)} \quad , \qquad i = 1, 2, \ldots, 2L \tag{5.11}$$

The weightings have to satisfy the relation given in Equation 5.12.

$$\sum_{i=0}^{2L+1} W_i = 1 \tag{5.12}$$

The same procedure is utilized to find the mean of the outputs seen in Equation 5.13.

$$\overline{\boldsymbol{y}}_{UT} = \sum_{i=0}^{2L+1} W_{i,x}\boldsymbol{\gamma}_{i,UT} \tag{5.13}$$

The state error covariance matrix can be calculated from the sigma points and the mean of the states by Equation 5.14.

$$\boldsymbol{P}_{x,UT} = \sum_{i,0}^{2L+1} W_{i,P}(\boldsymbol{\chi}_{i,UT} - \overline{\boldsymbol{x}}_{UT})(\boldsymbol{\chi}_{i,UT} - \overline{\boldsymbol{x}}_{UT})^T + \boldsymbol{Q} \tag{5.14}$$

with $\boldsymbol{Q}$ as the process covariance matrix and $W_{i,P}$ as the respective weightings defined in Equation 5.15 and Equation 5.16.

$$W_{0,P} = \frac{\lambda}{L+\lambda} \tag{5.15}$$

$$W_{i,P} = (1-\alpha^2+\beta) + \frac{1}{2(L+\lambda)} \quad , \qquad i = 1,2,\ldots,2L \tag{5.16}$$

here β is chosen from prior knowledge of Gaussian distribution. The next step is the update phase whereas the Kalman gain is calculated and utilized to update the states and the state error covariance matrix. To calculate the Kalman gain the output and cross error covariance matrices are defined in Equation 5.17 and Equation 5.18 respectively.

$$\boldsymbol{P}_{y_{UT}} = \sum_{i=0}^{2L+1} W_{i,P}(\boldsymbol{\gamma}_{i_{UT}} - \overline{\boldsymbol{y}}_{UT})(\boldsymbol{\gamma}_{i_{UT}} - \overline{\boldsymbol{y}}_{UT})^T + \boldsymbol{R} \tag{5.17}$$

$$\boldsymbol{P}_{xy_{UT}} = \sum_{i=0}^{2L+1} W_{i,P}(\boldsymbol{\chi}_{i_{UT}} - \overline{\boldsymbol{x}}_{UT})(\boldsymbol{\gamma}_{i_{UT}} - \overline{\boldsymbol{y}}_{UT})^T \tag{5.18}$$

here $\boldsymbol{R}$ is the output covariance matrix. From these the Kalman gain is calculated as in Equation 5.19 and utilized for the update process seen in Equation 5.20 and 5.21. [94, 95]

$$\boldsymbol{K} = \boldsymbol{P}_{xy_{UT}}\boldsymbol{P}_{y_{UT}}^{-1} \tag{5.19}$$

$$\hat{\boldsymbol{x}} = \overline{\boldsymbol{x}}_{UT} + \boldsymbol{K}(\boldsymbol{y} - \overline{\boldsymbol{y}}_{UT}) \tag{5.20}$$

$$\hat{\boldsymbol{P}}_x = \boldsymbol{P}_{x,UT} - \boldsymbol{K}\boldsymbol{P}_{y_{UT}}\boldsymbol{K}^{-1} \tag{5.21}$$

5.3 Implementation

The UKF which was implemented in MATLAB and Simulink is seen in the flowchart presented in Figure 5.2.

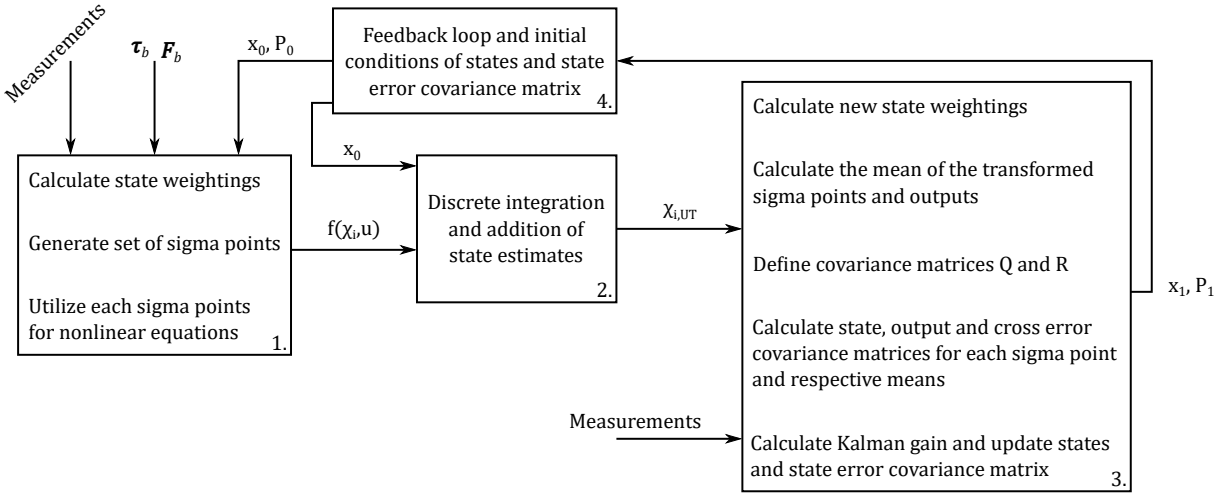


Figure 5.2: Progress of the UKF algorithm in the simulation model.

Within the first block in Figure 5.2 the thrust, torque, states and the state error covariance matrix are seen as inputs where the sigma points are generated and utilized through the nonlinear UT. The outputs of block 1 are the accelerations and velocities which are calculated through

the nonlinear equations. The accelerations and velocities are fed into block 2 together with the previous states where discrete integration is performed to find the change in velocities and positions. The previous states are added to the changes within block 2. Next, the outputs of block 2 are fed into block 3 together with the measurements of the simulation model outputs. Within block 3 the mean of the transformed sigma points and the outputs are calculated and utilized in the error covariance matrices. The Kalman gain is then calculated and utilized to update the states and the state error covariance matrix. The outputs of block 3 are fed back into block 1 and 2 through block 4 where the initial conditions of the states and the state error covariance matrix are defined.

The states chosen for the UKF to estimate are the translational x, y and z positions and the rotational positions given in quaternions. This results in a total of 7 states giving 15 sigma points to be generated. The additional variables within the nonlinear equations are chosen as the measurements which lower the computational cost as each additional state to be estimated adds 2 sigma points to be generated.
To tune the UKF, the parameters within the state and output covariance matrices are determined through an iterative algorithm sweeping a predetermined area of samples for candidates. With both the Q and R matrices being 7×7 they consists of 49 parameters each, but some simplifications are stated to make the tuning process easier. At first all the off-diagonal entries are set to zero reducing each matrix to 7 diagonal tuning parameters. Secondly, the coefficients correlated to translational position in both matrices are chosen to have the same penalty and the same goes for the rotational positions. Thereby the number of tuning parameters are reduced to 4 and the covariance matrices can be described as in Equation 5.22 and 5.23.

$$\boldsymbol{Q} = \begin{bmatrix} Q_T \cdot \boldsymbol{I}_{3\times3} & \boldsymbol{0}_{4\times3} \\ \boldsymbol{0}_{3\times4} & Q_R \cdot \boldsymbol{I}_{4\times4} \end{bmatrix} \tag{5.22}$$

$$\boldsymbol{R} = \begin{bmatrix} R_T \cdot \boldsymbol{I}_{3\times3} & \boldsymbol{0}_{4\times3} \\ \boldsymbol{0}_{3\times4} & R_T \cdot \boldsymbol{I}_{4\times4} \end{bmatrix} \tag{5.23}$$

with $\boldsymbol{I}_{n\times n}$ being identity matrices and Q_T, Q_R, R_T and R_R being the tuning parameters. The tuning algorithm consists of 3 grids of different resolutions for the parameters with the outer grid as the starting point. The different grids assign values for the parameters by using parameter loops before choosing the best candidate. An illustrative explanation of the parameter loops and the tuning algorithm are given in Figure 5.3 with a 2D example.

From the 2D example only 2 parameters are to be determined through an inner and an outer parameter loop. Starting by the outer grid, the outer loop is defined to run N number of iterations with an increment of 1. The first tuning parameter is then assigned the value of 10^{i-1} with i being the current outer loop iteration. Given the first parameter, the inner loop is initialized to run N number of iterations starting by assigning the value 10^{j-1} to the second parameter with j being the current inner loop iteration. Afterwards a simulation is run where the sum of squared errors (SSE) are calculated between the real and estimated outputs. The inner loop is executed for each iteration of the outer loop to attain the SSE of all combinations within the outer grid. The candidate with the lowest SSE is stored before moving to the middle grid whereas the candidate acts as the center value of the middle grid. The middle grid then repeats the process from the outer grid but with a finer resolution to only sweep points surrounding

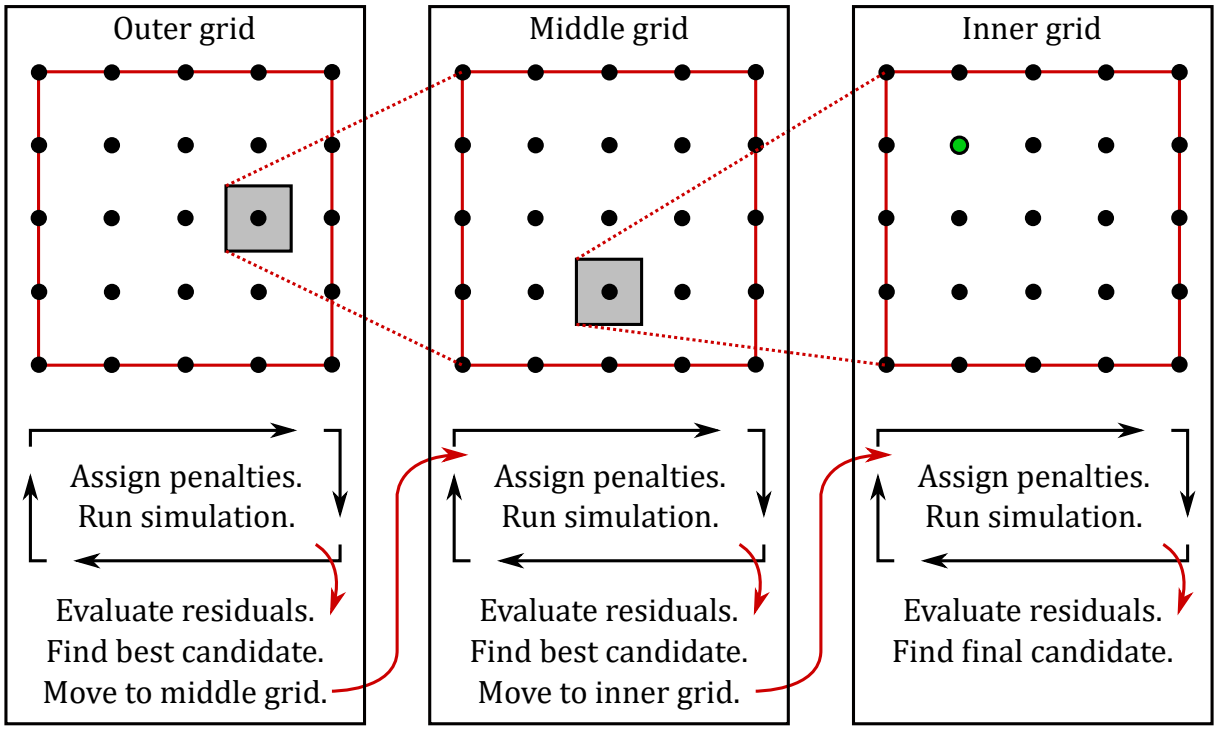


Figure 5.3: Pseudo code of the UKF tuning process.

the chosen candidate. After the process within the middle grid a new candidate is chosen and utilized in the inner grid repeating the process with a finer mesh from which the final tuning parameters are chosen and represented by the green point at Figure 5.3.

For the real algorithm 4 parameters are to be determined thus giving 4 parameter loops within each grid. The resolutions of the mesh grid are chosen as 1 for the outer grid, 0.25 for the middle and 0.05 for the inner grid. For the simulation used in the algorithm the UAV is supplied by hover thrust in the z direction and 0.01 $[N]$ in the y direction open loop to make the UAV fly into the WTB. The UAV is initialized at a length of 24.75 $[m]$ within the WTB thus, right at the limit before it hits GE/CE. No rotational disturbances are applied but the positive thrust in the y axis causes the UAV to fly into the translational disturbances. The simulation is run for 5 seconds and the SSE is based on all of the 7 outputs. The parameters used for the UKF and the final tuning coefficients are seen in Table 5.2.

N	α	κ_λ	β	Q_T	Q_R	R_T	R_R
5	10^{-4}	0	2	$10^{1.6}$	$10^{-0.7}$	10^{0}	10^{2}

Table 5.2: Parameters of the basic UKF.

The results of the UKF estimation can be seen in Figure 5.4.

The translational position estimates are seen to follow the real positions but with some fluctuations increasing with time as seen in Figure 5.4. These fluctuations are caused as the estimates tries to mimic the measurement noise. The rotational estimates are close to the reference and the fluctuations occurring are much smaller than the measurement noise and do not seem to increase with time as seen in Figure 5.4. The UKF is tuned for the UAV as it is affected by translational disturbances but without rotational disturbances. To test robustness

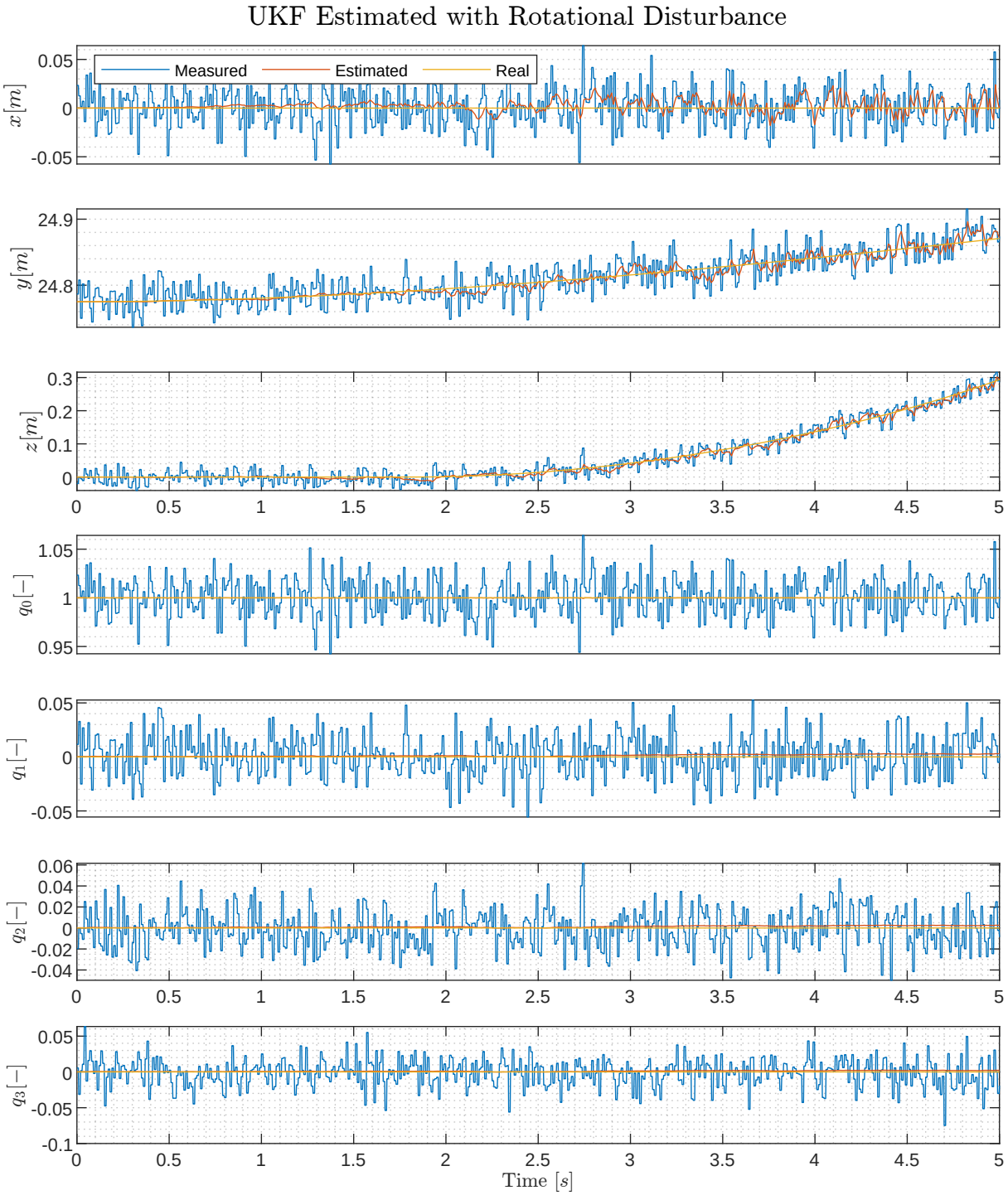


Figure 5.4: Results of the estimated translational positions compared to the real states.

against these disturbances Figure 5.5 shows the results of the UAV as it is initialized in hover position but affected by rotational disturbances.

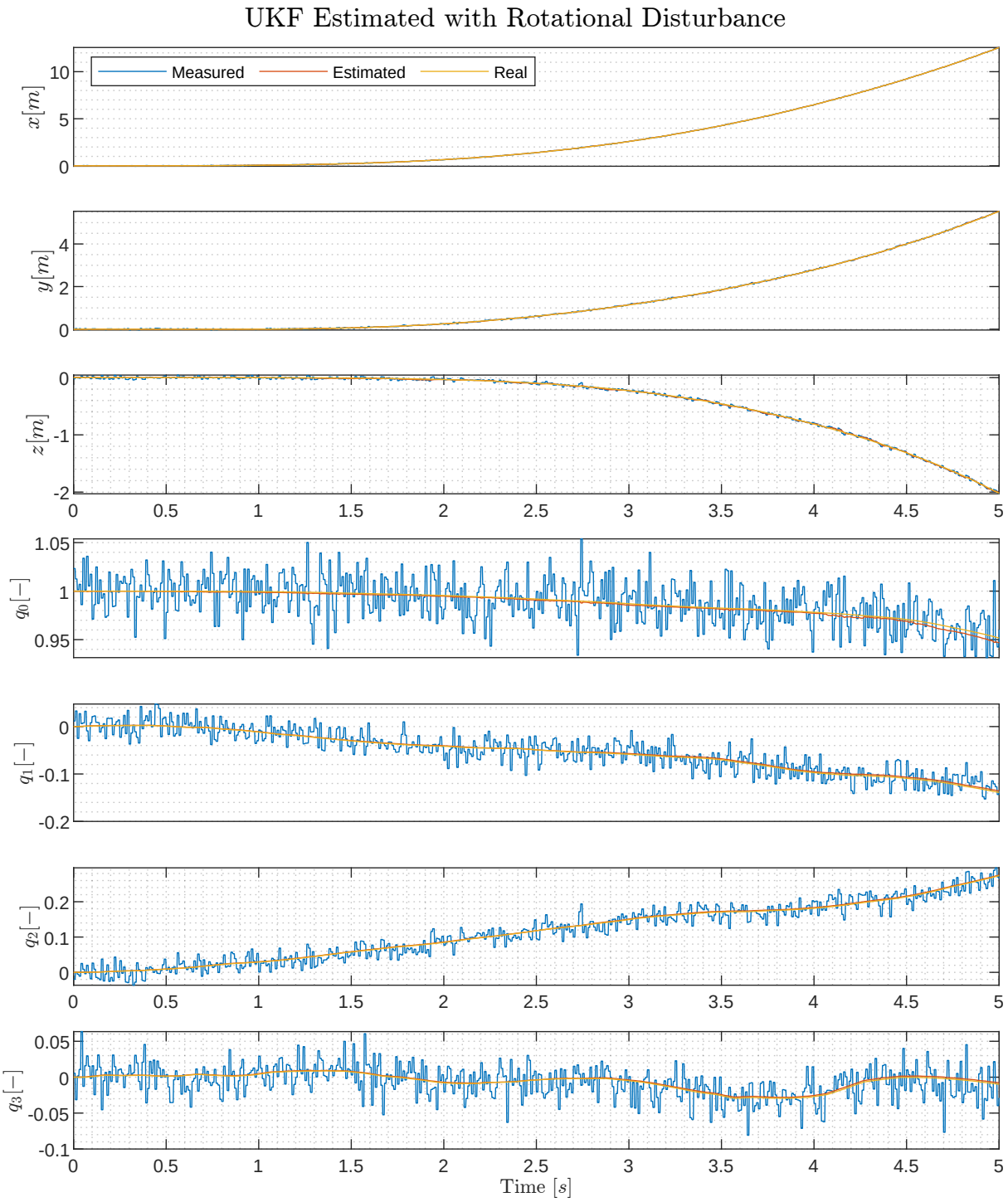


Figure 5.5: Results of the estimated translational positions compared to the real states.

For both the translational and rotational positions the estimates follow the real states though with a slight offset for q_3. The motion of the UAV for the translational positions is due to the thrust for hover being applied in the body frame and as the UAV starts to rotate the thrust is directed in the x and y directions.
From the results of the UKF estimates, it is concluded that the UKF is accurate as a state

observer for the UAV but for control purposes it is further desired to determine the additional thrust seen from the UAV as it enters the GE/CE. Therefore an additional observer is designed to estimate the translational disturbances. In Appendix G.5 the MATLAB code and the Simulink block diagram utilized in the UKF design is presented.

5.4 Augmented KF for Disturbance Estimation

In this section the mathematical procedure of the disturbance estimator design is presented. This observer is specifically designed to estimate the translational disturbance affecting the FAUAV in the z direction i.e. GE and CE. The purpose of such observer is to include the estimated disturbance in the controllers developed in Chapter 6.

5.4.1 General Kalman Filter Structure

The augmented Kalman filter (AKF) provides optimal state estimation based on output residuals between output measurements and estimated states while producing an estimate for the unknown disturbance. The structure of the AKF is based on the more general KF structure. The KF can be applied to systems where process noise ($\boldsymbol{w}$) and measurement noise ($\boldsymbol{v}$) is present and follows a normal distribution with a mean of zero and an associated variance ($\boldsymbol{\sigma}$) as stated in Equation 5.24.

$$\begin{aligned} w_j &\sim N(0, \sigma_j) \qquad j = 1, ..., n_x \\ v_l &\sim N(0, \sigma_l) \qquad l = 1, ..., n_y \end{aligned} \tag{5.24}$$

Where n_x is number of states of the KF and n_y is number of measurements used in the KF. A model is used for prediction of states and has the general form as in Equation 5.25.

$$\begin{aligned} \dot{\boldsymbol{x}}_n &= \boldsymbol{A}\boldsymbol{x}_n + \boldsymbol{B}\boldsymbol{u}_n + \boldsymbol{w}_n \\ \boldsymbol{y}_n &= \boldsymbol{C}\boldsymbol{x}_n + \boldsymbol{v}_n \end{aligned} \tag{5.25}$$

The subscript n denotes a discrete time instant. The mathematical algorithm is given in equation 5.26 through 5.30, from which a prediction phase and a correction phase are present within the filter. In the prediction phase the above prediction model is utilized to predict the next state estimates $\boldsymbol{n}_{n+1|n}$ based on the current input ($\boldsymbol{u}_{n|n}$) and state estimate ($\hat{\boldsymbol{x}}_{n|n}$). Next, the new predicted state error covariance matrix $\boldsymbol{P}_{n+1|n}$ is calculated from the locally linearized model equations and matrix $\mathbf{M}$ as seen in Equation 5.27. The correction phase involves the calculation of the Kalman gain $\boldsymbol{K}_{n+1}$, the correction of the state estimates $\hat{\boldsymbol{x}}_{n+1|n+1}$ and the corrected state error covariance matrix $\boldsymbol{P}_{n+1|n+1}$. The correction Equations are seen in Equations 5.28, 5.29 and 5.30 respectively.

Prediction:

$$\hat{\boldsymbol{x}}_{n+1|n} = \boldsymbol{A}_d\hat{\boldsymbol{x}}_{n|n} + \boldsymbol{B}_d\boldsymbol{u}_n \tag{5.26}$$

$$\boldsymbol{P}_{n+1|n} = \boldsymbol{A}_d\boldsymbol{P}_{n|n}\boldsymbol{A}_d^T + \boldsymbol{M} \tag{5.27}$$

Correction:

$$\boldsymbol{K}_{n+1} = \boldsymbol{P}_{n+1|n}\boldsymbol{C}^T(\boldsymbol{C}\boldsymbol{P}_{n+1|n}\boldsymbol{C}^T + \boldsymbol{N})^{-1} \tag{5.28}$$

$$\hat{\boldsymbol{x}}_{n+1|n+1} = \hat{\boldsymbol{x}}_{n+1|n} + \boldsymbol{K}_{n+1}(\boldsymbol{y}_{n+1} - \boldsymbol{C}\hat{\boldsymbol{x}}_{n+1|n}) \tag{5.29}$$

$$\boldsymbol{P}_{n+1|n+1} = (\boldsymbol{I} - \boldsymbol{K}_{n+1}\boldsymbol{C})\boldsymbol{P}_{n+1|n} \tag{5.30}$$

$\boldsymbol{M}$ is the covariance matrix for the states and $\boldsymbol{N}$ is the covariance matrix for the measured outputs. An important property of the KF is, that it will optimally determine a state estimate such that the squared estimation error is minimized [96, p. 237-243]. The algorithm is executed in discrete time, and therefore Equation 5.26 is the discrete time version of 5.25. The discretization can be performed using forward Euler method as illustrated in Equations 5.31.

$$\begin{aligned}
\dot{\boldsymbol{x}} &\approx \frac{\boldsymbol{x}_{n+1} - \boldsymbol{x}_n}{T_s} = \boldsymbol{A}\boldsymbol{x}_n + \boldsymbol{B}\boldsymbol{u}_n \\
\boldsymbol{x}_{n+1} - \boldsymbol{x}_n &= T_s\boldsymbol{A}\boldsymbol{x}_n + T_s\boldsymbol{B}\boldsymbol{u}_n \\
\boldsymbol{x}_{n+1} &= (\boldsymbol{I} + \boldsymbol{T_s}\boldsymbol{A})\boldsymbol{x}_n + T_s\boldsymbol{B}\boldsymbol{u}_n \\
\boldsymbol{x}_{n+1} &= \boldsymbol{A}_d\boldsymbol{x}_n + \boldsymbol{B}_d\boldsymbol{u}_n
\end{aligned} \tag{5.31}$$

As a result $\boldsymbol{A}_d$ and $\boldsymbol{B}_d$ can be obtained from Equations 5.32 and 5.33, respectively.

$$\boldsymbol{A}_d = \boldsymbol{I} + T_s\boldsymbol{A} \tag{5.32}$$

$$\boldsymbol{B}_d = T_s\boldsymbol{B} \tag{5.33}$$

5.4.2 Augmented Kalman Filter

Since the disturbance force $\hat{F}_{ext,z}$ is to be estimated is related to linear translational motion it is assumed that using the matrix stated in Equation 3.55 as prediction model is sufficient. It is hypothesized that only the third equation i.e. the equations describing translational acceleration in the z direction is sufficient. In this context it is assumed that Equation 5.34 is true (**Assumption 20**).

$$\boldsymbol{F}_{ext} = \begin{bmatrix} 0 \\ 0 \\ F_{ext,z} \end{bmatrix} \tag{5.34}$$

Using Equation 3.55 as basis for the prediction model requires slight modifications, the equation is restated for $\ddot{z}$ including the disturbance term $F_{ext,z}$.

$$\ddot{z}(\dot{z}, \hat{F}_{ext,z}, \mathbf{q}, F_z) = \frac{1}{m}\big((q_0^2 - q_1^2 - q_2^2 + q_3^2)F_z + \hat{F}_{ext,z}\big) - g - \frac{1}{m}C_d\dot{z} \tag{5.35}$$

In this context $\mathbf{q}$ and F_z is considered as inputs to the prediction model whereas $\dot{z}$ and $\hat{F}_{ext,z}$ are considered states of the KF. By inspection of Equation 2.6 and 2.19 an expression for F_z can be written as in Equation 5.36.

$$\begin{aligned} F_z &= 0.25f_{p,1} + 0.25f_{p,2} - 0.25f_{p,3} - 0.25f_{p,4} - 0.25f_{p,5} - 0.25f_{p,6} + 0.25f_{p,7} + 0.25f_{p,8} \\ &= 0.25(f_{p,1} + f_{p,2} + f_{p,7} + f_{p,8} - f_{p,3} - f_{p,4} - f_{p,5} - f_{p,6}) \end{aligned} \tag{5.36}$$

which is substituted for F_z. Lastly, $f_{p,i}$ represents the actual motor force which is influenced by the actuator dynamics, where $f_{des,i}$ is the inputs dictated by the controllers. More specifically $f_{p,i}$ must be calculated by the actuator response approximation described in Section 3.1.3 and restated in Equation 5.37.

$$\begin{aligned} \boldsymbol{f}_p(n) = \omega_n^2 * \boldsymbol{f}_{des}(n) * \left(\frac{1}{T^2} + \frac{2\sigma\omega_n^2}{T} + \omega_n^2\right)^{-1} + \left(\frac{2}{T^2} + \frac{2\sigma\omega_n^2}{T}\right) * \boldsymbol{f}_p(n-1) * \left(\frac{1}{T^2} + \frac{2\sigma\omega_n^2}{T} + \omega_n^2\right)^{-1} ... \\ - \frac{1}{T^2} * \boldsymbol{f}_p(n-2) * \left(\frac{1}{T^2} + \frac{2\sigma\omega_n^2}{T} + \omega_n^2\right)^{-1} \end{aligned} \tag{5.37}$$

where $(*)$ implies elementwise matrix multiplication. In conclusion, by substituting Equation 5.37 into 5.36 and lastly into 5.35 the acceleration equation to be used in the filters prediction model is obtained. To keep notation short, the result of the mathematical extensions are summarized in Equation 5.38, which can be observed to be a linear ordinary differential equation.

$$\ddot{z}(\dot{z}, F_{ext,z}, \mathbf{q}, \boldsymbol{f}_{des}, \boldsymbol{f}_{p(n-1)}, \boldsymbol{f}_{p(n-2)}) \tag{5.38}$$

It should be noted that the acceleration becomes a function of $\boldsymbol{f}_p$ instead of F_z. Velocity and position can be obtained from Equation 5.38, however for completeness the trivial equation for velocity is stated in Equation 5.39 and is in state space form equivalent to a free integrator.

$$\dot{z}(\dot{z}) = \dot{z} \tag{5.39}$$

Lastly, the prediction model must be augmented with an additional equation. As the dynamics of the disturbance is unknown a typical choice is denote the disturbance as given in Equation 5.40 [97].

$$\dot{F}_{ext,z}() = 0 \tag{5.40}$$

Next Equations 5.38, 5.39 and 5.40 are concatenated into vector form as seen in Equation 5.41, where states and acceleration are denoted as estimates.

$$\begin{bmatrix} \ddot{\hat{z}} \\ \dot{\hat{z}} \\ \dot{\hat{F}}_{ext,z} \end{bmatrix} = \boldsymbol{A} \begin{bmatrix} \dot{\hat{z}} \\ \hat{z} \\ \hat{F}_{ext,z} \end{bmatrix} + \boldsymbol{B}(\boldsymbol{q}, \boldsymbol{f}_{p(n-1)}, \boldsymbol{f}_{p(n-2)})\boldsymbol{f}_{des} \tag{5.41}$$

here $[\dot{\hat{z}} \ \hat{z} \ \hat{F}_{ext,z}]$ is the state vector, $\boldsymbol{f}_p$ is the input vector and $\boldsymbol{B} \in \mathbb{R}^{3\times 8}$ is the input matrix in which the parameters $\boldsymbol{q}, \boldsymbol{f}_{p(n-1)}$ and $\boldsymbol{f}_{p(n-2)}$ must be updated for each iteration of the filter. Matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are first order jacobian matrices where $\boldsymbol{A}$ is constructed by partial differentiation with respect to each of the filters states for each of the three equations. Likewise

$\boldsymbol{B}$ is obtained by partial differentiation with respect to each input. The procedure for deriving these are identical to the one presented in Section 4.5. The $\boldsymbol{A}$ and $\boldsymbol{C}$ matrices are given in Equation 5.42 where the notation of $\boldsymbol{B}$ is omitted due to its complexity.

$$\boldsymbol{A} = \begin{bmatrix} -\frac{C_d}{m} & 0 & \frac{1}{m} \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad ; \qquad \boldsymbol{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{5.42}$$

The discrete time version of Equation 5.41 is given in Equation 5.43.

$$\begin{bmatrix} \dot{\hat{z}}_{n+1} \\ \hat{z}_{n+1} \\ \hat{F}_{ext,z(n+1)} \end{bmatrix} = \boldsymbol{A}_d \begin{bmatrix} \dot{\hat{z}}_{n} \\ \hat{z}_{n} \\ \hat{F}_{ext,z(n)} \end{bmatrix} + \boldsymbol{B}_d(\boldsymbol{q}, \boldsymbol{f}_{p(n-1)}, \boldsymbol{f}_{p(n-2)})\boldsymbol{f}_{des(n)} \tag{5.43}$$

$\boldsymbol{M}$ and $\boldsymbol{N}$ are chosen as a diagonal positive definite matrices and are given in Equation 5.44. The diagonal elements are the tuning parameters of the filter.

$$\boldsymbol{M} = \begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_3 \end{bmatrix} \quad ; \qquad \boldsymbol{N} = \begin{bmatrix} n_1 & 0 \\ 0 & n_2 \end{bmatrix} \tag{5.44}$$

The filter was manually tuned since there are relatively few tuning parameters compared to e.g. the UKF. Tuning recommendations for the filter is to choose elements in $\boldsymbol{M} >> \boldsymbol{N}$. Also it is recommended to choose $m_2 > m_1 \wedge m_2$.

Further, it was observed that choosing $m_1 < m_3$ results in faster convergence of the estimated disturbance with the drawback of larger overshoot and more chattering on the estimated disturbance. The overshoot and chattering can be attenuated by choosing $m_1 > m_3$, however this results in a slower convergence and more phase lag of the estimated disturbance.

The chosen tuning parameters are summarized in Table 5.3.

Tuning Param.	**Value**
m_1	1.3
m_2	2
m_3	1
n_1	0.0004
n_2	0.001

Table 5.3: Chosen tuning parameters for the AKF.

A simulation is presented in Figure 5.6 where the AKF is to estimate the GE and CE while the FAUAV covers a small distance in the y direction equivalent to the direction of the WTB. The simulation is initialized just before the FAUAV encounters GE/CE. No control or additional disturbances are present in this simulation. The open loop input applied to the system is $\boldsymbol{F}_{gen} = [0\ 0.1\ mg\ 0\ 0\ 0]^T$.

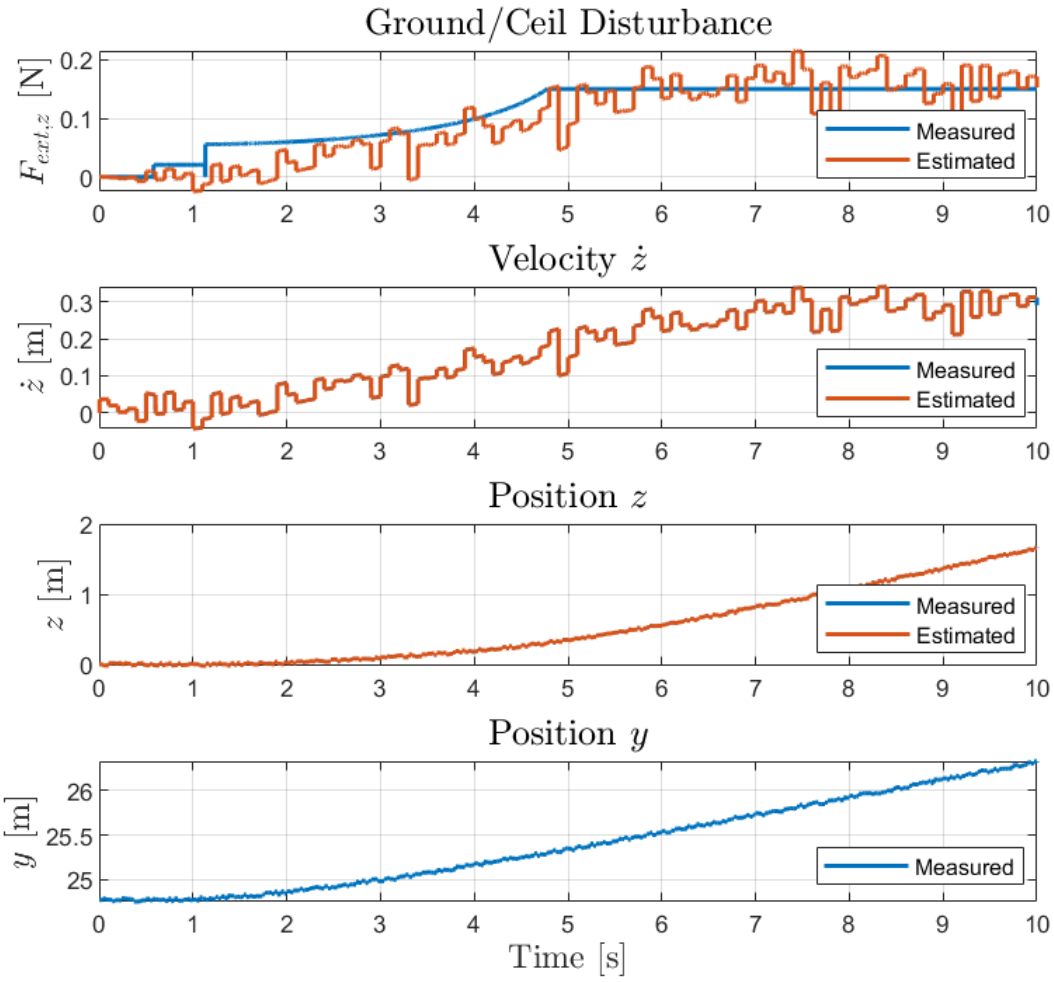


Figure 5.6: Estimated translational disturbance in the form of GE/CE utilizing the AKF.

