NTNU

Norwegian University of
Science and Technology

Automatic landing of multi-rotor on moving platform

Vuk Krivokapic

Department of Engineering Cybernetics
Norwegian University of Science and Technology

Submission date: December 18, 2018
Supervisor: Professor Tor Arne Johansen
Co-Supervisor: PhD candidate Martin Lysvand Sollie

Preface

This specialization project is a mandatory part of the 2.year masters program in Cybernetics and Robotics at Norwegian University of Science and Technology (NTNU), worth 7.5 credits. The project is a preparation for my masters thesis next semester.

All the equipment and software used in this project, and that will be used for the masters thesis, are provided by NTNU.

I would like to thank my supervisor Professor Tor Arne Johansen and co supervisor PhD candidate Martin Lysvand Sollie for valuable guidance and help through the semester.

Abstract

This thesis presents creation of a simulation environment for autonomous landing of multi-rotor on a moving platform. The simulation environment is created in Matlab/Simulink. HEXH20, manufactured by the British company QuadH20, is multi-rotor the environment is created for. The thesis is divided in three parts, modeling, control and simulation.

Mathematical models for the multi-rotor and the landing platform was developed and added to the simulation environment. Disturbances that affects landing, such as wind and waves are also modeled in order to make the simulation environment as realistic as possible. Several controllers are added to the system, to simulate the landing process. The controllers are implemented as a cascade, and they are tuned to work together.

The thesis also presents several logical algorithms that are developed to carry out a successful landing. The algorithms together complete a landing system, which is presented and simulated in the thesis. Through many simulations, algorithms are tested carefully. The goal was to test the robustness of the landing algorithms by performing simulations in several weather conditions.

Results of the simulations are presented in a own chapter in the report. Three dimensional plots of the landing approach are added to visualize the whole process. Some landing simulations showed satisfying results, while others failed due to different reasons. Both simulations that failed and that succeeded are discussed, and a plan for further development are presented at the very end of the report.

Table of Contents

List of Figures

List of Tables

Abbreviations

AUV	Autonomous Underwater Vehicle
DUNE	Unified Navigation Environment
etc	Et cetra
LOS	Line Of Sight
LQR	Linear Quadratic Regulator
LSTS	Laboratrio de Sistemas e Tecnologia Subaqutica
MPC	Model Predictive Control
NTNU	Norwegian University of Science and Technology
PID	Proportional Integral Derivative
PIV	Proportional Integral Velocity
PM Spectrum	Pierson-Moskowitz Spectrum
UAV	Unmanned Arial Vehicle
UK	United Kingdom

Chapter 1

Introduction

1.1 Motivation

The use of UAV is becoming increasingly popular in the recent years. A lot of money are invested in expanding fields of operation for these platforms. Years of development resulted into highly functional vehicles that can perform tasks as: geo referencing, surveillance of events, detection of forest fires, transport, etc. Still, there is a lot of room for improvements.

UAV is often used for missions at sea. It is desired to operate UAV's from the ships autonomous, without using pilots. In order to complete an autonomous mission, an autonomous landing system has to exist. In addition, an autonomous landing system enables landing on ships in rough weather conditions with limited view, that would not be possible to perform manually.

1.2 Problem formulation

To perform a safe, autonomous landing, a properly developed, reliable landing system has to exist. In order to design algorithms that can yield into a successful landing, a simulation environment has to be created. The simulation environment has to be realistic, taking in count all the disturbances that will affect a landing process. Creation of the simulation environment and development of the landing algorithms is the main tasks of this project. The project is divided in following sub tasks:

- Develop a realistic mathematical model of the hexa-rotor in Simulink.
- Develop control algorithms for the hexa-rotor.
- Investigate disturbances acting on the vehicle, and create a mathematical models of those.
- Perform testing of disturbance tolerance by simulating the model.
- Develop landing algorithms.

The simulation environment created in this project is supposed to work as a basis for later development in the masters thesis.

1.3 Related work

Development of a landing system for a UAV can be separated in three parts, mathematical modeling, control and landing algorithms.

A lot of work is done on modeling multi-rotors. Most of the work is related to quadcopters, but there also exist a lot of work on hexacopters. However, the dynamics of both quadcopters and hexacopters are similar, and can be used across each other. [2] has presented a detailed multi-rotor model, modeling even DC motor and rotor blade dynamics. Parameters are later confirmed by performing tests in a wind tunnel. [3] presents a simpler quadcopter model. [4] shows a detailed hexacopter model, presenting both model equations with descriptions and parameters found. Finding correct parameters can be a difficult task, [5] presents how the thrust parameters can be found by performing tests with a robotic arm, equipped with touch sensors.

Choice of control strategy depends on mission purposes. Several articles as [6], [2] and [7] does cascaded PID control. That implies controlling attitude and horizontal position separate, in a cascade. Another interesting control strategy, done by [8] and [9] is LQR control, an optimal control strategy that assigns control signals based on weight matrices. Both [9], that primary uses LQR control, and [7], that primary uses PID control, suggest testing of other control methods such as MPC, Backstepping and Adaptive control in order to improve controller performance. An example of backstepping control implementation is presented in [4]. Target tracking is a control problem that has to be considered before landing can be performed. One way of performing target tracking is PID controllers, but a, often more efficient, way is LOS guidance, presented in [10] and [9].

[7] has done work on landing a quadrotor on platform in motion using a vision system. The author has divided landing procedure into several parts. Controller gains are adjusted depending on which pre-defined part of the landing procedure the multi-rotor is located in. Like [3] and [11], different parts of the landing sequence are defined by boundary violation of imaginary boundaries around the target. Boundaries are defined to be maximal allowed distance from the target. The two sources mentioned last are not using gain adjustment during the landing sequence.

1.4 Outline

Chapter 2 gives an overview of the most important parts of the system. The chapter is divided into two parts, one for hardware and one for software used.

Chapter 3 shows how different parts of the system are modeled mathematically. All forces acting on the vehicle are described mathematically. Then the final mathematical model of the hexacopter is presented. The chapter also contains modeling of wind and waves.

Chapter 4 presents the developed control algorithms.

Chapter 5 starts by calculating necessary parameters in order to perform a realistic simulation. After that, limitations for the system are calculated. A test of maximal wind velocity that the system can tolerate follows, before tuning of the controllers are presented. The landing algorithms are described at the end of the chapter.

Chapter 6 shows final results of the simulation. The final results are shown in 3D plots.

Chapter 7 discuss the result from chapter 6. In addition, suggestions for further work are listed at the end of the chapter.

Chapter 8 presents conclusions made from the project.

Chapter 2

System Overview

2.1 Hardware

2.1.1 Hexacopter

The vehicle used in this project is a Hexacopter HEXH2O, produced by QuadH20, a drone manufactorer based in UK. It is known for being waterproof, and fits for missions involving water.[12]

Figure 2.1: HEXH20 [1]

Table 2.1: Hexacopter parameters

Frame weight without electrics	1.4 kg
Fully loaded frame weight	3.7 kg
Motor weight (all six motors)	1 kg
Frame length	0.4 m
Frame width	0.28 m
Frame heigth	0.18 m
Arm length	0.39 m
Forward speed	56 km/h

The parameters of hexacopter are listed in table 2.1. Most of the parameters are found in [1]. Dimension parameters are self- measured and may deviate a bit form real values.

2.1.2 Motors

The hexacopter is equipped with six E800 motors, from the famous UAV manufactur, DJI. DJI is a world leading UAV manufacture from China.

The motor parameters are found in [13]:

Table 2.2: Motor parameters

Motor weight	0.106 kg
Propeller weight	0.019 kg
ESC weight	0.03 kg
Total weight (Motor + Propeller + ESC)	0.155 kg
Max thrust	2.1 kg
KV	350 rpm/V
Maximum Allowable Voltage	26 V
Maximum Allowable Current	20 A

2.1.3 ArduPilot

ArduPilot is an open source auto pilot used to control UAV's. The ArduPilot is the most installed autopilot world wide [14]. The autopilot is capable of autonomous stabilization and way-point navigation. The way-point navigation part can support over 100 way-points in three dimensions. The ArduPilot uses Pixhawk as the flight controller. It is worth mentioning that ArduPilot software is continuously improving, as the open source software is in continuously development.

2.2 Software

2.2.1 ArduPilot - Mission planner

Mission planner is a ground station software for the ArduPilot (section 2.1.3). The named software allows the user to setup and configure the UAV in order to achieve most optimal results during the mission. As the name of the software implies, mission planner provides an opportunity to plan and upload missions to the autopilot. Another opportunity provided by the software is flight logging, which is useful for later mission analyzes.

2.2.2 DUNE

Unified Navigation Environment (DUNE) is a software used for sensor interaction, navigation, control, maneuvering and plan execution. The software is operative system independent. DUNE is mainly used for communication between different units of a larger system. DUNE is part of the LSTS tool-chain, developed at university of Porto [15].

2.2.3 Simulink

Simulink is an additional software to the computing program, Matlab. The main purpose of this software is modeling dynamical systems. Simulink is a software used in almost every modeling course at NTNU, which makes it a natural choice for this project as well. A significant amount of integrated functions makes Simulink a good modeling software for this project.

Chapter 3

Modeling

3.1 Theory

3.1.1 Coordinate frames

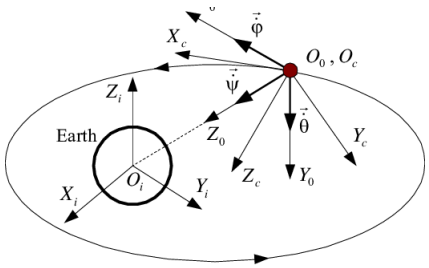


Figure 3.1: Coordinate frames relative to each other

To describe position and orientation of an aircraft, it is useful to use various coordinate frames (figure 3.1). The position of a body is often described relative to another, fixed frame. Attitude of a body is described as rotation about axis of a coordinate frame that has origin in the center of the body, known as body frame. Equations of motion for an aircraft are derived relative to a earth-fixed coordinate frame, while aerodynamic forces and torques are derived in the body coordinate frame. It is easy to do transformation between coordinate frames, by doing rotational and translational operations (section 3.1.3).

3.1.2 Euler angles

Euler angles is a principle of discribing rotations using three angles. The rotations are described in sequence,

1. Rotation about z-axis
2. Rotation about y-axis
3. Rotation about z-axis

with the rotation matrices,

$$\begin{aligned}\mathbf{R}_x &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \\ \mathbf{R}_y &= \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \\ \mathbf{R}_z &= \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}\end{aligned} \tag{3.1}$$

Euler angles are singular for angles equal to $\pm 90^\circ$.

3.1.3 Transformation between coordinate frames

Transformation between coordinate frames is done by multiplication of translational vector and rotational matrices (eq. 3.1). Rotational matrices represents rotation about each axis performed between coordinate frames, while the translational vector represents the distance between coordinate frames in three dimensions.

Rotation from body frame b to inertial frame i, where rotation is performed about all three axis, can be described as,

$$\mathbf{R}_b^i = \mathbf{R}_z \mathbf{R}_y \mathbf{R}_z \tag{3.2}$$

where

$$\left(\mathbf{R}_b^i\right)^{-1} = \left(\mathbf{R}_b^i\right)^T = \mathbf{R}_i^b \tag{3.3}$$

The total transformation matrix is:

$$\mathbf{T}_b^i = \begin{bmatrix} & & \text{x} \\ \mathbf{R}_b^i & & \text{y} \\ & & \text{z} \\ 000 & & 0 \end{bmatrix} \tag{3.4}$$

3.1.4 Pierson-Moskowitz Spectrum

Pierson-Moskowitz Spectrum (PM spectrum) is a model for energy distribution in sea, developed in 1964. Assumption made for the model is that wind blows steady for long time over large area, and that the waves will eventually reach point of equilibrium with the wind. Model is depended on two parameters, A and B.[16][17]

$$\begin{aligned} A &= 8.1 \cdot 10^{-3} \cdot g^2 \\ B &= 0.74(\frac{g}{V_{19.4}})^4 = \frac{3.11}{H_s^2} \end{aligned} \tag{3.5}$$

where $V_{19.4}$ is wind velocity at height of 19.4 meters, g is the gravity constant and H_s is the wave height. Wave height is depended on sea state, and can be roughly estimated using table 8.5 in [17].

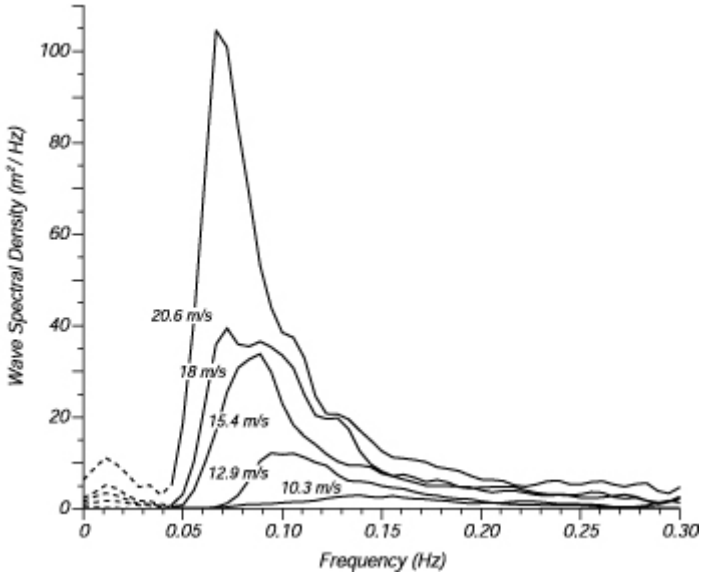


Figure 3.2: PM spectra for different $V_{19.4}$ velocities

Figure 3.2 shows fully developed PM spectra for different $V_{19.4}$ velocities.

3.2 Hexacopter dynamics

Hexacopter dynamics can be derived from the Euler angles. The reason why this can be done is that the hexacopter is assumed as a symmetrical and rigid body.

3.2.1 Rotation

Rotation from body to inertial frame is described by using series of rotations $\psi - \theta - \phi$, also known as yaw-pitch-roll. The result of the series of rotation is following rotation matrix:

$$\begin{aligned}\mathbf{R}_B^I &= \begin{bmatrix}\cos(\psi) & \sin(\psi) & 0\\ -\sin(\psi) & \cos(\psi) & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\cos(\theta) & 0 & -\sin(\theta)\\ 0 & 1 & 0\\ \sin(\theta) & 0 & \cos(\theta)\end{bmatrix}\begin{bmatrix}1 & 0 & 0\\ 0 & \cos(\phi) & \sin(\phi)\\ 0 & -\sin(\phi) & \cos(\phi)\end{bmatrix}\\ &= \begin{bmatrix}\cos\theta\cos\psi & \cos\psi\sin\theta\sin\phi-\sin\theta\cos\phi & \cos\psi\sin\theta\cos\phi+\sin\theta\sin\phi\\ \cos\theta\sin\psi & \sin\psi\sin\theta\sin\phi+\cos\theta\cos\psi & \cos\phi\sin\theta\sin\psi-\cos\psi\sin\phi\\ -\sin\theta & \sin\phi\cos\theta & \cos\theta\cos\phi\end{bmatrix}\end{aligned} \tag{3.6}$$

By using property from eq.3.3 it is possible to find rotation matrix from inertial to body frame.

$$\mathbf{R}_I^B = \begin{bmatrix}\cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta\\ \cos\psi\sin\theta\sin\phi-\sin\theta\cos\phi & \sin\psi\sin\theta\sin\phi+\cos\theta\cos\psi & \sin\phi\cos\theta\\ \cos\psi\sin\theta\cos\phi+\sin\theta\sin\phi & \cos\phi\sin\theta\sin\psi-\cos\psi\sin\phi & \cos\theta\cos\phi\end{bmatrix} \tag{3.7}$$

Angular velocity of the body frame relative to the inertial frame is described using the same series of rotations.

$$\begin{aligned}\omega_{ib}^b &= \mathbf{R}_{x,-\phi}\mathbf{R}_{y,-\theta}\begin{bmatrix}0\\0\\\dot{\psi}\end{bmatrix}+\mathbf{R}_{x,-\phi}\begin{bmatrix}0\\\dot{\theta}\\0\end{bmatrix}+\begin{bmatrix}\dot{\phi}\\0\\0\end{bmatrix}\\ &= \begin{bmatrix}-\sin\theta\dot{\psi}+\dot{\phi}\\ \sin\phi\cos\theta\dot{\psi}+\cos\phi\dot{\theta}\\ \cos\phi\cos\theta\dot{\psi}-\sin\phi\dot{\theta}\end{bmatrix}\\ &= \begin{bmatrix}1 & 0 & \sin\theta\\ 0 & \cos\phi & \sin\theta\cos\phi\\ 0 & -\sin\phi & \cos\phi\cos\theta\end{bmatrix}\begin{bmatrix}\dot{\phi}\\\dot{\theta}\\\dot{\psi}\end{bmatrix}\end{aligned} \tag{3.8}$$

3.2.2 Forces

All forces acting on the hexacopter can be decomposed in three directions, x,y and z, in the inertial frame. Beginning the force decomposition by decomposing gravity force, which is only acting in z-direction of the UAV, decomposed in the inertial frame. It is also worth mentioning that upwards direction is counted as positive in this model, which results into a negative sign of the z-component of the gravity force.

$$F_{gravity} = \begin{bmatrix}0\\0\\-mg\end{bmatrix} \tag{3.9}$$

When propellers of the hexacopter are rotating, they are creating a force that makes hexacopter fly. That force is called **thrust**. Thrust force is directly depended on the speed of each propeller. Total thrust is sum of all thrust forces produced by each propel. Differences in the thrust

produced by each propeller yields torque around the vehicle center of mass, resulting in rotation.

$$F_{thrust} = \begin{bmatrix} 0 \\ 0 \\ \mathbf{R}_B^I \sum_{i=1}^{6} b\Omega_i \end{bmatrix} \tag{3.10}$$

b is the thrust constant in eq. 3.10.

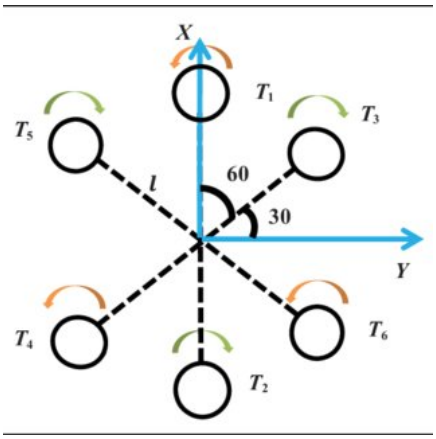


Figure 3.3: Thrust forces and rotation of propellers

Aerodynamic movement inducts drag forces in x, y and z direction. Mathematical model presenting the induced drag is,

$$F_{drag} = \begin{bmatrix} \frac{1}{2} C_d \rho V_x^2 A_x \\ \frac{1}{2} C_d \rho V_y^2 A_y \\ \frac{1}{2} C_d \rho V_z^2 A_z \end{bmatrix} \tag{3.11}$$

where C_d is the drag coefficient, found to be 1.05 by considering the shape of the hexacopter. [18] A_x, A_y and A_z are surface areas pointing i x,y and z direction respectively, while ρ is the air density. V represents the air velocity in given direction.

The drag force induced by the propellers is neglected in this project. The reason is the dominance of drag induced my movement in the air. Neglecting drag force induced by propellers also simplifies the model.

3.2.3 Torques

Torques are created by UAV's rotation about its own body axis. Moments about body axis of the UAV are created by adjusting angular velocity of each propel independently. Moments in roll, pitch and yaw can be modelled from the geometrical structure of the hexacopter, and angular velocities of the propellers (figure 3.3).

$$\tau_\phi = bl\big[-\Omega_2^2 + \Omega_5^2 + \frac{1}{2}(-\Omega_1^2 - \Omega_3^2 + \Omega_2^2 + \Omega_6^2)\big] \tag{3.12}$$

$$\tau_\theta = bl\frac{\sqrt{3}}{2}(-\Omega_1^2 + \Omega_3^2 + \Omega_4^2 - \Omega_6^2) \tag{3.13}$$

$$\tau_{psi} = d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2 - \Omega_5^2 + \Omega_6^2) \tag{3.14}$$

where b is the thrust constant, l is distance from propellers to the center of gravity and d is the drag factor.

3.2.4 Mathematical model

The mathematical model of the hexacopter can be divided into two parts, translational dynamics and rotational dynamics. Translational dynamics represent hexacopter movement along x,y and z axis of the inertial frame, while rotational dynamics represent rotation about x,y and z axis of the body frame. Frame of the landing platform is considered as inertial frame in this project, which makes sense since all controllers, described later in the repport, are designed to minimize error between the inertial and body frame.

Translational dynamics

Translational dynamics are modeled by considering all forces acting on the body, described i section 3.2.2.

$$ma = \sum F = F_{drag} + F_{thrust} + F_{gravity} \tag{3.15}$$

which gives following translational equations of motion:

$$\ddot{x} = \frac{1}{m}\left[(\cos\phi\cos\psi\sin\theta + \sin\phi\sin\psi)(\sum_{i=1}^{6} F_i) + f_{drag_x}\dot{x}\right] \tag{3.16a}$$

$$\ddot{y} = \frac{1}{m}\left[(\cos\phi\sin\psi\sin\theta - \sin\phi\cos\psi)(\sum_{i=1}^{6} F_i) + f_{drag_y}\dot{y}\right] \tag{3.16b}$$

$$\ddot{x} = \frac{1}{m}\left[(\cos\phi\cos\theta)(\sum_{i=1}^{6} F_i) + f_{drag_z}\dot{z}\right] \tag{3.16c}$$

Rotational dynamics

Rotational dynamics are modeled by considering all moments acting on the body, described in section 3.2.3.

$$\ddot{\phi} = \frac{1}{J_{xx}}\left[\dot{\theta}\dot{\psi}(J_{yy} - J_{zz}) + bl\Big[-\Omega_2^2 + \Omega_5^2 + \frac{1}{2}(-\Omega_1^2 - \Omega_3^2 + \Omega_2^2 + \Omega_6^2)\Big]\right] \tag{3.17a}$$

$$\ddot{\theta} = \frac{1}{J_{yy}}\left[\dot{\phi}\dot{\psi}(J_{zz} - J_{xx}) + bl\frac{\sqrt{3}}{2}(-\Omega_1^2 + \Omega_3^2 + \Omega_4^2 - \Omega_6^2)\right] \tag{3.17b}$$

$$\ddot{\psi} = \frac{1}{J_{zz}}\left[\dot{\phi}\dot{\theta}(J_{xx} - J_{yy}) + d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2 - \Omega_5^2 + \Omega_6^2)\right] \tag{3.17c}$$

Relation between control inputs and angular velocities of propellers

The system is controlled by four control inputs, u_1, u_2, u_3 and u_4. The first control input is used to control the vertical movement of the hexacopter, while the others are controlling roll, pitch and yaw movements, respectively.