To test robustness and stability of the filter a longer simulation is performed with an early stage version of the attitude controller presented in Section 6.2. Rotational disturbances are introduced as described in Section 3.5.2 with the variance values in Table 3.3 increased by a factor of 5. The FAUAV is set to hover throughout the simulation while a random disturbance signal is imposed on the system as seen in Figure 5.7.

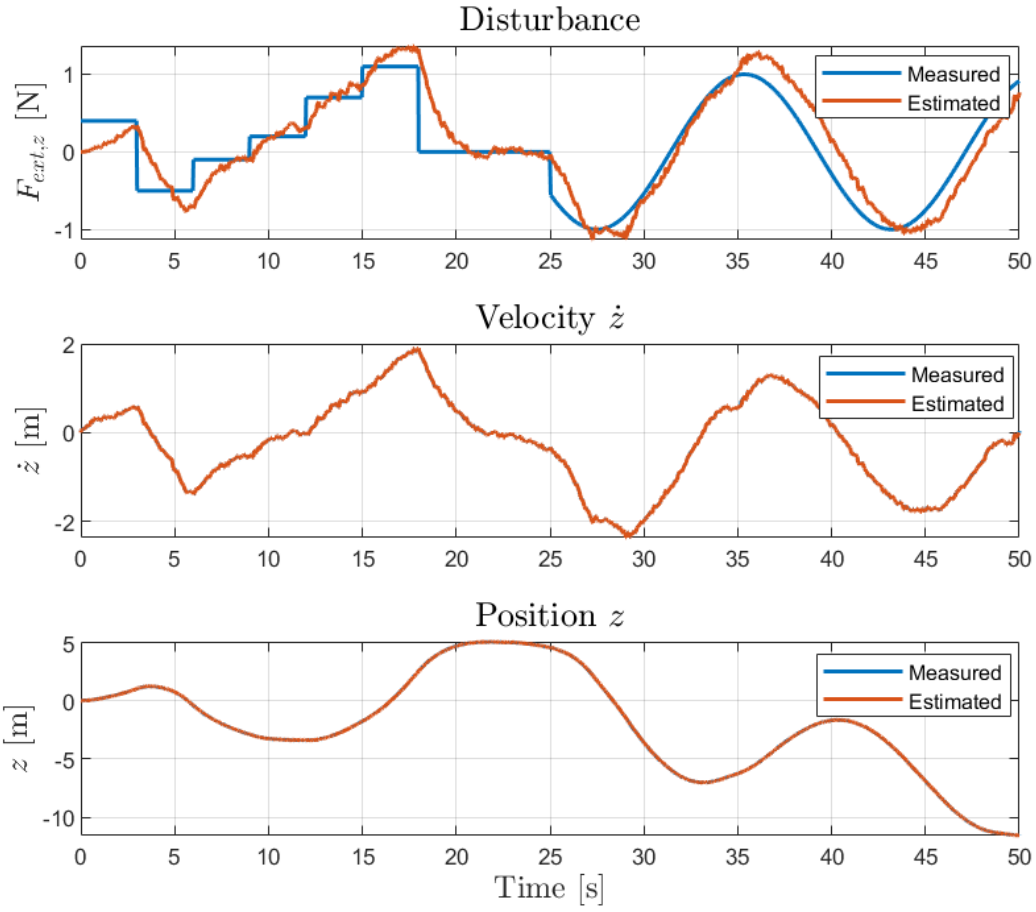


Figure 5.7: Estimated random translational disturbance utilizing the AKF.

It can be observed that the filter is able to estimate the imposed disturbance signal without being influenced by rotational disturbances. However, phase lag, overshoot and the filters response time results in inaccurate tracking when the disturbance is changing. The filter is found useful as a disturbance estimator as it will provide a disturbance estimate in the vicinity of the real disturbance.

6 | MPC and Trajectory Tracking

Abbreviation	Meaning
AKF	Augmented Kalman Filter
MPC	Model Predictive Control
OCP	Optimal Control Problem
QP	Quadratic Problem
RMS	Root Mean Square
SQP	Sequential Quadratic Programming
UAV	Unmanned Aerial Vehicle

Table 6.1: Abbreviations used in this chapter.

6.1 Introduction to MPC

In this section an overview of the basic functionality of MPC is given. This section is partly based on [98, p.1-3] and [99] as well as practical testing of an early stage MPC. Early stage MPC refers to the presented MPC design in Chapter 7 with manually tuned parameters, thus before utilizing optimization for the tuning process. MPC is an optimization based feedback control method applicable to both linear and nonlinear systems. The main applications of MPC is stabilization and trajectory tracking. Stabilization is equivalent to having a constant reference and trajectory tracking which deals with tracking a time varying reference. For a MPC process a control input $u(n)$ is found by the model predictive controller. The control input is a function of the system states $u(n) = f_u(x(n))$, where n indicates a discrete time instant $t_n, n = 0, 1, 2...$ The principle of MPC is to predict and optimize future system behavior by means of a system model. The system models used in this work are functions of states and inputs as seen in Equation 6.1.

$$\boldsymbol{\Theta}_p(k+1) = f(\boldsymbol{\Theta}_p(k), \boldsymbol{u}(k)) \tag{6.1}$$

where $\boldsymbol{\Theta}_p$ denotes predicted states and k refers to a specific discrete time step in the prediction. From Equation 6.1 the next state estimate is obtained from knowledge of current states, current inputs and system dynamics. A finite number of future state predictions is calculated where the number of predictions are dictated by a prediction horizon (N_p). Conclusively, Equation 6.1 can be iterated for $k = 0, ..., N-1$ and this is done for each time instant of n. The first set of states for the zeroth iteration of k is obtained from state measurements at time n as stated in Equation 6.2.

$$\boldsymbol{\Theta}_p(k=0) = \boldsymbol{\Theta}_p(n) \tag{6.2}$$

here n refers to discretized instances of real time. By applying optimal control a series of optimal control inputs is then obtained throughout the prediction horizon $\boldsymbol{u}(0), ..., \boldsymbol{u}(N-1)$ with corresponding series of estimated states. The corresponding discrete time steps then progresses as $t_n, ..., t_{n+N}$ throughout the prediction, where the interval between time steps are denoted $T_{s,MPC}$. The prediction horizon N_p and sampling time $T_{s,MPC}$ are tuning parameters of the controller. Additional tuning parameters are involved in the prediction such as solver method utilized to obtain $\boldsymbol{\Theta}_p(k+1)$. Further the plant model can be considered a tuning parameter. A cost function ℓ is defined for the optimization, typically such cost function is a function of predicted states, and input i.e. $\ell(\boldsymbol{\Theta}_p(k), u(k))$. The cost function is then accumulated for each prediction iteration given as $k = 0, ..., N-1$. An example of a commonly used cost function is given in 6.3 [98, p.2].

$$\ell(\boldsymbol{\Theta}_{err}(k), \mathbf{u}(k)) = \boldsymbol{\Theta}_{err}(k)^T \mathbf{Q}_{err} \boldsymbol{\Theta}_{err}(k) + \mathbf{u}(k)^T \mathbf{R}_u \mathbf{u}(k) \tag{6.3}$$

where $\boldsymbol{\Theta}_{err}(k) = \boldsymbol{\Theta}_p(k) - \boldsymbol{\Theta}_{ref}(k)$ and $\mathbf{Q}_{err}$ and $\mathbf{R}_u$ are diagonal weighting matrices for state errors and inputs respectively. As a result the accumulated cost function $J(\mathbf{x}_p(k), \mathbf{x}_{ref}(k), \mathbf{x}(n), \mathbf{u}(k))$ takes the form of Equation 6.4.

$$\min_{\mathbf{u}} \; J(\boldsymbol{\Theta}_p(k), \boldsymbol{\Theta}_{ref}(k), \boldsymbol{\Theta}(n), \mathbf{u}(k)) = \sum_{k=0}^{N-1} \ell(\mathbf{x}_p(k), \mathbf{u}(k)) \tag{6.4}$$

where $\min_{\mathbf{u}}$ implies that a minimization is to be performed on the function $J(\boldsymbol{\Theta}_p(k), \boldsymbol{\Theta}_{ref}(k), \boldsymbol{\Theta}(n), \mathbf{u}(k))$ with $\mathbf{u}$ being the manipulated variable. This is commonly referred to as an optimal control problem (OCP). The solution of the OCP can be written as in Equation 6.5.

$$\begin{aligned} \min_{\mathbf{u}(0),...,\mathbf{u}(N-1)} \; J(\boldsymbol{\Theta}_p(k), \boldsymbol{\Theta}_{ref}(k), \boldsymbol{\Theta}(n), \mathbf{u}(k)) &= \sum_{k=0}^{N-1} \ell(\boldsymbol{\Theta}^*_{err}(k), \mathbf{u}^*(k)) \\ &= \sum_{k=0}^{N-1} \ell(\boldsymbol{\Theta}^*_p(k), \boldsymbol{\Theta}_{ref}(k), \mathbf{u}^*(k)) \end{aligned} \tag{6.5}$$

In the solution, $\min_{\mathbf{u}(0),...,\mathbf{u}(N-1)}$, implies that an optimization has been performed for each input in the sequence. The optimal control sequence contains the optimized input values defined as $\mathbf{u}^*(k) \in \mathbb{R}^{N_u \times N_p}$, where N_u is number of inputs. The prediction error is defined as $\boldsymbol{\Theta}^*_{err}(k) \in \mathbb{R}^{N_x \times N_p}$ and contains the optimal errors computed from $\boldsymbol{\Theta}^*_{err}(k) = \boldsymbol{\Theta}_{ref}(k) - \boldsymbol{\Theta}^*_p(k)$. The optimal prediction trajectory is defined as $\boldsymbol{\Theta}^*_p(k) \in \mathbb{R}^{N_x \times N_p}$ and contains the systems behavior when applied with the optimal control sequence $\mathbf{u}^*(k)$. In Figure 6.1 an early stage simulation of the MPC controller is shown and further developed in Section 6.3. In the simulation the optimization algorithm is executed once and only global x position is displayed for simplicity. The optimal control sequence in the simulation is $u^*_x(t)$ and is equivalent to the desired body force in the x direction.

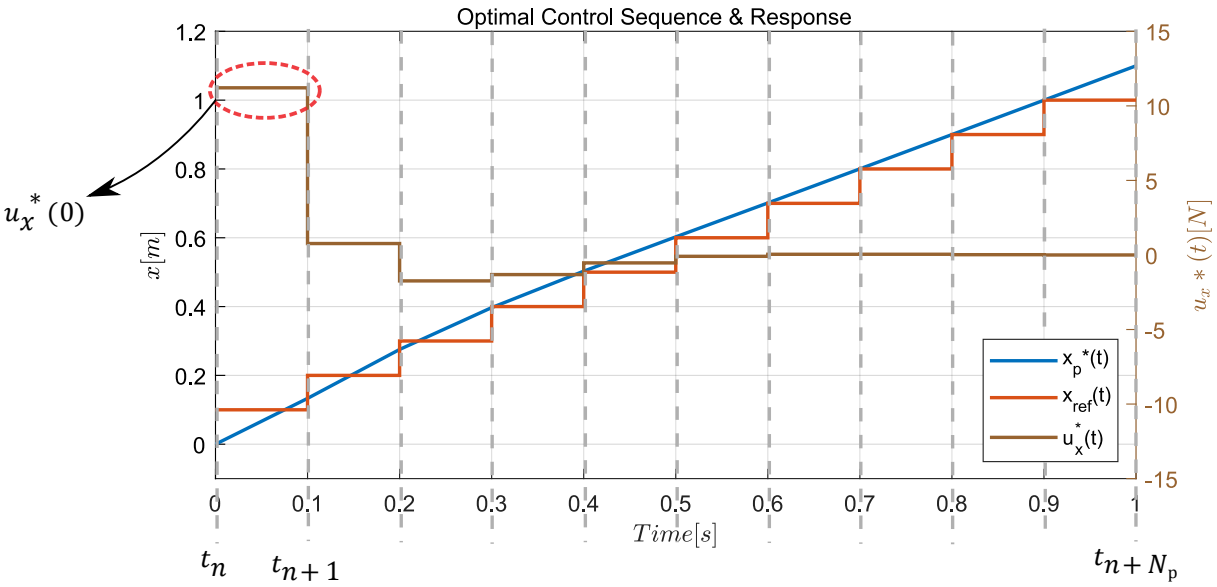


Figure 6.1: Single execution of the optimization algorithm. The tuning parameters used are: $N_p = 10, \quad N_c = 10, \quad T_{s,MPC} = 0.1, \; Q = diag([0,0,0,10,10,100]), \; R = diag([1,1,1])$.

It is to be noted, that usually only the first element of the optimal control sequence is applied to the system. The reason for this is that the further the MPC predicts into the future the less reliable the predictions become due to modelling inaccuracies.

In the example seen in Figure 6.1 the first element of the optimal control sequence corresponds to $u_x^*(0)$. Then, at time t_{n+1} the procedure is repeated with the next prediction horizon spanning the time interval $[t_{n+1}; t_{n+N+1}]$. In applications where a faster execution of the optimization algorithm is desired it can be beneficial to introduce a control horizon N_c. The control horizon dictates how many steps an optimization is to be performed within the prediction horizon. The control horizon must therefore fulfill $N_c < N_p$. In Figure 6.2 another simulation of the early stage MPC controller is shown where a prediction horizon of $N_c = 3$ is chosen.

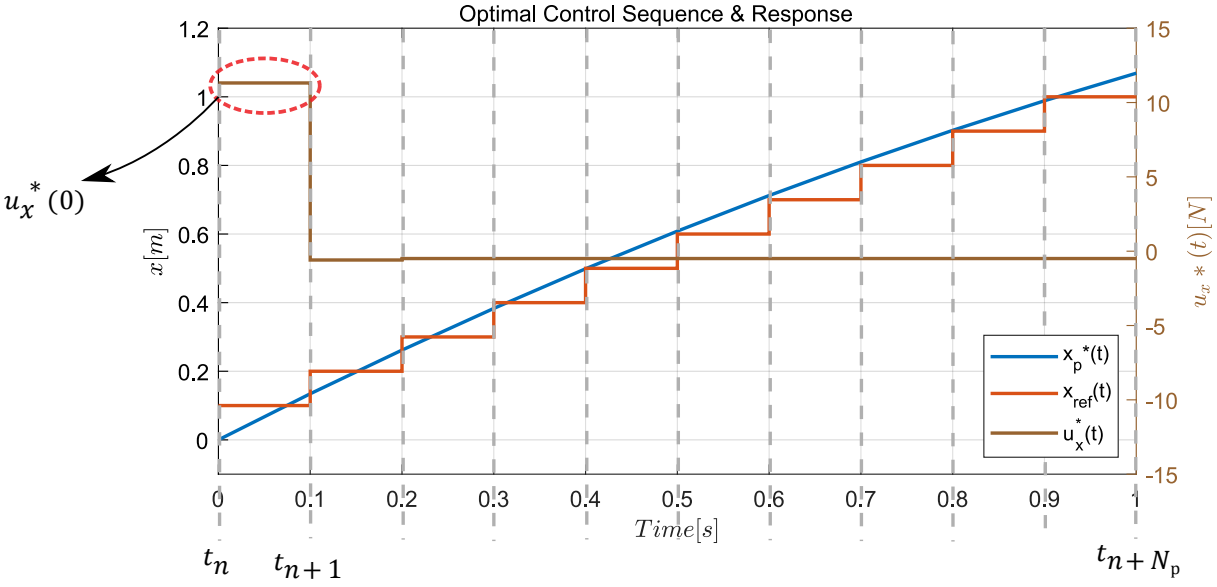


Figure 6.2: Single Execution of the Optimization Algorithm. The tuning parameters used are: $N_p = 10, \quad N_c = 3, \quad T_{s,MPC} = 0.1, \; Q = diag([0,0,0,10,10,100]), \; R = diag([1,1,1])$.

Furthermore, by including a control horizon is, that the significant part of the response is often contained within the first few samples [99]. By comparing Figure 6.1 and 6.2 it can be observed that the first element in the optimal control sequence ($u_x^*(0)$) is almost identical. In Table 6.2 a comparison of execution time and root mean square (RMS) error between predicted state and reference is shown. The values provided in the table are averages from 5 executions of the early stage position MPC.

	Average time $[s]$	**RMS Error** $[m]$	**RMS Error** $[m]$ **First 3 iterations**
Fig 6.1 $N_c = N_p = 10$	0.89	0.022	0.041
Fig 6.2 $N_c = 3$	0.60	0.026	0.05

Table 6.2

Introducing the control horizon becomes a trade off between computation time and accuracy. The MPC containing the control horizon is seen to decrease the computational time with approximately $\frac{1}{3}$, but results in lower accuracy when it tracks the defined reference. The calculation time and tracking error also dependent on the ratio between the prediction and the control horizon and the number of predictions for each optimization.

6.2 Attitude MPC

In this section the attitude controller is described. This involves defining the prediction model, optimization process and cost function. The purpose of the attitude controller is to achieve stabilization, i.e. tracking a constant reference and thus maintaining a fixed orientation.

6.2.1 Prediction Model

The prediction model is based on a subsystem of the full model described in Equation 3.59. The subsystem prediction model is based on Equation 3.57 and 3.58. The rotor response approximation given in Equation 3.8 is used in the prediction model in almost the same manner it was used in the AKF in Section 5.4.2. The rotor response approximation differs from the one used in the AKF in the sense, that the response is assigned to the desired moments. This is an assumption and does not reflect the true nature of the response, however it is found reasonable because the allocation happens instantaneously. Therefore it should not matter whether the response is introduced on the desired forces and moments or on the desired propeller forces. The adjusted response is given in Equation 6.6 (**Assumption 21**).

$$\begin{aligned}\boldsymbol{u}_{att}(k) =&\omega_n^2 * \boldsymbol{u}_{att,des}(k) * \left(\frac{1}{T^2} + \frac{2\sigma\omega_n^2}{T} + \omega_n^2\right)^{-1} \\ &+ \left(\frac{2}{T^2} + \frac{2\sigma\omega_n^2}{T}\right) * \boldsymbol{u}_{att}(k-1) * \left(\frac{1}{T^2} + \frac{2\sigma\omega_n^2}{T} + \omega_n^2\right)^{-1} \\ &- \frac{1}{T^2} * \boldsymbol{u}_{att}(k-2) * \left(\frac{1}{T^2} + \frac{2\sigma\omega_n^2}{T} + \omega_n^2\right)^{-1}\end{aligned} \tag{6.6}$$

In this context $\boldsymbol{u}_{att} = \boldsymbol{\tau}$ and $\boldsymbol{u}_{att,des} = \boldsymbol{\tau}_{des}$.

The discretized versions of Equation 3.57 and 3.58 are used in the controller and given in Equation 6.7

$$\begin{bmatrix} \boldsymbol{\omega}_{k+1} \\ \boldsymbol{q}_{k+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\omega}_k \\ \boldsymbol{q}_k \end{bmatrix} + \begin{bmatrix} \boldsymbol{J}^{-1}\big(\boldsymbol{u}_{att(k)} - (\boldsymbol{\omega}_k \times \boldsymbol{J}\boldsymbol{\omega}_k)\big) \\ \frac{1}{2}\boldsymbol{W}_k(\boldsymbol{\omega}_k)\boldsymbol{q}_k \end{bmatrix} T_{s,pos} \tag{6.7}$$

Where the prediction model state vector is denoted $\boldsymbol{\Theta}_{att} = [\boldsymbol{\omega}\ \ \boldsymbol{q}]^T$

6.2.2 Cost Function

For the MPC to choose the next optimal input, a cost function has to be defined for the optimization. The chosen cost function is presented in Equation 6.8.

$$\begin{aligned} \ell_{att}(\boldsymbol{\Theta}_{att,err}(k), \boldsymbol{u}_{att}(k), d\boldsymbol{u}_{att}(k)) = \Big(& \boldsymbol{\Theta}_{att,err}(k)^T \boldsymbol{Q}_{att} \boldsymbol{\Theta}_{att,err}(k) + \boldsymbol{u}_{att}(k)^T \boldsymbol{R} \boldsymbol{u}_{att}(k) \\ & + d\boldsymbol{u}_{att}(k)^t \boldsymbol{R}_{d,att} d\boldsymbol{u}_{att}(k) \Big) \end{aligned} \tag{6.8}$$

And the accumulated cost function is then given in Equation 6.9

$$J_{att}(\boldsymbol{\Theta}_{att,err}(k), \boldsymbol{u}_{att}(k), d\boldsymbol{u}_{att}(k)) = \sum_{k=0}^{N_{p,att}-1} \ell_{att}(\boldsymbol{\Theta}_{att,err}(k), \boldsymbol{u}_{att}(k), d\boldsymbol{u}_{att}(k)) \tag{6.9}$$

The cost function of the attitude MPC is a function of the state errors $\boldsymbol{\Theta}_{att,err} \in \mathbb{R}^{7\times1}$, the inputs $\boldsymbol{u}_{att} \in \mathbb{R}^{3\times1}$ and the input rates $d\boldsymbol{u}_{att} \in \mathbb{R}^{3\times1}$. The cost function stated in Equation 6.9 are a redefinition of the one presented in Equation 6.3. Here the weighting matrices for states and inputs are defined as $\boldsymbol{Q}_{att} \in \mathbb{R}^{7\times7}$ and $\boldsymbol{R}_{att} \in \mathbb{R}^{3\times3}$ respectively. Additionally, the input rates are considered with corresponding penalties contained in $\boldsymbol{R}_{d,att} \in \mathbb{R}^{3\times3}$. For the state errors the references are defined as $\boldsymbol{q}_{ref} = [1\ 0\ 0\ 0]^T$ and $\boldsymbol{\omega} = [0\ 0\ 0]^T$ thus the cost function increases as the UAV deviates from the original rotational position. Additionally the inputs are penalized to keep the operation cost low whereas the input rates are penalized to ensure a more smooth flight as the cost function increases if the current inputs deviate from the previous inputs.

6.2.3 Optimization Process

To initialize the optimization algorithm an initial guess must be supplied. The initial guess has influence on how fast and accurate the optimization algorithm converges. A proper guess results in few iterations of the optimization algorithm before convergence or very accurate results if the algorithm is performed for a specific number of calculations. A bad guess is either slow, inaccurate or might even diverge. For the MPC, the optimization could either utilize cold or warm start. The cold start simply means the initial guess is repeated for each iteration and the optimization algorithm has to search for the optimum from scratch over and over. The warm start remembers the solution from the previous iteration and utilizes this as the initial guess when the optimization algorithm is calculated again. The benefits of the 2 methods depends on the application. If the new optimum is close to the previous the warm start would be beneficial

as the algorithms search area can be reduced by utilizing the knowledge of the previous point to find the next solution. On the other hand, if the optimum frequently move at a great distance, the warm start would not be beneficial as the previous solution is placed far from the new optimum and the algorithm have to search in a large area. In this case the cold start might result in more accurate results. The 2 different scenarios of the warm start are shown in Figure 6.3 together with a cold start process, all presented as a 2D example.

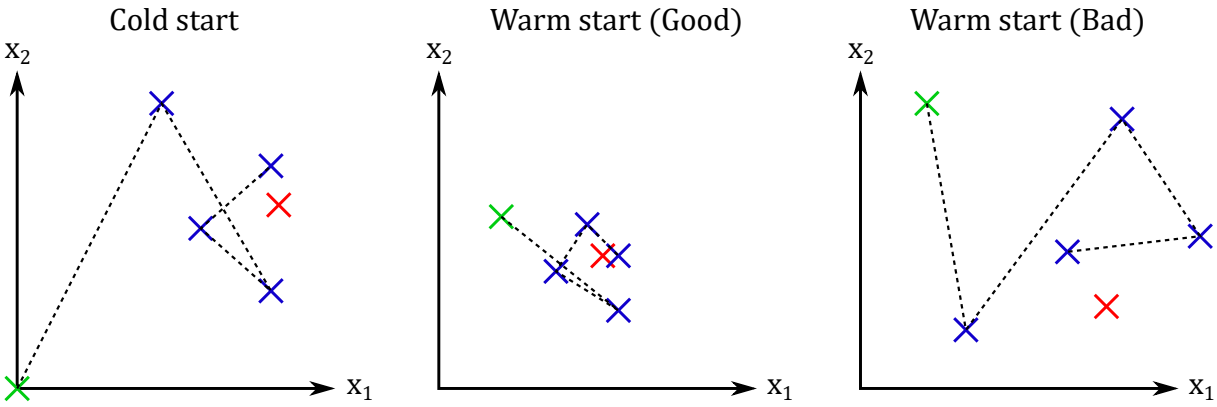


Figure 6.3: 2D example of the optimization with cold start to the left, warm start with closely related optimums in the middle and warm start with strong fluctuating optimums to the right.

The 3 different cases of initial guesses in Figure 6.3 are presented with the initial guess as the green cross, the current optimum as the red cross and new guesses from the algorithm as the blue crosses. The examples show an optimization algorithm with a fixed number of calculations stated as 5 where the dotted line show the direction of the new points. To the left, the cold start has to sweep points at a large area to move closer to the optimum. At the center warm start is utilized with an optimum closely related to the previous solution and the algorithm is seen to find a very suitable solution. On the other hand the warm start utilized to the right do not converge toward a very satisfying solution as the optimum is placed further from the previous solution than the zero point whereas cold start could be preferable.

For the attitude MPC the inputs to be chosen are assumed to be closely related between the optimization iterations due to the continuous nature of the physical system, and therefore warm start is used to presumably improve the optimization (**Assumption 22**).

It is common practise to state a compact form of the OCP as in Equation 6.10 below.

$$\begin{aligned} \min_{\mathbf{u}} \quad & J(\boldsymbol{\Theta}_{att,err}(k), \boldsymbol{u}_{att}(k), d\boldsymbol{u}_{att}(k)) \\ \text{subject to} \quad & \textit{Equation } 6.7 \\ & \boldsymbol{u}_{att}(k) \in \mathbb{U} \\ & \boldsymbol{\Theta}_{att}(k=0) = \boldsymbol{\Theta}_{att}(n) \end{aligned} \tag{6.10}$$

The set $\mathbb{U}$ contains all feasible input values. These values are constrained to avoid actuator saturation thus an upper and lower bound is imposed on each element in the input vector $\boldsymbol{u}_{att}$ as shown in Equation 6.11.

$$l_{b,i} \geq u_{att,i} \geq u_{b,i} \qquad \text{where } i = 1, 2, 3 \tag{6.11}$$

A pseudo code of how the optimization process is structured in MATLAB is given in Algorithm 1.

Algorithm 1 NMPC Pseudo Code

Load references ($\boldsymbol{\Theta}_{ref,k}$) current measured plant outputs ($\boldsymbol{\Theta}_k$) and other a priori knowledge.
Assign values for tuning parameters $\boldsymbol{Q}, \boldsymbol{R}, \boldsymbol{R}_d, N_p, N_c, T_s$
Assign input constraints $\boldsymbol{l}_b$ and $\boldsymbol{u}_b$
Provide Initial input guess $\boldsymbol{u}_{(n)}$
Import or calculate future reference points
A Target function, with a function handle ($@(\boldsymbol{u})$) is defined and executed:
- $TargetFcn = @(\boldsymbol{u})CostFcn(\boldsymbol{u}, \boldsymbol{Q}, \boldsymbol{R}, \boldsymbol{R}_d, \boldsymbol{u}_{(n)}, \boldsymbol{\Theta}_k, \boldsymbol{\Theta}_{ref,k}, N_p, N_c, T_s)$;
- The target function specifies $\boldsymbol{u}$ to be the variable of manipulation

The cost function is defined and executed:
- This triggeres the prediction loop to start

$\boldsymbol{function}$ $J = CostFcn(\boldsymbol{u}, \boldsymbol{Q}, \boldsymbol{R}, \boldsymbol{R}_d, \boldsymbol{u}_{(n)}, \boldsymbol{\Theta}_k, \boldsymbol{\Theta}_{ref,k}, N_p, N_c, T_s)$;
$\boldsymbol{\Theta}_k = \boldsymbol{\Theta}_{(n)}$;
$\boldsymbol{u}_{array} = reshape(\boldsymbol{u}_{(n)}, N_u, N_c)$;
$J = 0$;
$\boldsymbol{for}$ $i = 0 : N_p - 1$
$\boldsymbol{if}$ $\leq N_c - 1$
$\boldsymbol{u}_k = \boldsymbol{u}_{array}(:, i)$;
$\boldsymbol{else}$
$\boldsymbol{u}_k = \boldsymbol{u}_{array}(:, N_c)$;
$\boldsymbol{end}$
$\boldsymbol{\Theta}_{k+1} = SystemFcn(\boldsymbol{\Theta}_k, \boldsymbol{u}_k, T_s)$;
$J = J + \boldsymbol{\Theta}_{err}^T \boldsymbol{Q} \boldsymbol{\Theta}_{err} + \boldsymbol{u}_k^T \boldsymbol{R} \boldsymbol{u}_k + d\boldsymbol{u}_k^T \boldsymbol{R}_d d\boldsymbol{u}_k$
$\boldsymbol{end}$

- Where $SystemFcn$ contains discretized dynamic equations
- Note, both $CostFcn$ and $SystemFcn$ are contained in $TargetFcn$

Perform optimization using $fmincon$:
$\boldsymbol{u}_{ctrl}(n+1) = fmincon(TargetFcn, \boldsymbol{u}_{(n)}, [], [], [], [], \boldsymbol{l}_b, \boldsymbol{u}_b)$;
- Note that $\boldsymbol{u}_{ctrl}(n+1) \in \mathbb{R}^{(N_p \times N_u)}$

Repeat

This code is executed in MATLAB, the code requires usage of the '*optimization toolbox*' from which the optimization function '*fmincon*' is utilized. Wrapping a function-handle around the cost functions allows '*fmincon*' to continuously optimize the cost function by manipulating any given variables. In this case $\boldsymbol{u}$ is the manipulated variable.

In Appendix G.1 MATLAB code is provided for the attitude MPC. By converting the MATLAB based MPC controller into C code, the calculation time of the optimization is significantly reduced. The optimization time was found to be in the interval $150 - 300\ [ms]$ in MATLAB code, where the generated C code decrease optimization time with approximately a factor of 0.01. In Appendix G.1.1 MATLAB code is provided for converting the underlying MPC functions into C code for faster optimization.

In section 6.2.4 we address the optimization method used for the OCP. Controller tuning and results are presented in Chapter 7.

6.2.4 Optimization Method

In this section a short description of the optimization method utilized within the *fmincon* function in MATLAB is described. The method utilized is the sequential quadratic programming (SQP) which is a constraint optimization method based on polynomial approximation.

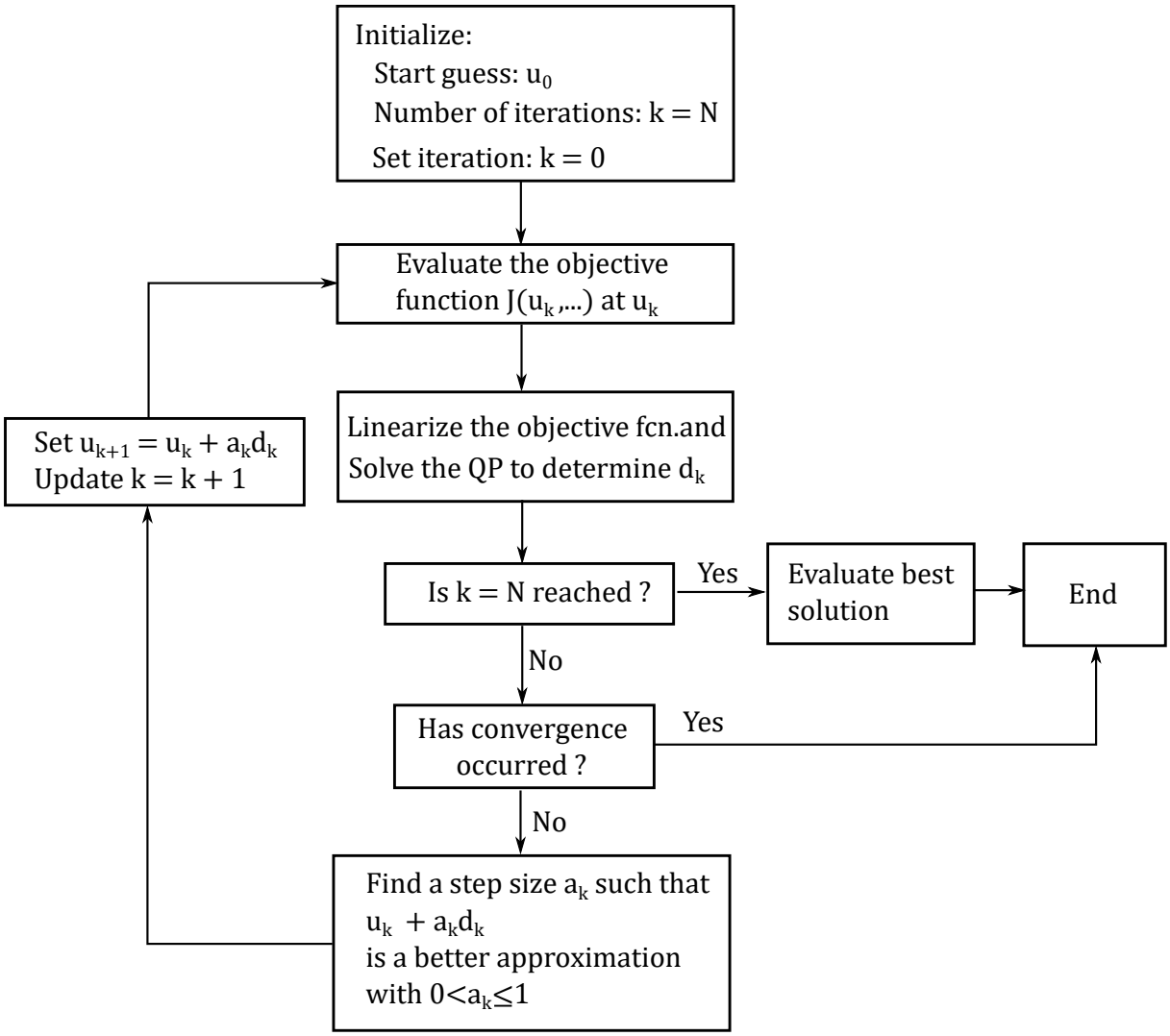


Figure 6.4: Flowchart of the SQP algorithm.

Figure 6.4 illustrates the concept of the SQP algorithm. Firstly, the algorithm is initialized with a start guess, number of iterations and the iteration counter is set to 0. The next step is to evaluate the nonlinear objective function in the point u_k. Thereafter, the function is linearized and the Hessian is determined such that the quadratic problem (QP) can be solved for the gradient d_k. If either the criteria for maximum number of iterations or for convergence are reached, the best solution is chosen and the algorithm terminates. Otherwise, a step size a_k is found by e.g. linear search method and a new guess is calculated along with updating the iteration counter and the loop continuous until the number of defined iterations or convergence is reached.

6.3 Position Tracking MPC

In this section the position tracking controller is described. The purpose of this controller is to track the time varying reference described in Chapter 2 Section 2.2.

6.3.1 Prediction Model

The prediction model used for the position tracking MPC is the discrete version of Equation 3.55. Again the rotor response approximation given in Equation 3.8 is used to provide a realistic response of the input $\boldsymbol{u}_{pos}$ as shown in Equation 6.12.

$$\begin{aligned}\boldsymbol{u}_{pos}(k) =&\omega_n^2 * \boldsymbol{u}_{pos,des}(k) * \left(\frac{1}{T^2} + \frac{2\sigma\omega_n^2}{T} + \omega_n^2\right)^{-1} \\ &+ \left(\frac{2}{T^2} + \frac{2\sigma\omega_n^2}{T}\right) * \boldsymbol{u}_{pos}(k-1) * \left(\frac{1}{T^2} + \frac{2\sigma\omega_n^2}{T} + \omega_n^2\right)^{-1} \\ &- \frac{1}{T^2} * \boldsymbol{u}_{pos}(k-2) * \left(\frac{1}{T^2} + \frac{2\sigma\omega_n^2}{T} + \omega_n^2\right)^{-1}\end{aligned} \tag{6.12}$$

Where $\boldsymbol{u}_{pos} = \boldsymbol{F}$ and $\boldsymbol{u}_{pos,des} = \boldsymbol{F}_{des}$. The discretized version of Equation 3.55 is given in Equation 6.13.

$$\begin{bmatrix}\dot{\boldsymbol{p}}_{k+1} \\ \boldsymbol{p}_{k+1}\end{bmatrix} = \begin{bmatrix}\dot{\boldsymbol{p}}_k \\ \boldsymbol{p}_k\end{bmatrix} + \begin{bmatrix}\ddot{\boldsymbol{p}}_k(\mathbf{q}_k, \dot{\mathbf{p}}_k, \mathbf{u}_{pos,(k)}, \hat{\boldsymbol{F}}_{ext,(k)}) \\ \dot{\boldsymbol{p}}_k\end{bmatrix} T_{s,pos} \tag{6.13}$$

Where $\ddot{\boldsymbol{p}}_k(\mathbf{q}_k, \dot{\mathbf{p}}_k, \mathbf{u}_{pos,(k)}, \hat{\boldsymbol{F}}_{ext,(k)})$ is given in discrete form in Equation 6.14.

$$\ddot{\mathbf{p}}_k(\mathbf{q}_k, \dot{\mathbf{p}}_k, \mathbf{u}_{\mathbf{pos},(\mathbf{k})}, \hat{\boldsymbol{F}}_{ext,(k)}) = \frac{1}{m}\Big(\mathbf{Q}(\boldsymbol{q}_k)\, \mathbf{u}_{pos,(k)} + \hat{\boldsymbol{F}}_{ext,(k)}\Big) - g\, \mathbf{e} - \frac{1}{m} C_d \dot{\mathbf{p}}_k \tag{6.14}$$

The state vector is denoted $\boldsymbol{\Theta}_{pos} = [\dot{\boldsymbol{p}}\ \boldsymbol{p}]^T$.

6.3.2 Cost Function

The cost function used for this controller is equivalent to the attitude MPC cost function. The accumulated cost function is given in Equation 6.15.

$$J_{pos}(\boldsymbol{\Theta}_{pos,err}(k), \boldsymbol{u}_{pos}(k), d\boldsymbol{u}_{pos}(k)) = \sum_{k=0}^{N_{p,pos}-1} \ell_{pos}(\boldsymbol{\Theta}_{pos,err}(k), \boldsymbol{u}_{pos}(k), d\boldsymbol{u}_{pos}(k)) \tag{6.15}$$

Where $\boldsymbol{\Theta}_{pos,err} \in \mathbb{R}^{6\times 1}$, $\boldsymbol{u}_{pos} \in \mathbb{R}^{3\times 1}$ and $d\boldsymbol{u}_{pos} \in \mathbb{R}^{3\times 1}$.

6.3.3 Optimization Process

The optimization process uses warm start and the pseudo code presented in Algorithm 1 applies to this controller as well. The optimization method is equivalent to the one described in Section 6.2.4.

The OCP for the position tracking MPC is stated in compact form in Equation 6.16.

$$\begin{aligned} \min_{\mathbf{u}} \quad & J_{pos}(\boldsymbol{\Theta}_{pos,err}(k), \boldsymbol{u}_{pos}(k), d\boldsymbol{u}_{pos}(k)) \\ \text{subject to} \quad & \textit{Equation } 6.13 \\ & \boldsymbol{u}_{pos}(k) \in \mathbb{U} \\ & \boldsymbol{\Theta}_{pos}(k=0) = \boldsymbol{\Theta}_{pos}(n) \\ & \hat{\boldsymbol{F}}_{ext}(k=0) = \hat{\boldsymbol{F}}_{ext}(n) \\ & \hat{\boldsymbol{F}}_{ext}(k) = \hat{\boldsymbol{F}}_{ext}(k+1) \end{aligned} \tag{6.16}$$

The set $\mathbb{U}$ contains all feasible input values. The set is defined by imposing constraints in the same manner as for the attitude MPC as seen in Equation 6.11. This optimization process differs from the attitude MPC as the estimated disturbance is used in the prediction model. In the OCP it is implied that the estimated disturbance at time instant n is assumed to be constant for the duration of the prediction. Controller tuning is addressed in Chapter 7 Section 7.3.

7 | Bayesian Optimization and Controller Tuning

This chapter presents the tuning methods used for determining the control parameters of the MPCs presented in Chapter 6. The different considerations concerning the tuning parameters are discussed and results are then presented for the final attitude and position tracking MPC. Finally, simulation results are presented for flight of the UAV within the WTB under different scenarios.

Abbreviation	**Meaning**
AF	Acquisition Function
BO	Bayesian Optimization
EI	Expected Improvement
EIP	Expected Improvement Plus
GP	Gaussian Process
GPR	Gaussian Process Regression
MPC	Model Predictive Control
SSE	Sum of Squared Erroes
UAV	Unmanned Aerial Vehicle
WTB	Wind Turbine Blade

Table 7.1: Abbreviations used in this chapter.

The usage of MPC controllers has a strong advantage when following a trajectory efficiently by solving a cost function as a trade off between performance and operational cost. On the other hand the MPCs contain numerous controller parameters to be tuned and the tuning process is often not intuitive, resulting in a time consuming and inaccurate iterative procedure. It is ideal for the tuning process to utilize algorithms to determine the control parameters that minimizes or maximizes a predefined cost. One approach is to obtain and analyse data by conducting experiments or running simulations. Conducting experiments with a UAV for controller tuning could damage the UAV as an inadequate controller can cause the UAV to crash. Running simulations is less accurate due to the lack of unmodelled dynamics but safer, cheaper and not as time consuming. Though for controllers with many parameters the number of simulations required could be extensive and thus still result in a time consuming process. Another approach for tuning the controller parameters is to utilize hyperparameter tuning methods. Hyperparameter tuning is when the optimization problem is solved by adjusting hyperparameters seen as inputs for a surrogate model. A surrogate model is a statistical approximation of the real system constructed only from input and output data. The surrogate model can be used as a fast approach to predict the output from a given set of control coefficients.

3 different search algorithm is usually utilized for choosing suitable controller coefficients by hyperparameter tuning. These algorithms are the grid search, the random search and Bayesian optimization all illustrated by a 2D example in Figure 7.1.

At the illustration of the search algorithms seen in Figure 7.1 the green circles is the optimal point of the tuning parameters. The blue and red areas illustrate the influence on the output for the given parameters x_1 and x_2 respectively and the red crosses are samples. The grid search demonstrated at the left of Figure 7.1 is a simple structured search algorithm where the samples are placed in the full grid with equal distance between one another. The downside to this method is the symmetry as the 9 samples only covers 3 values of x_1 and 3 values of x_2. To overcome the symmetry, random search can be utilized where the 9 samples are placed random in the full grid. From this, up to 9 different values of x_1 and x_2 can be tested. The 2 methods are easy to implement but do only provide a vague idea of the distribution of states compared to determining the optimum. A more complex algorithm is the Bayesian optimization (BO) which uses a compromise of exploitation and exploration. This is described further in section 7.1 but from the right of Figure 7.1 it can be seen, that the algorithm places some points in the outer regions of the grid but also concentrate samples around the optimum.

For tuning the attitude and position tracking MPCs a Bayesian optimization approach is utilized due to the fast approach of using hyperparameter tuning and the intelligent prediction of the samples. Section 7.1 covers the Bayesian optimization principles, theory and implementation whereas Section 7.2 and 7.3 presents the considerations, tuning process and results of the attitude and position tracking MPCs respectively.

7.1 Bayesian Optimization

BO is a black-box optimization method where a probabilistic surrogate model is established as a statistical approximation of the system as a set of tuning parameters are applied as inputs and corresponding cost function values are supplied as outputs. The surrogate models utilized for the tuning of the MPCs are Gaussian process regressions (GPR) from which the BO utilizes the model and adjusts the hyperparameters to minimize the cost function and retrieves the corresponding optimal controller parameters. A key advantage of using BO is the reduction of the time needed to tune the controllers. Even by usage of simulation-based tuning, an iterative parameter determination is costly as a simulation has to be run for each change of

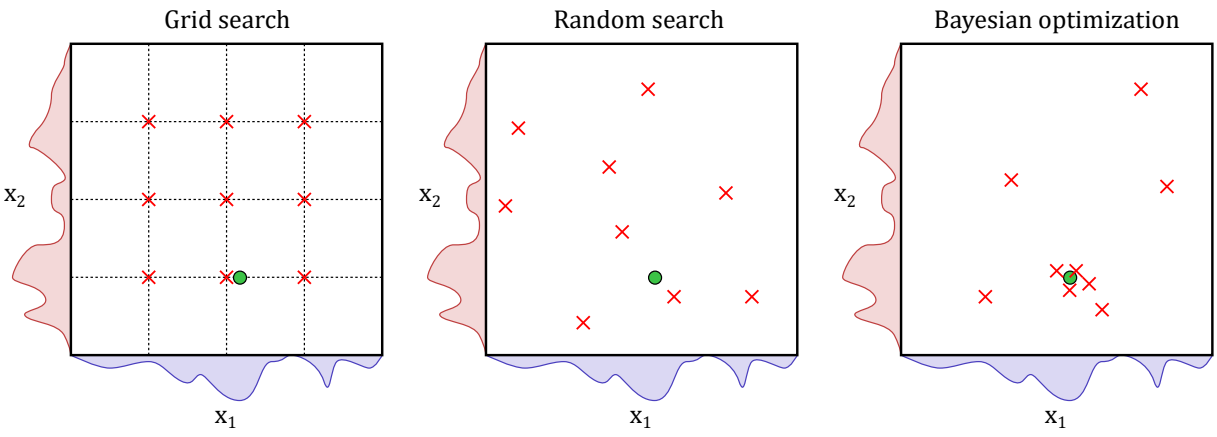


Figure 7.1: Different methods for hyperparameter tuning.

the parameters. By approximating the simulation model with a GPR the computational cost is significantly reduced. Another key advantage of using BO is the choice of the next parameter change. When tuning controllers, especially with an extensive number of tuning parameters, the change of tuning parameters do not always have an intuitive effect on the response especially the relation between the parameters for controllers with multiple coefficients. Methods like grid search and random search are simple to incorporate for the tuning process, but it is not guaranteed that the solution is in the near vicinity of the optimum. The BO uses an acquisition function to evaluate prior sets of parameters and corresponding cost function values to obtain the next guess of parameters by maximizing the probability of the next parameters reaching an optimum. For BO to operate, an acquisition function must be chosen. Further the Gaussian process (GP) together with the hyperparameters and corresponding cost function have to be established. These terms are explained in the following.

GP is described through a mean function $\mu(\boldsymbol{x})$ and a covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$ giving $f(\boldsymbol{x}) \sim GP(\mu(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}'))$. Here $\boldsymbol{x}$ and $\boldsymbol{x}'$ are 2 different sets of parameters seen as inputs for the GP, the mean is the expected function value $E[f(\boldsymbol{x})]$ and the covariance function is the expected covariance between set $\boldsymbol{x}$ and $\boldsymbol{x}'$ given by $E[(f(\boldsymbol{x}) - \mu(\boldsymbol{x}))(f(\boldsymbol{x}') - \mu(\boldsymbol{x}'))]$.

Different covariance functions have already been developed for GPs. The covariance function is utilized to determine similarity between sets of $\boldsymbol{x}$ and corresponding y. The similarity is based on the assumption, that points in the same vicinity of $\boldsymbol{x}$ would likely result in a similar y as the actual point $\boldsymbol{x}$. The covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$ is called a kernel which must satisfy the condition of kernel symmetry given as $k(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x}', \boldsymbol{x})$ and the condition of the matrix $\boldsymbol{K}$ being positive semidefinite. The matrix $\boldsymbol{K}$ is the gram matrix with entries given as $K_{ij} = k(x_i, x_j)$. The kernel used for the BO is chosen as the squared exponential given in Equation 7.1.

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma_{sd}^2 exp\left(-\frac{||\boldsymbol{x} - \boldsymbol{x}'||^2}{2\sigma_l}\right) \tag{7.1}$$

where σ_{sd} is the signal standard deviation and σ_l as the length-scale. The length-scale is describing the range for the correlation between different $\boldsymbol{x}$. If the length-scale attain a large value, the correlation between points $\boldsymbol{x}$ and $\boldsymbol{x}'$ is strong even if they are placed far apart. Opposite if the length-scale attain a small value, the point $\boldsymbol{x}$ do only have a strong correlation to points in the near vicinity.

As mentioned, the BO utilizes an acquisition function (AF) to determine the inputs for the next prediction. The AF utilizes the knowledge from prior points to determine in which direction and length to search for a new point. AFs have already been developed, all building upon the terms exploration and exploitation. The term exploitation is when the AF exploits the knowledge of previous points. The next point is chosen in the near vicinity of a prior points yielding a low cost to search for a local minimum. The term exploration on the other hand is when the AF choose the next point to be placed in an otherwise unexplored area. The more the AF is utilizing exploration the more it turns into a random search algorithm. The AF used for the MPC tuning is the expected-improvement-plus (EIP). The method of expected improvement (EI) determines the next point from the predictive distribution calculated from the hyperparameters of the surrogate model, namely the mean $\mu(\boldsymbol{x})$ and the standard deviation $\sigma(\boldsymbol{x})$. The goal of the BO is to maximize the EI by decreasing the mean as seen in Equation 7.2.

$$EI(\boldsymbol{x}) = \begin{cases} (f(\boldsymbol{x}^+) - \mu(\boldsymbol{x}))\boldsymbol{\Phi}(Z) + \sigma(\boldsymbol{x})\phi(Z), & if\ \sigma(\boldsymbol{x}) > \mathbf{0} \\ 0, & if\ \sigma(\boldsymbol{x}) = \mathbf{0} \end{cases} \tag{7.2}$$

where $f(\boldsymbol{x}^+)$ is the best cost function value so far and $\boldsymbol{\Phi}(Z)$ and $\phi(\boldsymbol{Z})$ is the cumulative and probability distribution respectively. Z is defined as in Equation 7.3.

$$Z = \begin{cases} \frac{f(\boldsymbol{x}^+) - \mu(\boldsymbol{x})}{\sigma(\boldsymbol{x})}, & if\ \sigma(\boldsymbol{x}) > \mathbf{0} \\ 0, & if\ \sigma(\boldsymbol{x}) = \mathbf{0} \end{cases} \tag{7.3}$$

To maximize this AF, the BO chooses points that reduce the mean thus increasing the first term of Equation 7.2. The EI uses a mixture of exploration and exploitation to find the next point but it is possible for the function to overuse exploitation and thus only focusing on finding a local minimum instead of searching unexplored territories. To reduce this effect, the AF is extended to EIP whereas the plus establishes an exploitation ratio t_σ. To choose the next point the guess $\boldsymbol{x}$ must satisfy the condition $\sigma_f(\boldsymbol{x}) < t_\sigma\sigma$ with $\sigma_f(\boldsymbol{x})$ and σ being the standard deviation of the posterior cost function and additive process noise respectively. If $\boldsymbol{x}$ do not satisfy this condition, a new point is evaluated.

The implementation and execution of the BO is conducted in 3 different processes: Acquiring training data, generation of the GPR and the actual BO.

Acquiring training data: The first step to acquire the training data is to specify number of training data sets. This number assign the number of loops to define parameter guesses and run simulations. To assign values for the parameters, the individual parameter value spans have to be established whereas the ones used for the attitude MPC can be seen in Table 7.2 together with the respective data type. These are further described in Section 7.2.

Parameter	T_s	N_p	N_c	$\boldsymbol{Q}_p$	$\boldsymbol{Q}_w$	$\boldsymbol{R}_M$	$\boldsymbol{R}_{dM}$
Value span	$8:1:16[ms]$	$6:1:15$	$2:1:5$	$[10^3\ 10^5]$	$[10^0\ 10^1]$	$[10^3\ 10^5]$	$[10^3\ 10^5]$
Data type	Integer	Integer	Integer	Double	Double	Double	Double

Table 7.2: Parameter value span of the attitude MPC.

With the parameter spans, the parameters are assigned a value randomly within the respective span with the MATLAB command *rand*. As an example, the time constant T_s and the position penalties Q_p are calculated as in Equation 7.4 and 7.5 respectively.

$$T_s = round(rand \cdot 8) + 8 \tag{7.4}$$

$$Q_p = 10^{(rand \cdot 2 + 3)} \tag{7.5}$$

The *rand* command takes a random number between 0 and 1. For T_s the data type is an integer thus the random number generated is rounded to the nearest integer. For Q_p the coefficient is of data type real and can attain all values within the span to double precision.
With all the parameters assigned, a simulation is then run for the simulation model utilizing the chosen values. The cost function for the BO is calculated by the data acquired from the simulation. The cost function for the attitude MPC is based on the quaternion errors and the required control inputs as seen in Equation 7.6.

$$J = \sum_{i=1}^{N} \left((q_j(i) - ref_j)^2 + u_k(i)^2 \right) \tag{7.6}$$

with N being the number of data samples, $j = 0, 1, 2, 3$ being the respective quaternion and $k = 1, 2, 3$ defining the respective input. It has been observed that some of the combinations of parameters results in unstable controllers thus causing the drone to spin. This causes the cost function to skyrocket and sometimes cause trouble for the BO. The outliers are thereby removed from the input matrix and output vector before moving to the GPR.

Gaussian process regression: The data acquired are further utilized to fit a GPR for the BO. The GPR are generated by use of the $fitrpg(\boldsymbol{X}, \boldsymbol{y}, 'options')$ in MATLAB. Here $\boldsymbol{X}$ is the input parameters thus the tuning parameters from the training data, $\boldsymbol{y}$ is the corresponding cost functions and within '*options*', the desired kernel is defined. The GPR is then written as in Equation 7.7.

$$gprMdl = fitrgp(\boldsymbol{X}, \boldsymbol{y}, \text{'KernelFunction', 'squaredexponential'}) \tag{7.7}$$

Bayesian optimization: With the GPR, the actual optimization algorithm can be initialized. The first part of the algorithm is to define the tuning parameters as optimizable variables whereas the respective parameter value span and the data type are established. As an example, T_s and Q_p are used again as seen in Equation 7.8 and 7.9.

$$T_s = optimizableVariable('T_s', [8, 16], 'Type', 'integer') \tag{7.8}$$
$$Q_p = optimizableVariable('Q_p', [10^3, 10^5], 'Type', 'real') \tag{7.9}$$

Next part is to write the BO where the MATLAB command *bayesOpt* is utilized. Within this command the prediction function with corresponding function handles are defined together with the desired AF. The command is shown in Equation 7.10.

$$\begin{aligned} bayesObject = & bayesOpt(@(tbl)mdlfun(tbl, gprMdl), vars, ... \\ & 'AcquisitionFunctionName', 'expected - improvement - plus') \end{aligned} \tag{7.10}$$

The *bayesOpt* calls the function $mdlfun(tbl, gprMdl)$ which is the prediction function using the GPR and the optimizable variables collected in the function handle *tbl*. The final part is to establish the prediction function utilizing the GPR and the new point chosen from the AF. The prediction function is described in Equation 7.11.

$$f = predict(gprMdl, vars) \tag{7.11}$$

Where the *bayesOpt* minimizes f by implementing the new parameters collected in *vars* into the GPR and predict the corresponding cost function value. The MATLAB code of the BO is seen in Appendix G.6.

The following sections presents the tuning and final results of the MPCs.

7.2 Attitude MPC Tuning

Before starting BO the different parameters to be tuned and corresponding parameter spans must be determined to generate proper training data sets for the GPR. The adjustable parameters in the attitude MPC are the operation time $T_{s,att}$, the prediction and control horizons $N_{p,att}$ and $N_{c,att}$, the state penalties for quaternions and body rates contained in $\boldsymbol{Q}_{p,att} \in \mathbb{R}^{4\times4}$ and $\boldsymbol{Q}_{w,att} \in \mathbb{R}^{3\times3}$ respectively, the input penalties for the torques are contained in $\boldsymbol{R}_{att} \in \mathbb{R}^{3\times3}$ and the penalties for the input rates contained in $\boldsymbol{R}_{du,att} \in \mathbb{R}^{3\times3}$. To reduce the number of tuning parameters the penalties for the quaternions are denoted the same value by the assumption, that the response of the different quaternions are similar (**Assumption 23**). The same goes for the penalties for the body rates, the inputs and the input rates. This reduces the number of tuning parameters from 16 to 7 and thus assumed to reduce the number of training data sets required to find a suitable solution. As mentioned in Section 7.1 the parameters used for the training data are attained randomly between a stated intervals seen in Table 7.2. These intervals are established from knowledge obtained from prior simulations to define a decent grid to search for an optimal solution. An exception is the calculation time $T_{s,att}$, where the upper and lower limits are determined from a mixture of control theory and practical implementation. First the largest bandwidth of the rotations was calculated in Section 4.5 whereas the upper limit of the calculation time was stated as a frequency of 10 times the bandwidth. This ensures, that the controller is at least 10 times faster than the largest closed loop bandwidth. The lower bound was due to the possible calculation time observed from simulations where the slowest calculations were at the vicinity of 8 $[ms]$.

With the parameter spans defined, the parameters can be assigned values and simulations can be performed to generate the training data required for establishing a GPR. Within the simulations the designed UAV is constantly affected by rotational disturbances and the attitude MPC is thus desired to be robust towards these disturbances. It is thereby chosen to generate the training data from simulations containing the rotational disturbances multiplied by a factor of 10. By this it is assumed, that if the designed UAV is capable of stabilizing while exposed to greater disturbances the same would be the case for lesser disturbances and thus ensure a more robust controller. Results are presented for the case of both the normal and increased disturbances in Figure 7.2. Additionally for the GPR, the number of training data sets must de stated. With 7 parameters to be determined it was assumed, that numerous simulations were required to establish a suitable surrogate model. To test this hypothesis 2 different GPR models were generated, one with 10 sets of training data and one with 100. These models were used in the BO whereas the final controller coefficients can be seen in Table 7.3 and the results are seen in Figure 7.2.

Training data	$T_{s,att}$	$N_{p,att}$	$N_{c,att}$	$\boldsymbol{Q}_{p,att}$	$\boldsymbol{Q}_{w,att}$	$\boldsymbol{R}_{att}$	$\boldsymbol{R}_{d,att}$
10	12 $[ms]$	9	3	10^(4.37)	10^(4.9)	10^(3.04)	10^(4.21)
100	15 $[ms]$	10	4	10^(4.75)	10^(4.83)	10^(3.09)	10^(5)

Table 7.3: Final controller coefficients of the attitude MPC.

At the left column of Figure 7.2 the results of the 2 attitude MPCs are seen in the presence of attitude disturbances amplified by a factor of 10 while the right column shows the results in the presence of the formerly established attitude disturbances. For all of the results the controllers are seen to be stable within a period of 100 $[s]$. With the increased disturbances, the rotations

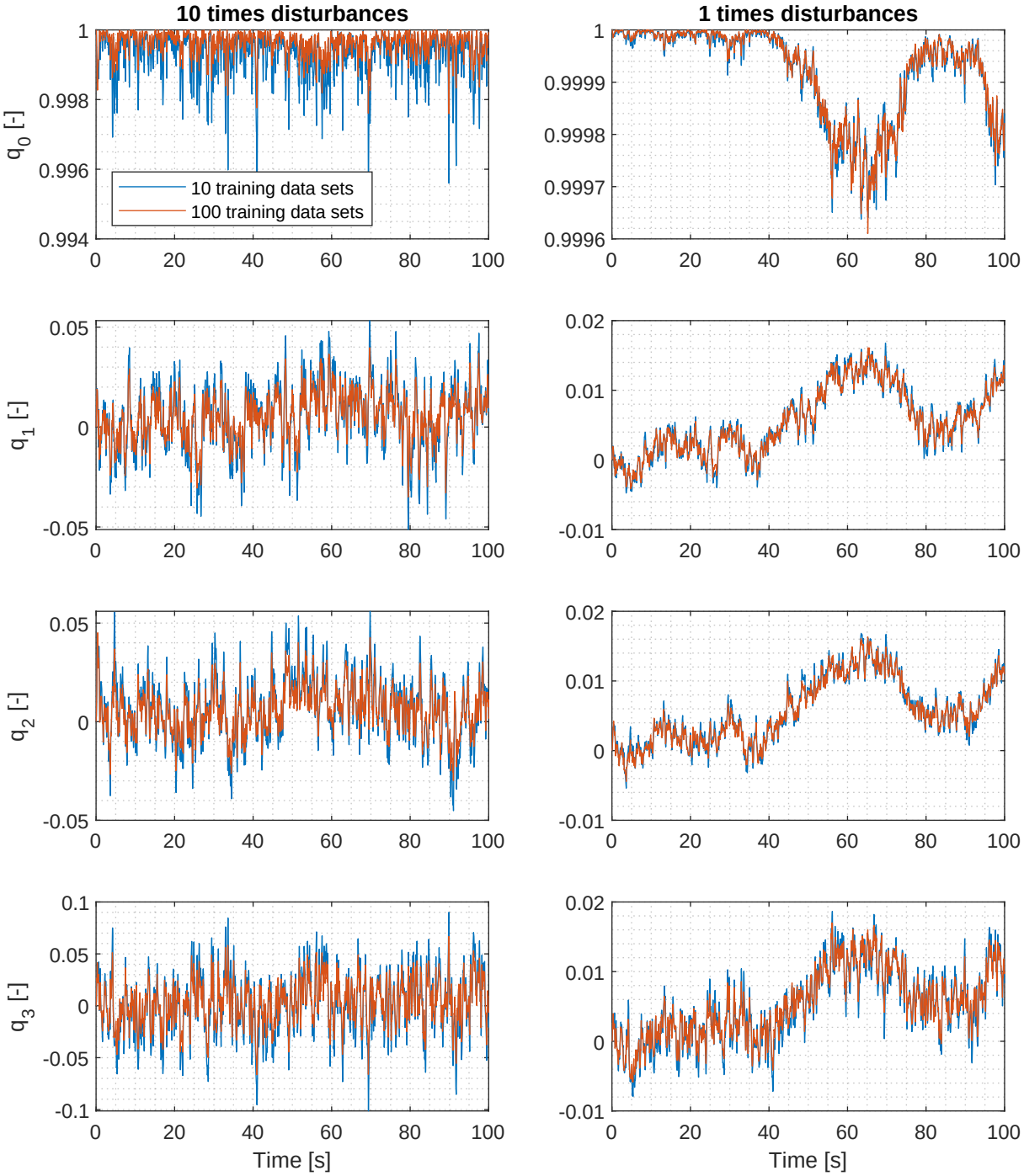


Figure 7.2: Results of the attitude MPC.

of the UAV are seen greater than for the results with normal disturbances. Both controllers are fluctuating around the references but at the left column the MPC based upon 10 training data sets are seen to have larger spikes than the MPC based upon 100 training data sets. The same is the case for the right column with normal disturbances but here the difference is much less noticeable. To gain a more comparable measure of the controllers a SSE analysis of all 4 results are performed and summarized in Table 7.4.

Training data	**Increased dis.**	**Normal dis.**
10	5185	1788
100	3761	1744

Table 7.4: SSE of the attitude MPC results.

As seen from Table 7.4 the SSE is larger for the MPC based on 10 training data sets. With the increased disturbances the MPC of 10 data sets have around a 1.4 times larger SSE than for the MPC of 100 sets. For applying normal disturbances the SSE results are almost equal. It can thus be argued that for the controller design with small disturbances only 10 training data sets (or maybe less) is sufficient to obtain a suitable solution but with larger disturbances it is desired to utilize more training data for the GPR. To see the effects on the translational position from the rotations still occurring with the attitude MPC, the x, y and z positions are seen in Figure 7.3. A 10 second simulation is run with normal rotational disturbances and utilizing the attitude MPC generated from the 100 training data sets, while no attempts are made to control position.

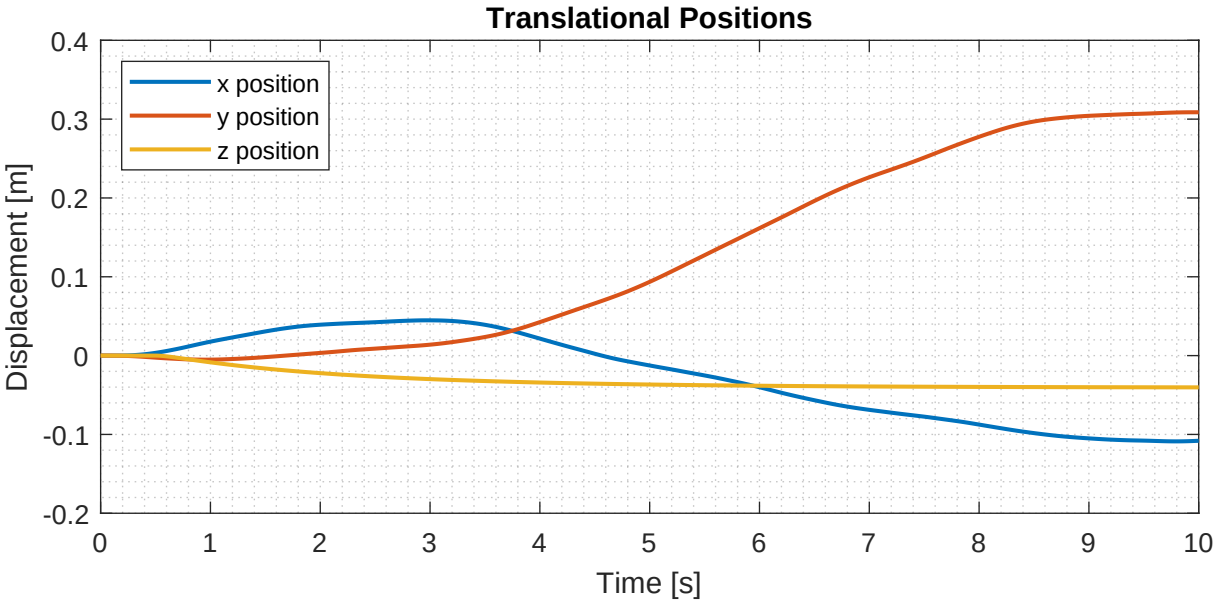


Figure 7.3: Translational positions utilizing the MPC based on 100 training data sets.

Even though the attitude MPC correct for the rotational disturbances, the translational position is seen to deviate from the original position. For further work the attitude MPC based on the 100 training data sets is utilized. For the next part, the position tracking MPC is tuned utilizing BO.

7.3 Position Tracking MPC Tuning

For tuning the position tracking MPC, the combining of the penalties in the $\boldsymbol{Q}$ and $\boldsymbol{R}$ matrices was not found sufficient and all penalties were assigned a value independently. As for the attitude MPC the respective parameter spans are determined from prior simulations and can be seen in Table 7.5.

Parameter	$T_{s,pos}$	$N_{p,pos}$	$N_{c,pos}$	Q_x	Q_y
Value span	$80:10:160[ms]$	10	4	$[10^0\ 10^5]$	$[10^0\ 10^5]$
Data type	Integer	Integer	Integer	Double	Double

Parameter	Q_z	Q_{dx}	Q_{dy}	Q_{dz}	R_x
Value span	$[10^0\ 10^5]$	$[10^3\ 10^8]$	$[10^3\ 10^8]$	$[10^3\ 10^8]$	$[10^0\ 10^5]$
Data type	Double	Double	Double	Double	Double
Parameter	R_y	R_z	R_{dx}	R_{dy}	R_{dz}
Value span	$[10^0\ 10^5]$	$[10^0\ 10^5]$	$[10^0\ 10^5]$	$[10^0\ 10^5]$	$[10^0\ 10^5]$
Data type	Double	Double	Double	Double	Double

Table 7.5: Parameter value span of the attitude MPC.

Within Table 7.5 Q_i indicates the penalties of the translational positions, Q_{di} are the penalties for the translational velocities, R_i are the penalties for the thrust inputs and R_{di} denotes the penalties of the thrust input rates. The calculation time parameter span is chosen as 10 times the span of the attitude MPC. This is chosen from the bandwidth assessment of the position compared to the attitude bandwidth presented in Section 4.5. This though implies, that future reference points in the maximum period of $T_{s,pos}N_p$ must be obtainable for trajectory tracking. It is thus assumed, that a trajectory can be generate 1.6 $[s]$ in the future at any given time step (**Assumption 24**). The objective function for the position tracking MPC utilized in the BO are seen in Equation 7.12.

$$J = \sum_{i=1}^{N} \left(5((i) - x_{ref})^2 + (y(i) - y_{ref})^2 + 5(z(i) - z_{ref})^2\right) \tag{7.12}$$

The objective function is only defined from the position errors. Due to the frequency of the WTBs oscillations and the possibility of colliding with the inner walls of the WTB, the x and z position errors are penalized by a factor of 5. The simulations for the training data are run for 25 seconds but it was deemed necessary to utilize 100 training data sets to obtain an accurate surrogate model. The simulation used for the training data was performed under the following conditions:

- Rotational disturbances are amplified by a factor of 10 to obtain a robust controller.
- The y reference was defined as a ramp with a slope of 1 $[m/s]$ for the designed UAV to mimics a flight through the WTB.
- The simulation was performed under what is referred to as worst case conditions. This implies that the x and z reference are given by Equation 2.1 and 2.2, respectively. The amplitude of the oscillations are chosen at a wind speed of 12 $[m/s]$ in accordance with Figure 1.6.

The resulting tuning parameters are stated in Table 7.6, however it should be noted that 2 tuning parameters were adjusted manually. The immediate solution from the BO resulted in a good tracking of the y and z references but the x position had phase lag and only reached $1/4$ amplitude of the oscillating reference. Due to the resemblance of the x and z references it was assumed, that the x position would track the reference with higher accuracy if the penalties related to x attained similar penalties as for the penalties related to z. By adjusting the penalties for the x input and input rate the reference tracking was significantly improved.

$T_{s,pos}$	$N_{p,pos}$	$N_{c,pos}$	Q_x	Q_y	Q_z	Q_{dx}	Q_{dy}
120 $[ms]$	10	4	10^(4.64)	10^(4.51)	10^(4.87)	10^(7.73)	10^(7.69)
Q_{dz}	R_x	R_y	R_z	R_{dx}	R_{dy}	R_{dz}	
10^(7.81)	10^(4.96)	10^(4.74)	10^(4.91)	10^(2.73)	10^(4.74)	10^(2.9)	

Table 7.6: Position tracking MPC coefficients designed for moving y reference.

The final controller design are utilized for testing under different conditions as presented and evaluated in Section 7.4.

7.4 Results

In this section simulation results are presented in order to evaluate controller performance under different conditions. Three different scenarios are presented referred to stabilization, normal conditions and worst case scenario.

Stabilization: In these simulations constant references are given for $\boldsymbol{q}$ and $\boldsymbol{p}$, which corresponds to hovering. The designed UAV is affected by an increased rotational disturbance by a factor of 10 in Figures 7.4 and 7.5 and a factor of 30 for Figures 7.6 and 7.7.

Normal Conditions: Deflection corresponding to a wind speed of 5 $[m/s]$ in accordance with Figure 1.6 and Equations 2.1 and 2.2. The oscillation frequency of the wind turbine wing was reduced with 50%, i.e. $\omega_{WTB} = 0.3495[Hz]$. The y reference corresponds to a speed in the y direction of $1.5[m/s]$. The designed UAV is affected by an increased rotational disturbance by a factor of 10 in Figures 7.4 and 7.5 and a factor of 30 for Figures 7.6 and 7.7.

Worst case scenario: The worst case scenario is defined as a flight through the WTB at a speed of $1\,[m/s]$ in the y direction while the WTB is oscillating with the frequency and maximum deflection stated in Section 1.6 Additionally, the UAV is exerted for translational and rotational disturbances. The rotational disturbances are enhanced with a factor of 10 as seen in Figures 7.12 and 7.13 and a factor of 30 as seen in Figures 7.14 and 7.15.

For all the results below, simulation 2 contains disturbances amplified by a factor of 30. The variances for the rotational disturbances are stated in Table

Multiplication factor	$M_x\,[Nm]$	$M_y\,[Nm]$	$M_z\,[Nm]$
10	$1.87 \cdot 10^{-4}$	$1.18 \cdot 10^{-3}$	$4.28 \cdot 10^{-4}$
30	$5.61 \cdot 10^{-4}$	$3.54 \cdot 10^{-3}$	$1.28 \cdot 10^{-3}$

Table 7.7: Values used for the rotational disturbance variances.

7.4.1 Stabilization Problem with Disturbances

Stabilization Simulation 1:

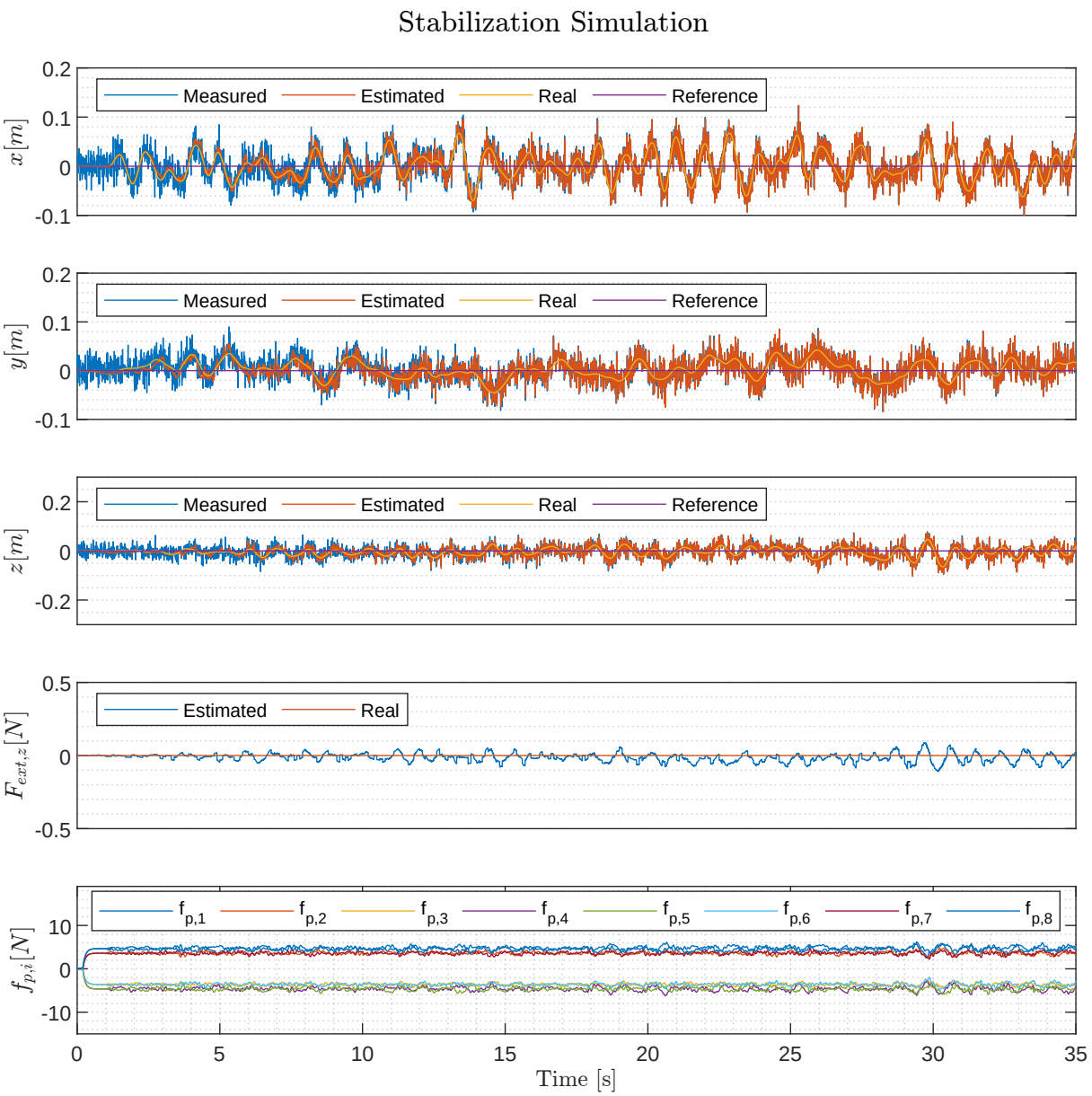


Figure 7.4

From Figure 7.4 it can be seen that the position controller remains stable throughout the simulation. A maximum deviation for x, y and z of approximately $0.06[m]$ was observed for the estimated states which is used to calculate the error inside the position MPC.