Thrust force acting in z-direction of the body frame is induced by angular velocity of all the propellers. Velocity of the vertical movement depends on the force produced by the propellers. Control input u_1 directly controls thrust in body z-axis by controlling angular velocities of the propellers. Roll, pitch and yaw movement about body axis are obtained by combinations of different angular velocities of the six propellers. Mapping between control inputs u_1, u_2, u_3 and propellers angular velocities are presented in this section.

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} b & b & b & b & b & b \\ -\frac{bl}{2} & -bl & -\frac{bl}{2} & \frac{bl}{2} & bl & -\frac{bl}{2} \\ -\frac{bl\sqrt{3}}{2} & 0 & \frac{bl\sqrt{3}}{2} & \frac{bl\sqrt{3}}{2} & 0 & -\frac{bl\sqrt{3}}{2} \\ -d & d & -d & d & -d & d \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \\ \Omega_5^2 \\ \Omega_6^2 \end{bmatrix} \tag{3.18}$$

Eq. 3.18 presents control inputs as functions of the propeller velocities. By taking the psaudo-inverse of the 4x6 matrix in eq. 3.18, it is possible to find propeller velocities as function of control inputs.

$$\begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \\ \Omega_5^2 \\ \Omega_6^2 \end{bmatrix} = \frac{1}{6bl} \begin{bmatrix} l & 2 & 0 & -\frac{bl}{d} \\ l & 1 & -\sqrt{3} & \frac{bl}{d} \\ l & -1 & -\sqrt{3} & -\frac{bl}{d} \\ l & -2 & 0 & \frac{bl}{d} \\ l & -1 & \sqrt{3} & -\frac{bl}{d} \\ l & 1 & \sqrt{3} & \frac{bl}{d} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \tag{3.19}$$

Total system equations [4]

Model for the total system is presented by combining translational dynamics of the system, rotational dynamics of the system and control inputs.

$$\begin{aligned}
\ddot{\phi} &= \frac{1}{J_{xx}}\left[\dot{\theta}\dot{\psi}(J_{yy}-J_{zz}) + bl\Big[-\Omega_2^2+\Omega_5^2+\frac{1}{2}(-\Omega_1^2-\Omega_3^2+\Omega_2^2+\Omega_6^2)\Big]\right] \\
&= \frac{1}{J_{xx}}\left[\dot{\theta}\dot{\psi}(J_{yy}-J_{zz}) + u_2\right] && (3.20a) \\
\ddot{\theta} &= \frac{1}{J_{yy}}\left[\dot{\phi}\dot{\psi}(J_{zz}-J_{xx}) + bl\frac{\sqrt{3}}{2}(-\Omega_1^2+\Omega_3^2+\Omega_4^2-\Omega_6^2)\right] \\
&= \frac{1}{J_{yy}}\left[\dot{\phi}\dot{\psi}(J_{zz}-J_{xx}) + u_3\right] && (3.20b) \\
\ddot{\psi} &= \frac{1}{J_{zz}}\left[\dot{\phi}\dot{\theta}(J_{xx}-J_{yy}) + d(-\Omega_1^2+\Omega_2^2-\Omega_3^2+\Omega_4^2-\Omega_5^2+\Omega_6^2)\right] \\
&= \frac{1}{J_{zz}}\left[\dot{\phi}\dot{\theta}(J_{xx}-J_{yy}) + u_4\right] && (3.20c) \\
\ddot{x} &= \frac{1}{m}\left[(\cos\phi\cos\psi\sin\theta+\sin\phi\sin\psi)(\sum_{i=1}^{6}F_i) + f_{drag_x}\dot{x}\right] \\
&= \frac{1}{m}\left[(\cos\phi\cos\psi\sin\theta+\sin\phi\sin\psi)u_1 + f_{drag_x}\dot{x}\right] && (3.20d) \\
\ddot{y} &= \frac{1}{m}\left[(\cos\phi\sin\psi\sin\theta-\sin\phi\cos\psi)(\sum_{i=1}^{6}F_i) + f_{drag_y}\dot{y}\right] \\
&= \frac{1}{m}\left[(\cos\phi\sin\psi\sin\theta-\sin\phi\cos\psi)u_1 + f_{drag_y}\dot{y}\right] && (3.20e) \\
\ddot{x} &= \frac{1}{m}\left[(\cos\phi\cos\theta)(\sum_{i=1}^{6}F_i) + f_{drag_z}\dot{z}\right] \\
&= \frac{1}{m}\left[(\cos\phi\cos\theta)u_1 + f_{drag_z}\dot{z}\right] && (3.20f)
\end{aligned}$$

The system can further be simplified. The 12 states will be named x_1 - x_{12}, for the sake of order.

$$\begin{aligned}
&x_1 = \phi && x_2 = \dot{x}_1 = \dot{\phi} && x_3 = \theta && x_4 = \dot{x}_3 = \dot{\theta} \\
&x_5 = \psi && x_6 = \dot{x}_5 = \dot{\psi} && x_7 = x && x_8 = \dot{x}_7 = \dot{x} \\
&x_9 = y && x_{10} = \dot{x}_9 = \dot{y} && x_{11} = z && x_{12} = \dot{x}_{11} = \dot{z}
\end{aligned} \qquad (3.21)$$

$$a_1 = \frac{J_{yy} - J_{zz}}{J_{xx}} \qquad a_2 = -\frac{J_r}{J_{xx}} \qquad a_3 = \frac{J_{zz} - J_{zz}}{J_{yy}} \qquad a_4 = \frac{J_r}{J_{yy}}$$
$$a_5 = \frac{J_{xx} - J_{yy}}{J_{zz}} \qquad a_6 = -\frac{f_{drag_x}}{m} \qquad a_7 = -\frac{f_{drag_y}}{m} \qquad a_8 = -\frac{f_{drag_z}}{m}$$
$$b_1 = \frac{1}{J_{xx}} \qquad b_2 = \frac{1}{J_{yy}} \qquad b_3 = \frac{1}{J_{zz}}$$

The final, simplified, model is:

$$\begin{aligned}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= a_1 x_4 x_6 + b_1 u_2 \\
\dot{x}_3 &= x4 \\
\dot{x}_4 &= a_3 x_2 x_6 + b_2 u_3 \\
\dot{x}_5 &= x_6 \\
\dot{x}_6 &= a_5 x_2 x_4 + b_3 u_4 \\
\dot{x}_7 &= x_8 \\
\dot{x}_8 &= a_6 x_8 + \frac{1}{m}(cos\phi cos\psi sin\theta + sin\phi sin\psi) u_1 \\
\dot{x}_9 &= x_{10} \\
\dot{x}_{10} &= a_7 x_{10} + \frac{1}{m}(cos\phi sin\psi sin\theta - sin\phi cos\psi) u_1 \\
\dot{x}_{11} &= x_{12} \\
\dot{x}_{12} &= a_8 x_{12} + \frac{cos\phi cos\theta}{m} u_1 - g
\end{aligned}$$

Figure 5.2 shows a simplified block diagram of the total system.

3.3 Wind

It is important to consider wind disturbances while creating a simulation environment. There exist many techniques of modelling wind. Dryden gust wind model is used in this project.

$$V_{total\ wind} = V_{constant\ wind} + V_{wind\ gust} \tag{3.22}$$

Dryden gust model consist three transfer functions, one for each wind direction (x, y and z). Input to the transfer functions is white noise, while the output are wind gusts. In order to make model realistic, a constant wind has to be added. Transfer functions contains three parameters. V_a is the nominal airspeed of the vehicle, which is typically 2-4 m/s for the hexacopter used in this task. The same nominal airspeed is used in all three directions. Intensities of turbulence in

each direction are represented by σ_x, σ_y and σ_z, and their values can be found in Table 4.1 in [19]. L_x, L_y and L_z are representing the spatial wavelengths, their values can also be found in the same table.

Usually, transfer function for the wind in x direction, which is often defined as forward direction of a vehicle, is a first order transfer function, while transfer functions in y and z directions are second order functions. The reason is that the UAV's usually moves with higher velocity in x-direction. In this project, there are no defined a specific forward direction, therefore second order transfer functions will be used for wind gusts in all three directions.

$$H_x(s) = \sigma_x \sqrt{\frac{3V_a}{L_x}} \left(\frac{s + \frac{V_a}{\sqrt{3}L_x}}{s + \frac{V_a}{L_x}} \right) \tag{3.23a}$$

$$H_y(s) = \sigma_y \sqrt{\frac{3V_a}{L_y}} \left(\frac{s + \frac{V_a}{\sqrt{3}L_y}}{s + \frac{V_a}{L_y}} \right) \tag{3.23b}$$

$$H_z(s) = \sigma_z \sqrt{\frac{3V_a}{L_z}} \left(\frac{s + \frac{V_a}{\sqrt{3}L_z}}{s + \frac{V_a}{L_z}} \right) \tag{3.23c}$$

Wind gusts are given in body frame [19], while constant winds are given i inertial frame. Since the whole system is modeled relative to the inertial frame, wind gusts has to be transformed from body to inertial frame. Transformation is done by multiplying wind vector with rotation matrix from equation 3.6.

$$V^I_{wind\ gust} = R^I_B \cdot V^B_{wind\ gust} \tag{3.24}$$

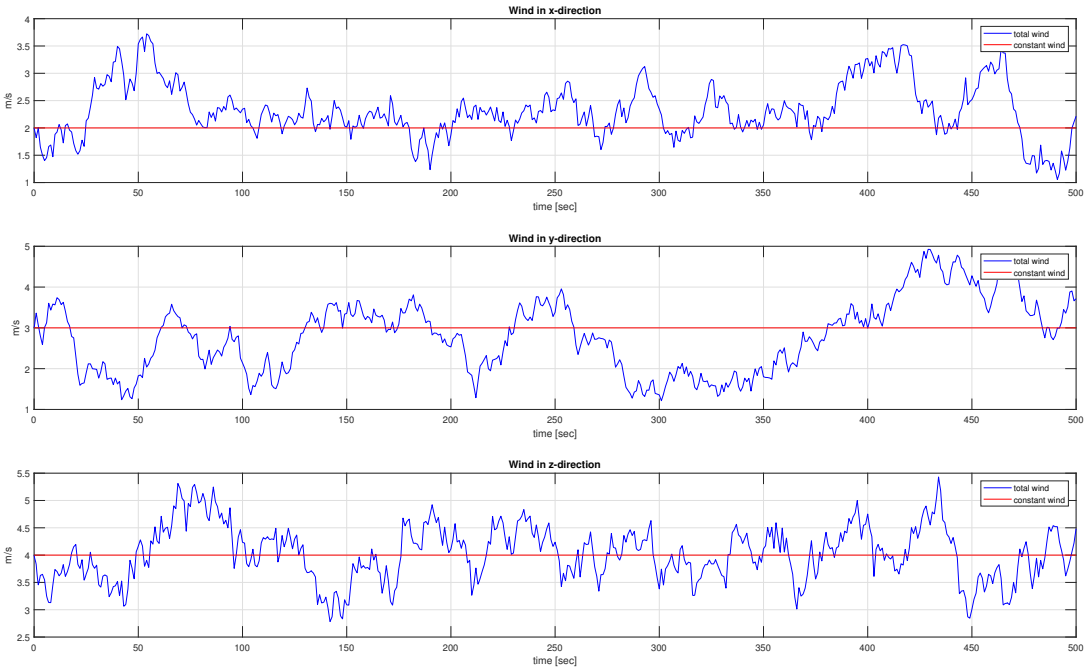


Figure 3.4: Response of the Dryden gust wind model

3.4 Platform

The platform model is just a simple integrator in all three dimensions, with velocity input. Outputs of the integrators are platform positions in three dimensions. Platform is assumed to be the inertial frame in this task, and the UAV moves relative to the platform position. In addition, wave disturbances are added to the platform (section 3.4.1).