The UKF only reduces noise on x, y and z in the very beginning of the simulation. This can be seen by comparing the estimated and real states whereas variance on the estimated states approaches the variance on the measured states as the simulation progresses.

The estimated disturbance is fluctuating around zero. As z is influenced by the rotational disturbances this could in turns influence the estimators ability to predict $F_{ext,z}$.

The inputs remain stable throughout the simulation and only changes slightly.

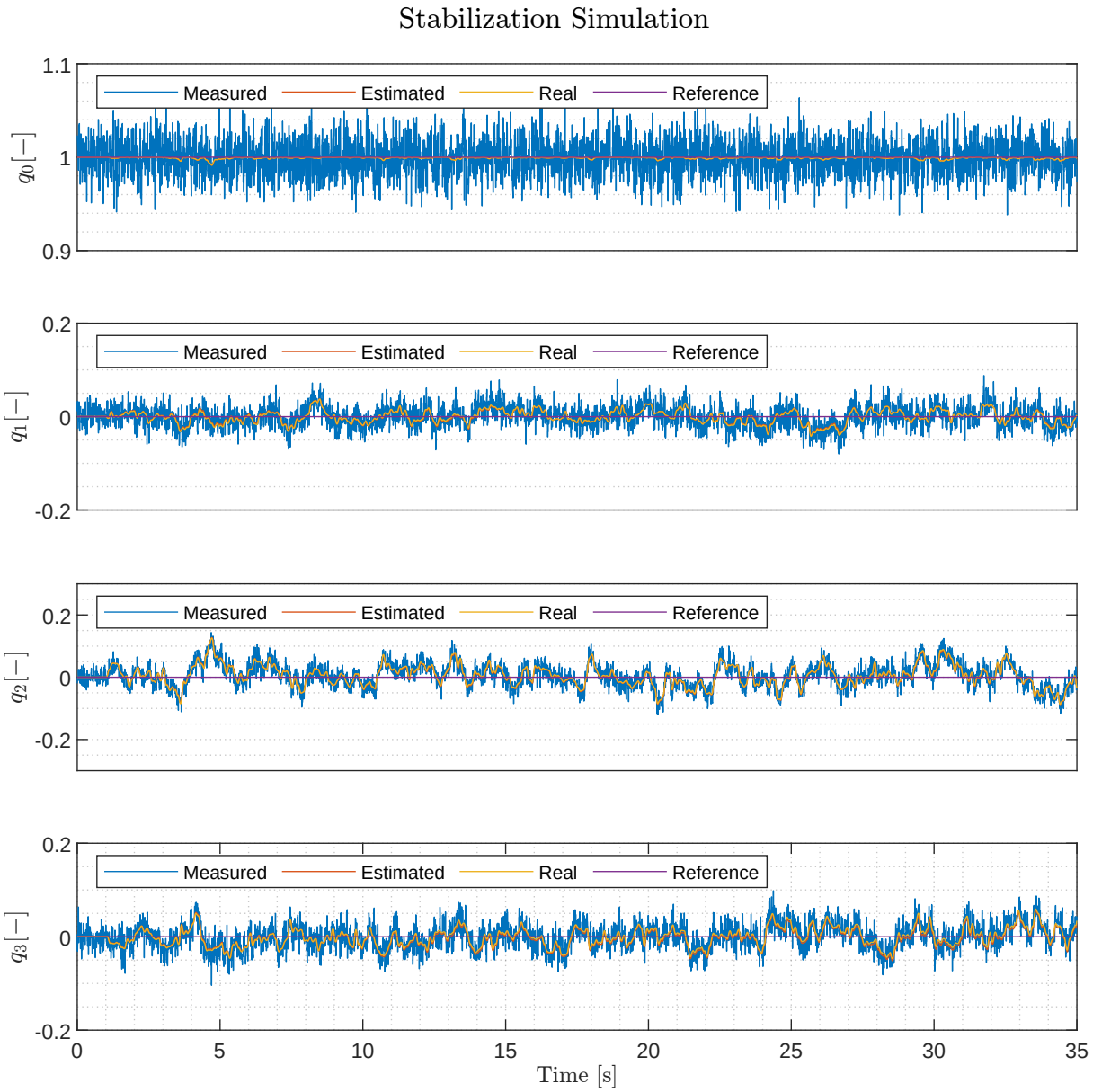


Figure 7.5

The quaternion states and thus the attitude dynamics are stabilized throughout the full simulation. A peak error on the estimated quaternions can be observed to be around 0.1[−] for q_2.

The UKF manages to perform satisfactory noise reduction on the quaternions throughout the simulation. Again this can be observed by comparing estimated states with real states. This remains a general tendency in all results presented in this section.

Stabilization Simulation 2:

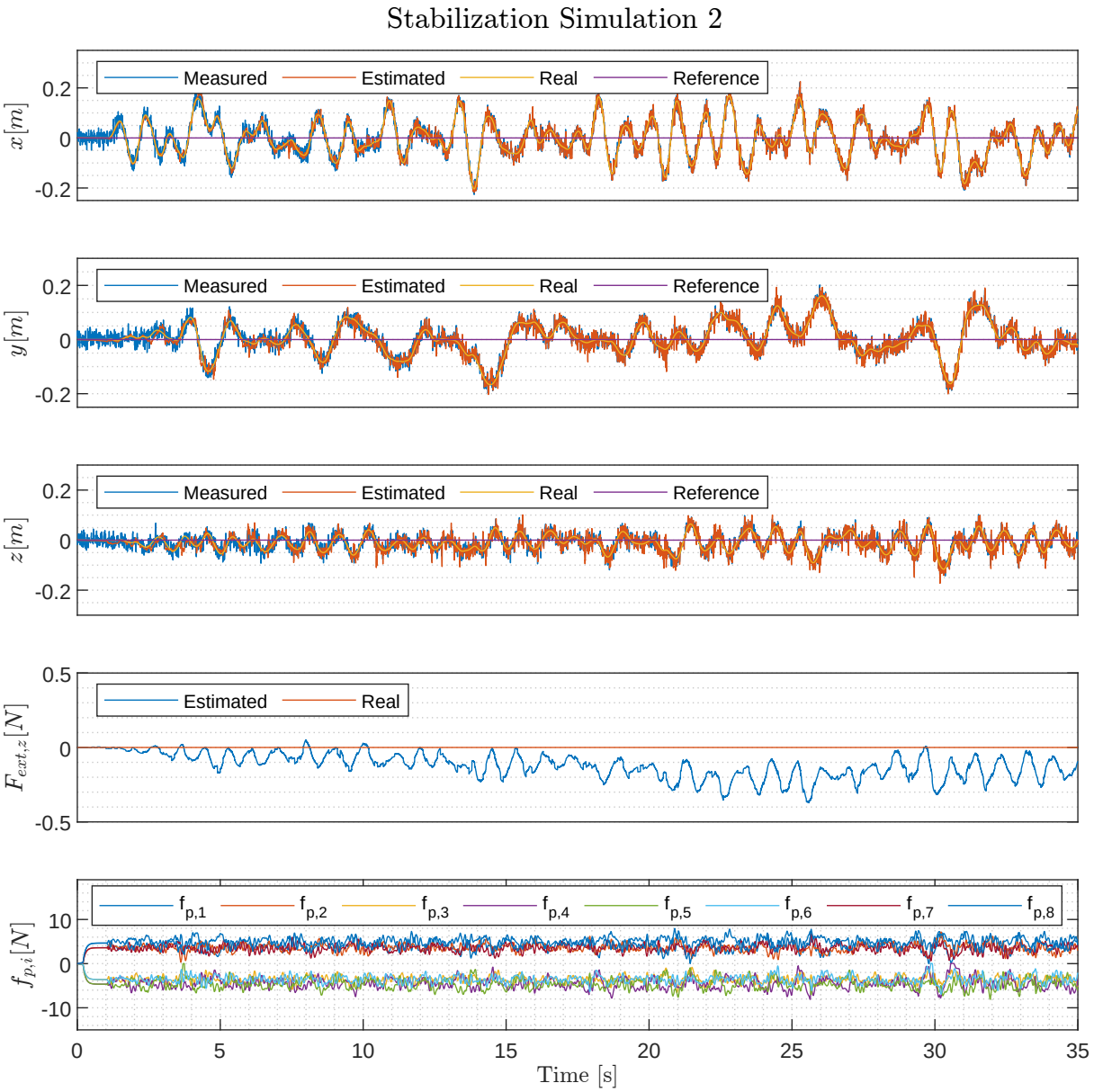


Figure 7.6

Due to the increased rotational disturbances x, y and z deviations increase when compared to the first stabilization simulation with a peak error of around $0.2\,[m]$ for the estimated states.

The estimated disturbance from the AKF is fluctuating more than in the previous simulation, presumably due to the increased rotational disturbances.

In the input graph, a rotor direction change can be observed at around $30\,[s]$ into the simulation. The rotors $f_{p,1}, f_{p,4}$ and $f_{p,6}$ experience slow rotor response associated with direction change without compromising stability.

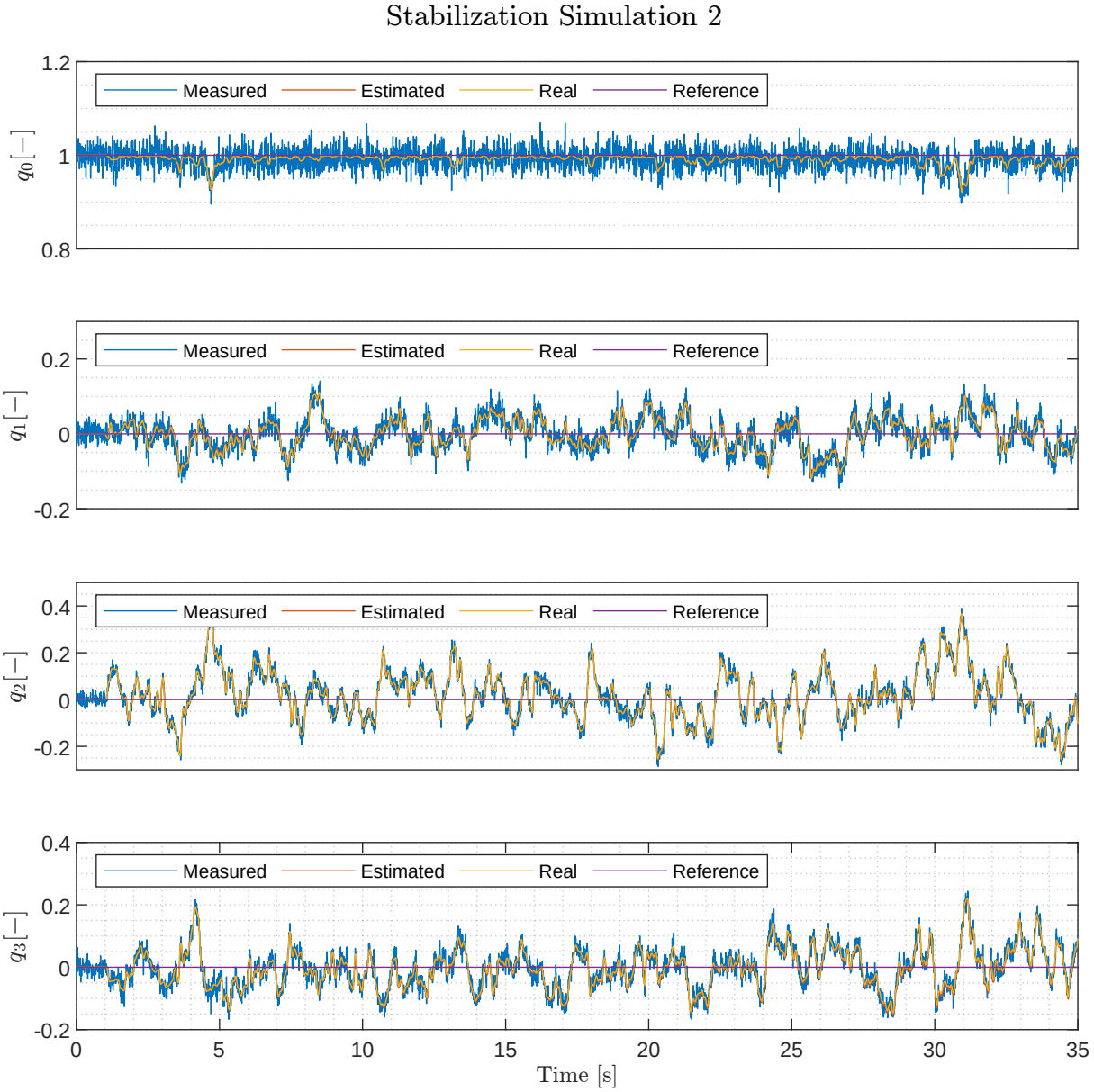


Figure 7.7

The attitude dynamics displays stable behavior, as each quaternion fluctuates around the reference. However, by inspecting the estimated quaternions large errors can be observed with a peak error of around $0.4[-]$ for q_2. The peak error occurs at the instance where the aforementioned rotors change spinning direction as seen in Figure 7.6.

7.4.2 Normal Conditions Simulation with disturbances

Normal Conditions Simulation 1:

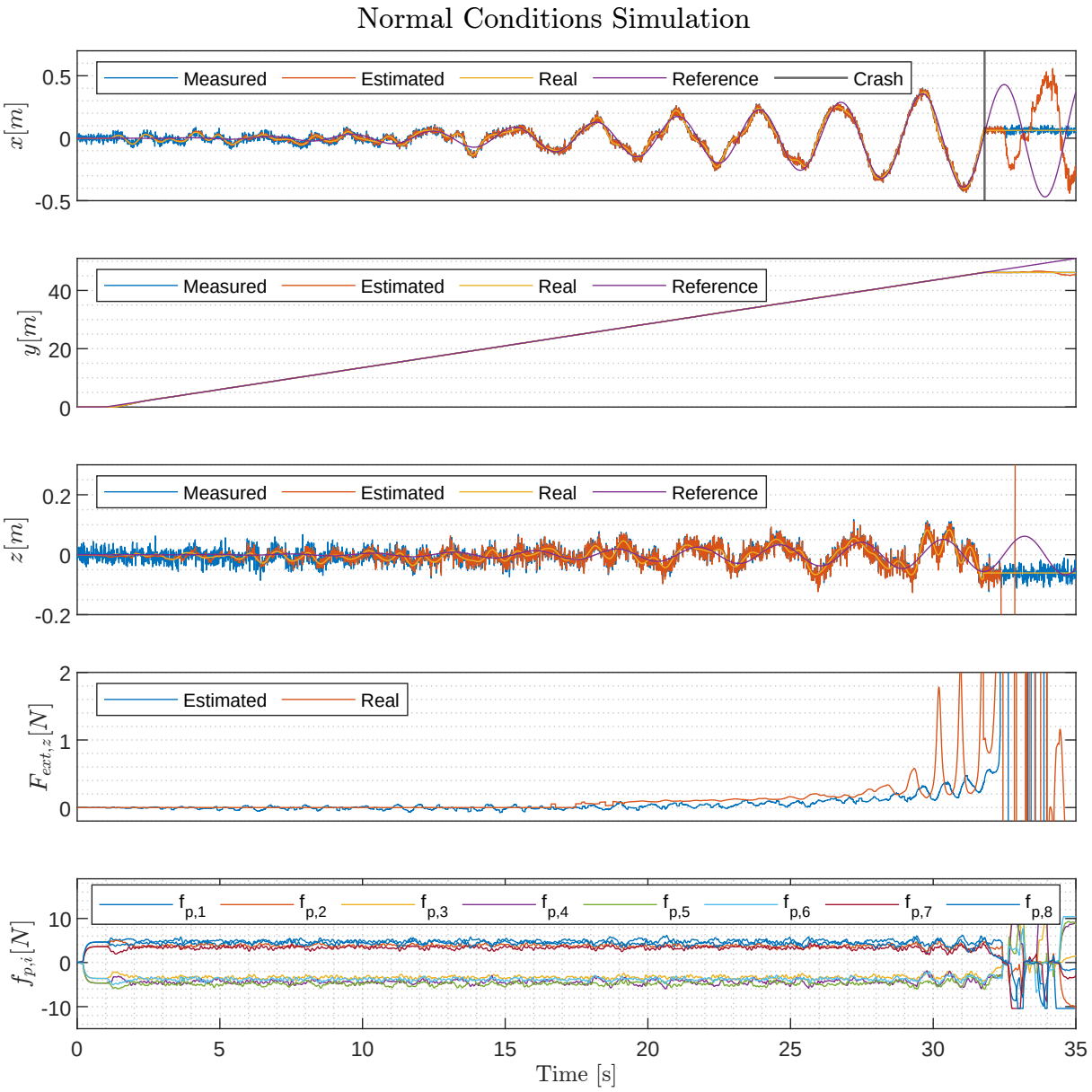


Figure 7.8

The control performs well during the simulation until the UAV collides with the inner walls of the WTB at around 33 $[s]$ into the simulation. The UAV covers a distance of 45 $[m]$ into the WTB as seen in the plot for y position.

Prior to collision with the inner wall the GE and CE increases drastically and the AKF fails to react to the violent disturbances.

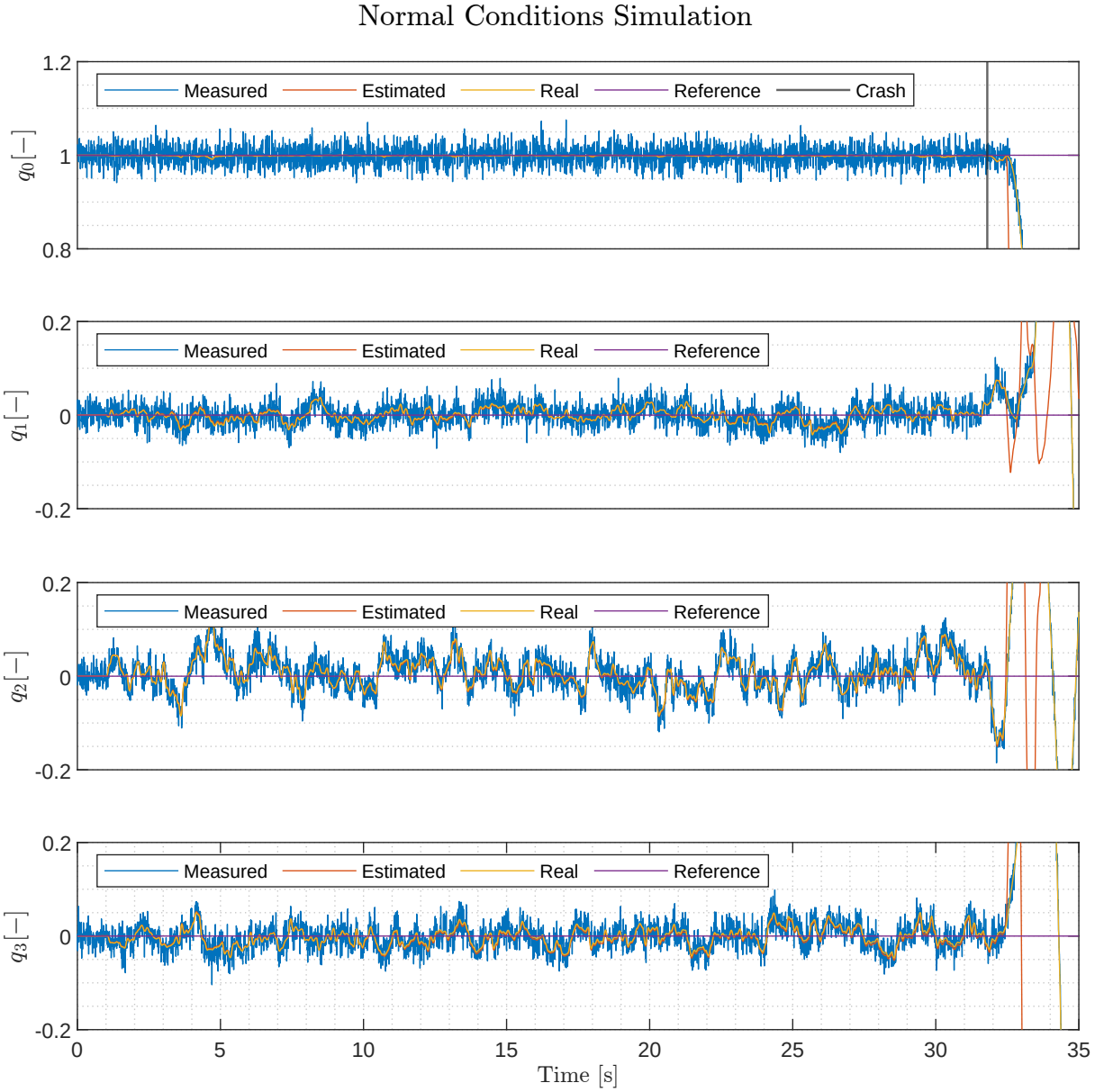


Figure 7.9

The quaternions are kept stable by the attitude controller until the collision of the UAV occurs. A peak error is observed to be approximately $0.12[-]$ for q_2.

Normal Conditions Simulation 2:

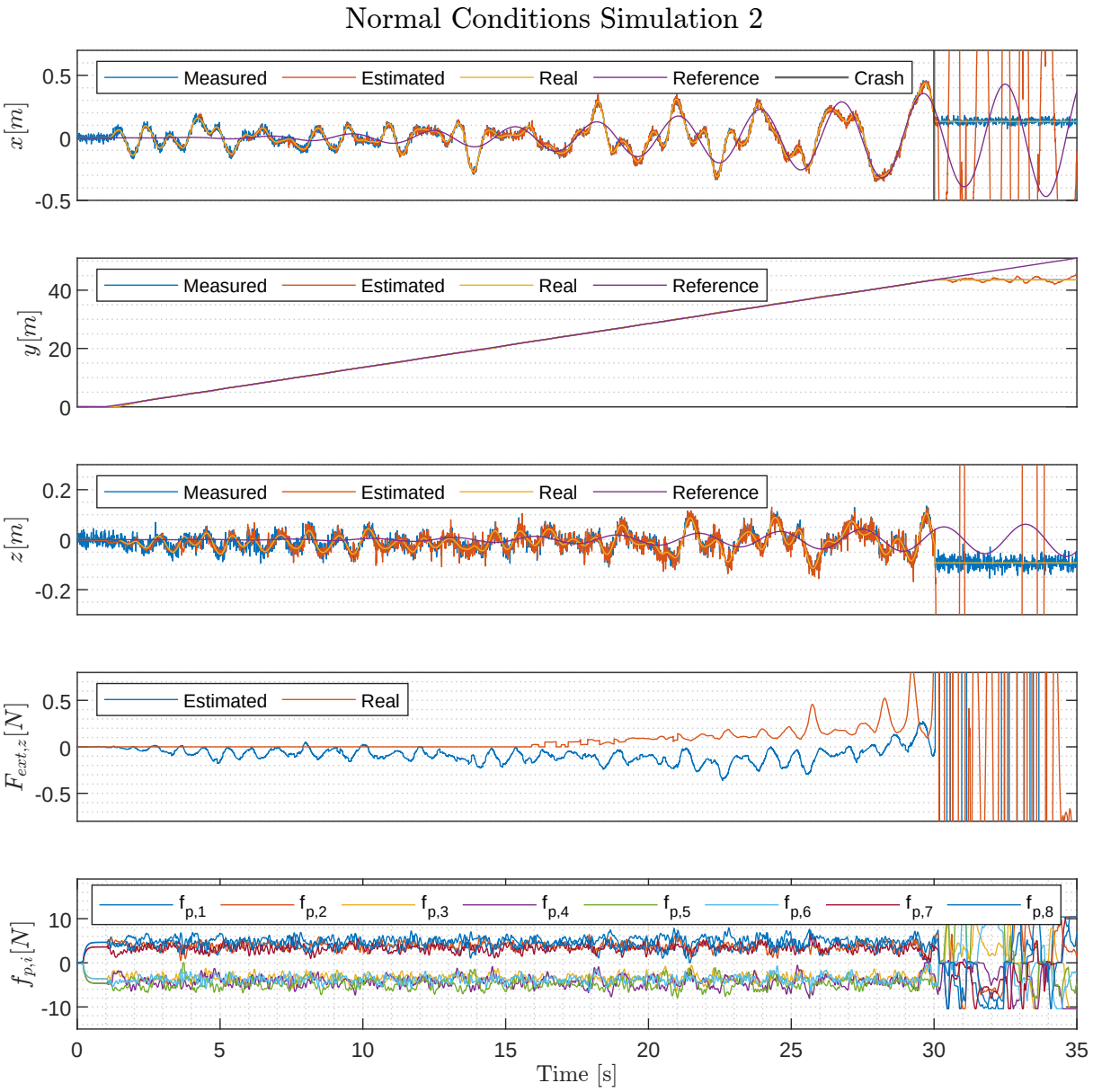


Figure 7.10

The increased rotational disturbances cause larger fluctuations on x, y and z compared to the previous simulation. Further, the crash occurs earlier with increased rotational disturbances at around 30 $[s]$ into the simulation. This corresponds to a distance of approximately 39 $[m]$ into the WTB.

The estimated disturbance from the AKF is drifting. This is an issue since the disturbance estimate is used in the position tracking MPC.

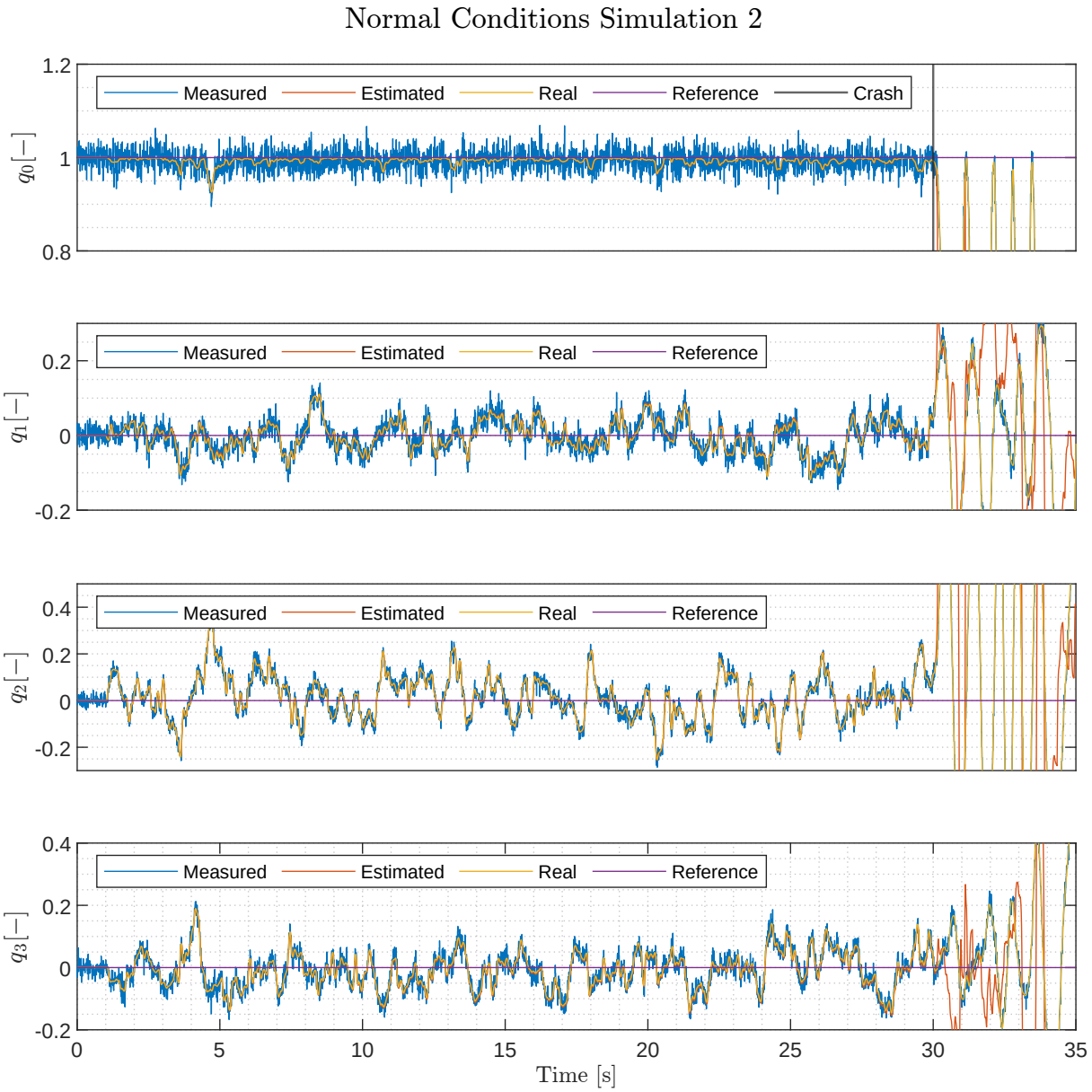


Figure 7.11

At Figure 7.11 the quaternions fluctuate with a peak error of around $0.35[-]$ for q_2. The fluctuations are concentrated around the reference and the attitude MPC is still deemed stable until collision occurs.

7.4.3 Worst Case Conditions with Disturbances

Worst Case Simulation 1:

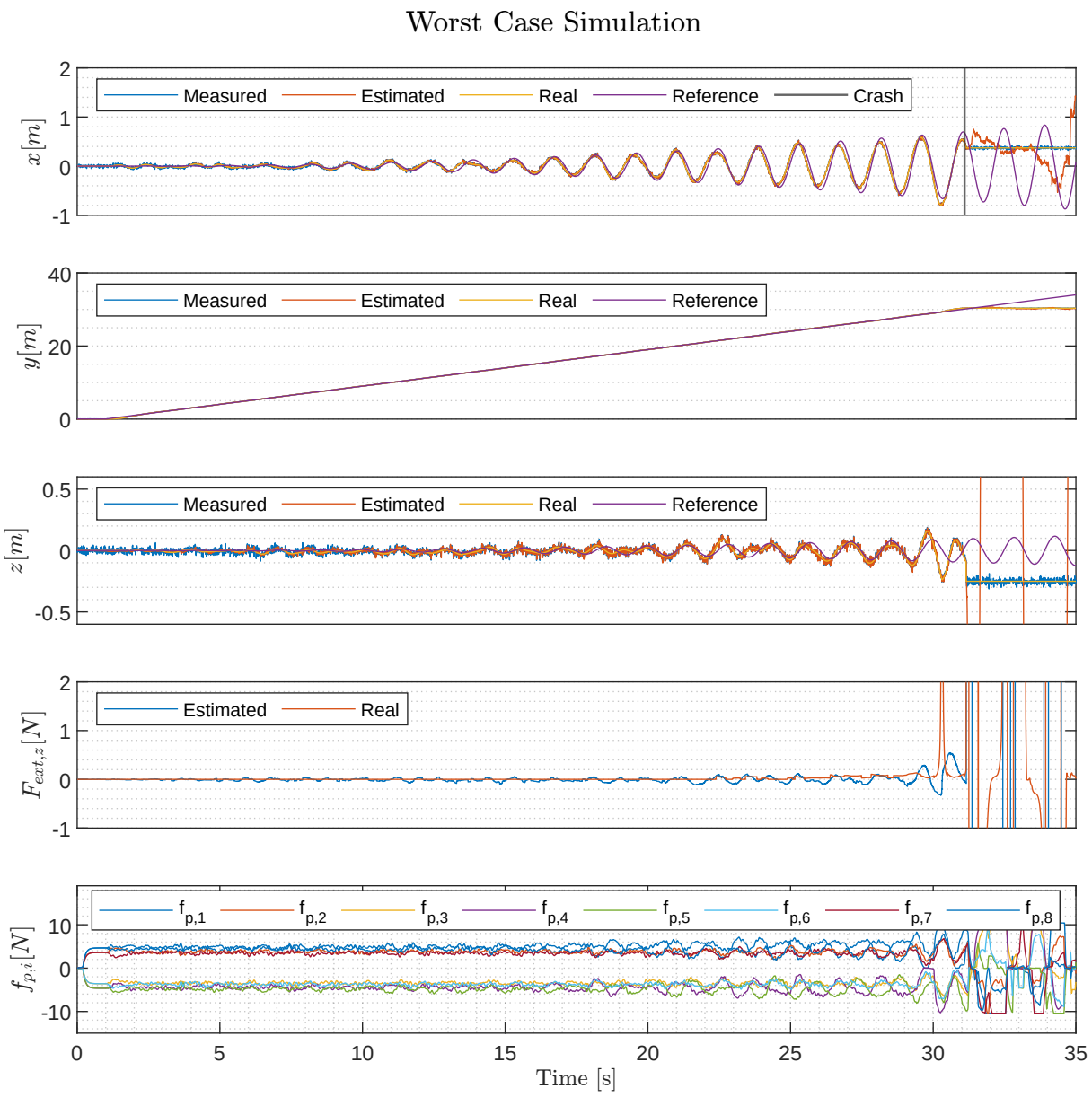


Figure 7.12

It can be observed that x and z estimates slightly leads the aggressive trajectory reference.

The collision occurs at around 31 $[s]$ into the simulation which corresponds to 30 $[m]$ into the WTB. The collision happens after 2 rotors change direction at approximately 29 $[s]$ into the simulation. The collision is caused by a combination of the aggressive trajectory and the slow rotor responses occurring prior to the collision.

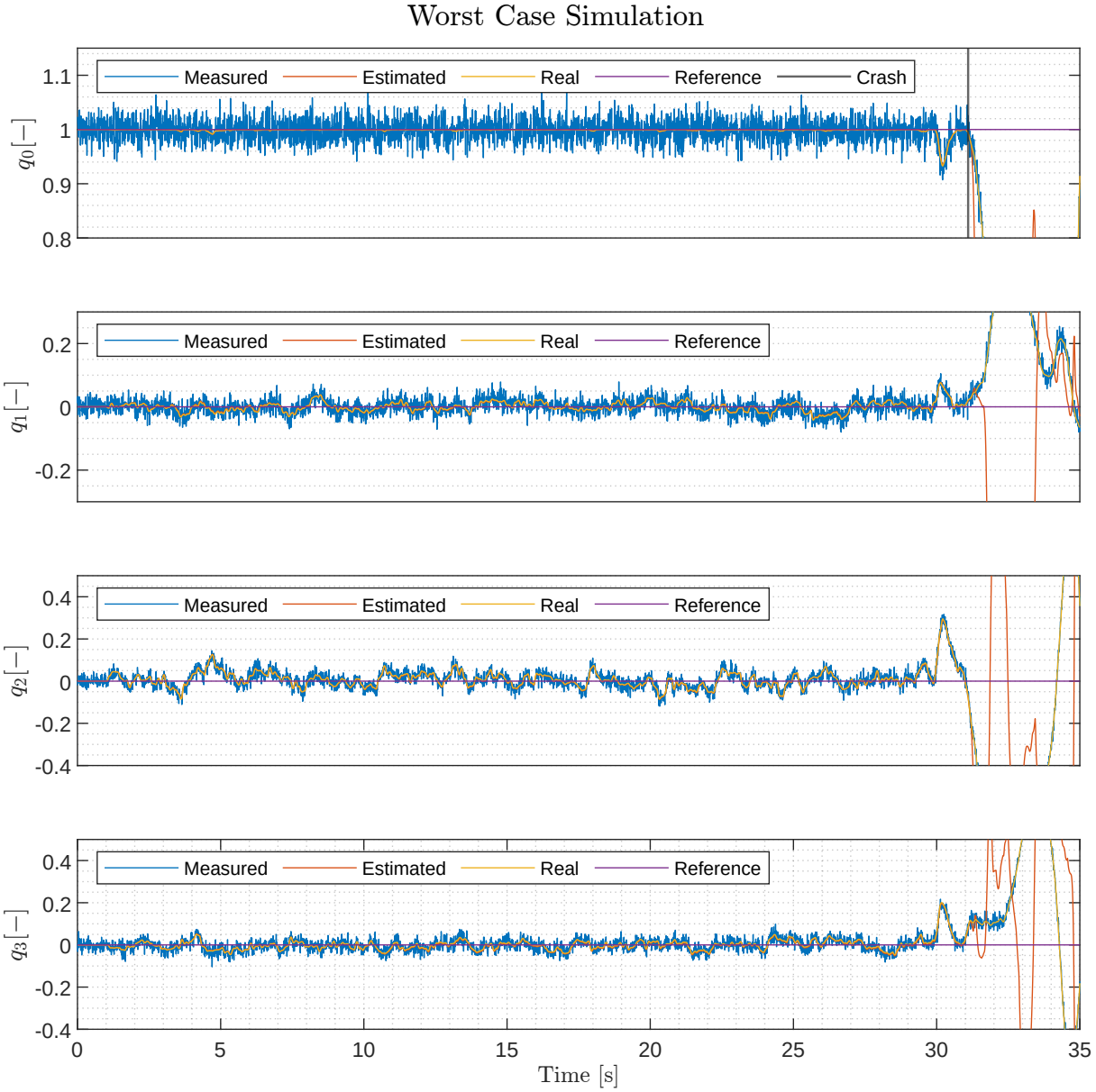


Figure 7.13

The quaternions are stabilized throughout the simulation with a maximum error of around 0.15 $[-]$ for q_2 at approximately 5 $[s]$ into the simulation prior to the collision.

Worst Case Simulation 2:

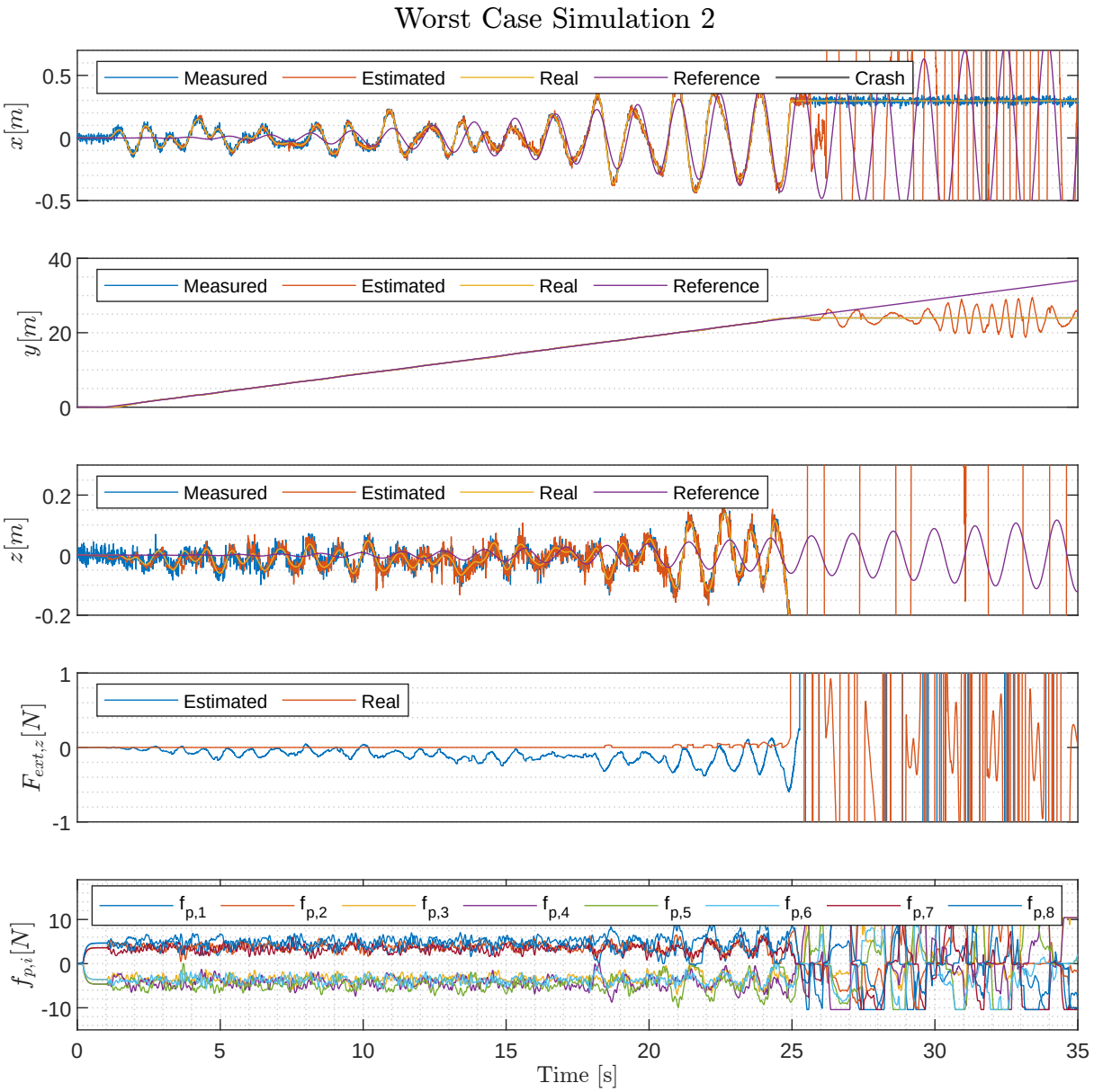


Figure 7.14

Due to the increased disturbances the collision occurs earlier at Figure 7.14 than in the previous simulation. The UAV collides with the inner wall at around 26 $[s]$ which corresponds to a flight of 24 $[m]$ into the WTB.

By inspecting the input plot, rotors are located in the slow response area multiple times prior to the collision without compromising stability. The first zero crossing of the inputs are seen at time 18 $[s]$.

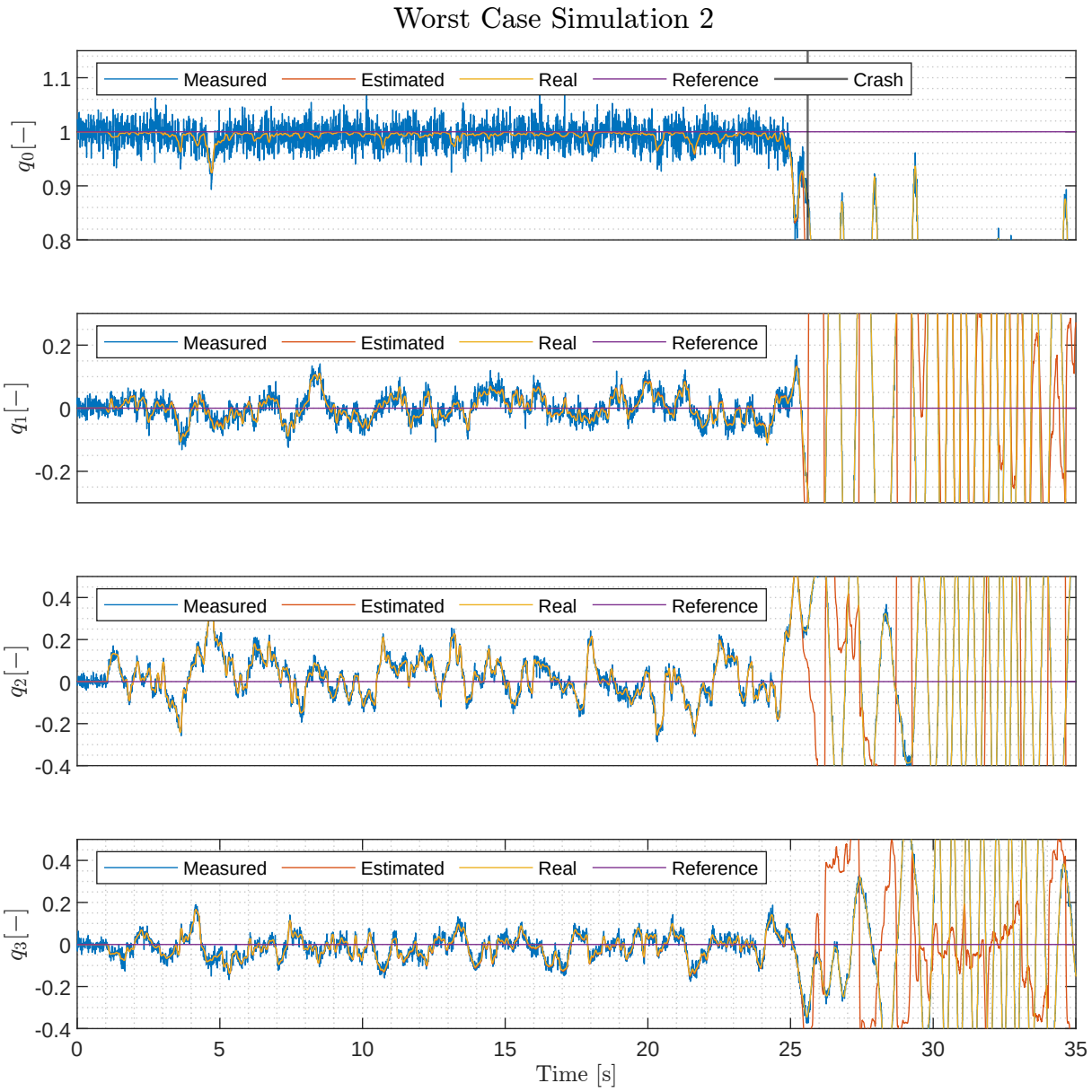


Figure 7.15

The quaternions are stabilized, but fluctuate around the reference with a peak error of around 0.3 [−] for q_2.

8 | Discussion

8.1 On Assumptions Regarding Measurements and Trajectory Generation

Assumption 3 states that the test trajectory developed in Chapter 2, Section 2.2 is constructed to represent worst case conditions during a hypothetical inspection. It should be noted that it is not realistic that the WTB should oscillate in this manner. A WTB has several modes of vibration [100, p.11-12], and therefore oscillations differ in frequency throughout the length of the WTB. In fact some of the modes might have higher frequency of oscillation than what was considered in this work. In fairness inspections should not be conducted when weather conditions are rough. Further, the WTB can be tilted to reduce the surface area of the WTB that is normal to the approximate wind direction and thereby external forces acting on the WTB can be reduced when an inspection is desired.

A method of online trajectory generation was not investigated, but remains an important issue to be handled for a robotic solution that can inspect confined areas such as the inside of a WTB. It was stated in Assumption 16 that such solution was assumed to be possible by means of the tracking and depth cameras in combination with an intelligent image processing algorithm. However, it is not known how accurate such a solution will be in practise.

It should also be mentioned that Assumption 12 is made based on recommendations from Intel RealSense [101], but it has not been tested in practise in this work. In regards to the depth and tracking cameras it was assumed that a pose update in fusion with the BNO055 (IMU) can be obtained at a rate of 100 $[Hz]$. This relates to Assumption 14 and 15 this is neither tested in practise and whether it is possible, cannot be verified.

Lastly Assumption 13 was necessary because we did not have the opportunity to perform tests on the chosen candidate sensors. Fortunately, we previously worked with the BNO055 (IMU) and performed tests to obtain variance of the measured euler angles as well as on linear acceleration. Also the BNO055 is well documented in the provided datasheet. As result variance values had to be assumed for position, velocity and angular body rate. Sensor drift was not considered but could result in further implications regarding accuracy of measurements.

A consequence of the assumptions considered in this discussion it can not be concluded whether the developed design and control structure can stabilize a prototype UAV flying in similar conditions.

8.2 Design Considerations

(**Assumption 4**) The basic idea of designing a fully actuated UAV was the capability of controlling all 6DOF independently. Choosing the design consisting of 8 rotors allowed the UAV to generate thrust in the 3 translational directions and torque for the 3 rotations by choosing the right configuration of the rotors. A drawback of the 8 rotors, compared to the traditional quadcopter, was the increased size and as the UAV was desired to fly within a confined space, the increased size reduced the maneuverable space and limited the flyable length into the WTB. Even though the drone size is increased, the UAV has the advantage of moving the the x and y direction without performing either roll or pitch.
To gain the most efficiency compared to usable force and torque from the rotors the total CAM was desired to be maximized utilizing a grid search approach to determine the optimum rotor orientations. By only maximizing the CAM different concerns arose related to coupling of the outputs and the physical limits and behaviours of the inputs. From the initial optimum of rotor orientations the required force inputs for some of the rotors were seen to surpass 3 times the saturation limit and additionally commanding motion for some DOF resulted in motion for other DOF thus indicating coupling between the outputs. To locate the cause of the large inputs and the coupling appearing for the design it was chosen to analyze the singular values to determine the effort of each inputs on the different outputs. Here the difference between the lowest and highest singular value was above a factor 1000 and some input thereby required a high effort to be commanded compared to others. By implementing an inequality constraint for the condition number the required inputs and the coupling were significantly reduced though other problems arose from early stage simulations concerning the bidirectional rotors. As bidirectional rotors are utilized this allowed the rotors to generate force in both directions as they could change spinning direction. The problem occurring from the constraint of the singular values was, that some of the inputs were close to zero while following the defined references. As the oscillations of the WTB increased the rotors producing low thrust started to change direction frequently. If a rotor changed spinning direction the given rotor response time was increased which caused an asymmetry for the thrust and torque distribution. The frequent changing of spinning directions of some of the rotors would eventually result in the UAV would crash. To avoid the frequent change of spinning direction the optimization was further subject to the inequality constraints regarding elements of the CAM being different from zero and forcing the error of the input away from either zero or the saturation limit. These constraints were a trade-off with the singular values as increasing the error on the specific inputs resulted in a slack of the constraint for the condition number. This meant, that by forcing the inputs closer to the center of zero and saturation the condition number had to increase to find a possible solution. Increasing the condition number required larger inputs to follow the WTBs oscillations and would in some cases result in rotors changing spinning direction. The effects of the constraints can be seen in Figure 8.1.

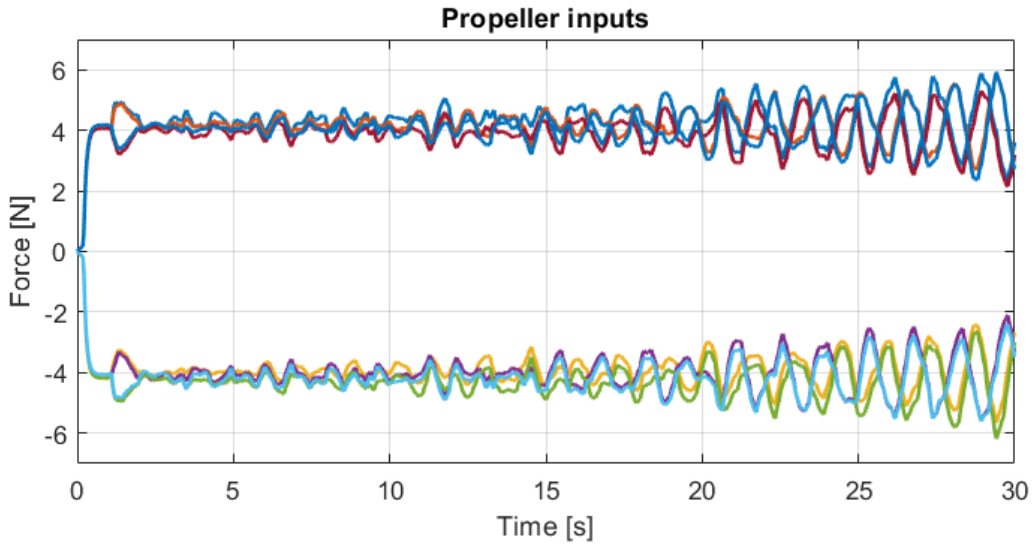


Figure 8.1: Rotor inputs from the final flight with the full trajectory.

Figure 8.1 shows the 8 rotor inputs from the worst case simulation presented in Figure 7.12 and 7.13. Within the first second the rotors are initialized to reach the hover thrust. within the period $1-2$ seconds the inputs deviated from the hover thrust as the UAV had to accelerate in the y direction. For the rest of the simulation the behaviour of the inputs can be divided into 2 parts. The first part is the non periodic fluctuations due to the compensations for the attitude MPC for the rotational disturbances. The second part is the sinusoidal oscillations required to follow the x and z references. The further the UAV moved into the WTB the more the amplitude of the sinusoidal inputs increased. At the end of the simulation the UAV had reached 29 $[m]$ into the WTB and the inputs do not reach either across zero or the saturation limit but for flight further into the WTB the required inputs would cause the rotors to change spinning direction. At this point the clearance to the walls would be small and the slow response of the rotors as they change direction could cause the UAV to hit the wall. The bidirectional rotors do not need a complex optimization algorithm for calculation of the inverse of the CAM as the limits can be accommodated in the control. On the other hand, as the changing spinning direction causes problems for the control it might be beneficial to utilize unidirectional rotors.

8.3 On The Disturbance Models

Two disturbance models were constructed, one related to translational disturbances, seen in Assumption 9, and one related to rotational disturbances, seen in Assumption 11.
The translational disturbances have been modelled as GE and CE. Those disturbances were simplified to only affect in the global z direction to construct disturbance rejection in the position tracking MPC. This assumption is though a rough approximation of the translational disturbances occurring inside a confined space and especially as the 8 rotors are directed in different x, y and z orientations. In real life the translational disturbances are assumed to affect differently for the 8 rotors depending on each rotors respective orientation, how close they are placed to the walls, the individual spinning direction and the UAVs rotational position. The translational disturbance model is thus not seen as a realistic model between the UAV and the environment but is mostly used to evaluated the necessity for disturbance rejection. The extension of a translational disturbance model is further elaborated in Chapter 10.

For the rotational disturbances it is assumed, that the UAV would be affected by at least wall effect and interfering air streams generated from the rotors. To mimic these disturbances, a normal distributed random noise was generated for each rotational direction. As no experiments were conducted to determine these disturbances the variances were chosen to be 5 % of the torque the rotors were capable of generating. Once again the disturbances were introduced to the simulation model to test how sensitive the UAV structure was and the robustness of the controller structure. For a more realistic rotational disturbance model experiments should be conducted for the UAV build.

8.4 On Results and Bayesian Optimization

As seen from the results in Figure 7.2 the attitude MPC was capable of stabilizing the rotational orientation of the UAV with only slight deviation even when the rotational disturbances were increased by a factor of 10. It was though seen more difficult to tune the position tracking MPC just for stabilizing the translational position whereas the results in Figure 7.4.1 were seen to fluctuate. One of the causes for these fluctuation was the measurement noise not being reduced by the UKF and the estimated states for the position tracking MPC were influenced by noise as seen in Section 7.4. Another cause was the coupling effects where the translational motions were seen sensitive to rotations. The combination of these could explain the deviation of the translational positions at $4 - 6$ $[cm]$ as seen in Figure 7.4. The UAV was still capable of maintaining a position in the vicinity of the references but due to the confined space it had to fly within, these fluctuations had a crucial role for how far into the WTB the UAV could perform inspection.
Another aspect of the controllers was the usage of the AKF disturbance estimator for the purpose of improving disturbance rejection. By inspecting the results in Section 7.4 it can be seen, that the AKF has a slow response when estimating the disturbance force $F_{ext,z}$, which could have a negative impact on the ability to reject the disturbance. Further, it can observed that the AKF is significantly influenced by rotational disturbances which in turns influences the positions MPCs ability to track a given reference. This can be seen by comparing the fluctuations of $\hat{F}_{ext,z}$ from simulation 1 with simulation 2 for both stabilization, normal conditions and and worst case conditions.
The MPCs were seen fit for the UAV control structure but each controller contained multiple parameters to be determined and thereby BO was utilized to ease the tuning process. The BO proved highly sufficient for the tuning process and in some cases required much less training data than first anticipated. The number of training data required of a suitable solution would depend on different factors. The first being the prior knowledge of the simulation model as the parameter spans were to be defined. For the attitude MPC, it was fairly simple to manually determine the parameter spans and narrow the field of suitable controller designs but for the position tracking MPC the parameters were assigned bigger spans. This together with the increased number of parameters required an increased number of training data sets. Further the complexity of the task the UAV had to perform seemed to influence the number of training data sets necessary to attain some proper controller coefficients. This is especially the case for the full trajectory whereas 100 training data sets were required to gain the needed insight of the controller parameters to manually adjust the coefficients.

9 | Conclusion

Capability of producing thrust and torque for all DOF: The designed UAV consisted of 8 rotors with only 6DOF to control. This overactuation combined with the presented rotor placements and orientations allowed independent control of all translational and rotational motions.

Determine optimal rotor locations and orientations based on the application: An optimization algorithm was utilized to determine the optimum orientations of the rotors. The optimization was subject to inequality constraints regarding coupling, efficiency of inputs and physical considerations as rotors changing spinning direction or inputs reaching saturation. The chosen rotor orientations resulted in slight coupling from rotations to translational motions. Contrary the design was seen capable of flying 24 $[m]$ into a WTB in the worst case scenario and 45 $[m]$ for the normal condition scenario when combined with the control structure. For the worst case scenario, the UAV is even kept stabilized as the rotors change spinning direction.

Realistic proposal of UAV build: For the build both the general UAV structure and a size and weight estimate were presented. The work was fully simulation based and a prototype is still to be constructed but components such as the rotor configuration has been evaluated for the respective design. Additionally, a camera solution consisting of a depth and tracking camera has been proposed in combination with an 9DOF IMU for establishing the designed UAVs position and orientation during flight.

Derive a realistic simulation model capturing dominant dynamics for the purpose of control design: The designed UAV was modelled by rigid body dynamics for the global translational and body rotational accelerations. The input dynamics were modelled as an approximation of the rotor dynamics established from static and dynamic experiments obtained from [43] and [57]. Additionally measurement noise and both translational and rotational disturbances were introduced to mimic real life flight conditions.

Follow aggressive and smooth trajectory of sinusoidal tendency with a frequency corresponding to ω_{WTB}: The combination of the depth and tracking cameras was assumed capable of providing a trajectory for the design UAV to follow within the WTB. For small amplitudes of the WTBs oscillations the flight of the UAV didn't follow the trajectory as smooth due to the fluctuations being the dominant factor. As the oscillations increased further into the WTB the UAV was seen to follow the frequency of the sinusoidal waveforms smoothly. For normal conditions the UAV reaches a length of 45 $[m]$ and 39 $[m]$ into the WTB in simulation 1 and 2 respectively. In simulation 1, only small fluctuations are present for the reference tracking of the x and z position, where the UAV follows the WTBs oscillations. For simulation 2, larger fluctuations occur which causes an earlier collision with the inner walls of the WTB than simulation 1. For worst case conditions, the UAV fly 30 $[m]$ and 24 $[m]$ inside the WTB before collision with the inner wall seen at simulation 1 and 2 respectively. For both simulations, the x and z references

are smoothly followed though the x position is seen to lead the reference.

Keep the orientation aligned with inertial frame: The attitude MPC was capable of stabilizing the UAVs orientation at all the presented scenarios. The quaternions are though fluctuating substantially due to the strong rotational disturbances. The peak error is observed for q_2 at $0.3\,[-]$ corresponding to 22° deviation for roll, pitch and yaw when exerted for simulation 2 of the worst case scenario. The attitude MPC is concluded robust toward disturbances but not aggressive enough to compensate for the amplitude of the disturbances resulting in rotational fluctuations.

Robustness towards unknown and unmodelled disturbances: The disturbances introduced to the simulation model were random normal distributed pulses for rotation and GE/CE for the translational z motion. The attitude MPC was concluded to be robust toward the rotational disturbances even as these were increased by factors of 10 and 30. The UAV was exposed to translational disturbances beyond $18\,[m]$ into the WTB. The UAV still achieved to cover a distance of $6\,[m]$ under the influence of disturbances before colliding with the inner walls of the WTB for the worst case scenario and up to $27\,[m]$ for normal conditions. The response of the AKF was too slow to in regards to estimate the translational disturbance and thus a more aggressive disturbance estimator is desired.

10| Future Work

Continuation of Simulation Studies

Observer choice and design:

The presented disturbance models are only a rough simplification of the reality and the observers are designed thereby. This is especially concerning the AKF with the purpose of estimating the translational disturbance in the global z direction. First off the AKF was seen to lack the real disturbance and thus it is recommended to test observer designs such as sliding mode observers to improve the disturbance estimation speed. Additionally the disturbance estimator(s) should concern disturbances appearing for all the translational motions.

Controller considerations:

For the control structure two different MPCs are used each operating at their respective iteration frequency. Within each calculation the previous output is kept constant thus giving a ZOH until the next calculation is performed. This gives a continuous representation of the discrete control signal but have very abrupt changes. To smooth the continuous input a spline or bezier curve could be constructed from previous and current controller outputs. Additionally, the performance could be improved as the spline or bezier curves approximates the control outputs in between optimizations. Another aspect regarding further testing of the controllers and the observers is to introduce model inaccuracies in the prediction models to test robustness. This is recommended as the presented prediction models are based on the same equations as stated for the plant, thus only the approximation of the rotors are deviates. In practise the prediction models are assumed to differ from the UAVs motions and the model inaccuracies. These inaccuracies could mimic the deviations occurring between the prediction model and the UAVs flight. Before experimental testing it is also advised to further investigate robustness towards increased noise measurements and augmenting the GE/CE model to produce translational disturbances in all directions.

Prototyping of the Designed UAV

The Designed UAV build:

A main point for future work is the realization of the presented UAV design. This work is completely simulation based but to see if the presented design, control structure and optimization algorithms for tuning actually works, it requires the designed UAV to be build in real life.

Extensive work on sensors:

As the presented work is purely simulation based the needed for handling sensors have been reduced to simply determining a simple measurement noise model. For the realization it is recommended to perform an elaborate evaluation of the sensor measurements as the designed UAV is desired to fly within a WTB. One point for future work is the need of a trajectory generation algorithm with obstacle avoidance. Secondly the sensors are capable of providing the

same measurements but from different locations on the UAV and with different frequencies. To gain more accuracy of the measurements and to reduce the noise it is recommended to utilize sensor fusion. Lastly due to local magnetic fields generated from within the WT, electromagnetic disturbances are expected and must be accounted for to ensure an accurate 9DOF pose.

Test of the designed UAVs robustness toward different environments:
If the designed UAV is realized it is recommended to test the robustness of the design in different environments. The inside of a WTB is a small confined moving space involving a number of different disturbances and this is not the ideal starting point for testing the UAV. As a suggestion the initial flight testing of the design could be at a large indoor space with plenty of opportunity of moving around. Next could be outdoor testing where the UAV is exerted for disturbances such as the wind. Next could be a stationary confined space as a tube or pipe before testing in a moving confined space. By tuning the controllers to the different conditions, the robustness of the designed UAV and control structure can be determined for different environments.

Experimental Validation of the Designed UAV Simulation Model
Experiments and test within a confined environment:
It is advised to thoroughly test a prototype in laboratory setups in different hollow structures, such as rectangular, cylindrical or other types of pipes, because the shape of box spars vary with the type of WTB. Furthermore such a laboratory setup could be used to derive a more realistic disturbance model. The translational disturbance model presented in this work only see the translational disturbance for the drone in the global z direction, whereas the rotational disturbances were modelled as normal distributed random pulses. For more accurate simulations it is recommended to extend the disturbance models from experimental work. The translational disturbance could involve the effects from each rotor by orientation, relation to walls, spinning direction and partial GE/CE. The rotational disturbances could be modelled from tests of wall effect and the crossing air streams of the rotors.

| Bibliography

[1] Globe News Wire. Company. Last adressed: 9 February 2021.

[2] Ying Du, Shengxi Zhou, Zianjian Jing, Yeping Peng, Hongkun Wu, and Ngaiming Kwok. Damage detection techniques for wind turbine blades: A review. *Mechanical Systems and Signal Processing*, 141, 2020.

[3] ASM S., Xiao C., Vladimir F., Anders N. C., Nicolai A. B. R., Kim B., Anders B. D., and Rasmus R. P. Wind turbine surface damage detection by deep learning aided drone inspection analysis. *Maintenance management of wind turbines*, 12:676, 2019.

[4] Jaclyn Solimine, Christopher Niezrecki, and Murat Inalpolat. An experimental investigation into passive acoustic damage detection for structural health monitoring of wind turbine blades. *Structural Health Monitoring*, 19(6):1711–1725, 2020.

[5] Zhen Li, Arthur Haigh, Constantinos Soutis, Andrew Gibson, and Robin Sloan. Microwaves sensor for wind turbine blade inspection. *Appl Compos Mater*, 24:495–512, 2017.

[6] Lin Yonggang, Tu Le, Liu Hongwei, and Li Wei. Fault analysis of wind turbines in china. *Renewable and Sustainable Energy Reviews*, 55:482–490, 2016.

[7] EUDP. Instruction blade inspections. *Wind turbine blade inspection*, 2014.

[8] Anne Jüngert. Damage detection in wind turbine blades using two different acoustic techniques. *The e-Journal of Nondestructive Testing*, 2008.

[9] Force Techonolgy. Company.

[10] Clobotics. Company.

[11] Andrius Kulsinskas, Petar Durdevic, and Daniel Ortiz-Arroyo. Internal wind turbine blade inspections using uavs: Analysis and design issues. *Energies*, 14(2):294, 2021.

[12] Cathiness Wind Farm. Summary of wind turbine accident data.

[13] Bryan Bridge Tariq P. Sattar, Hernando Leon Rodriguez. Climbing ring robot for inspection of offshore wind turbines. *Industrial Robot*, 36, 2019.

[14] Sun Lim; Chang-Woo Park; Jung-Hoon Hwang; Dong-Yeop Kim; Tae-Keun Kim. The inchworm type blade inspection robot system. *International Conference on Ubiquitous Robots and Ambient Intelligence*, 2012.

[15] Jianxin Gao Jan Seton, Vitor Frosas. Advanced shearography kit and a robotic deployment platform for on-site inspection of wind turbine blades. *EU research results*, 2019.

[16] Øyvind Netland; Gunnar Jenssen; Hilde Marie Schade; Amund Skavhaug. An experiment on the effectiveness of remote, robotic inspection compared to manned. *IEEE International Conference on Systems, Man and Cybernetics*, 2014.

[17] Chris Rizos. A positioning system for indoor and outdoor applications where gnss does not work. *Association of Public Authority Surveyors Conference*, pages 73–83, 2014.

[18] Gérard Lachapelle. Gnss indoor location technologies. *Journal of Global Positioning Systems*, 3, 2005.

[19] Stephen Austin Conyers. Empirical evaluation of ground, ceiling, and wall effect for smallscale rotorcraft. *Electronic Theses and Dissertations*, page 1570, 2019.

[20] E. Kuantama, T. Vesselenyi, S. Dzitac, and R. Tarca. Pid and fuzzy-pid control model for quadcopter attitude with disturbance parameter. *INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS AND CONTROL*, 12(4):519–532, 2017.

[21] Moayad H. Y. Abu Rmilah, Mohsen A. Hassan, and Noor Azizi Bin Mardi. A pc-based simulation platform for a quadcopter system with self-tuning fuzzy pid controllers. *A SIMULATION PLATFORM FOR QUADCOPTER SYSTEMS*, 24:934–950, 2016.

[22] Lucas M. Argentim, Willian C. Rezende, Paulo E. Santos, and Renato A. Aguiar. Pid, lqr and lqr-pid on a quadcopter platform. *2013 International Conference on Informatics, Electronics and Vision (ICIEV)*, 2013.

[23] Rumit Kumar, Matthew Dechering, Abhishek Pai, Austin Ottaway, M. Radmanesh, and Manish Kumar. Differential flatness based hybrid pid/lqr flight controller for complex trajectory tracking in quadcopter uavs. *2017 IEEE National Aerospace and Electronics Conference (NAECON)*, 2017.

[24] N. Cibiraj and M. Varatharajan. Chattering reduction in sliding mode control of quadcopters using neural networks. *Energy Procedia*, 117:885–892, 2017.

[25] Ngoc Phi Nguyen and Sung Kyung Hong. Fault-tolerant control of quadcopter uavs using robust adaptive sliding mode approach. *Energies*, 12:95, 2019.

[26] Safeer Ullah, Adeel Mehmood, Qudrat Khan, Sahki Rehman, and Jamshed Iqbal. Robust integral sliding mode control design for stability enhancement of under-actuated quadcopter. *International Journal of Control, Automation and Systems*, 18(7):1671–1678, 2020.

[27] Yadira Quiñonez, Fernando Barrera, Ian Bugueño, and Juan Bekios-Calfa. Simulation and path planning for quadcopter obstacle avoidance in indoor environments using the ros framework. *Trends and Applications in Software Engineering*, 688:305–316, 2017.

[28] Sergio Garcia, M. Elena López, Rafael Barea, Luis M. Bergasa, Alejandro Gómez, and Eduardo J. Molinos. Indoor slam for micro aerial vehicles control using monocular camera and sensor fusion. *International Conference on Autonomous Robot Systems and Competitions*, 2016.

[29] Shibo Zhao and Zheng Fang. Direct depth slam: Sparse geometric feature enhanced direct depth slam system for low-texture environments. *Sensors*, 18:3390, 2018.

[30] Mina Kamel, Michael Burri, and Roland Siegwart. Linear vs nonlinear mpc for trajectory tracking applied to rotary wing micro aerial vehicles. *IFAC PapersOnLine*, 50(1):3463–3469, 2017.

[31] Bekir Sait Ciftler, Adem Tuncer, and Ismail Güvenc. Indoor uav navigation to a rayleigh fading source using q-learning. *Networking and Internet Architecture*, 2017.

[32] Tung-Cheng Wu, Shau-Yin Tseng, Chin-Feng Lai, Chia-Yu Ho, and Ying-Hsun Lai. Navigating assistance system for quadcopter with deep reinforcement learning. *Conference*, 2018.

[33] Yusuke Inagaki; Hiroshi Ikeda; Patrik Ken Takeuchi; Yoshihiko Yato; Takayuki Sawai. An effective measure for evaluating sewer condition: Uav screening in comparison with cctvs and manhole cameras. *Water Practice and Technology*, 15(2):482–488, 2020.

[34] Chengbiao Wang Jiaojiao Gao, Yusen Yan. Research on the application of uav remote sensing in geologic hazards investigation for oil and gas pipelines. *International Conference on Pipelines and Trenchless Technology 2011*, 2011.

[35] FLYABILITY. Company.

[36] droneflight. Company.

[37] Yangbo Long and David J. Cappelleri. Linear control design, allocation, and implementation for the omnicopter mav. *2013 IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[38] Yangbo Long and David J. Cappelleri. Complete dynamic modeling, control and optimization for an over-actuated mav. *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[39] Yangbo Long and David J. Cappelleri. Omnicopter: A novel overactuated micro aerial vehicle. *Advances in Mechanisms, Robotics and Design Education and Research*, 14:215–226, 2013.

[40] Ahmed Alkamachi and Ergun Ercelebi. A proportional derivative sliding mode control for an overactuated quadcopter. *Journal of aerospace engineering*, 233(4):1354–1363, 2019.

[41] G. Jiang and R. Voyles. Hexrotor uav platform enabling dextrous interaction with structures-flight test. *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–6, 2013.

[42] Danie Conradie. Six degrees of freedom omnicopter with ardupilot. Last accessed: 8 February 2021.

[43] Dario Brescianini and Raffaello D'Andrea. Design, modeling and control of an omni-directional aerial vehicle. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3261–3266, 2016.

[44] E. Dyer, S. Sirouspour, and M. Jafarinasab. Energy optimal control allocation in a redundantly actuated omnidirectional uav. *2019 International Conference on Robotics and Automation (ICRA)*, pages 5316–5322, 2019.

[45] Ramy Rashad, Jelmer Goerres, Ronald Aarts, Johan B.C. Engelen, and Stefano Stramigioli. Fully actuated multirotor uavs. *IEEE Robotics and Automation Magazine*, 27(3):97–107, 2020.

[46] C. Lindenburg. Aeroelastic modelling of the lmh64-5 blade. *Technical Report*, 2002.

[47] D. Pardo and K. Branner. Finite element analysis of the cross-section of wind turbine blades; a comparison between shell and 2d-solid models. *Wind Engineering*, 2005.

[48] Bladena and KIRTxTHOMSEN. Wind turbine baldes handbook, 2019. Last accessed: 11/2-2021.

[49] Ziying Yu, Zhenhong Hu, Xing Zheng, Qingwei Ma, and Hongbin Hao. Aeroelastic performance analysis of wind turbine in the wake with a new elastic actuator line model. *Technical Report*, 2020.

[50] Luis David Avenda˜no-Valencia and Spilios D. Fassois. Damage/fault diagnosis in an operating wind turbine under uncertainty via a vibration response gaussian mixture random coefficient model based framework. *Technical Report*, 91:326–353, 2017.

[51] Sangyul Park, Jongbeom Her, Juhyeok Kim, and Dongjun Lee. Design, modeling and control of omni-directional aerial robot. *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1570–1575, 2016.

[52] Mina Kamel, Sebastian Verling, Omar Elkhatib, Christian Sprecher, Paula Wulkop, Zachary Taylor, Roland Siegwart, , and Igor Gilitschenski. The voliro omniorientational hexacopter. *IEEE Robotics and Automation Magazine*, 25(4):34 – 44, 2018.

[53] Alexandros Nikou, Georgios C. Gavridis, and Kostas J. Kyriakopoulos. Mechanical design, modelling and control of a novel aerial manipulator. *International Conference on Robotics and Automation (ICRA)*, pages 4699–4703, 2015.

[54] Dario Brescianini and Raffaello D'Andrea. An omni-directional multirotor vehicle. *Mechatronics*, 55:76–93, 2018.