3.4.1 Waves

Waves are modeled according to chapter 8.2.6 in [17]. A linear approximation is done during wave modeling. Waves are modeled as second order transfer functions.

$$h_{wave}(s) = \frac{K_w s}{s^2 + 2\lambda\omega_0 s + \omega_0^2} \tag{3.25}$$

where

$$K_w = 2\lambda\omega_0\sigma$$

Parameters in equation 3.25 can be found by looking a Pierson-Moskowitz Spectrum (Section 3.1.4). To create spectra plot, which is later used to compute parameters, $V_{19.4}$ has to be computed. Formula used is,

$$H_s = \frac{2.06}{g^2} V_{19.4}^2$$
$$V_{19.4} \approx V_{20} = g \cdot \sqrt{\frac{H_s}{2.06}} \tag{3.26}$$

where H_s is average wave height value, depending on chosen sea state. Values of wave height intervals of different sea states are based on Table 8.5 in [17]. Further, wave spectra is created by slightly modifying functions from MSS toolbox.[20]

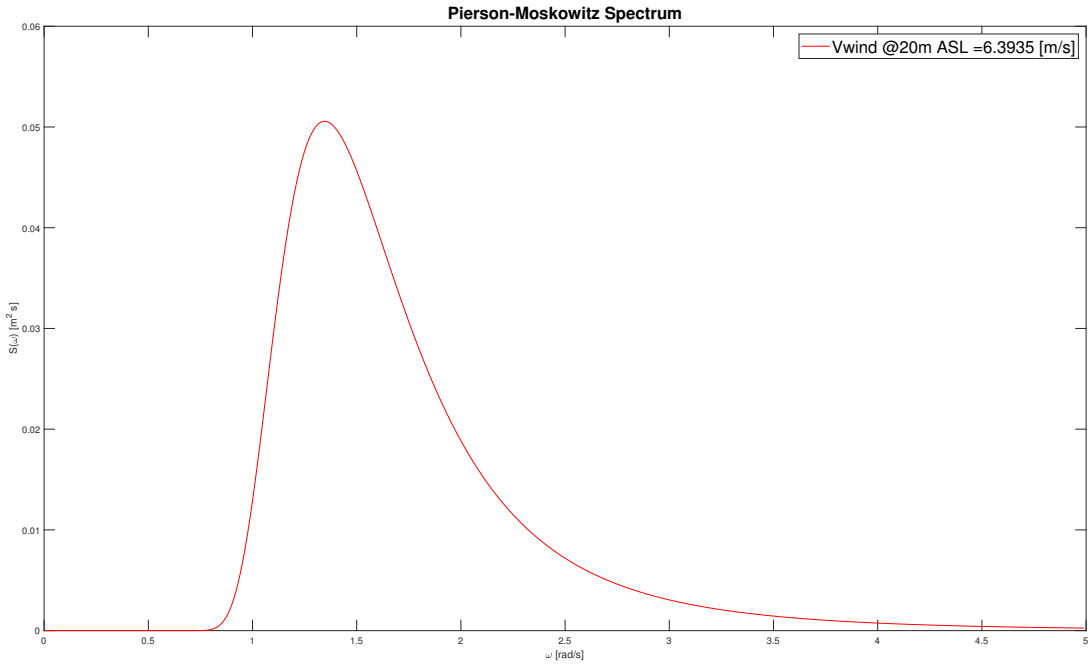


Figure 3.5: PM spectrum for sea state 4

Figure 3.5 shows PM specter for sea state 4. The specter is further used to calculate parameters for equation 3.25.

$$\sigma^2 = m_0 = S(w_0) = S_{max}$$
$$\sigma = \sqrt{S_{max}} \tag{3.27}$$

Further, λ can be calculated by using Matlab command:

```
lscurvefit('Slin',0.1,\omega,S)
```

where **Slin** is a function from MSS toolbox.

Wave densities are assumed to be equal in all directions. There are made one transfer function for each direction, but parameters inside functions are equal. A constant value is added in all three directions, representing current.

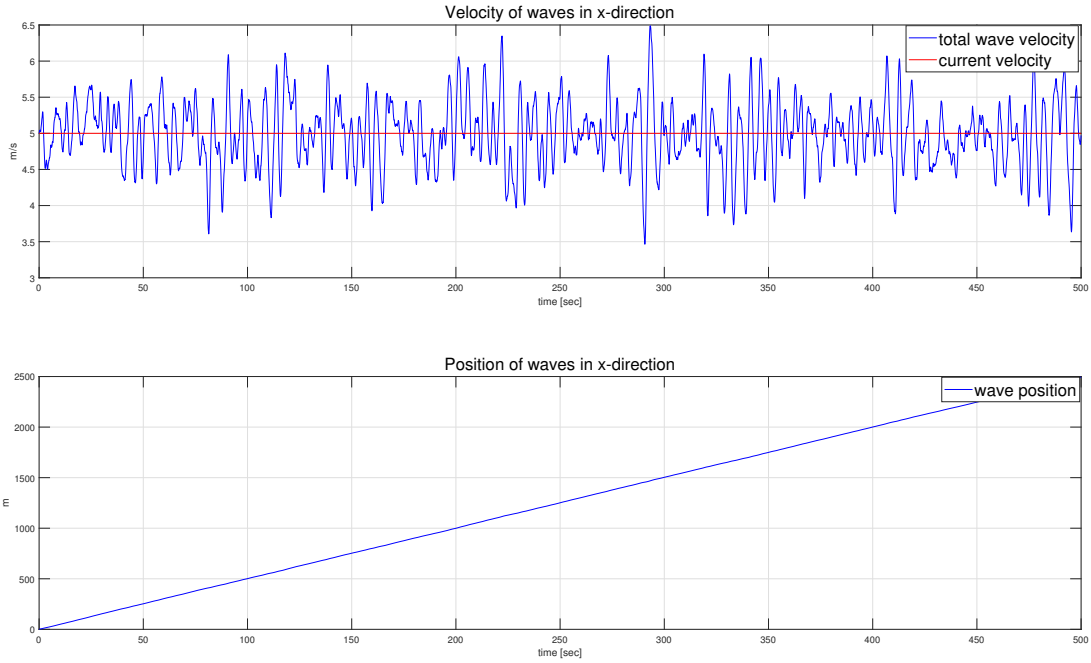


Figure 3.6: Waves in x-direction for sea state = 4

Figure 3.6 shows response of wave simulation for sea state = 4. A current on 5 m/s is added to the simulation shown in the figure.

Chapter 4

Control

4.1 Theory

4.1.1 PID controller

Proportional Integral Derivative controller (PID) is an algorithm in control engineering used to control a process. Algorithm calculates the control signal based on the error between measured process value and the reference value. The main goal is to remove the error between process value and the reference value. The control algorithm contains mainly three parts, which all have a different role in the calculation of the control signal.

- **Proportional** part - receives error between reference and process measurement, and gains the error with a pre-tuned parameter, K_p in order to remove it.
- **Integral** part - sums up error between reference and process measurement over time, and removes the offset based on the error sum. The sum is gained with a pre-tuned parameter, K_i.
- **Derivative** part - calculates error slope based on the error derivative. This part is used to remove error between process measurement and reference faster. Error slope is gained with a pre-tuned value, K_d.

The final control value is a sum of all three parts. Other combinations of the named parts are also possible, such as P, PI or PD.

$$U = K_p \cdot e(t) K_i \int_0^t e(t)\, dt + K_d \cdot \frac{de(t)}{dt} \tag{4.1}$$

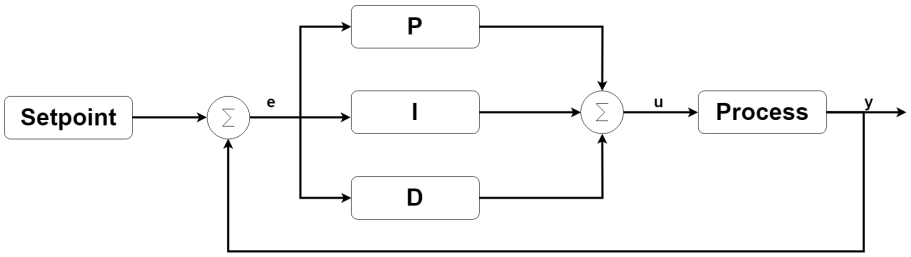


Figure 4.1: PID controller block diagram

Adding a feed-forward gain with desired velocity, can help the control loop to predict the next error. The feed-forward is added outside the feed-back loop, and will therefore not influence the control loop stability. Feed-forward can also added to control loops to remove the disturbance faster.

4.2 Method

There are six different controllers implemented in the total in this project. All of the controllers are different versions of PID control algorithm, described i section 4.1.1.

4.2.1 Altitude controller

Altitude controller is a PID controller that compares desired altitude with current altitude measurement. The output signal of this controller is named u_1.

$$u_1 = K_{pz}e_z(t) + K_{dz}\dot{e}_z(t) + K_{iz}\int e_z(t)dt \tag{4.2}$$

where $e_z(t) = z - z_d$.

The controller consist two equal controllers, where one takes desired position as input, while the other takes integral of the desired velocity as input. Reason why the controller is designed this way is described later in the report. The controllers are not designed to work simultaneously. Since integral of velocity and position are the same input, the controllers are tuned equally (see section 5.4.4).

4.2.2 Attitude controllers

The attitude control contains three controllers, one for each rotation about the body axis, roll, pitch and yaw. As the attitude controllers are the last part of the horizontal control cascade, it is important that they are designed properly. All three controllers are using the PID algorithm. Reference for the roll and pitch controller is generated by mapping structure, described in section 4.2.4. Reference for the yaw controller is fed in directly. Yaw controller is not used in this project, since direction of the hexacopter movement is not a significant factor. Names of the roll, pitch and yaw control signals are u_2, u_3 and u_4, respectively.

Roll

$$u_2 = k_{p\phi}e_\phi(t) + k_{d\phi}\dot{e}_\phi(t) + k_{i\phi}\int e_\phi(t)dt \tag{4.3}$$

Pitch

$$u_3 = k_{p\theta}e_\theta(t) + k_{d\theta}\dot{e}_\theta(t) + k_{i\theta}\int e_\theta(t)dt \tag{4.4}$$

4.2.3 Horizontal controller

Horizontal controller contains two controllers, one for movement in x-direction and another for movement in y-direction. Both controllers are designed equally, with tuning constants being the only difference. Control algorithm used for the horizontal control is called Proportional Integral Velocity (PIV). PIV control differs from PID control in the way that the D - part is omitted from the position controller, and added as a velocity controller. Horizontal controller is a cascade with the first part being a PI position controller and the second part being a PI velocity controller. Both controllers has a feed-back loop, feeding back measured position to the position controller and measured velocity to the velocity controller. Reference signal of the position controller is the platform position (see platform model in section 3.4), while the reference signal for the velocity controller is the output of the position controller. In addition, feed-forward velocity is added to the velocity control loop, where platform velocity is the fed signal. Output of the horizontal controllers are horizontal forces, which can easily be transformed to accelerations and used as reference values of the attitude controllers after a mapping is done (section 4.2.2 and 4.2.4).

This type of control gives a faster response, since velocity is both controlled and fed forward. The goal is to design a controller that can respond fast to platform maneuvers.