[55] Hyunsoo Yang, Sangyul Park, Jeongseob Lee, Joonmo Ahn, Dongwon Son, and Dongjun Lee. Lasdra: Large-size aerial skeleton system with distributed rotor actuation. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7017–7023, 2018.

[56] Sangyul Park, Jeongseob Lee, Joonmo Ahn, Myungsin Kim, Jongbeom Her, Gi-Hun Yang, and Dongjun Lee. Odar: Aerial manipulation platform enabling omnidirectional wrench generation. *IEEE/ASME Transactions on Mechatronics*, 23(4):1907–1918, 2018.

[57] RCbenchmark. Rcbenchmark database. Last accessed: 18 Marts 2021.

[58] A. Alkamachi and E. Erçelebi. A proportional derivative sliding mode control for an overactuated quadcopter. *Journal of Aerospace Engineering*, 233(4):1354–1363, 2019.

[59] Leiba Rodman. *Topics in Quaternion Linear Algebra*. International series of monographs on physics. Princeton University Press, 2014.

[60] Moti Ben-Ari. A tutorial on euler angles and quaternions. Last accessed: 17/2-2021.

[61] Basile Graf. Quaternions and dynamics. *arXiv.org*, 2008.

[62] J. L. Merian, L.G. Kraige, and J.N. Bolton. *Engineering Mechanics Dynamics*, volume 2(8). WILEY, 2016.

[63] Dantam N.T. Practical exponential coordinates using implicit dual quaternions. *Algorithmic Foundations of Robotics XIII*, 14:639–655, 2020.

[64] A. Alaimo, V. Artale, C. Milazzo, A. Ricciardello, and L. Trefiletti. Mathematical modeling and control of a hexacopter. *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1043–1050, 2013.

[65] Antonio Matus-Vargas, Gustavo Rodriguez-Gomez, and Jose Martinez-Carranza. Ground effect on rotorcraft unmanned aerial vehicles: a review. *Intelligent Service Robotics*, 2021.

[66] Xiang He, Gordon Kou, Marc Calaf, and Kam K. Leang. In-ground-effect modeling and nonlinear-disturbance observer for multirotor unmanned aerial vehicle control. *Journal of Dynamic Systems, Measurement, and Control*, 141, 2019.

[67] Bosch Sensortec. Bno055: Intelligent 9-axis absolute orientation sensor, 2016. Available at: https://learn.adafruit.com/adafruit-bno055-absolute-orientation-sensor/downloads.

[68] intel. Intel® realsense™ tracking camera, 2019. Available at: https://www.intel.com/content/www/us/en/support/articles/000032422/emerging-technologies/intel-realsense-technology.html.

[69] intel. Intel® realsensetm product family d400 series, 2020. Available at: https://dev.intelrealsense.com/docs/intel-realsense-d400-series-product-family-datasheet.

[70] Christian Neil R. Katigbak, John Raymond B. Garcia, Jonn Edric D. Gutang, Jonathan B. De Villa, Ace Dominic C. Alcid, Ryan Rhay P. Vicerra, Angelo R. Dela Cruz, Edison A. Roxas, and Kanny Krizzy D. Serrano. Autonomous trajectory tracking of a quadrotor uav using pid controller. *IEEE International Conference Humanoid, Nanotechnology, Information Technology Communication and Control, Environment and Management*, 2015.

[71] Changlong Liu, Jian Pan, and Yufang Chang. Pid and lqr trajectory tracking control for an unmanned quadrotor helicopter: Experimental studies. *Chinese Control Conference*, 2016.

[72] Benoit Landry. Planning and control for quadrotor flight through cluttered environments. *Massachusetts Institute of Technology*, 2015.

[73] J. Xiong, N. Guo, Y. Hong, and E. Zheng. Improved position and attitude tracking control for a quadrotor uav. *2019 Chinese Automation Congress (CAC)*, pages 4197–4201, 2019.

[74] JIA SUN, YUANDA WANG, YAO YU, and CHANGYIN SUN. Nonlinear robust compensation method for trajectory tracking control of quadrotors. *IEEE Access*, 7:26766–26776, 2019.

[75] Markus Achtelik Mina Kamel, Kostas Alexis and Roland Siegwart. Fast nonlinear model predictive control for multicopter attitude tracking on so(3). *IEEE Conference on Control Applications*, 2015.

[76] Mohamed Nasr, Mostafa Ashraf, Mahmoud S. Hussein, Ahmad S. Salem, Catherine M. Elias, Omar M. Shehata, and Elsayed I. Morgan. A comparitive study on the control of uavs for trajectory tracking by mpc, smc, backstepping, and fuzzy logic controllers. *IEEE International Conference on Vehicular Electronics and Safety*, 2018.

[77] Davide Invernizzi Member IEEE, Mattia Giurato, Paolo Gattazzo, , and Marco Lovera Member IEEE. Comparison of control methods for trajectory tracking in fully actuated unmanned aerial vehicle. *IEEE Transactions on Control Systems Technology*, 2020.

[78] Tomoki Anzai, Moju Zhao, Masaki Murooka, Fan Shi, Kei Okada, and Masayuki Inaba. Design, modeling and control of fully actuated 2d transformable aerial robot with 1 dof thrust vectorable link module. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS*, 2019.

[79] Denis Kotarski, Petar Piljek, Hrvoje Brezak, and Josip Kasać. Design of a fully actuated passively tilted multirotor uav with decoupling control system. *International Conference on Mechanical and Aerospace Engineering*, 2017.

[80] Ramy Rashad, Federico Califano, and Stefano Stramigioli. Port-hamiltonian passivity-based control on se(3) of a fully actuated uav for aerial physical interaction near-hovering. *IEEE ROBOTICS AND AUTOMATION LETTERS, VOL. 4, NO. 4*, 2019.

[81] Maximilian Brunner, Karen Bodie, Mina Kamel, Michael Pantic, Weixuan Zhang, Juan Nieto, and Roland Siegwart. Trajectory tracking nonlinear model predictive control for an overactuated mav. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[82] Shakeeb Ahmad and Rafael Fierro. Real-time quadrotor navigation through planning in depth space in unstructured environments. *019 International Conference on Unmanned Aircraft Systems*, 2020.

[83] R. (Rens) Werink. On the control allocation of fully-actuated and over-actuated multirotor uavs. *University of Twente: Robotics and Mechatronics*, 2019.

[84] Guorong Wang, Yimin Wei, and Sanzheng Qiao. *Generalized Inverses: Theory and Computations*. Volume 53. Springer, 2018.

[85] Björn Wittenmark Karl J. Åström. *Computer controlled systems*. Prentice Hall, 3rd. edition, 1997.

[86] A. Radke and Zhiqiang Gao. A survey of state and disturbance observers for practitioners. *2006 American Control Conference*, page 6, 2006.

[87] Stanisław Konatowski, Piotr Kaniewski, and Jan Matuszewski. Comparison of estimation accuracy of ekf, ukf and pf filters. *Annual of navigation*, 23(1):69–87, 2016.

[88] M. Mallick, M. Morelande, and L. Mihaylova. Continuous-discrete filtering using ekf, ukf, and pf. *2012 15th International Conference on Information Fusion*, pages 1087–1094, 2012.

[89] Syamantak Datta Gupta, Jun Ye Yu, Mahendra Mallick, Mark Coates, and Mark Morelande. Comparison of angle-only filtering algorithms in 3d using ekf, ukf, pf, pff, and ensemble kf. *2015 18th International Conference on Information Fusion (Fusion)*, pages 1649–1656, 2015.

[90] Ting Yuan, Yaakov Bar-Shalom, Peter Willett, R. Ben-Dov, and S. Pollak. A comparison of multiple-imm estimation approaches using ekf, ukf, and pf for impact point prediction. *Signal and Data Processing of Small Targets 2014*, 9092, 2014.

[91] D. V. A. N. Ravi Kumar, S. Koteswara Rao, and K. Padma Raju. Estimate-merge-technique-based algorithms to track an underwater moving target using towed array bearing-only measurements. *Sādhanā*, 42:1617–1628, 2017.

[92] S. J. Julier and J. K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

[93] Sinan Xiao, Zhenzhou Lu, and Feifei Qin. Estimation of the generalized sobol's sensitivity index for multivariate output model using unscented transformation. *Journal of structural engineering*, 143(5):6016005, 2017.

[94] E. A. Wan and R. Van Der Merwe. The unscented kalman filter for nonlinear estimation. *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium*, pages 153–158, 2000.

[95] Gabriel A. Terejanu. Unscented kalman filter tutorial. *Department of Computer Science and Engineering University*, 2021.

[96] Rolf Isermann. *Fault-Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance*. Institut fur Automatisierungstechnik. Springer Link, 2006.

[97] Jesper Liniger. System diagnosis: Kalman filter and ekf. Lecture Slides, 2020.

[98] Jürgen Pannek Lars Grüne. *Nonlinear Model Predictive Control, Theory and Algorithms*. Communications and Control Engineering. Springer Link, 2016.

[99] Mathworks. Understanding model predictive control. Last accessed: 03 May 2021. https://www.mathworks.com/videos/understanding-model-predictive-contro-part-3-mpc-design-parameters-1530607670393.html.

[100] Aral Sarrafi, Zhu Mao, Christopher Niezrecki, and Peyman Poozesh. Vibration-based damage detection in wind turbine blades using phasebased motion estimation and motion magnification. *Journal of Sound and Vibration*, 2018.

[101] Intel RealSense. Intel realsense tracking camera t265 and depth cameras d400 series – better together. Last accessed: 24 May 2021. https://www.intelrealsense.com/depth-and-tracking-combined-get-started/.

[102] J. Jonkman, S. Butterfield, W. Musial, , and G. Scott. Definition of a 5-mw reference wind turbine for offshore system development. *Technical Report*, 2009.

[103] Qblade. Qblade: Wind turbine design and simulation, 2021. Last accessed: 5/3-2021.

[104] Eric A. Wan and Rudolph van der Merwe. *Kalman Filtering and Neural Networks*. Wiley, 2001.

[105] Kwang Woo Ahn and Kung-Sik Chan. On the convergence rate of the unscented transformation. *Annals of the Institute of Statistical Mathematics*, 65(5):889–912, 2013.

[106] Zhijian Qiu, Xiaomei Qu, Yue Wang, and Delfim F. M. Torres. An improved unscented kalman filter for discrete nonlinear systems with random parameters. *Discrete dynamics in nature and society*, pages 1–10, 2017.

[107] Sigurd Skogestad and Ian Postletwaite. *Multivariable Feedback Control*, volume 2. WILEY, 2010.

A | Assumptions

Assumption 1 *The inner structure of the WTB is a box spar and the cross section is assumed square.*

Assumption 2 *The bidirectional rotors have approximately double the response time when changing direction.*

Assumption 3 *The approximated trajectory already accounts for the WTB oscillations and they are modelled as sinusoidal for x and z.*

Assumption 4 *Attaining values for all entries in the CAM combined with a low condition number contribute to a more decoupled system.*

Assumption 5 *The force and torque generated from the rotors are proportional to the rotational speed squared.*

Assumption 6 *The input voltage for the rotor configuration is proportional to the rotational speed at settled conditions.*

Assumption 7 *The UAV is described through rigid body dynamics and thus effects from vibrations and jerk are assumed negligible.*

Assumption 8 *The moment of inertia can be described as a cuboid.*

Assumption 9 *The translational disturbances are only affecting in the global z direction.*

Assumption 10 *The ground and ceiling effect equations are formed from experimental data of a single rotor but assumed applicable for multi rotor configurations.*

Assumption 11 *The variance of the rotational disturbances are a factor of* 2% *of the maximum torque the rotors can provide.*

Assumption 12 *The depth camera can be used to increase accuracy of the tracking camera.*

Assumption 13 *Values of missing measurement noise variances are given a value.*

Assumption 14 *Measurements from the camera solution can be obtained with a frequency of* $100\,[Hz]$

Assumption 15 *The measurements of the BNO055 can be fused with the camera solutions measurement to improve state measurements.*

Assumption 16 *It is possible to use and implement an online trajectory generation algorithm that can identify the desired direction of flight and also provide estimates of a finite amount of future reference points in the WTB, by means of the candidate sensors.*

Assumption 17 *SISO closed loop bandwidths are assumed accurate for control design considerations.*

Assumption 18 *The controllers must be at least 10 times faster than the closed loop bandwidths.*

Assumption 19 *The simulation model is accurate to describe the UAVs dynamics and modern observer design can be applied such as the KF.*

Assumption 20 *As translational disturbance only affects the global z direction, disturbance estimation is obtainable from the z acceleration.*

Assumption 21 *The 2. order rotor model is considered sufficient for establishing prediction models for the observers and controllers.*

Assumption 22 *Warm start is chosen for the MPCs as the next optimum solution is assumed closely related to the current optimum.*

Assumption 23 *Combination of penalties for attitude MPC is sufficient as the influences are assumed close to equal.*

Assumption 24 *It is possible to obtain future reference points for trajectory tracking in a period of* $1.6\,[s]$ *at any given time step.*

B | Design Tables

This appendix presents table values which were utilized in the design process. Section B.1 holds the dimensions of the reference WTB. Further, Section B.2 shows the values of the CAM at different angles. Lastly, Section B.3 presents the rotor data of each of the rotor candidate which was investigated along with the efficiency at the optimal rotor speed at a given thrust.

B.1 WTB Dimensions

x	y	z
-0.633	10.60	1.85
-0.596	12.72	1.84
-0.593	14.84	1.83
-0.598	16.97	1.83
-0.601	19.09	1.82
-0.572	21.21	1.81
-0.578	23.33	1.80
-0.531	25.45	1.85
-0.485	27.57	1.88
-0.495	29.69	1.87
-0.507	31.81	1.88
-0.492	33.93	1.91
-0.504	36.05	1.91
-0.514	38.17	1.91
-0.525	40.29	1.91
-0.468	42.41	1.92
-0.481	44.53	1.92
-0.494	46.66	1.91
-0.508	48.78	1.91
-0.521	50.90	1.90
-0.535	53.02	1.90
-0.549	55.14	1.89
-0.568	57.26	1.88
-0.624	59.38	1.87

Table B.1: x, y and z coordinates for the center points of each WTB cross section.

Length $[m]$	**Twist** $[°]$	H_{cs} $[m]$
10.60	13.63	1.23
12.72	12.83	1.18
14.84	12.05	1.13
16.97	11.30	1.08
19.09	10.56	1.03
21.21	9.84	0.98
23.33	9.14	0.93
25.45	8.47	0.88
27.57	7.81	0.83
29.69	7.17	0.78
31.81	6.55	0.73
33.93	5.95	0.68
36.05	5.37	0.63
38.17	4.81	0.58
40.29	4.27	0.53
42.41	3.75	0.47
44.53	3.25	0.42
46.66	2.77	0.37
48.78	2.31	0.32
50.90	1.87	0.27
53.02	1.45	0.22
55.14	1.05	0.17
57.26	0.67	0.12
59.38	0.31	0.07

Table B.2: Dimensions of the WTB and the height of the quadratic tube. [102, 103]

B.2 Feasibility

Angle	v_x	v_y	v_z	v_ϕ	v_θ	v_ψ
$-90°$	6.71	2	3.87	1.08	1.64	2.72
$-80°$	6.71	1.97	3.81	1.08	1.61	2.68
$-70°$	6.71	1.88	3.64	1.08	1.54	2.55
$-60°$	6.71	1.94	3.35	1.08	1.42	2.35
$-50°$	6.71	2.49	2.97	1.08	1.75	2.08
$-40°$	6.71	2.97	2.49	1.08	2.08	1.75
$-30°$	6.71	3.35	1.94	1.08	2.35	1.42
$-20°$	6.71	3.64	1.88	1.08	2.55	1.54
$-10°$	6.71	3.81	1.97	1.08	2.68	1.61
$0°$	6.71	3.87	2	1.08	2.72	1.64
$10°$	6.71	3.81	1.97	1.08	2.68	1.61
$20°$	6.71	3.64	1.88	1.08	2.55	1.54
$30°$	6.71	3.35	1.94	1.08	2.35	1.42
$40°$	6.71	2.97	2.49	1.08	2.08	1.75
$50°$	6.71	2.49	2.97	1.08	1.75	2.08
$60°$	6.71	1.94	3.35	1.08	1.42	2.35
$70°$	6.71	1.88	3.64	1.08	1.54	2.55
$80°$	6.71	1.97	3.81	1.08	1.61	2.68
$90°$	6.71	2	3.87	1.08	1.64	2.72

Table B.3: The CAM at different angles with 0° as the initial design orientation.

B.3 Rotor Candidates

Mr Steele 2306 2345Kv with Carbon 5030			
RPM	kgf	N	Nm
0	0	0	0
4998	0,016	0,1568	0,00205
11783	0,092	0,9016	0,00995
16151	0,17	1,666	0,01729
19451	0,242	2,3716	0,02421
21741	0,3	2,94	0,02869
23413	0,348	3,4104	0,03337
25008	0,393	3,8514	0,03681
27239	0,463	4,5374	0,04257
28042	0,487	4,7726	0,04464
Hover	M_{eff}	Tot_{eff}	
4,13 $[N]$	62 [%]	59 [%]	
25953 $[RPM]$			

Hypertrain Blaster 2207 2450Kv Gemfan 5050BN			
RPM	***kgf***	***N***	***Nm***
0	0	0	0
8648	0,1176	1,15248	0,03702
11969	0,2254	2,20892	0,05208
14486	0,3336	3,26928	0,06884
16304	0,4243	4,15814	0,08323
17995	0,5101	4,99898	0,09467
19399	0,5867	5,74966	0,10628
20555	0,6642	6,50916	0,11901
21855	0,7421	7,27258	0,12951
24305	0,9179	8,99542	0,15172
25938	1,0483	10,27334	0,16818
Hover	M_{eff}	Tot_{eff}	
4,13 $[N]$	62 [%]	68 [%]	
19085 $[RPM]$			

DYS Samguk Shu HQ 5050			
RPM	***kgf***	***N***	***Nm***
0	0	0	0
6330	0,0696	0,68208	0,0104
7642	0,1053	1,03194	0,0142
8914	0,1445	1,4161	0,01821
10053	0,1862	1,82476	0,02236
11116	0,2336	2,28928	0,027
12207	0,2771	2,71558	0,0313
13283	0,3328	3,26144	0,03688
14290	0,388	3,8024	0,04251
15288	0,4462	4,37276	0,04852
16181	0,5019	4,91862	0,05424
16793	0,542	5,3116	0,05827
17680	0,6047	5,92606	0,06462
18498	0,6703	6,56894	0,07113
19360	0,7291	7,14518	0,07735
20090	0,7927	7,76846	0,08395
20878	0,8484	8,31432	0,09013
21611	0,9036	8,85528	0,09735
22178	0,9578	9,38644	0,10292
22776	1,0151	9,94798	0,10829
23468	1,077	10,5546	0,11512
Hover	M_{eff}	Tot_{eff}	
4,13 $[N]$	63 [%]	75 [%]	
13933 $[RPM]$			

Hypertrain Blaster 2207 2450Kv Gemfan 5040R			
RPM	***kgf***	***N***	***Nm***
0	0	0	0
8411	0,0785	0,7693	0,00952
9873	0,109	1,0682	0,01208
11015	0,134	1,3132	0,01415
12076	0,162	1,5876	0,01672
12884	0,185	1,813	0,01898
13712	0,212	2,0776	0,02124
14448	0,233	2,2834	0,02312
15152	0,256	2,5088	0,02544
15892	0,282	2,7636	0,02759
16343	0,301	2,9498	0,02936
16932	0,322	3,1556	0,03113
17439	0,343	3,3614	0,03291
17896	0,364	3,5672	0,0347
18355	0,384	3,7632	0,03633
18739	0,401	3,9298	0,0376
19151	0,415	4,067	0,03884
19537	0,427	4,1846	0,03992
19895	0,441	4,3218	0,04138
20192	0,458	4,4884	0,04295
20451	0,471	4,6158	0,04381
Hover	$\boldsymbol{M_{eff}}$	$\boldsymbol{Tot_{eff}}$	
4,13 $[N]$	72 [%]	76 [%]	
19332 $[RPM]$			

DYS Samguk Shu HQ 5045			
RPM	***kgf***	***N***	***Nm***
0	0	0	0
6507	0,0624	0,61152	0,01318
7878	0,0933	0,91434	0,01649
9199	0,131	1,2838	0,02039
10446	0,17	1,666	0,02417
11677	0,214	2,0972	0,02844
12778	0,258	2,5284	0,03242
13774	0,307	3,0086	0,03674
14950	0,363	3,5574	0,042
15940	0,415	4,067	0,04691
16744	0,462	4,5276	0,05128
17713	0,521	5,1058	0,05672
18583	0,58	5,684	0,06223
19565	0,65	6,37	0,06865
20469	0,717	7,0266	0,07489
21299	0,779	7,6342	0,08091
22187	0,836	8,1928	0,08718
22859	0,897	8,7906	0,09336
23526	0,951	9,3198	0,09843
24317	1,017	9,9666	0,1048
24994	1,081	10,5938	0,1107
Hover	$\boldsymbol{M_{eff}}$	$\boldsymbol{Tot_{eff}}$	
4,13 $[N]$	71 [%]	77 [%]	
13639 $[RPM]$			

C | Quaternions: Definition and Mathematics

A common method to describe rotation of UAVs in 3 dimensional space is to utilize Euler angles. The Euler angles are intuitively easy to understand but have some drawbacks as Gimbal lock resulting in loss of DOF and the usage of the nonlinear term sine and cosine. For the UAV to move freely in the 3 dimensional space one could use quaternions.
Quaternions are represented as 4 dimensional vectors consisting of a real part (scalar part) and 3 complex parts (vector part) as seen in Equation C.1.

$$q = q_0 + iq_1 + jq_2 + kq_3 \tag{C.1}$$

The coefficients i, j and k represent complex unit vectors describing the rotation in a 3 dimensional space illustrated in Figure C.1. The additional components of the quaternion q_i has to satisfy the normality condition as defined in Equation C.2 giving the coordinates for the 4 dimensional hyperspace.

$$q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1 \tag{C.2}$$

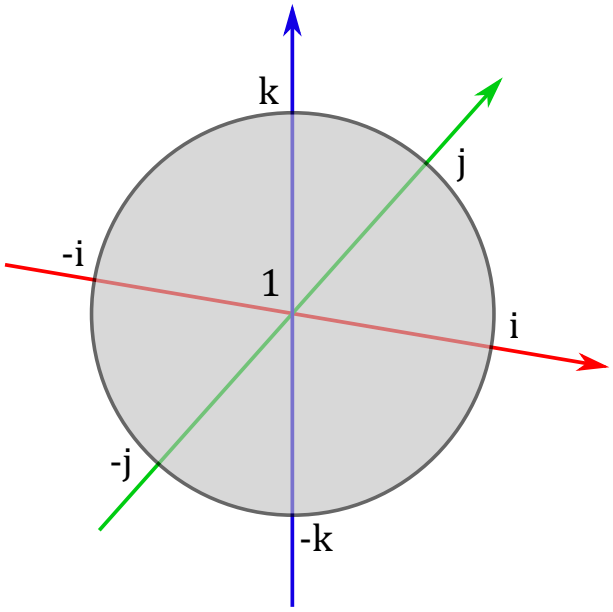


Figure C.1: 3 dimensional representation of the 4 dimensional quaternion vector.

At Figure C.1 the axes are seen as the complex i, j and k elements whereas the grey circle represents the unit sphere for pure rotation. Pure rotation is described by the real part of the quaternion being zero $q_0 = 0$ and the vector parts are satisfying the normality condition. At points within the unit sphere the real component attain a value between 0 and 1 with 1 being the center of the sphere. At points outside the unit sphere the real component attain a value between -1 and 0 where -1 is obtained by moving toward infinity at any direction in the complex 3 dimensional space. If the real part attain -1 or 1 no rotation occur.

To satisfy the normality condition, the quaternion has to be normalized for which the complex conjugated q_R^* is need. The complex conjugate is seen in Equation C.4.

$$q_R = q_0 + iq_1 + jq_2 + kq_3 \tag{C.3}$$
$$q_R^* = q_0 - iq_1 - jq_2 - kq_3 \tag{C.4}$$

The normalization is attained by dividing the quaternion by the norm of the quaternion calculated as in Equation C.5

$$|q_R| = \sqrt{q_R^* q_R} = \sqrt{(q_0 - iq_1 - jq_2 - kq_3)(q_0 + iq_1 + jq_2 + kq_3)} = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} \tag{C.5}$$

The multiplication of the complex components are following standard multiplication of complex number with $i^2 = -1$ but due to the quaternions are containing 3 different complex component additional products are introduced as stated in Equation C.6 and C.7. [59–61]

$$i^2 = j^2 = k^2 = ijk = -1 \tag{C.6}$$
$$ij = -ji = k\,, \quad jk = -kj = i\,, \quad ki = -ik = j \tag{C.7}$$

The multiplication rules can be visualized as seen in Figure C.2.

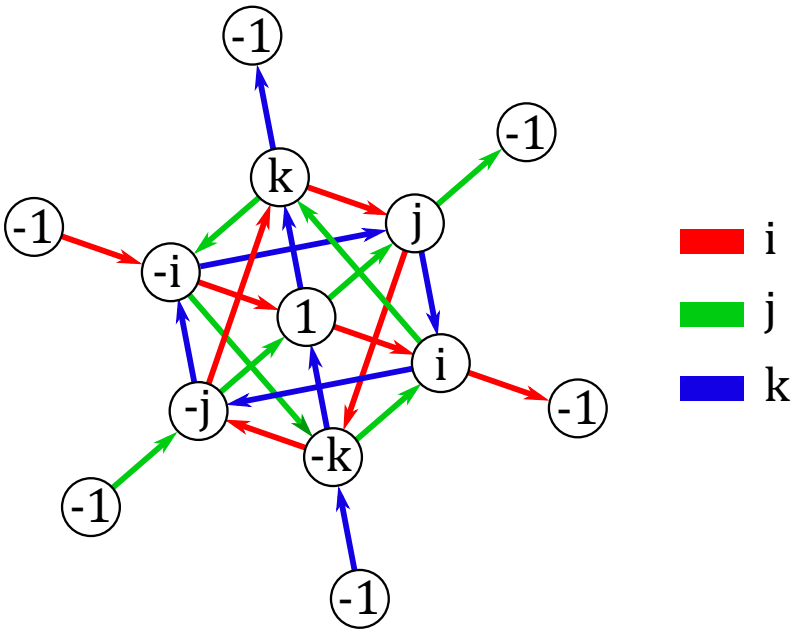


Figure C.2: Multiplication scheme of complex components. The arrows correspond to the direction in which the respective component is multiplied to the previous coefficient.

C.1 Rotational matrix

If q_R from Equation C.3 is a normalized quaternion the rotational matrix can be derived from q_R, q_R^* and a quaternion of pure rotation $q_A = 0 + ix + jy + kz$ as seen in Equation C.8.

$$\begin{aligned} q_B = q_R q_A q_R^* &= (q_0 + iq_1 + jq_2 + kq_3)(0 + ix + jy + kz)(q_0 - iq_1 - jq_2 - kq_3) \\ &= ((ix + jy + kz)q_0 + (-x + ky - jz)q_1 + (-kx - y + iz)q_2 \\ &\quad +(-jx - iy - z)q_3)(q_0 - iq_1 - jq_2 - kq_3) \\ &= (ix + jy + kz)q_0^2 + 2(ky - jz)q_0q_1 + 2(-kx + iz)q_0q_2 + 2(jx - iy)q_0q_3 \\ &\quad +(ix - jy - kz)q_1^2 + 2(jx + iy)q_1q_2 + 2(kx + iz)q_1q_3 + (-ix + jy - kz)q_2^2 \\ &\quad +2(ky + jz)q_2q_3 + (-ix - jy + kz)q_3^2 \end{aligned} \tag{C.8}$$

All of the real parts of Equation C.8 cancels out leaving only the imaginary parts. Equation C.8 is reshaped to matrix form as seen in Equation ... where the columns are related to x, y and z respectively and the rows are related to i, j and k respectively. [60]

$$q_B = q_R q_A q_R^* = \begin{bmatrix} (q_0^2 + q_1^2 - q_2^2 - q_3^2)ix & 2(q_1q_2 - q_0q_3)iy & 2(q_0q_2 + q_1q_3)iz \\ 2(q_0q_3 + q_1q_2)jx & (q_0^2 - q_1^2 + q_2^2 - q_3^2)jy & 2(q_2q_3 - q_0q_1)jz \\ 2(q_1q_3 - q_0q_2)kx & 2(q_0q_1 + q_2q_3)ky & (q_0^2 - q_1^2 - q_2^2 + q_3^2)kz \end{bmatrix} \tag{C.9}$$

C.2 Motor Mixing Algorithm

To define a unit vector in 3 dimensional space a gain of 1/sqrt(3) is multiplied to the coordinates [1; 1; 1]

$$uv = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \end{bmatrix} \tag{C.10}$$

$$|uv| = \sqrt{\frac{1}{\sqrt{3}^2} + \frac{1}{\sqrt{3}^2} + \frac{1}{\sqrt{3}^2}} = 1 \tag{C.11}$$

D | Simulation Results of the Mathematical Model

This appendix presents additional simulation results of the mathematical equation describing the UAVs motions by applying thrust and torque at the different DOF separately in open loop.

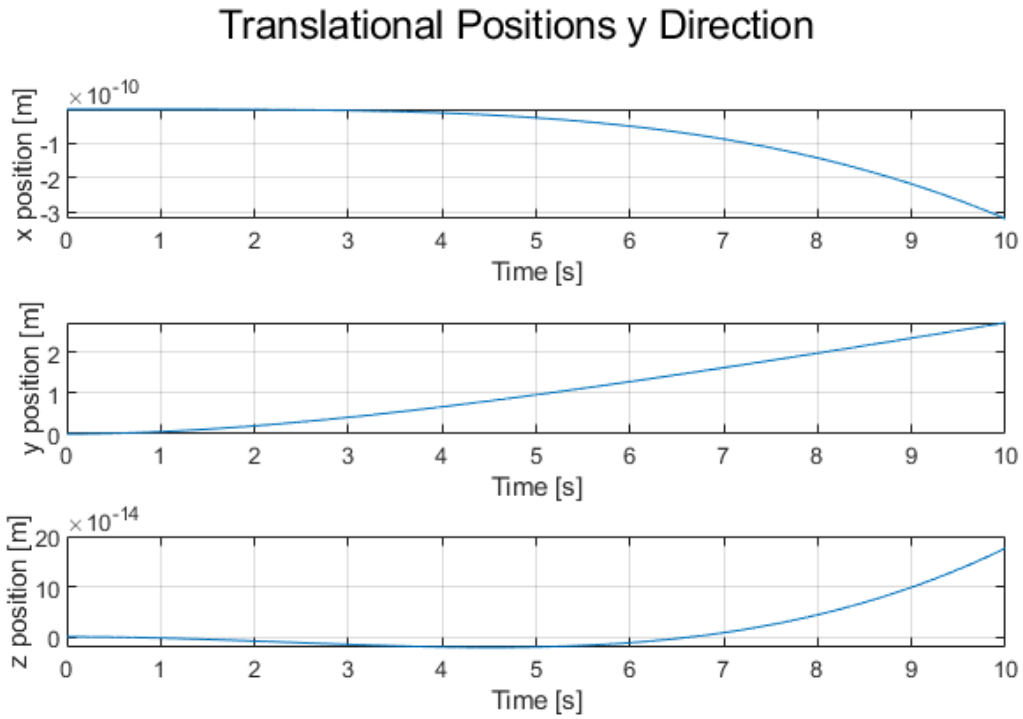


Figure D.1: Translational positions applying hovering thrust and $0.1N$ for positive y direction.

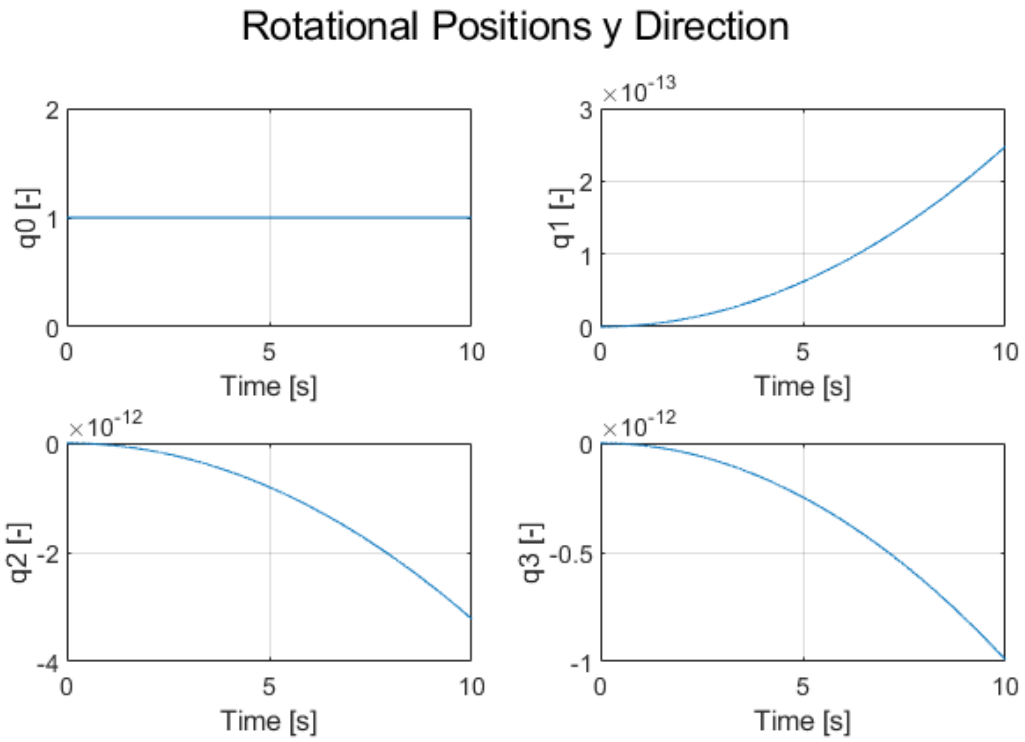


Figure D.2: Rotational positions applying hovering thrust and $0.1N$ for positive y direction.

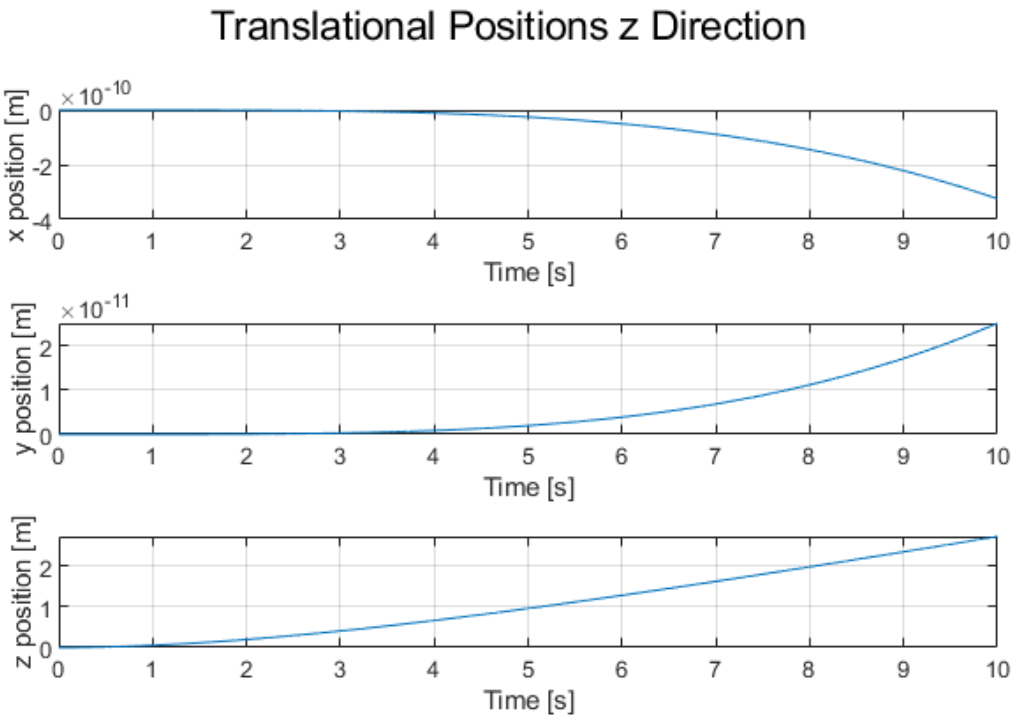


Figure D.3: Translational positions applying hovering thrust and $0.1N$ for positive z direction.

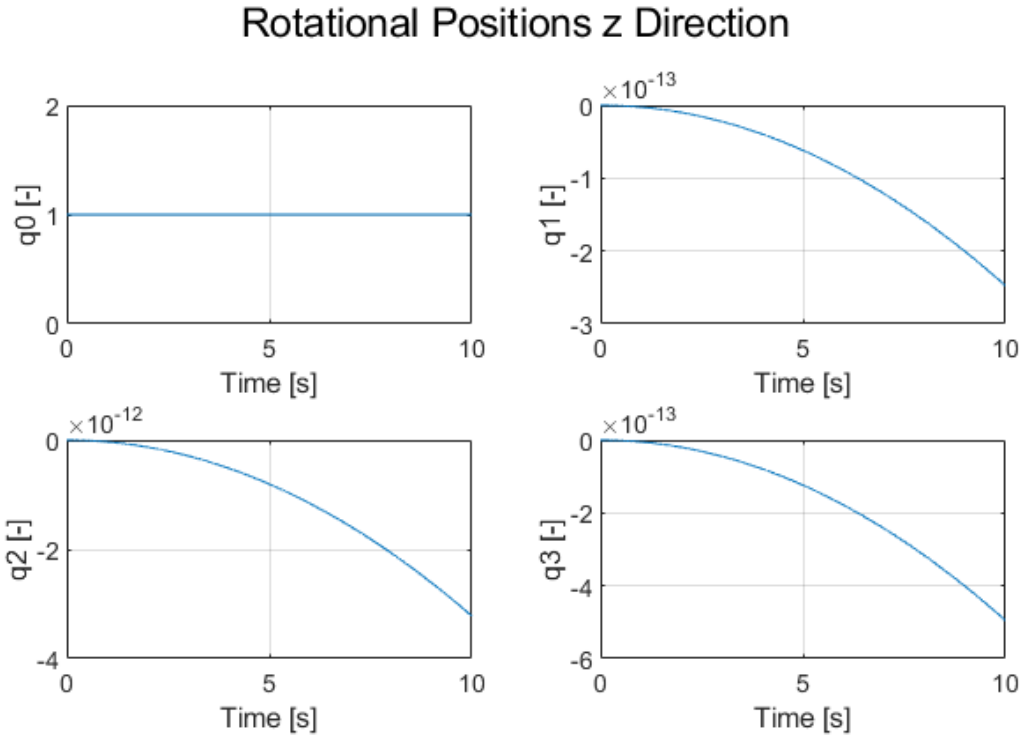


Figure D.4: Rotational positions applying hovering thrust and $0.1N$ for positive z direction.

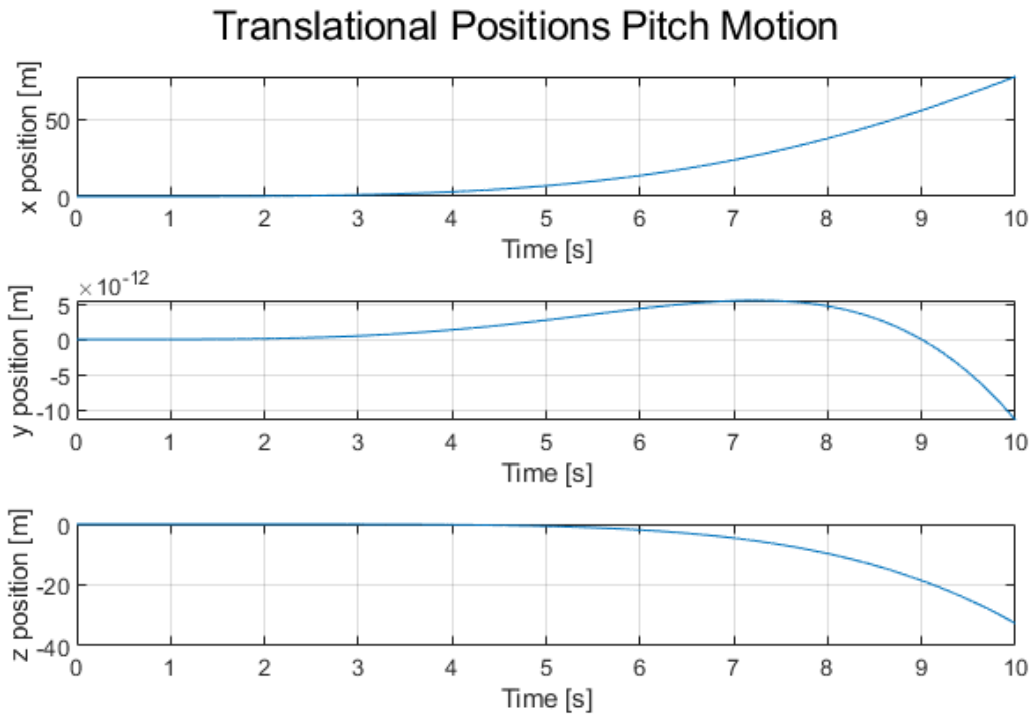


Figure D.5: Translational positions applying hovering thrust and $0.0001Nm$ for positive pitch.

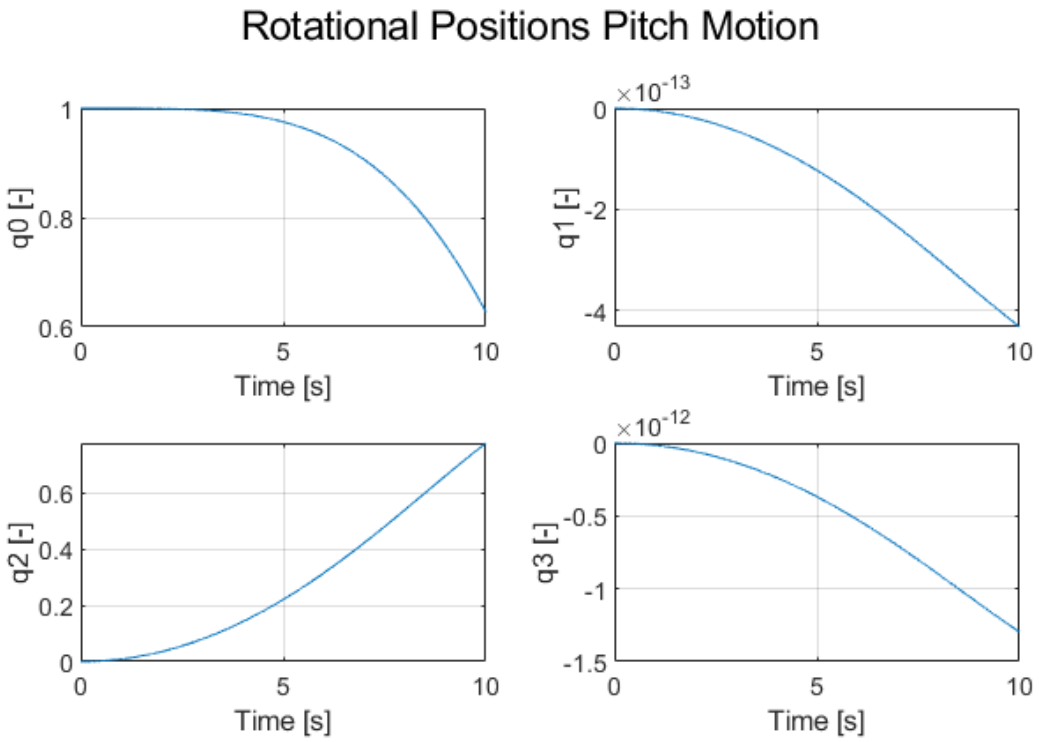


Figure D.6: Rotational positions applying hovering thrust and $0.0001Nm$ for positive pitch.

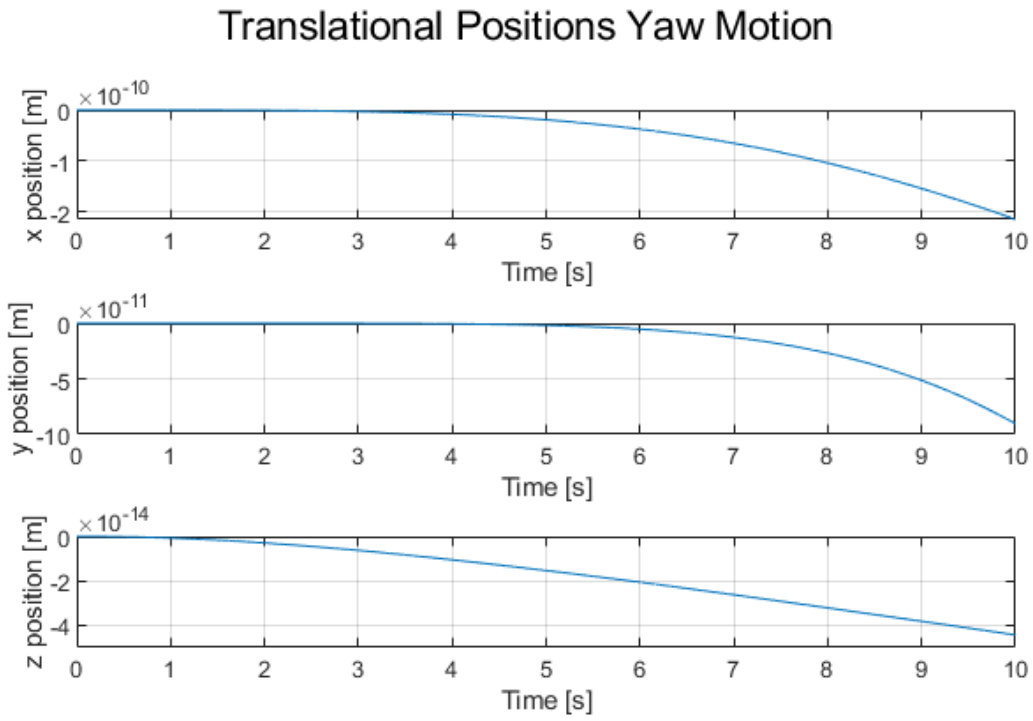


Figure D.7: Translational positions applying hovering thrust and $0.0001Nm$ for positive yaw.

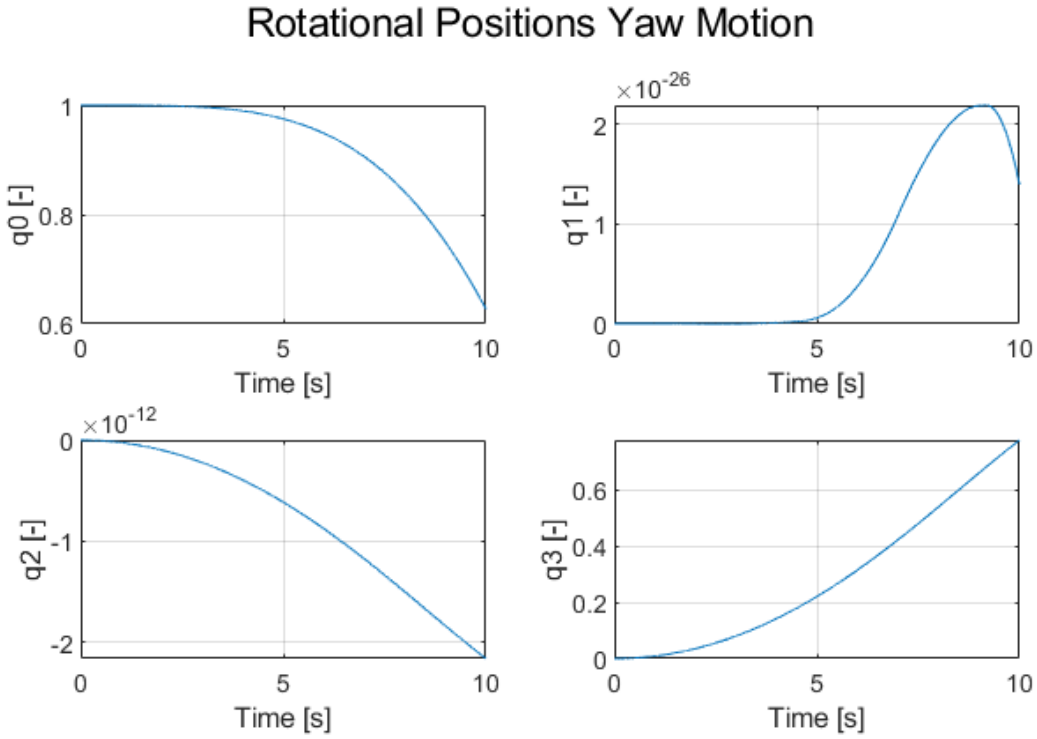


Figure D.8: Rotational positions applying hovering thrust and $0.0001Nm$ for positive yaw.

E | 2nd Order Accuracy of UKF

As formerly mentioned, the UKF can estimate the mean and covariance to the 2nd order, and this can be proven through the Taylor series expansion. At first, the real system can be approximated as in Equation E.1.

$$\overline{y} = f(\overline{x} + \delta x) = \sum_{n=0}^{\infty} \left[\frac{(\delta x \cdot \nabla_x)^n}{n!} f(x) \right]_{x=\overline{x}} \tag{E.1}$$

The expression with the partial derivatives is then defined as in Equation E.2.

$$D_{\delta x}^n = \frac{(\delta x \cdot \nabla_x)^n}{n!} \tag{E.2}$$

Substituting Equation E.2 into Equation E.1 the real system can be approximated as in Equation E.3.

$$\overline{y} = f(\overline{x}) + D_{\delta x} f(x) + D_{\delta x}^2 f(x) + D_{\delta x}^3 f(x) + \dots \tag{E.3}$$

Due to the symmetrical distribution of the states x, all odd terms become zero leaving the expression in Equation E.4.

$$\overline{y} = f(\overline{x}) + D_{\delta x}^2 f(x) + D_{\delta x}^4 f(x) + D_{\delta x}^6 f(x) + \dots \tag{E.4}$$

The covariance of the states can then be defined as in Equation E.5.

$$P_x = \delta x (\delta x)^T \tag{E.5}$$

Thereby the 2nd order term can be described as in Equation E.6.

$$D_{\delta x}^2 \triangleq \frac{\nabla_x^T \delta x (\delta x)^T \nabla_x}{2} = \frac{\nabla_x^T P_x \nabla_x}{2} \tag{E.6}$$

Equation E.4 can thereby be described as in Equation E.7.

$$\overline{y} = f(\overline{x}) + \frac{\nabla_x^T P_x \nabla_x}{2} f(x) + \mathbb{E}(D_{\delta x}^4 f(x) + D_{\delta x}^6 f(x) + \dots) \tag{E.7}$$

Here it is evident that the terms to a 2nd order degree is known whereas the higher order terms of 4th order and above is unknown variables contained as expected values in the Euclidean space $\mathbb{E}$. This proves the mean is correctly approximated to the 2nd order.

For the output covariance accuracy, the approximated output is defined as $\overline{y} = \mathbb{E}[y]$ meaning $\overline{y}$ is the expected value of y where y is defined in Equation E.8.

$$y = f(x) + D_{\delta x}f(x) + D^2_{\delta x}f(x) + D^3_{\delta x}f(x) + \dots \tag{E.8}$$

The output covariance can then be described as in Equation E.9.

$$P_y = \mathbb{E}[(y - \overline{y})(y - \overline{y})^T] = \mathbb{E}[y(y)^T] - \overline{y}(\overline{y})^T \tag{E.9}$$

By taking each term of Equation E.9 piecewise, Equation E.10 and E.11 are obtained.

$$y(y)^T = f(\overline{x}) + \nabla_x^T P_x \nabla_x^T + \mathbb{E}\left(\sum_{i=1}^{\infty}\sum_{j=1}^{\infty}\frac{1}{i!j!}D^i_{\delta x}f(x)(D^j_{\delta x})^T\right) \quad , \qquad i \neq j = 1 \tag{E.10}$$

$$\begin{aligned}\overline{y}(\overline{y})^T = {} & f(\overline{x}) + \frac{1}{4}[\nabla_x^T P_x \nabla_x^T][\nabla_x^T P_x \nabla_x^T]^T \\ & + \sum_{i=1}^{\infty}\sum_{j=1}^{\infty}\frac{1}{2i!2j!}\mathbb{E}[D^{2i}_{\delta x}f(x)]\mathbb{E}[(D^{2j}_{\delta x})^T] \quad , \qquad i \neq j = 1\end{aligned} \tag{E.11}$$

For the calculation of the term $\overline{y}(\overline{y})^T$ it is important to remember that odd terms are zero and thus neglected in the calculation. Thereby the output covariance is defined as in Equation E.12.

$$\begin{aligned}P_y = {} & \nabla_x^T P_x \nabla_x^T - \frac{1}{4}[\nabla_x^T P_x \nabla_x^T][\nabla_x^T P_x \nabla_x^T]^T + \mathbb{E}\left(\sum_{i=1}^{\infty}\sum_{j=1}^{\infty}\frac{1}{i!j!}D^i_{\delta x}f(x)(D^j_{\delta x})^T\right) \\ & - \sum_{i=1}^{\infty}\sum_{j=1}^{\infty}\frac{1}{2i!2j!}\mathbb{E}[D^{2i}_{\delta x}f(x)]\mathbb{E}[(D^{2j}_{\delta x})^T] \quad , \qquad i \neq j = 1\end{aligned} \tag{E.12}$$

From Equation E.12 all terms of 3rd order or higher is unknown and thus neglected giving a 2nd order accuracy as seen in Equation E.13.

$$P_y = \nabla_x^T P_x \nabla_x^T - \frac{1}{4}[\nabla_x^T P_x \nabla_x^T][\nabla_x^T P_x \nabla_x^T]^T \tag{E.13}$$

For the UT the proofs of the estimation accuracy are extended for each set of sigma points, but the conclusion of 2nd order accuracy is the same.[104–106]

F | SVD Design

The maximum CAM assure the most thrust and torque of the rotors are obtained. On the other side the given design only maximizing the CAM do not concern how much effort is required of each input to affect the outputs and thus an additional design is proposed by maximizing the CAM under a certain constraint of the singular values. Singular values give an idea of the direction from the inputs to the outputs by analyzing the system gains. For a singular-input singular-output (SISO) system represented by $y = Gu$ the gain can simply be calculated from the magnitude of the output and input as seen in Equation F.1.

$$\frac{|y(\omega)|}{|u(\omega)|} = \frac{|G(j\omega)u(\omega)|}{|u(\omega)|} = |G(j\omega)| \tag{F.1}$$

For the UAV with multiple inputs and multiple outputs the 2-norm can be utilized to get the magnitudes from which the gains can be determine as in Equation F.2.

$$\frac{||\boldsymbol{y}(\omega)||_2}{||\boldsymbol{u}(\omega)||_2} = \frac{||\boldsymbol{G}(j\omega)\boldsymbol{u}(\omega)||_2}{||\boldsymbol{u}(\omega)||_2} \tag{F.2}$$

The gains are frequency dependent as for the SISO case but too depends on the direction of the inputs as aforementioned. An intuitive understanding of the input directions can be illustrated graphically by a 2D MIMO system given 2 normalized inputs and 2 normalized outputs depicted in Figure F.1

At Figure F.1 the normalized input form the blue unit circle whereas the red ellipsis shows the effects of the inputs to the outputs. It is obvious that the output is highly affected by the input direction as it requires much more input to increase output 2 than it does for output 1. The smallest effect on the output is given at the smallest singular value $\underline{\sigma}(\boldsymbol{G})$ with the input direction of $u1 = 0$, $u2 = 1$ whereas the highest output is given at the highest singular value $\overline{\sigma}(\boldsymbol{G})$ with the input direction of $u1 = 1$, $u2 = 0$. The minimum and maximum singular values can be found by Equation F.3 and F.4

$$\max_{\boldsymbol{u}\neq\boldsymbol{0}} \frac{||\boldsymbol{G}\boldsymbol{u}||_2}{||\boldsymbol{u}||_2} = \max_{||\boldsymbol{u}||_2=1} ||\boldsymbol{G}\boldsymbol{u}||_2 = \overline{\sigma}(\boldsymbol{G}) \tag{F.3}$$

$$\min_{\boldsymbol{u}\neq\boldsymbol{0}} \frac{||\boldsymbol{G}\boldsymbol{u}||_2}{||\boldsymbol{u}||_2} = \min_{||\boldsymbol{u}||_2=1} ||\boldsymbol{G}\boldsymbol{u}||_2 = \underline{\sigma}(\boldsymbol{G}) \tag{F.4}$$

This skewed distribution of the singular values is to avoided if equal actuation is desired for all DOF thus for the second proposed design the optimization is still to maximize the CAM but

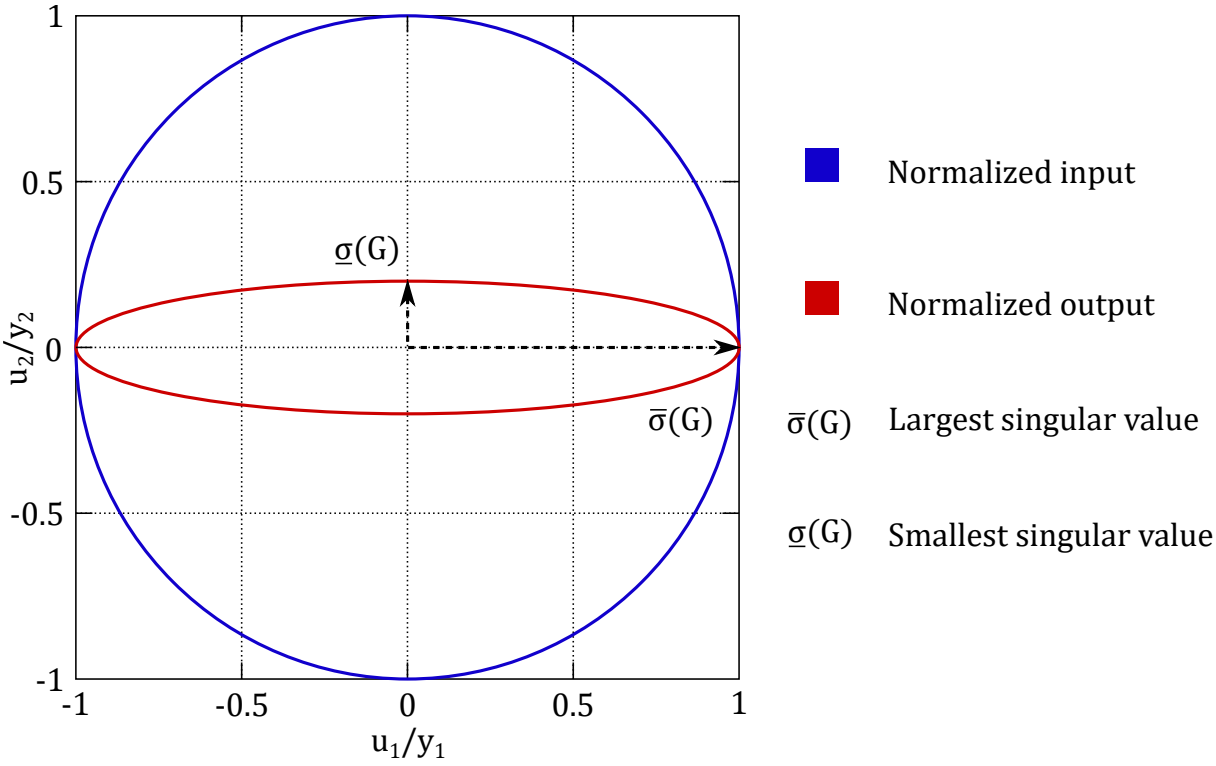


Figure F.1: Illustration of the input directions and definition of the smallest and largest singular values.[107]

under the constraint of the maximum singular value being no more than 3 times the minimum singular value. [107]

G | MATLAB Code

G.1 Attitude MPC MATLAB Code

In this Appendix the practical implementation of the developed attitude MPC controller in MATLAB is presented. It is a slightly simplified version as the MPC only executes a single prediction based on initial conditions.

By following instructions in this section it is possible to copy provided MATLAB codes directly into MATLAB and see the result of a single execution of the model predictive attitude controller. Note that 'Optimization Toolbox' is required.

Firstly the Main script must be saved in a MATLAB file.

Secondly the three function i.e. *MPC_ att.m*, *CostFcn_ att.m* and *SystemFcn_ att.m* must be saved in separate MATLAB files as well.

When this has been done, the Main script can be run. After the optimization is finished a plot will be printed of chosen predicted states and their corresponding references.

All aforementioned codes are given below:

Main Script:

```
% Matlab Code For a Single Execution of the Optimization Algorithm.
clc; clear all; close all;


%% Main Script (Functions: Sim_MPC_att , CostFcn_Sim_att, SystemFcn_Sim_att. Are called from the Main
    Script)

%Load Measuments x_0 = [p_0 q_0 r_0 q0_0 q1_0 q2_0 q3_0]^T
x0 = transpose([0.05 0 0 0.95 -0.05 0 0]); % Errors are introduced on p, q0, q1
x_ref = transpose([0 0 0 1 0 0 0]);

%Load Previously optimized input
u_prev = transpose([0 0 0]);

% Define Tuning Parameters
Ts_att = 0.0150; % Timestep
Np_att = 100;    % Pred. Horizon
Nc_att = 20;     % Ctrl Horizon
Nx_att = 7;      % #states
Ny_att = 3;      % #controlled outputs
Nu_att = 3;      % #inputs
Q_att = diag([67.473,67.473,67.473,55620*100,55620,55620,55620]); %Penalty on states
R_att = diag([1219.3,1219.3,1219.3]);                    %Penalty on inputs
R_attRate = diag([99642,99642,99642]);                   %Pen. on rate of change of inputs
LB_att = (-15);                                          %Inputs lower bound
UB_att = (15);                                           %Inputs upper bound
```

```
lb_att = repmat(LB_att, Nu_att*Nc_att, 1);              %Lower bound in respective dimensions
ub_att = repmat(UB_att, Nu_att*Nc_att, 1);              %Upper bound in respective dimensions



uk = u_prev; %3x1
u_init = repmat(uk, Nc_att,1); %making a Nu*Nc vector so fmincon can understand the input


%Repeating the konstant reference throughout the prediction horizon
xref = repmat(x_ref,1,Np_att);
u_ctrl = u_prev;

u_ctrl = MPC_att(lb_att, ub_att, Np_att, Nx_att, Nu_att, Ny_att, Nc_att, Ts_att, Q_att, R_att, u_init, uk,
     x0, xref, R_attRate);



%% Plotting
% As intermediate values cannot be accessed when parsing fmincon it is
% necessary to re-simulate the prediction model using the optimal control
% sequence, i.e. u_ctrl and inital conditions x0.

xk = x0;


for i = 1:Np_att
    if i <= Nc_att
        uk = u_ctrl(:,i);
    end
    xk = SystemFcn_att(xk, uk, Ts_att);
    xkn(:,i) = xk;
end

t = 0:Ts_att:Np_att*Ts_att-Ts_att;

% As a visual example p, q0, q1 are displayed in the figure
% along with their reference
figure(1)
subplot(3,1,1)
plot(t, xkn(1,:),'linewidth',1)
hold on
plot(t, xref(1,:),'linewidth',1)
ylabel('p body rate [rad/s]')
title('p Body Rate')
legend('p_{pred}','p_{ref}')

subplot(3,1,2)
plot(t, xkn(4,:),'linewidth',1)
hold on
plot(t, xref(4,:),'linewidth',1)
ylabel('quaternion q0 [-]')
title('q0 quaternion')
legend('q0_{pred}','q0_{ref}')

subplot(3,1,3)
plot(t, xkn(5,:),'linewidth',1)
hold on
plot(t, xref(5,:),'linewidth',1)
ylabel('quaternion q1 [-]')
xlabel('time [s]')
title('q1 quaternion')
legend('q1_{pred}','q1_{ref}')
```

MPC_att.m:

```
function u_ctrl = MPC_att(lb_att, ub_att, Np_att, Nx_att, Nu_att, Ny_att, Nc_att, Ts_att, Q_att, R_att,
    u_init, uk, x0, xref, R_attRate)

% Defining the TargetFcn
TargetFcn = @(u) CostFcn_att(u, Q_att, R_att, uk, x0, xref, Ts_att, Np_att, Nc_att, Nu_att, R_attRate);

% Setting Opti-options
options = optimoptions('fmincon','Algorithm','sqp','Display','off');

% Performing Optimization
u_vec = fmincon(TargetFcn, u_init, [], [], [], [], lb_att, ub_att, [], options);

% Reshaping Output into (Nu_att,Nc_att)-vector
u_ctrl = reshape(u_vec, Nu_att, Nc_att);
end
```

CostFcn_att.m:

```
function [J] = CostFcn_att(u, Q_att, R_att, uk, x0, xref, Ts_att, Np_att, Nc_att, Nu_att, R_attRate)

% Reshaping optimized variable u into (Nu_att,Nc_att)-vector
u_array = reshape(u, Nu_att, Nc_att);
% Defining initial Cost
J = 0;
%Loading initial conditions
xk = x0;
for i = 1:Np_att                        % Using the optimized inputs
    if i <= Nc_att
        uk = u_array(:,i);
    else
        uk = u_array(:,Nc_att);         % Keeping the last optimized input for the remainder of Np_att
    end
    xk =  SystemFcn_att(xk, uk, Ts_att); % Executing Prediction model
    x_er = xk - xref(:,i);
    du = uk - u_array(:,1);
    J = J + x_er'*Q_att*x_er + uk'*R_att*uk + du'*R_attRate*du; % Accumulating cost
end
end
```

SystemFcn_att.m

```
function xk = SystemFcn_att(x, uk, Ts_att)
% Unpacking states (x) to be compatible with system equations
p = x(1);
q = x(2);
r = x(3);
q0 = x(4);
q1 = x(5);
q2 = x(6);
q3 = x(7);

%Constants:
Ix = 0.0028;
J = [Ix 0 0; 0 Ix 0; 0 0 Ix];

% System equations:
w = transpose([p q r]);
dw = inv(J)*(uk-(cross(w,J*w))); % Rotational equations
w = w + dw*Ts_att;

dq0 = - p*q1 - q*q2 - q3*r;   % Quaternion equations
dq1 = p*q0 - q*q3 + q2*r;
dq2 = p*q3 + q*q0 - q1*r;
dq3 = q*q1 - p*q2 + q0*r;

q0 = q0 + dq0*Ts_att;
```

```
q1 = q1 + dq1*Ts_att;
q2 = q2 + dq2*Ts_att;
q3 = q3 + dq3*Ts_att;
xk = [w(1); w(2); w(3); q0; q1; q2; q3]; % Predicted states
end
```

G.1.1 C Code Generation of Attitude MPC

The code provided in this section allows to convert the three functions *MPC_att.m*, *CostFcn_att.m* and *SystemFcn_att.m* into C code.

It should be noted that when the C code has been successfully compiled, a new file will emerge in the workspace directory named *file_mex.mexw64*. In order to call the compiled function simply add *_mex* to you're function call.

```
% C Code Generation
clc; clear all; close all;

cfg = coder.config('mex');
cfg.IntegrityChecks = false;
cfg.SaturateOnIntegerOverflow = false;
cfg.ResponsivenessChecks = false;
cfg.EchoExpressions = false;

% Defining proper dimension for input parameters:
Np_att = 100;    % Pred Horizon
Nc_att = 20;     % Ctrl Horizon
Nx_att = 7;      % #states
Ny_att = 3;      % #controlled outputs
Nu_att = 3;      % Number of inputs

lb_att = zeros(Nu_att*Nc_att,1); % Assigning lower bound throughout control horizon
ub_att = zeros(Nu_att*Nc_att,1); % Assigning upper bound throughout control horizon
Ts_att = 0;                      % Discrete prediction time step
Q_att = zeros(7,7);              % State penalty
R_att = zeros(3,3);              % Input penalty
R_attRate = zeros(3,3);          % Rate of change input penalty
u_init = zeros(Nu_att*Nc_att,1); % Repeating initial guess on input throughout control horizon
uk = zeros(3,1);                 % Dimension of input vector
x0 = zeros(7,1);                 % Dimension of initial condition state vector
xref = zeros(Nx_att, Np_att);  % Dimension of the state reference repeated throughout prediction horizon

% C Code conversion arguments:
Argument = {lb_att ub_att Np_att Nx_att Nu_att Ny_att Nc_att Ts_att Q_att R_att u_init uk x0 xref
    R_attRate};

% Specifying the file to be converted:
codegen -config cfg MPC_att -args Argument
```

G.2 Position Tracking MPC

This section presents the MATLAB code used for the position tracking MPC. The 3 functions *Sim_MPCRe.m*, *CostFcn_Re.m* and *SystemFcn_Re.m* are utilized through the main script. The main script is the direct implementation within a MATLAB Function block in Simulink and thus has to be implemented in Simulink for a working example.

Main Script:

```
function [u_ctrl,u_m]  = fcn(xmeas, u_prev, u_m1, u_m2, t, yr, R, Q, Ts_MPC, Nc, Ny, Nu, Nx, Np, ub, lb,
    L, zdef, xdef, w_WTB, Wait, F_dist_z, R_du)
coder.extrinsic('Sim_MPCRe_mex');
```

```
% packing certain measurements from xmeas
x0 = transpose([xmeas(1) xmeas(2) xmeas(3) xmeas(4) xmeas(5) xmeas(6)]);
konst = transpose([xmeas(7) xmeas(8) xmeas(9) xmeas(10)]); %Used in SystemFcn

%Take care of the trajectory reference

uk = u_prev; %3x1

u_init = repmat(uk,Nc,1); %making a Nu*Nc vector so fmincon can understand the input

xx = zeros(1,Np);
yy = zeros(1,Np);
zz = zeros(1,Np);

for i = 1:Np %Final horizon or continous?
    if t < Wait
        xx(i) = 0;
        yy(i) = 0;
        zz(i) = 0;
    else

        yy(i) = 1*((t-Wait)+Ts_MPC/1000*i); % Try of final set point
        xx(i) = xdef/2*(yy(i)/L)^2*sin(w_WTB*((t-Wait)+Ts_MPC/1000*i)+pi/2);
        zz(i) = zdef/2*(yy(i)/L)^2*sin(w_WTB*((t-Wait)+Ts_MPC/1000*i));

    end
end

yref = [xx;yy;zz];

xref = [zeros(3,Np) ;yref];

u_ctrl = u_prev;
u_ctrl = Sim_MPCRe_mex(R, R_du, Q, Ts_MPC, Nc, Ny, Nu, Nx, Np, ub, lb, u_init, uk, x0, konst, xref,
     u_prev, u_m1, u_m2, F_dist_z);

T = Ts_MPC/1000;
omg = 19;
sigma = 1;
u_m = (omg^2*u_ctrl + (2/T^2 + (2*sigma*omg)/T)*u_m1 - 1/T^2 * u_m2)/(1/T^2 + (2*sigma*omg)/T + omg^2);
```

Sim_MPCRe.m:

```
function u_ctrl = Sim_MPCRe(R, R_du, Q, Ts_MPC, Nc, Ny, Nu, Nx, Np, ub, lb, u_init, uk, x0, konst, xref,
     u_prev, u_m1, u_m2, F_dist_z)
%% Defining the TargetFcn
TargetFcn = @(u) CostFcn_Re(u, Q, R, R_du, uk, x0, xref, konst, Ts_MPC, Np, Nc, Nu, u_prev, u_m1, u_m2,
     F_dist_z);
%% Calling Optimization
options = optimoptions('fmincon','Algorithm','sqp','Display','off');

u_vec = fmincon(TargetFcn, u_init, [], [], [], [], lb, ub, [], options);

u_ctr = reshape(u_vec, Nu, Nc);
u_ctrl = u_ctr(:,1);
end
```

CostFcn_Re.m:

```
function J = CostFcn_Re(u, Q, R, R_du, uk, x0, xref, konst, Ts_MPC, Np, Nc, Nu, u_prev, u_m1, u_m2,
     F_dist_z)
xk = x0;
u_array = reshape(u, Nu, Nc);
J = 0;
```

```
for i = 1:Np
    if i <= Nc
        uk = u_array(:,i);
    else
        uk = u_array(:,Nc);
    end
    xk = SystemFcn_Re(xk, uk, konst, Ts_MPC, u_prev, u_m1, u_m2, F_dist_z);
    du = uk - u_array(:,1);
    x_er = xk - xref(:,i);
    J = J + x_er'*Q*x_er + uk'*R*uk + du'*R_du*du;
end
end
```

SystemFcn_Re.m:

```
function xk = SystemFcn_Re(x, uk, konst,Ts_MPC, u_prev, u_m1, u_m2, F_dist_z)
q0 = konst(1);
q1 = konst(2);
q2 = konst(3);
q3 = konst(4);
g=9.8;
m=0.8422;
DampT=0.5;
%states[dx dy dz x y z]
Q_rot = [(q0^2+q1^2-q2^2-q3^2) 2*(q1*q2-q0*q3) 2*(q0*q2+q1*q3);
     2*(q1*q2+q0*q3) (q0^2-q1^2+q2^2-q3^2) 2*(q2*q3-q0*q1);
     2*(q1*q3-q0*q2) 2*(q0*q1+q2*q3) (q0^2-q1^2-q2^2+q3^2)];

tau = 0.15;
T = Ts_MPC;

%tau = 0.085;
T = Ts_MPC;
omg = 19;
sigma = 1;
%u_n = 1/(1+tau/T)*uk + 1/(1+T/tau)*u_prev; %First order motor response %approx
u_n = (omg^2*uk + (2/T^2 + (2*sigma*omg)/T)*u_m1 - 1/T^2 * u_m2)/(1/T^2 + (2*sigma*omg)/T + omg^2);
    %Second order motor response %approx

dx = [(1/m*(Q_rot*u_n + [0;0;F_dist_z]) - [0; 0; g] - 1/m*DampT*[x(1); x(2); x(3)]); x(1); x(2); x(3)];
xk = x + dx*Ts_MPC/1000;
end
```