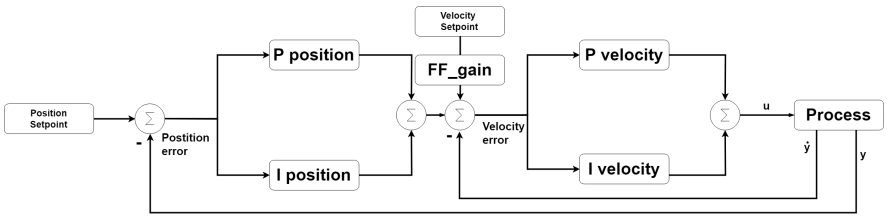


Figure 4.2: PIV controller block diagram

4.2.4 Mapping between controllers

As mentioned in section 4.2.2 a mapping has to be done between horizontal movement and rotation angles of the UAV. The horizontal controllers generates desired horizontal forces, but the actual movement is performed by torques about UAV's body axis and thrust forces created by the propellers. Mapping between horizontal acceleration and roll and pitch rotations will be described in this section.

From equations 3.2.4 and 3.21 we have:

$$\begin{aligned} \ddot{x} &= -\frac{f_{drag_x}}{m}x + \frac{1}{m}(cos\phi cos\psi sin\theta + sin\phi sin\psi)u_1 \\ \ddot{y} &= -\frac{f_{drag_y}}{m}y + \frac{1}{m}(cos\phi sin\psi sin\theta - sin\phi cos\psi)u_1 \end{aligned} \tag{4.5}$$

Assuming equilibrium, the drag forces are equal to zero. Equation 4.5 than becomes:

$$\begin{aligned} \ddot{x} &= \frac{1}{m}(cos\phi cos\psi sin\theta + sin\phi sin\psi)u_1 \\ \ddot{y} &= \frac{1}{m}(cos\phi sin\psi sin\theta - sin\phi cos\psi)u_1 \end{aligned} \tag{4.6}$$

Further, small roll (ϕ) and pitch (θ) angles are assumed, as UAV is assumed to be stabilized and at equilibrium during these calculations.

Assumption that $\phi = 0$ and $\theta = 0$ gives:

$$\begin{aligned} \ddot{x} &= \frac{u_1}{m}(cos\psi\theta + sin\psi\phi) \\ \ddot{y} &= \frac{u_1}{m}(sin\psi\theta - cos\psi\phi) \end{aligned} \tag{4.7}$$

At equilibrium, when the UAV is hovering over a place, a known fact is that the altitude control signal $u_1 = mg$.

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = g \begin{bmatrix} sin\psi & cos\psi \\ -cos\psi & sin\psi \end{bmatrix} \begin{bmatrix} \phi \\ \theta \end{bmatrix} \tag{4.8}$$

Calculating the inverse of the 2x2 matrix in equation 4.8, equation that maps desired roll and pitch angle from horizontal accelerations can be found.

$$\begin{bmatrix} \phi_d \\ \theta_d \end{bmatrix} = \frac{1}{g} \begin{bmatrix} sin\psi & -cos\psi \\ cos\psi & sin\psi \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} \tag{4.9}$$

As the mapping is assuming a UAV at equilibrium, yaw angle is assumed to be zero in equation 4.9, which gives a more simplified mapping between horizontal accelerations an roll and pitch angles.

$$\begin{bmatrix} \phi_d \\ \theta_d \end{bmatrix} = \frac{1}{g} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} \tag{4.10}$$

$$\phi_d = -\frac{1}{g}\ddot{y} \tag{4.11a}$$

$$\theta_d = \frac{1}{g}\ddot{x} \tag{4.11b}$$

Even that equation 4.11 gives the desired mapping, equation 4.9 has been implemented to the model, because it keeps the opportunity for future implementation of the heading control open.

Chapter 5

Simulation

5.1 Parameters

5.1.1 Inertia

Using parameters from table 2.1 to find moments of inertia about all three body axis of the hexarotor. For simplicity reasons, motors are assumed to be point loads attached to mass less arms. Body is assumed to be a cube, with mass evenly spread around the whole volume. From figure 2.1 it can be seen that the assumptions makes sense, as the body has approximately cubical shape and the motors are a lot smaller then the body itself.

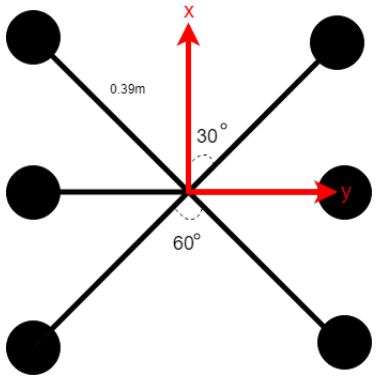


Figure 5.1: Dimensions and angles of hexarotor

Inertia about each axis is calculated using formulas from [21].

$$\begin{aligned} I_{xx} &= 4(lsin30°)^2 m_{motor} + 2l^2 m_{motor} + \frac{m_{body}}{12}(h^2 + w^2) \\ &= 4(0.39 \cdot sin30°)^2 \cdot 0.155 + 2 \cdot 0.39^2 \cdot 0.155 + \frac{3.7}{12}(0.18^2 + 0.28^2) \\ &= 0.023576 + 0.047151 + 0.034163 \\ &= 0.10489 \text{ kgm}^2 \end{aligned} \tag{5.1}$$

$$\begin{aligned} I_{yy} &= 4(lcos30^\circ)^2 m_{motor} + \frac{m_{body}}{12}(h^2 + l^2) \\ &= 4(0.39 \cdot cos30^\circ)^2 \cdot 0.155 + \frac{3.7}{12}(0.18^2 + 0.4^2) \\ &= 0.070727 + 0.059323 \\ &= 0.13005 \text{ kgm}^2 \end{aligned} \tag{5.2}$$

$$\begin{aligned} I_{zz} &= 4 \cdot l^2 \cdot m_{motor} + \frac{m_{body}}{12}(w^2 + l^2) \\ &= 4 \cdot 0.39^2 \cdot 0.155 + \frac{3.7}{12}(0.28^2 + 0.4^2) \\ &= 0.141453 + 0.073507 \\ &= 0.21496 \text{ kgm}^2 \end{aligned} \tag{5.3}$$

Total inertia matrix than becomes:

$$I = \begin{bmatrix} 0.10489 & 0 & 0 \\ 0 & 0.13005 & 0 \\ 0 & 0 & 0.21496 \end{bmatrix} \tag{5.4}$$

5.1.2 Thrust constant

Mapping between input variables and angular velocities is shown in equation 3.18. Thrust constant b can be found by looking at the relation between maximum angular velocity of one motor Ω_{max} and maximum value of the control signal $u_{1_{max}}$, obtained in section 5.2.3 and 5.2.4.

$$\begin{aligned} u_{1_{max}} &= 6 \cdot b \cdot \Omega_{max}^2 \\ b &= \frac{u_{1_{max}}}{6 \cdot \Omega_{max}^2} \\ &= \frac{124}{6 \cdot 953^2} \qquad = 2.28 \cdot 10^{-5} \text{ Ns}^2 \end{aligned} \tag{5.5}$$

5.2 Limitations

5.2.1 Angular rate

Since this project only takes care of simulation model and does not include real testing with the UAV, it was hard to measure maximum angular rate about body axis. An approximation is done by looking at another, similar UAV in [22]. Maximum angular rate is sat to be less then the reference drone, to be sure that the limitations are within valid range of hexacopter used in this project.

Maximal angular rate for the UAV is sat to be $130\frac{\text{deg}}{\text{sec}}$ about all three body axis. Since rotation about z axis ψ is not considered in this project, the interesting rotation rate limitations are:

$$\dot{\phi}_{max} = \dot{\theta}_{max} = \pm 130 \ \frac{\text{deg}}{\text{sec}} \tag{5.6}$$

5.2.2 Maximal vertical velocity

From [1], maximal vertical velocity is listed to be 6 m/s in ascend direction and 4.5 m/s in descend direction. One has to be aware of the fact that [1] is using other type of motors, and that vertical velocity limitations for hexacopter used in this task may deviate from ones implemented in this project. Still, an assumption that the deviation is negligible is made.

$$\begin{aligned} v_{z_{max,ascend}} &= 6 \ \frac{\text{m}}{\text{sec}} \\ v_{z_{max,descend}} &= 4.5 \ \frac{\text{m}}{\text{sec}} \end{aligned} \tag{5.7}$$

5.2.3 Control Inputs

Thrust

u_1 is control input for the controller that controls vertical movement of the UAV. Control output of the controller is thrust force (section 5.1.2). Maximal thrust for each motor is 2.1 kg. [13], which tells us that each rotor can lift 2.1 kg weight. Since the hexacopter is equipped with six rotors, the maximum total thrust becomes:

$$T_{max} = u_{1_{max}} = (2.1 \cdot 6)kg = 12.6kg \approx 124 \text{ N} \tag{5.8}$$

Roll

Maximum roll input is also obtained by using relation from equation 3.18 and parameters from section 5.2.4 and 5.1.2. In addition, parameters from table 2.1 are used.

$$\begin{aligned} u_{2_{max}} &= bl\Omega_{max}^2 + 2 \cdot \frac{bl}{2}\Omega_{max}^2 \\ &= 2bl\Omega_{max}^2 \\ &= 2 \cdot 2.28 \cdot 10^{-5} \cdot 0.39 \cdot 953^2 = 16.15 \text{ Nm} \end{aligned} \tag{5.9}$$

Pitch

Same equations and tables obtained as in calculation of maximal roll input.

$$\begin{aligned} u_{3_{max}} &= 2 \cdot \frac{bl}{2}\sqrt{3}\Omega_{max}^2 \\ &= bl\sqrt{3}\Omega_{max}^2 \\ &= 2 \cdot 2.28 \cdot 10^{-5} \cdot 0.39 \cdot \sqrt{3} \cdot 953^2 = 13.99 \text{ Nm} \end{aligned} \tag{5.10}$$

5.2.4 Maximal angular velocity of motors

Maximal angular velocity of each rotor is calculated analytically by including known limitations for the hexacopter combined with equations that maps input parameters with propeller rotations (equation 3.18).

$$\Omega_{max} = KV \cdot V_{max} \frac{2\pi}{60} \tag{5.11}$$

where KV is motor constant in $\frac{\text{rpm}}{\text{V}}$, V_{max} is maximum working voltage and $\frac{2\pi}{60}$ is conversion factor. Maximum motor angular velocity is calculated to be:

$$\Omega_{max} = 350 \cdot 26 \frac{2\pi}{60} = 953 \, \frac{\text{rad}}{\text{sec}} \tag{5.12}$$

5.3 Simulink model

The Simulik model shows connections between different parts of the system graphically. The system is large, with a lot of details. For simplicity reasons, and to make the system more understandable, it is divided in so called *subsystems*. Model with subsystems and connections between those is shown in Figure 5.2.

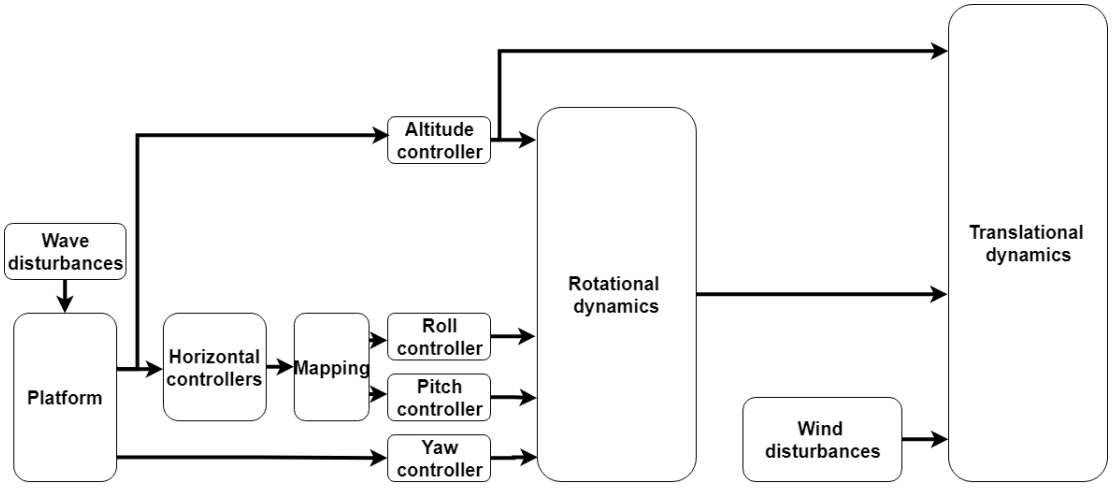


Figure 5.2: Simplified Simulink model

5.4 Tuning

5.4.1 Roll controller

Roll controller, designed as described in equation 4.2.2, has been assigned control parameters listed in table 5.2. Control parameters are tuned after trial-error approach.

Table 5.1: Tuning parameters roll controller

Parameter	Value
K_p	18.9
K_i	5.6
K_d	8.8

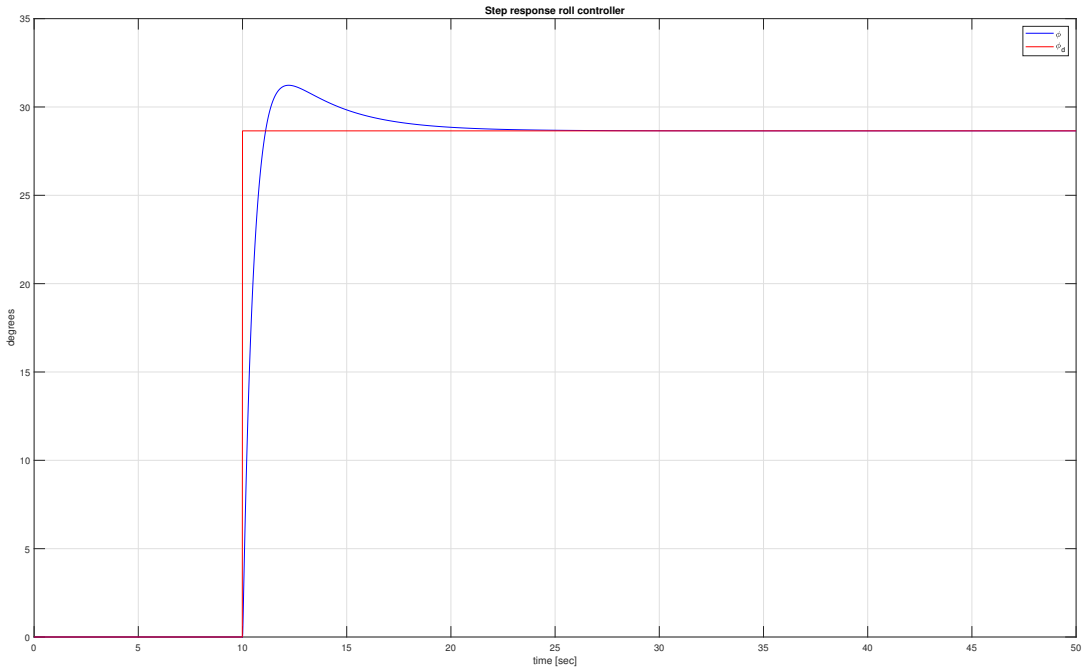


Figure 5.3: Step response roll controller

5.4.2 Pitch controller

Because inertia is almost equal about x and y axis (section 5.1.1), control parameters are tuned equally.

Table 5.2: Tuning parameters pitch controller

Parameter	Value
K_p	18.9
K_i	5.6
K_d	8.8

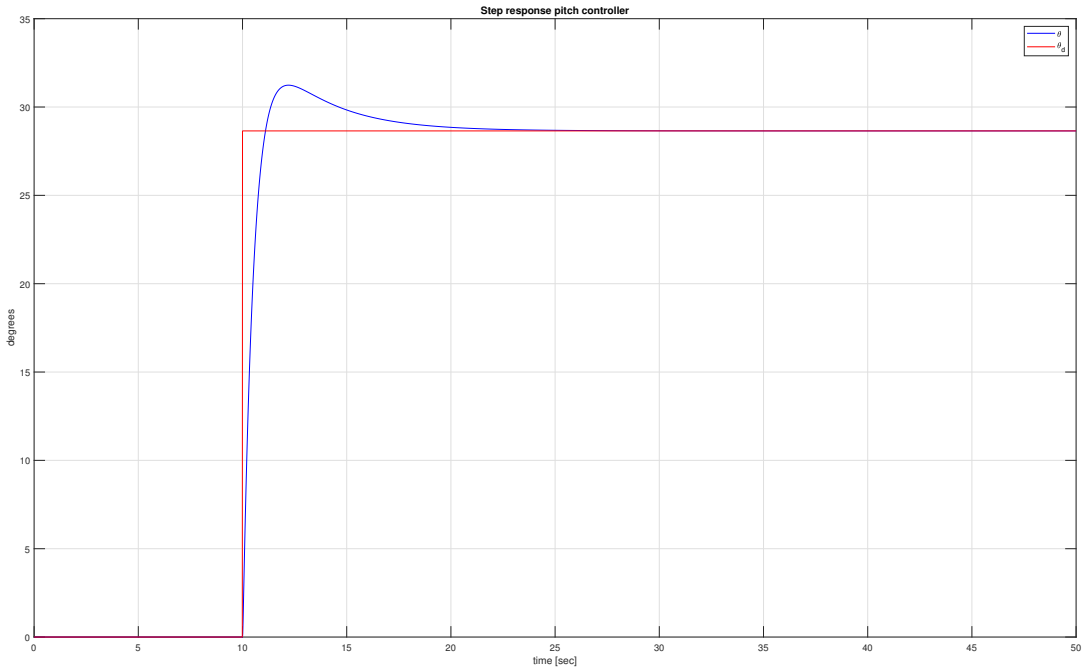


Figure 5.4: Step response pitch controller

Both pitch and roll controller give a satisfying response. Tuning controllers became harder after the limitations from section 5.2 was considered. The limitation that decides tuning parameters is maximal angular rate from equation 5.6. With higher maximal rate, it would be possible to tune a more aggressive controller, which gives a faster response.

5.4.3 Horizontal controllers

Horizontal controllers are tuned with two set of variables. First set of variables are tuned to make a stable, but slow varying process. The other set are tuned in to make the controller aggressive, and give a process that responds fast to changes. Downside with the slow varying process is that it takes time to get back to the desired position, while downside with the aggressive process is that it can get unstable quickly.

The idea is to change variable sets depending on which state the UAV is located in (Section 5.5.3).

After tuning, following parameters were found to fit a robust controller:

Table 5.3: Parameters of the robust horizontal controller

Parameter	Value x-controller	Value y-controller
K_p	0.400	0.300
K_i	0.058	0.058
$K_{p_{velocity}}$	0.880	0.780
$K_{i_{velocity}}$	0.180	0.080
$K_{feed-forward}$	1.000	1.000

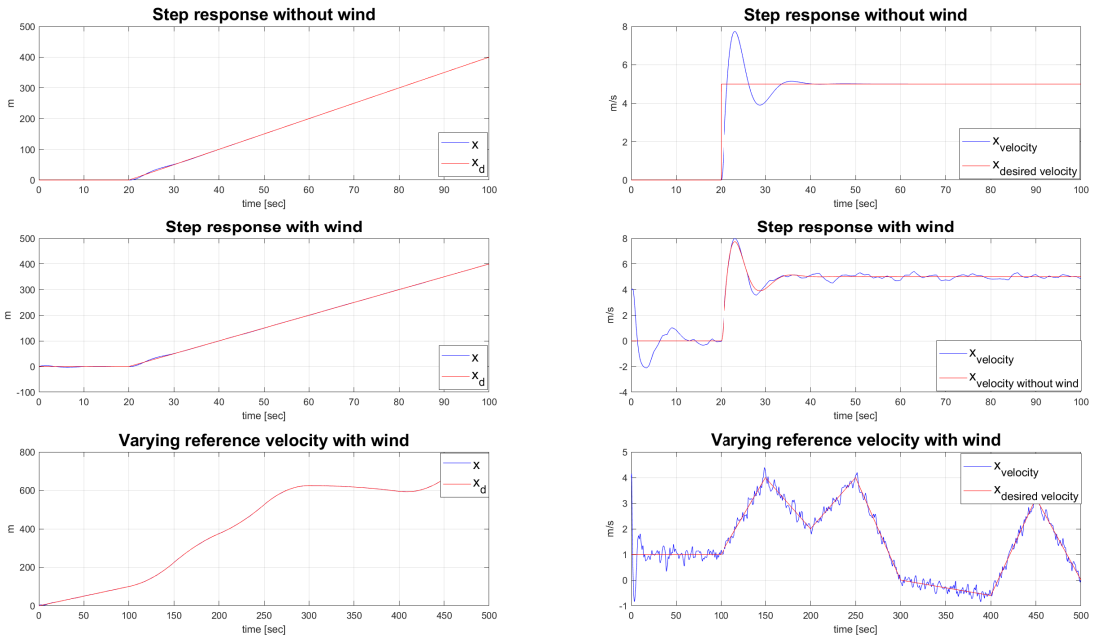


Figure 5.5: Responses of robust x-controller

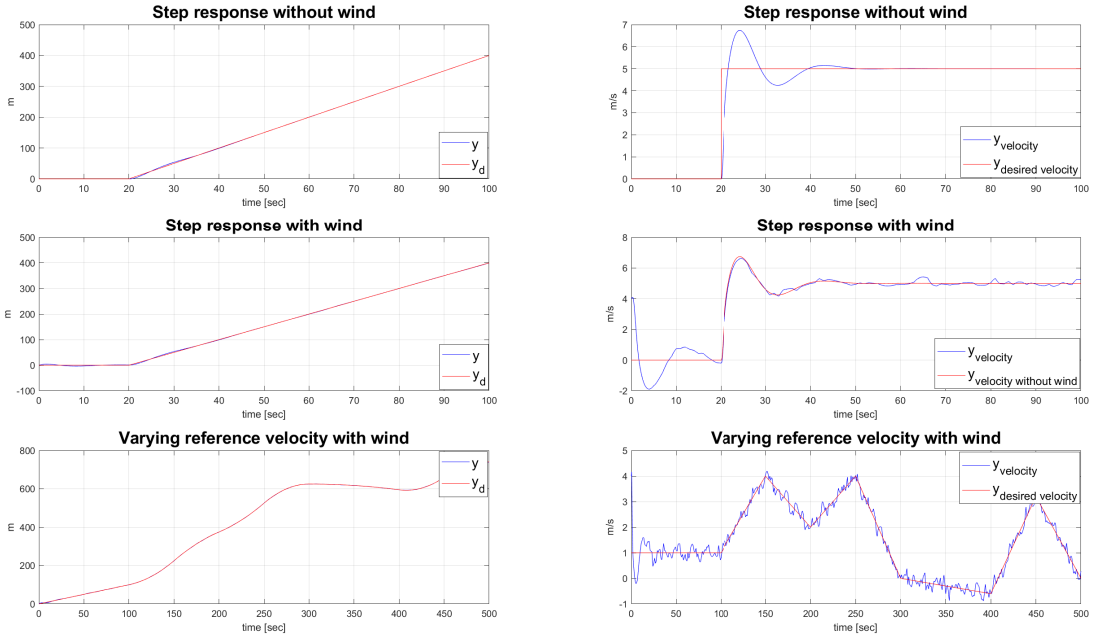


Figure 5.6: Responses of robust y-controller

Figures 5.5 and 5.6 shows different responses of the robust horizontal controllers. Responses are applied in the velocity, while desired position is simply a integral of desired velocity. In addition, wind is added to the two lowermost rows in both figures. The wind has a constant velocity on 4 m/s in all three directions. Wind implementation is described in section 3.3.