G.3 Design Optimization Algorithm

This section presents the MATLAB code used for the UAV design procedure. The code is a working example, can be adjusted for desired criteria and prints out plots for the 3 2D planes together with a 3D plot of the final design of the UAV.

```
clc; clear all; close all;
Ite = 5;
Itn = Ite-1;
KF = 10.414;
Ktau = 0.1123;
kappa = Ktau/KF;
count = 0;
point = 0;
float = 0;

lrod = 1;

P1 = lrod/sqrt(3)*[1 -1 1 -1 1 -1 1 -1; 1 1 1 1 -1 -1 -1 -1; 1 1 -1 -1 -1 -1 1 1];

for ic = 1:1:Ite
```

```
c1 = (ic-1)/Itn;
c2 = c1;
for ia = 1:1:Ite
    b1max = sqrt(1-c1^2);
    b1 = b1max*(ia-1)/Itn;
    b2 = b1;
    a1 = sqrt(1-b1^2-c1^2);
    a2 = -a1;
    for jc = 1:1:Ite
        c3 = -(jc-1)/Itn;
        c4 = c3;
        for ja = 1:1:Ite
            b3max = sqrt(1-c3^2);
            b3 = b3max*(ja-1)/Itn;
            b4 = b3;
            a3 = sqrt(1-b3^2-c3^2);
            a4 = -a3;
            for kc = 1:1:Ite
                c5 = -(kc-1)/Itn;
                c6 = c5;
                for ka = 1:1:Ite
                    b5max = sqrt(1-c5^2);
                    b5 = -b5max*(ka-1)/Itn;
                    b6 = b5;
                    a5 = sqrt(1-b5^2-c5^2);
                    a6 = -a5;
                    for lc = 1:1:Ite
                        c7 = (lc-1)/Itn;
                        c8 = c7;
                        for la = 1:1:Ite
                            count = count+1;
                            Percent = round(count/Ite^8*100,4);
                            b7max = sqrt(1-c7^2);
                            b7 = -b7max*(la-1)/Itn;
                            b8 = b7;
                            a7 = real(sqrt(1-b7^2-c7^2));
                            a8 = -a7;
                            X1 = real([a1 a2 a3 a4 a5 a6 a7 a8; b1 b2 b3 b4 b5 b6 b7 b8; c1 c2 c3 c4 c5
                                 c6 c7 c8]);

                            M1 = [X1; cross(P1,X1)+kappa*X1];
                            SVD = svd(M1);
                            MIN = min(SVD);
                            MAX = max(SVD);
                            Val(count) = MAX/MIN;
                            Mabs = abs(M1);
                            MINx = min(Mabs(1,:));
                            MINy = min(Mabs(2,:));
                            MINz = min(Mabs(3,:));

                            if MAX < 7.5*MIN & MINx > 0.1 & MINy > 0 & MINz > 0
                                Minv = pinv(M1);
                                hoverDiff(:,count) = ones(8,1)*10.414/2-abs(Minv*[0;0;9.8*0.8422;0;0;0]);
                                if hoverDiff(:,count) < 2.5 & hoverDiff(:,count) > -2.5
                                    MAKS(count) = sum(sum(abs(M1)));
                                    if MAKS(count) == max(MAKS)
                                        Beta = M1;
                                        Xf1 = Beta(1:3,:);
                                        sing = SVD;
                                        Hover = hoverDiff(:,count);
                                    end
                                end
                            end
                        end
                    end
                end
            end
        end
```

```
            end
        end
        float = float + 1
    end
end
HOVER = hoverDiff(hoverDiff~=0);

%%
figure(1)
plot3([0 P1(1,1)],[0 P1(2,1)],[0 P1(3,1)],'k','linewidth',1.5);
hold on;
plot3([0 P1(1,2)],[0 P1(2,2)],[0 P1(3,2)],'k','linewidth',1.5);
hold on;
plot3([0 P1(1,3)],[0 P1(2,3)],[0 P1(3,3)],'k','linewidth',1.5);
hold on;
plot3([0 P1(1,4)],[0 P1(2,4)],[0 P1(3,4)],'k','linewidth',1.5);
hold on;
plot3([0 P1(1,5)],[0 P1(2,5)],[0 P1(3,5)],'k','linewidth',1.5);
hold on;
plot3([0 P1(1,6)],[0 P1(2,6)],[0 P1(3,6)],'k','linewidth',1.5);
hold on;
plot3([0 P1(1,7)],[0 P1(2,7)],[0 P1(3,7)],'k','linewidth',1.5);
hold on;
plot3([0 P1(1,8)],[0 P1(2,8)],[0 P1(3,8)],'k','linewidth',1.5);
hold on;
xlim([-2 2]);
ylim([-2 2]);
zlim([-2 2]);
arrow3([P1(1,1) P1(2,1) P1(3,1)],[P1(1,1)+Xf1(1,1) P1(2,1)+Xf1(2,1) P1(3,1)+Xf1(3,1)],'r')
hold on;
arrow3([P1(1,2) P1(2,2) P1(3,2)],[P1(1,2)+Xf1(1,2) P1(2,2)+Xf1(2,2) P1(3,2)+Xf1(3,2)],'r')
hold on;
arrow3([P1(1,3) P1(2,3) P1(3,3)],[P1(1,3)+Xf1(1,3) P1(2,3)+Xf1(2,3) P1(3,3)+Xf1(3,3)],'r')
hold on;
arrow3([P1(1,4) P1(2,4) P1(3,4)],[P1(1,4)+Xf1(1,4) P1(2,4)+Xf1(2,4) P1(3,4)+Xf1(3,4)],'r')
hold on;
arrow3([P1(1,5) P1(2,5) P1(3,5)],[P1(1,5)+Xf1(1,5) P1(2,5)+Xf1(2,5) P1(3,5)+Xf1(3,5)],'r')
hold on;
arrow3([P1(1,6) P1(2,6) P1(3,6)],[P1(1,6)+Xf1(1,6) P1(2,6)+Xf1(2,6) P1(3,6)+Xf1(3,6)],'r')
hold on;
arrow3([P1(1,7) P1(2,7) P1(3,7)],[P1(1,7)+Xf1(1,7) P1(2,7)+Xf1(2,7) P1(3,7)+Xf1(3,7)],'r')
hold on;
arrow3([P1(1,8) P1(2,8) P1(3,8)],[P1(1,8)+Xf1(1,8) P1(2,8)+Xf1(2,8) P1(3,8)+Xf1(3,8)],'r')
xlabel('x');
ylabel('y');
zlabel('z');
grid;

%%
figure(2)
subplot(1,3,1)
for i = 1:1:8
plot([0 P1(1,i)],[0 P1(2,i)],'k','linewidth',1);
hold on;
quiver(P1(1,i),P1(2,i),Xf1(1,i),Xf1(2,i),'r','linewidth',1);
hold on;
end
xlabel('x');
ylabel('y');
xlim([-1.5 1.5]);
ylim([-1.5 1.5]);
grid;
subplot(1,3,2)
for i = 1:1:8
plot([0 P1(1,i)],[0 P1(3,i)],'k','linewidth',1);
hold on;
quiver(P1(1,i),P1(3,i),Xf1(1,i),Xf1(3,i),'r','linewidth',1);
```

```
hold on;
end
xlabel('x');
ylabel('z');
xlim([-1.5 1.5]);
ylim([-1.5 1.5]);
grid;
subplot(1,3,3)
for i = 1:1:8
plot([0 P1(2,i)],[0 P1(3,i)],'k','linewidth',1);
hold on;
quiver(P1(2,i),P1(3,i),Xf1(2,i),Xf1(3,i),'r','linewidth',1);
hold on;
end
xlabel('y');
ylabel('z');
xlim([-1.5 1.5]);
ylim([-1.5 1.5]);
grid;
```

G.4 Ground/Ceiling Effect Code

This section presents the MATLAB coding of the ground and ceiling effects. This can be copied into a MATLAB Function block in Simulink.

```
function T = fcn(osci,Tn, z, y)

R = 0.0635;       %Propeller radius
a1 = 0.9;         %Tuning parameter
a2 = 0.5;         %Tuning parameter
rho = 1;          %Tuning parameter
Rge = 5*R;        %Ground effect kicks in
Rce = 5*R;        %Ceiling effect kicks in

z_wtbGrou = 0.012*y - 0.615 + osci(3,1);
z_wtb = -0.012*y + 0.615 + osci(3,1);

z_ref = 0;
z_p = z_ref + z;
z_e = -(z_ref - z_p); %

z1 = (z_wtbGrou - z_e)*-1;

if z1 > R/4
    z1 = z1;
else
    z1 = R/4 + 0.0000001; %Avoid singularity
end

z2 = z_wtb - z_e;

if z2 > 0
    z2 = z2;
else
    z2 = 0.001;
end

if z1 < Rge && z2 < Rce && z1 > 0 && z2 > 0                    %Both ground and ceiling effect
    T = 1/(((1-rho*(R/(4*z1))^2))*((1-rho*1/a1*(R/(a2+z2))^2)))*Tn;
elseif z1 < Rge && z1 > 0                                      %Only ground effect
    T = (1/(1-rho*(R/(4*z1))^2))*Tn;
elseif z2 < Rce && z2 > 0                                      %Only Ceiling effect
    T = (1/(1-rho*(1/a1*(R/(a2+z2))^2)))*Tn;
else
    T = Tn;                                                    %Neither
```

```
end
```

G.5 UKF Design

This section presents the MATLAB codes for the UKF structure together with the grid search algorithm established for the UKF tuning process. These are not stand alone scripts and have to be combined in the same manor as seen in Figure G.1. The grid search algorithm further requires a working simulation model to obtain data for the cost function.

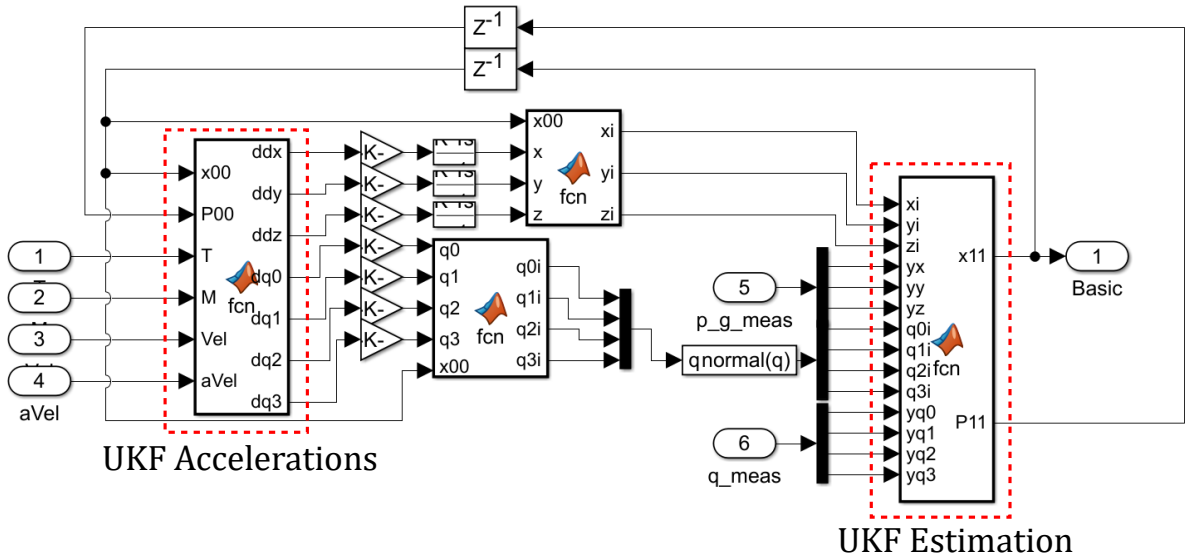


Figure G.1: Structure of the UKF within Simulink.

G.5.1 UKF Accelerations

```
function OUT = AccBasic_Try(x00,P00,T,M,m,g,Ixx,Iyy,Izz,DampT,Vel,aVel)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here

% Parameters
Sta = 7;
alpha = 10^(-4);  % The alpha value is often a small positive number
kappa = 0;        % The kappa value is often chosen as zero

Mx = M(1,1);
My = M(2,1);
Mz = M(3,1);

Velx = Vel(1,1);
Vely = Vel(2,1);
Velz = Vel(3,1);

% Weightings
gamma = alpha^2*(Sta+kappa) - Sta;
c = Sta+gamma;

% Delta for states
[V,D] = eig(c*P00);
dxp = real(V*(D.^(1/2))*inv(V));
dxn = -real(V*(D.^(1/2))*inv(V));

xi00 = zeros(length(x00),length(x00)*2+1);
dX = zeros(length(x00),length(x00)*2+1);
```

```
ddx00 = zeros(1,length(x00)*2+1);
ddy00 = zeros(1,length(x00)*2+1);
ddz00 = zeros(1,length(x00)*2+1);
dq000 = zeros(1,length(x00)*2+1);
dq100 = zeros(1,length(x00)*2+1);
dq200 = zeros(1,length(x00)*2+1);
dq300 = zeros(1,length(x00)*2+1);
dx00 = zeros(1,length(x00)*2+1);
dy00 = zeros(1,length(x00)*2+1);
dz00 = zeros(1,length(x00)*2+1);
dpw00 = zeros(1,length(x00)*2+1);
dqw00 = zeros(1,length(x00)*2+1);
drw00 = zeros(1,length(x00)*2+1);
for i = 1:1:2*Sta+1
    if i == 1
        xi00(:,i) = x00;
    else
        if i < Sta+2
            dX(:,i) = dxp(:,i-1);
        else
            dX(:,i) = dxn(:,i-Sta-1);
        end
        xi00(:,i) = x00 + dX(:,i);
    end

    % Calculate outputs from sigma points evaluated at initial state
    % prediction
    q0 = xi00(4,i);
    q1 = xi00(5,i);
    q2 = xi00(6,i);
    q3 = xi00(7,i);

    dx00(i) = Velx;
    dy00(i) = Vely;
    dz00(i) = Velz;

    dd_p = -g*[0; 0; 1] + inv(m)*[(q0^2+q1^2-q2^2-q3^2) 2*(q1*q2-q0*q3) 2*(q0*q2+q1*q3);
        2*(q1*q2+q0*q3) (q0^2-q1^2+q2^2-q3^2) 2*(q2*q3-q0*q1);
        2*(q1*q3-q0*q2) 2*(q0*q1+q2*q3) (q0^2-q1^2-q2^2+q3^2)]*T -
             1/m*DampT*eye(3)*[dx00(i);dy00(i);dz00(i)];

    ddx00(i) = dd_p(1,1);
    ddy00(i) = dd_p(2,1);
    ddz00(i) = dd_p(3,1);

    pw = aVel(1,1);
    qw = aVel(2,1);
    rw = aVel(3,1);

    dq000(i) = 0.5*[0 -pw -qw -rw]*[q0; q1; q2; q3];
    dq100(i) = 0.5*[pw 0 rw -qw]*[q0; q1; q2; q3];
    dq200(i) = 0.5*[qw -rw 0 pw]*[q0; q1; q2; q3];
    dq300(i) = 0.5*[rw qw -pw 0]*[q0; q1; q2; q3];

    dpw00(i) = Mx/Ixx + qw*rw*(Izz-Iyy)/Ixx;
    dqw00(i) = My/Iyy + pw*rw*(Ixx-Izz)/Iyy;
    drw00(i) = Mz/Izz + pw*qw*(Iyy-Ixx)/Izz;
end

ddx = transpose(ddx00);
ddy = transpose(ddy00);
ddz = transpose(ddz00);
dq0 = transpose(dq000);
dq1 = transpose(dq100);
dq2 = transpose(dq200);
dq3 = transpose(dq300);
OUT = [ddx ddy ddz dq0 dq1 dq2 dq3];
```

```
end
```

G.5.2 UKF Estimation

```
function OUT = UKFbasic_Try(xi,yi,zi,yx,yy,yz,q0i,q1i,q2i,q3i,yq0,yq1,yq2,yq3,R_ukf,Q_ukf)
%UNTITLED4 Summary of this function goes here
%   Detailed explanation goes here

% Parameters
Sta = 7;
OP = 7;

% Tuning parameters
alpha = 10^(-4);       % The alpha value is often a small positive number
kappa = 0;         % The kappa value is often chosen as zero
beta = 2;          % For Gaussian disturbuted noise beta is optimal at a value of 2

% Weightings
gamma = alpha^2*(Sta+kappa) - Sta;
Wm = gamma/(Sta+gamma);
Wc = Wm + (1-alpha^2+beta);
W = 1/(2*(Sta+gamma));
c = Sta+gamma;

xh00 = zeros(Sta,1);
yh00 = zeros(OP,1);
Wi = zeros(1,Sta*2+1);
xi00 = zeros(Sta,Sta*2+1);
yi00 = zeros(OP,Sta*2+1);

for i = 1:1:Sta*2+1
    if i == 1
        Wi(i) = Wm;
    else
        if i < Sta+2
            Wi(i) = W;
        else
            Wi(i) = W;
        end
    end
    xi00(:,i) = [xi(i); yi(i); zi(i); q0i(i); q1i(i); q2i(i); q3i(i)];
    yi00(:,i) = [xi(i); yi(i); zi(i); q0i(i); q1i(i); q2i(i); q3i(i)];
    xh00 = xh00 + Wi(i)*xi00(:,i);
    yh00 = yh00 + Wi(i)*yi00(:,i);
end

PH00 = zeros(Sta,Sta);
PY = zeros(OP,OP);
PXY = zeros(Sta,OP);
Wj = zeros(1,Sta*2+1);
Ph00 = zeros(Sta,Sta,Sta*2+1);
Py = zeros(OP,OP,Sta*2+1);
Pxy = zeros(Sta,OP,Sta*2+1);
for j = 1:1:2*Sta+1
    if j == 1
        Wj(j) = Wc;
    else
        Wj(j) = W;
    end
    Ph00(:,:,j) = Wj(j)*(xi00(:,j)-xh00)*transpose(xi00(:,j)-xh00);
    PH00 = PH00 + Ph00(:,:,j);
    Py(:,:,j) = Wj(j)*(yi00(:,j)-yh00)*transpose(yi00(:,j)-yh00);
    PY = PY + Py(:,:,j);
    Pxy(:,:,j) = Wj(j)*(xi00(:,j)-xh00)*transpose(yi00(:,j)-yh00);
    PXY = PXY + Pxy(:,:,j);
end
```

```
Ph0 = PH00 + Q_ukf;
Pyy = PY + R_ukf;

K = PXY*inv(Pyy);
P11 = Ph0 - K*Pyy*transpose(K);
y = [yx;yy;yz;yq0;yq1;yq2;yq3];
res = y - yh00;
x11 = xh00 + K*res;

OUT = [x11 P11];
```

G.5.3 UKF Grid Search Algorithm

```
count = 0;
opti = 5;
for j1 = 1:opti
    K1 = 10^(-1+j1);
    for j2 = 1:opti
        K2 = 10^(-1+j2);
        for j3 = 1:opti
            count = count + 1
            K3 = 10^(-1+j3);
            R_ukf = [eye(3)*K1 zeros(3,4); zeros(4,3) eye(4)*K2];
            Q_ukf = [eye(3)*10^(0) zeros(3,4); zeros(4,3) eye(4)*K3];
            sim('Full_Model_try');

            xr = ans.UKFBreadings.Data(:,1);
            xe = ans.UKFBreadings.Data(:,8);
            yr = ans.UKFBreadings.Data(:,2);
            ye = ans.UKFBreadings.Data(:,9);
            zr = ans.UKFBreadings.Data(:,3);
            ze = ans.UKFBreadings.Data(:,10);
            q0r = ans.UKFBreadings.Data(:,4);
            q0e = ans.UKFBreadings.Data(:,11);
            q1r = ans.UKFBreadings.Data(:,5);
            q1e = ans.UKFBreadings.Data(:,12);
            q2r = ans.UKFBreadings.Data(:,6);
            q2e = ans.UKFBreadings.Data(:,13);
            q3r = ans.UKFBreadings.Data(:,7);
            q3e = ans.UKFBreadings.Data(:,14);

            for i = 1:1:length(xr)
                Dang(i) = (xr(i) - xe(i))^2 + (yr(i) - ye(i))^2 + (zr(i) - ze(i))^2 + (q0r(i) - q0e(i))^2 +
                     (q1r(i) - q1e(i))^2 + (q2r(i) - q2e(i))^2 + (q3r(i) - q3e(i))^2;
            end
            SUM(count) = sum(Dang);
            Penalty(count,:) = [K1 K2 K3];
            Iterations(count,:) = [j1-1 j2-1 j3-1];
        end
    end
end
for l = 1:length(SUM)
    if SUM(l) == min(SUM)
        S = SUM(l);
        P = Penalty(l,:);
        ite = Iterations(l,:);
    end
end
%%
count2 = 0;
opti = 5;
for mm1 = 1:opti
    K1 = 10^(ite(1,1)-0.5+(mm1-1)/4);
    for mm2 = 1:opti
        K2 = 10^(ite(1,2)-0.5+(mm2-1)/4);
        for mm3 = 1:opti
            count2 = count2 + 1
```

```
            K3 = 10^(ite(1,3)-0.5+(mm3-1)/4);
            R_ukf = [eye(3)*K1 zeros(3,4); zeros(4,3) eye(4)*K2];
            Q_ukf = [eye(3)*10^(0) zeros(3,4); zeros(4,3) eye(4)*K3];
            sim('Full_Model_try');

            xr = ans.UKFBreadings.Data(:,1);
            xe = ans.UKFBreadings.Data(:,8);
            yr = ans.UKFBreadings.Data(:,2);
            ye = ans.UKFBreadings.Data(:,9);
            zr = ans.UKFBreadings.Data(:,3);
            ze = ans.UKFBreadings.Data(:,10);
            q0r = ans.UKFBreadings.Data(:,4);
            q0e = ans.UKFBreadings.Data(:,11);
            q1r = ans.UKFBreadings.Data(:,5);
            q1e = ans.UKFBreadings.Data(:,12);
            q2r = ans.UKFBreadings.Data(:,6);
            q2e = ans.UKFBreadings.Data(:,13);
            q3r = ans.UKFBreadings.Data(:,7);
            q3e = ans.UKFBreadings.Data(:,14);

            for i = 1:1:length(xr)
                Dang2(i) = (xr(i) - xe(i))^2 + (yr(i) - ye(i))^2 + (zr(i) - ze(i))^2 + (q0r(i) - q0e(i))^2
                    + (q1r(i) - q1e(i))^2 + (q2r(i) - q2e(i))^2 + (q3r(i) - q3e(i))^2;
            end
            SUM2(count2) = sum(Dang2);
            Penalty2(count2,:) = [K1 K2 K3];
            Iterations2(count2,:) = [ite(1,1)-0.5+(mm1-1)/4 ite(1,2)-0.5+(mm2-1)/4 ite(1,3)-0.5+(mm3-1)/4];
        end
    end
end
for l = 1:length(SUM2)
    if SUM2(l) == min(SUM2)
        S2 = SUM2(l);
        P2 = Penalty2(l,:);
        ite2 = Iterations2(l,:);
    end
end

count3 = 0;
opti = 5;
for nn1 = 1:opti
    K1 = 10^(ite2(1,1)-0.2+(nn1-1)/10);
    for nn2 = 1:opti
        K2 = 10^(ite2(1,2)-0.2+(nn2-1)/10);
        for nn3 = 1:opti
            count3 = count3 + 1
            K3 = 10^(ite2(1,3)-0.2+(nn3-1)/10);
            R_ukf = [eye(3)*K1 zeros(3,4); zeros(4,3) eye(4)*K2];
            Q_ukf = [eye(3)*10^(0) zeros(3,4); zeros(4,3) eye(4)*K3];
            sim('Full_Model_try');

            xr = ans.UKFBreadings.Data(:,1);
            xe = ans.UKFBreadings.Data(:,8);
            yr = ans.UKFBreadings.Data(:,2);
            ye = ans.UKFBreadings.Data(:,9);
            zr = ans.UKFBreadings.Data(:,3);
            ze = ans.UKFBreadings.Data(:,10);
            q0r = ans.UKFBreadings.Data(:,4);
            q0e = ans.UKFBreadings.Data(:,11);
            q1r = ans.UKFBreadings.Data(:,5);
            q1e = ans.UKFBreadings.Data(:,12);
            q2r = ans.UKFBreadings.Data(:,6);
            q2e = ans.UKFBreadings.Data(:,13);
            q3r = ans.UKFBreadings.Data(:,7);
            q3e = ans.UKFBreadings.Data(:,14);

            for i = 1:1:length(xr)
```

```
            Dang3(i) = (xr(i) - xe(i))^2 + (yr(i) - ye(i))^2 + (zr(i) - ze(i))^2 + (q0r(i) - q0e(i))^2
                + (q1r(i) - q1e(i))^2 + (q2r(i) - q2e(i))^2 + (q3r(i) - q3e(i))^2;
        end
        SUM3(count3) = sum(Dang3);
        Penalty3(count3,:) = [K1 K2 K3];
        Iterations3(count3,:) = [ite2(1,1)-0.2+(nn1-1)/10 ite2(1,2)-0.2+(nn2-1)/10
            ite2(1,3)-0.2+(nn3-1)/10];
    end
  end
end
for l = 1:length(SUM3)
  if SUM3(l) == min(SUM3)
    S3 = SUM3(l);
    P3 = Penalty3(l,:);
    ite3 = Iterations3(l,:);
  end
end
```

G.6 Bayesian Optimization

This appendix contains a BO test code, as well as the codes used for tuning the attitude and position MPC controllers.

G.6.1 Bayesian Optimization Test Code

The BO test code can be copied directly into MATLAB and executed. The results of running the test code is a plot of how the objective function is minimized. It should be noted that the training data provided for this code is randomly generated and the only purpose of this code is to provide an understanding of how BO for controller tuning was implemented.

```
% Load training data (In this example we generate random data)
epse = rand(10,7); % 7 tuning parameters and 10 experiments.
JJJ = rand(10,1); % Values of cost for each experiment.

% Fit a random gaussian process model
gprMdl = fitrgp(epse,JJJ,'KernelFunction','squaredexponential'); %Also called a surrogate model

% Define desired variables to be optimized and the span of values they are allowed to attain
Ts_att = optimizableVariable('Ts',[0 1],'Type','integer'); % Also reffered to as hyperparameters
Np_att = optimizableVariable('Np',[0 1],'Type','integer');
Nc_att = optimizableVariable('Nc',[0 1],'Type','integer');
Q_p = optimizableVariable('Qp',[0 1],'Type','real');
Q_w = optimizableVariable('Qw',[0 1],'Type','real');
R_M = optimizableVariable('RM',[0 1],'Type','real');
R_dM = optimizableVariable('RdM',[0 1],'Type','real');
vars = [Ts_att Np_att Nc_att Q_p Q_w R_M R_dM];

% Perform bayesian optimization
bayesObject = bayesopt(@(tbl)mdlfun(tbl,gprMdl),vars,'Verbose',1,...
    'AcquisitionFunctionName','expected-improvement-plus',...
    'PlotFcn',@plotMinObjective);

% The following function utilizes the new guess of hyperparameters given from the BO to predict the
    corresponding cost f.
function f = mdlfun(tbl,gprMdl) % By default this executes 30 times
Ts_att = tbl.Ts;
Np_att = tbl.Np;
Nc_att = tbl.Nc;
Q_p = tbl.Qp;
Q_w = tbl.Qw;
R_M = tbl.RM;
R_dM = tbl.RdM;
vars = [Ts_att Np_att Nc_att Q_p Q_w R_M R_dM];
```

```
f = predict(gprMdl,vars);
end
```

G.6.2 Bayesian Optimization Attitude MPC

```
clc; clear all; close all;
Count = 0;
Samples = 10;
for i = 1:Samples
    Ts_att = round(rand*8)+8; % Must be lesser than 0.018 [0.008 0.01 0.012 0.014 0.016]
    Np_att = round(rand*5)+6; % Between 6 and 15
    Nc_att = round(rand*3)+2; % Between 2 and 5
    Q_p = rand*9+1;
    Q_w = 10^(rand*2+3);
    Q_att = diag([Q_p,Q_p,Q_p,Q_w,Q_w,Q_w,Q_w]);
    R_M = 10^(rand*2+3);
    R_dM = 10^(rand*2+3);
    R_att = diag([R_M,R_M,R_M]);
    R_attRate = diag([R_dM,R_dM,R_dM]);
    eps(i,:) = [Ts_att Np_att Nc_att Q_p Q_w R_M R_dM];
    sim('Full_Model_Final_Att',10);
    Cost(:,:) = [ans.Opt.Data(:,:) ans.In.Data(:,:)];
    for ii = 1:length(Cost(:,1,1))
        Jcost(ii) = (1-Cost(ii,1))^2 + Cost(ii,2)^2 + Cost(ii,3)^2 + Cost(ii,4)^2 + Cost(ii,5)^2 +
            Cost(ii,6)^2 + Cost(ii,7)^2;
    end
    Jc(i,1) = sum(Jcost);
    Count = Count + 1
end
%%
epse = eps;
JJJ = Jc;
for jj = 1:length(Jc)
    if Jc(jj) > 10^(5)
        epse(jj,:) = zeros(1,width(eps));
        JJJ(jj,1) = 0;
    else
        epse(jj,:) = eps(jj,:);
        JJJ(jj,1) = Jc(jj,1);
    end
end
epse(epse==0) = [];
epse = reshape(epse,length(epse)/7,7);
JJJ = JJJ(JJJ~=0);

gprMdl = fitrgp(epse,JJJ,'KernelFunction','squaredexponential');

Ts_att = optimizableVariable('Ts',[8,16],'Type','integer');
Np_att = optimizableVariable('Np',[6,11],'Type','integer');
Nc_att = optimizableVariable('Nc',[2,5],'Type','integer');
Q_p = optimizableVariable('Qp',[10^(0),10^(2)],'Type','real');
Q_w = optimizableVariable('Qw',[10^(3),10^(5)],'Type','real');
R_M = optimizableVariable('RM',[10^(3),10^(5)],'Type','real');
R_dM = optimizableVariable('RdM',[10^(3),10^(5)],'Type','real');
vars = [Ts_att Np_att Nc_att Q_p Q_w R_M R_dM];

bayesObject = bayesopt(@(tbl)mdlfun(tbl,gprMdl),vars,'Verbose',1,...
    'AcquisitionFunctionName','expected-improvement-plus',...
    'PlotFcn',@plotMinObjective);

X_best = [bayesObject.XAtMinEstimatedObjective.Ts bayesObject.XAtMinEstimatedObjective.Np
     bayesObject.XAtMinEstimatedObjective.Nc bayesObject.XAtMinEstimatedObjective.Qp
     bayesObject.XAtMinEstimatedObjective.Qw bayesObject.XAtMinEstimatedObjective.RM
     bayesObject.XAtMinEstimatedObjective.RdM]
Ts_att = X_best(1,1);
Np_att = X_best(1,2);
```

```
Nc_att = X_best(1,3);
Q_p = X_best(1,4);
Q_w = X_best(1,5);
Q_att = diag([Q_p,Q_p,Q_p,Q_w,Q_w,Q_w,Q_w]);
R_M = X_best(1,6);
R_dM = X_best(1,7);
R_att = diag([R_M,R_M,R_M]);
R_attRate = diag([R_dM,R_dM,R_dM]);
sim('Full_Model_Final_Att');
Data(:,:) = [ans.Opt.Data(:,:) ans.In.Data(:,:)];
%%
figure(10)
subplot(2,2,1)
plot(1:length(Data(:,1)),Data(:,1))
subplot(2,2,2)
plot(1:length(Data(:,2)),Data(:,2))
subplot(2,2,3)
plot(1:length(Data(:,3)),Data(:,3))
subplot(2,2,4)
plot(1:length(Data(:,4)),Data(:,4))

function f = mdlfun(tbl,gprMdl)
Ts_att = tbl.Ts;
Np_att = tbl.Np;
Nc_att = tbl.Nc;
Q_p = tbl.Qp;
Q_w = tbl.Qw;
R_M = tbl.RM;
R_dM = tbl.RdM;
vars = [Ts_att Np_att Nc_att Q_p Q_w R_M R_dM];
f = predict(gprMdl,vars);
end
```

G.6.3 Bayesian Optimization Position Tracking MPC

```
clc; clear all; close all;
Count = 0;
Samples = 100;
for i = 1:Samples
    Ts_mul = round(rand*8)+8;
    Q_posX = 10^(rand*5);
    Q_posY = 10^(rand*5);
    Q_posZ = 10^(rand*5);
    Q_velX = 10^(rand*5+3);
    Q_velY = 10^(rand*5+3);
    Q_velZ = 10^(rand*5+3);
    Q = diag([Q_velX,Q_velY,Q_velZ,Q_posX,Q_posY,Q_posZ]);
    R_MX = 10^(rand*5-2);
    R_MY = 10^(rand*5-2);
    R_MZ = 10^(rand*5-2);
    R_dMX = 10^(rand*5-2);
    R_dMY = 10^(rand*5-2);
    R_dMZ = 10^(rand*5-2);
    R = diag([R_MX,R_MY,R_MZ]);
    R_du = diag([R_dMX,R_dMY,R_dMZ]);
    eps(i,:) = [Ts_mul Q_posX Q_posY Q_posZ Q_velX Q_velY Q_velZ R_MX R_MY R_MZ R_dMX R_dMY R_dMZ];
    sim('PosTrackTune',25);
    Cost(:,:) = [ans.Opt.Data(:,:) ans.In.Data(:,:)];

    for ii = 1:length(Cost(:,1,1))
        Jcost(ii) = 5*(Cost(ii,4)-Cost(ii,1))^2 + (Cost(ii,5)-Cost(ii,2))^2 +
             5*(Cost(ii,6)-Cost(ii,3))^2;% + (Cost(ii,4)^2 + Cost(ii,5)^2 +
             (9.8*0.8422-Cost(ii,6))^2)*0.001;
    end
    Jc(i,1) = sum(Jcost);
    Count = Count + 1
end
```

```
epse = eps;
JJJ = Jc;
for jj = 1:length(Jc)
    if Jc(jj) > 10^(6)
        epse(jj,:) = zeros(1,width(eps));
        JJJ(jj,1) = 0;
    else
        epse(jj,:) = eps(jj,:);
        JJJ(jj,1) = Jc(jj,1);
    end
end
epse(epse==0) = [];
epse = reshape(epse,length(epse)/13,13);
JJJ = JJJ(JJJ~=0);

gprMdl = fitrgp(epse,JJJ,'KernelFunction','squaredexponential');

Ts_mul = optimizableVariable('Ts',[8,16],'Type','integer');
Q_posX = optimizableVariable('QpX',[10^(0),10^(5)],'Type','real');
Q_posY = optimizableVariable('QpY',[10^(0),10^(5)],'Type','real');
Q_posZ = optimizableVariable('QpZ',[10^(0),10^(5)],'Type','real');
Q_velX = optimizableVariable('QwX',[10^(3),10^(8)],'Type','real');
Q_velY = optimizableVariable('QwY',[10^(3),10^(8)],'Type','real');
Q_velZ = optimizableVariable('QwZ',[10^(3),10^(8)],'Type','real');
R_MX = optimizableVariable('RMX',[10^(0),10^(5)],'Type','real');
R_MY = optimizableVariable('RMY',[10^(0),10^(5)],'Type','real');
R_MZ = optimizableVariable('RMZ',[10^(0),10^(5)],'Type','real');
R_dMX = optimizableVariable('RdMX',[10^(0),10^(5)],'Type','real');
R_dMY = optimizableVariable('RdMY',[10^(0),10^(5)],'Type','real');
R_dMZ = optimizableVariable('RdMZ',[10^(0),10^(5)],'Type','real');
vars = [Ts_mul Q_posX Q_posY Q_posZ Q_velX Q_velY Q_velZ R_MX R_MY R_MZ R_dMX R_dMY R_dMZ];

bayesObject = bayesopt(@(tbl)mdlfun(tbl,gprMdl),vars,'Verbose',1,...
    'AcquisitionFunctionName','expected-improvement-plus',...
    'PlotFcn',@plotMinObjective);

X_best = [bayesObject.XAtMinEstimatedObjective.Ts bayesObject.XAtMinEstimatedObjective.QpX
     bayesObject.XAtMinEstimatedObjective.QpY bayesObject.XAtMinEstimatedObjective.QpZ
     bayesObject.XAtMinEstimatedObjective.QwX bayesObject.XAtMinEstimatedObjective.QwY
     bayesObject.XAtMinEstimatedObjective.QwZ bayesObject.XAtMinEstimatedObjective.RMX
     bayesObject.XAtMinEstimatedObjective.RMY bayesObject.XAtMinEstimatedObjective.RMZ
     bayesObject.XAtMinEstimatedObjective.RdMX bayesObject.XAtMinEstimatedObjective.RdMY
     bayesObject.XAtMinEstimatedObjective.RdMZ]
Ts_mul = X_best(1,1);
Q_posX = X_best(1,2);
Q_posY = X_best(1,3);
Q_posZ = X_best(1,4);
Q_velX = X_best(1,5);
Q_velY = X_best(1,6);
Q_velZ = X_best(1,7);
Q = diag([Q_posX,Q_posY,Q_posZ,Q_velX,Q_velY,Q_velZ]);
R_MX = X_best(1,8);
R_MY = X_best(1,9);
R_MZ = X_best(1,10);
R_dMX = X_best(1,11);
R_dMY = X_best(1,12);
R_dMZ = X_best(1,13);
R = diag([R_MX,R_MY,R_MZ]);
R_du = diag([R_dMX,R_dMY,R_dMZ]);
sim('PosTrackTune',55);
Data(:,:) = [ans.Opt.Data(:,:) ans.In.Data(:,:)];

figure(10)
subplot(3,1,1)
plot(1:length(Data(:,1)),Data(:,1))
subplot(3,1,2)
```

```
plot(1:length(Data(:,2)),Data(:,2))
subplot(3,1,3)
plot(1:length(Data(:,3)),Data(:,3))

function f = mdlfun(tbl,gprMdl)
Ts_mul = tbl.Ts;
Q_posX = tbl.QpX;
Q_posY = tbl.QpY;
Q_posZ = tbl.QpZ;
Q_velX = tbl.QwX;
Q_velY = tbl.QwY;
Q_velZ = tbl.QwZ;
R_MX = tbl.RMX;
R_MY = tbl.RMY;
R_MZ = tbl.RMZ;
R_dMX = tbl.RdMX;
R_dMY = tbl.RdMY;
R_dMZ = tbl.RdMZ;
vars = [Ts_mul Q_posX Q_posY Q_posZ Q_velX Q_velY Q_velZ R_MX R_MY R_MZ R_dMX R_dMY R_dMZ];
f = predict(gprMdl,vars);
end
```