It can be noticed that controllers in both directions give almost perfect tracking of the position. Still, the interesting part is looking at the velocity responses, especially with the wind present. Both controllers creates small oscillations that damps out the error smoothly. It is also noticeable that the velocity starts in 4 m/s in the plots with wind present. The reason is simply just the constant wind velocity.

Table 5.4: Parameters of the aggressive horizontal controller

Parameter	Value x-controller	Value y-controller
K_p	0.250	0.450
K_i	0.088	0.098
$K_{p_{velocity}}$	4.790	4.630
$K_{i_{velocity}}$	0.380	0.380
$K_{feed-forward}$	1.100	1.150

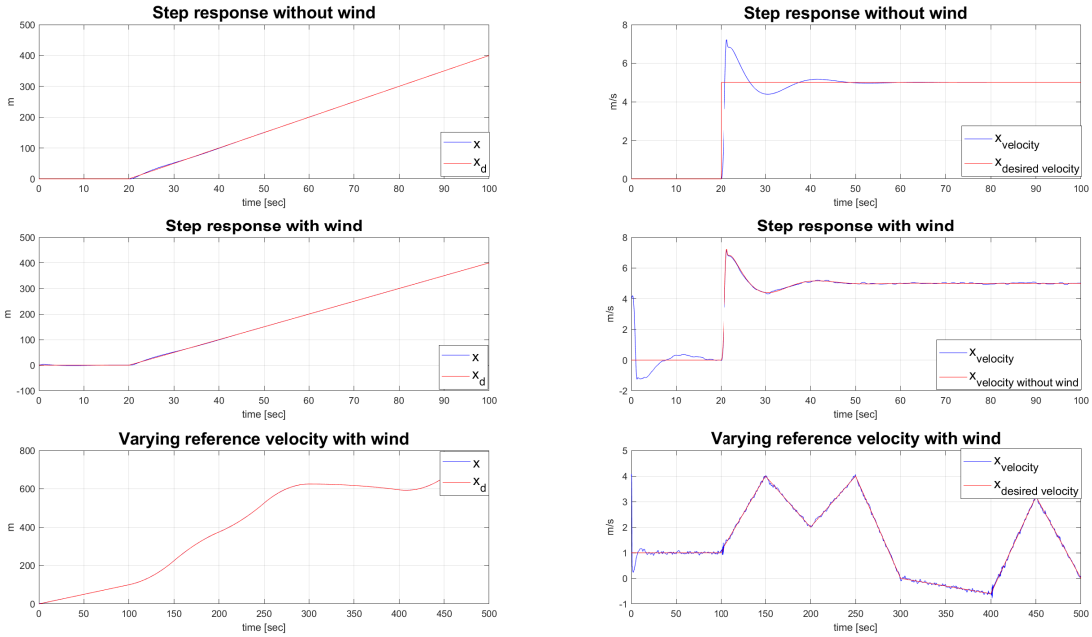


Figure 5.7: Responses of aggressive x-controller

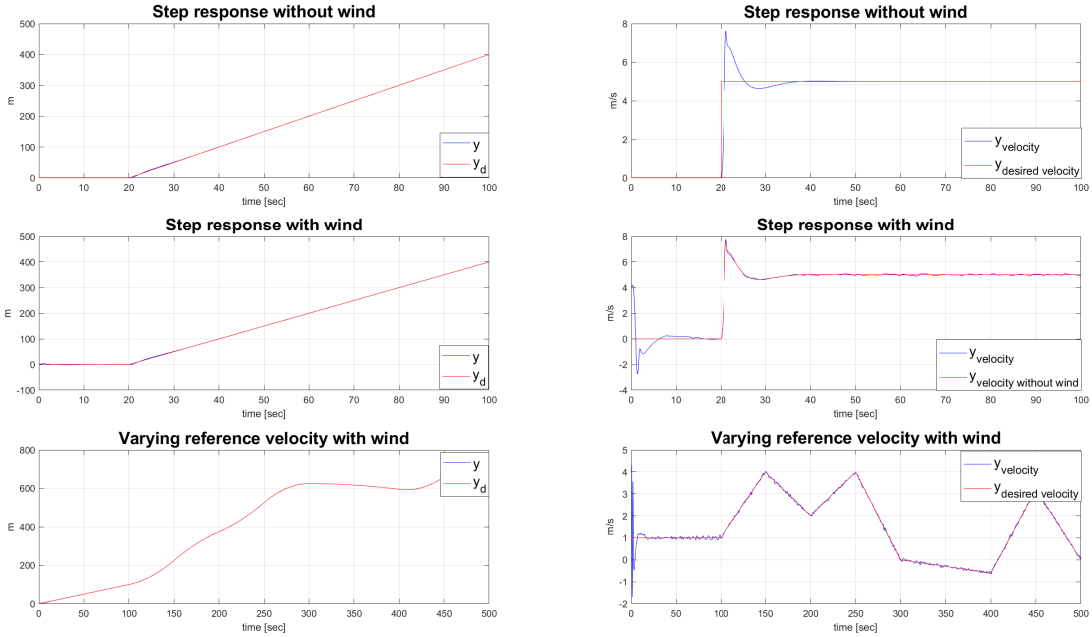


Figure 5.8: Responses of aggressive y-controller

Figures 5.7 and 5.8 shows responses of the aggressive horizontal controllers. The overshoot edges of the aggressive controllers are pointy, compared to the robust controller. The aggressive controllers tries to kill the error as fast as possible. It is also noticeable that the aggressive controller is more efficient in demolishing disturbance oscillations. Although the robust con-

troller gives faster response it has a downside. Sharp maneuvers of UAV in the horizontal plane, which are undesirable during descend movement, can cause instability.

Tuning parameters of the aggressive controllers are listed in table 5.4.

5.4.4 Altitude controller

The altitude controller is tuned with only one set of variables, unlike the horizontal controllers. Following parameters are found to give satisfying response:

Table 5.5: Tuning parameters altitude controller

Parameter	**Value**
K_p	4.7
K_i	0.23
K_d	44.82

5.4.5 Maximal wind

Several simulations are done in order to check the wind tolerance of the hexacopter. [1] says that maximal wind/gust the UAV can tolerate is 25 miles per hour, or approximately 11 meters per second. It is important to be skeptical about the information on the manufacturer website, as it is most likely underestimated a lot in order to avoid problems with costumers liking to pushing the limits and breaking the product.

Tests are performed with both sets of horizontal controllers (see section 5.4.3).

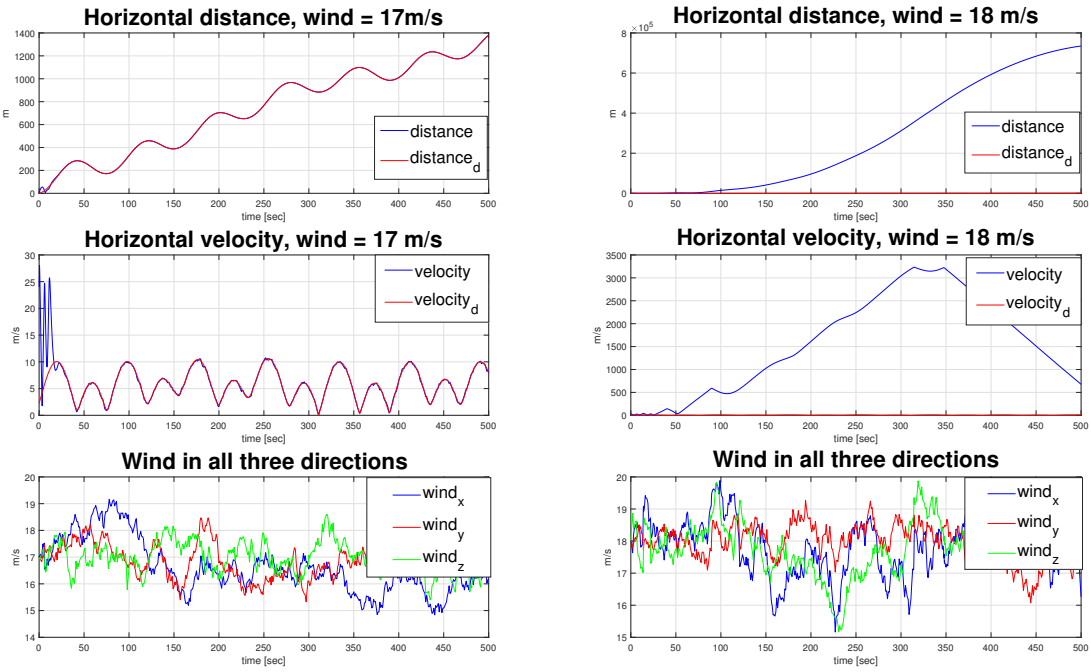


Figure 5.9: Wind tolerance test of the robust controller

Figure 5.9 shows a wind test performed on the robust controller. The test is simulated with same wind intensity in all three directions. Wind is modeled as described in section 3.3.

Left hand side shows response when a wind with constant velocity on 17 m/s is added, while the right hand side shows response with 18 m/s constant wind. It is noticeable that the controller manages to resist the 17 m/s wind, but not 18 m/s.

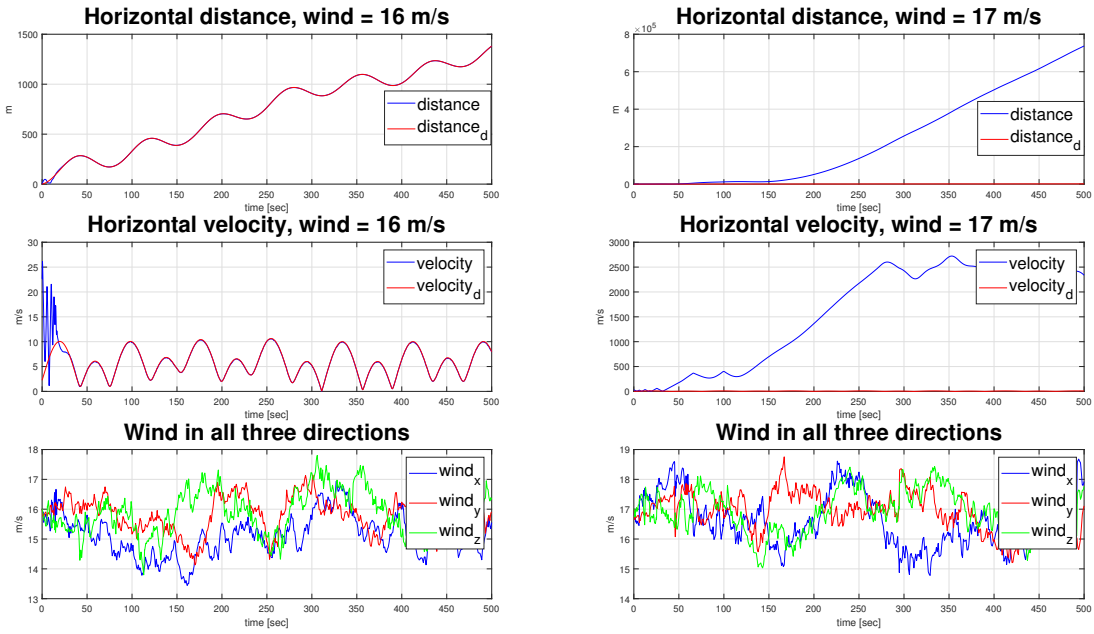


Figure 5.10: Wind tolerance test of the aggressive controller

Figure 5.10 shows a wind tolerance test performed on the aggressive controller. The same procedure is done, as in the test performed on the robust controller. As expected, the aggressive controller tolerates less wind than the robust. From the simulation results, it can be seen that maximal constant wind the aggressive controller can tolerate is 16 m/s, which is 1 m/s less than the robust controller.

Since both controllers are used during a landing (section 5.5.3), the conclusion is that the maximal constant wind during a landing can be no higher than 16 m/s. According to Beafurt scale [23] 16 m/s is considered as **High wind**, which brings whole threes in motion on land, and heaps up the sea. Another interesting fact is that the maximal wind speed found by simulation only deviates 5 m/s from the manufacturer advice. [1]

5.5 Landing

Landing itself is divided into different states. Different control parameters are used in different states. State allocation depends on boundary violations and vertical velocity. Imaginary boundaries are made in order to do the landing precise and prevent failure.

5.5.1 Boundaries

The landing algorithms are developed to always see three imaginary cylinders which extends straight up from the platform. All three cylinders are centered over the landing platform (origin of the inertial frame). The size of the cylinders are different. The main purpose of these cylinders is to make imaginary boundaries that decides how far away from the center of platform the UAV is allowed to be during the landing. Depending on how close the UAV is the platform in vertical direction, the criteria of horizontal error relative to the platform changes. Simplified,

the closer the UAV is the platform, both horizontal and vertical direction, the smaller is the cylinder that determines boundaries.

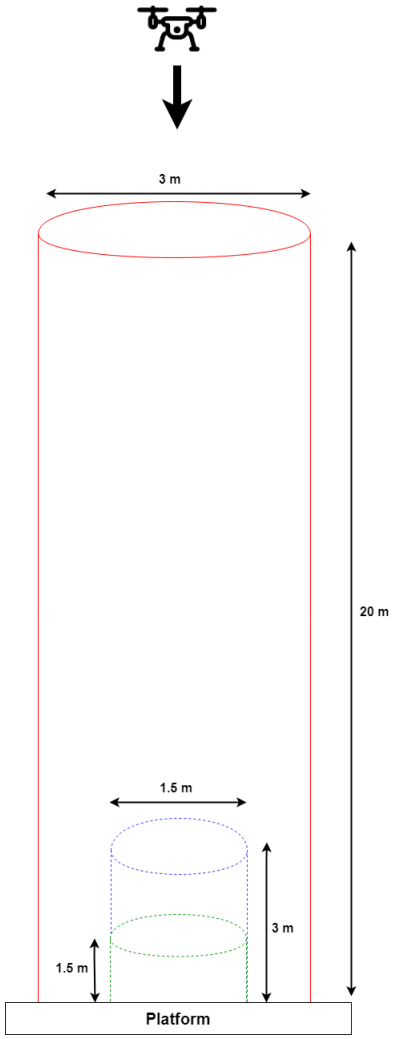


Figure 5.11: Cylinder boundaries illustrated

The largest cylinder (red) has diameter three times hexacopter length, including arms, which is approximately 3 meters. That gives the UAV opportunity to move one length to the side, and still stay inside the cylinder. Height of the largest cylinder is 20 meters, which is the decided hovering altitude of the hexacopter. The largest cylinder is typically used as boundary during descend movement between hovering and landing states.

Blue cylinder is the medium size cylinder, and has the diameter two times hexacopter length, including arms. That allows the hexacopter to deviate by a half length relative to the center of landing platform. Height of the medium cylinder is three meters. The cylinder is used as boundary to make sure that the UAV is ready for landing.

The smallest cylinder, green, has the same diameter as the medium cylinder. Height of this cylinder is 1.5 meters. The cylinder is used during the landing state.

Note that the cylinder dimensions are not final. The three cylinders above describes the main principle of setting boundaries during the landing. Final dimensions of the cylinders will be set after testing.

5.5.2 State machine

As the UAV approaches the platform with downwards velocity, it moves through different states. Each state has own requirements when it comes to the importance of holding position and reaction time. In addition, there exist requirements for entrance to every state as well. Based on requirements of the different states, a state machine is designed to decide which state that is the next the UAV should move to during a landing approach. The state machine contains in total seven states, where each state has it's own set of control parameters.

A flow chart of the state machine is presented in Appendix A.

The whole state machine will now be described. Every state will be described separately. Note that the vertical velocities in the different states are not final, and can be changed.

1) Hover

Hover is the state where UAV flies right over the platform position on a given height, waiting for a landing signal. When the landing signal is given, the UAV has to get inside the largest cylinder, in order to move to the next state. It is only possible to move further to one state from **Hover**, which is **Down**. So, when the distance gets smaller than the radius of the largest cylinder, the state machine switches state to **Down**.

2) Down

Name of the state where UAV is approaching the platform in constant downwards velocity is **Down**. The approach velocity is sat to be -0.6 m/s. When in this state, the state machine continuously checks if the UAV is within the imaginary boundaries. If the UAV is inside the medium cylinder (both radius and height), the state changes to **Final**. In other case, if the UAV is under the height of the medium cylinder, but in between the walls of the medium and large cylinder, the state changes to **Up**.

The last case considers violation of boundaries represented by the large cylinder. A violation that big can be critical, and the state changes to **Up fast**.

3) Up

If a boundary somehow is violated, the state machine is built in the way that it always sends the UAV up one state. State **Up** is the state that does the transportation from a state to previous. At this state, upwards velocity is sat to be constant at 0.6 m/s. If the UAV is in **UP** state, and the altitude is between the height of the large and medium cylinder in addition to the horizontal position being inside the large cylinder, the state changes to **Down**. If altitude is higher than the height of the large cylinder, the state changes to **Hover**, and the whole landing process starts over again.

4) Up fast

Up fast is an emergency state used when boundary represented by the walls of the large cylinder are violated. Upwards velocity in this state is sat to 2 m/s in order to get fast up to a safer point. It is only possible to move to **Hover** state from **Up fast state**. That happens when the altitude is above the largest cylinder height.

5) Final

This is the state where the state machine makes sure that the UAV manages to stay inside the medium cylinder for a while. Downwards velocity is -0.4 m/s in this state. If the UAV violates boundaries represented by walls of the medium cylinder, state changes to **Up**. If boundaries of the large cylinder are violated, state changes to the emergency state, **Up fast**. At the end, if UAV manages to get inside the smallest cylinder, state changes to **Land**.

6) Wait for landing

To complete a safe landing , the UAV has to be sure that its landing on the platform that does not move towards the UAV. The velocity is sat to zero, and the vehicle is hovering at very low altitude while waiting for landing signal from algorithm in section 5.5.4. When the landing signal arrives, state changes to **Land**.

7) Land

Land is the last state of the state machine, and is a state with no return. Landing velocity is sat to be -0.1 m/s. All motors turns off about 20 cm over the platform, as the landing is completed.

5.5.3 Parameter allocation

The UAV is planned to fly in rough environment with large disturbances, often represented by strong winds. Two sets of control parameters are tuned in, where the difference between them is the ability to resist disturbances. The first set of parameters are tuned to make a robust controller, while the second set is tuned to make an aggressive controller (section 5.4.3).

Usage of the sets depends on which state of the landing process the UAV is located in. In states where the boundaries are represented by walls of the largest cylinder, it is more important having a robust controller than having a aggressive controller. Radius of the large cylinder gives the UAV time to get back to the desired position, when dragged away by the disturbances. Therefore, a robust, slow varying controller can be used. On the other hand, when boundaries are reprensented by walls of the smallest cylinder, it is important having a fast, aggressive controller, with fast reaction to disturbances.

Control parameters are allocated depending on the state UAV is located in during the landing process. Robust controllers are used in: **Hover, Down, Up** and **Up fast** states. Aggressive controllers are used in: **Final**, **Wait for landing** and **Land** states.

5.5.4 Landing timing

When landing, it is important to make sure that the platform is not moving towards the vehicle itself. Waves causes platform movement in upward direction, called heave. To make sure that the UAV is hitting the platform on the way down, an algorithm is designed.

The algorithm checks the vertical platform position continuously, and remembers the lowest and highest measured platform position. If the platform position is higher than the previous highest or lower than the previous lowest, the algorithm updates the remembered positions. The algorithm checks if current wave is within 0.2 m from the top of the highest measured wave. If the statement is true, landing approval is given.
Figure 5.12 shows a landing using the described algorithm. Notice that vehicle hits the platform on the way down.

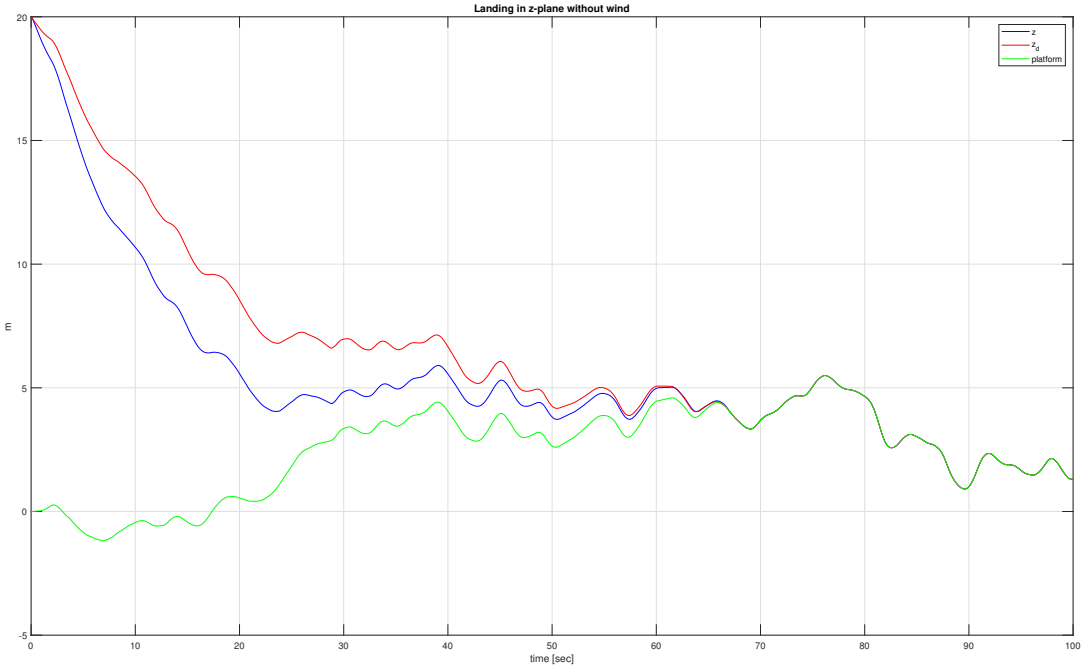


Figure 5.12: Simulation of landing in z-plane without wind present and sea state = 4

Note that landing is simulated without wind present, to make sure that the landing does not get aborted due to wind disturbances. Note that the system is not able to perform landing for sea state 6 and 8, which can cause problem in a realistic situation. The algorithm is far from perfect, as it does not work if the highest waves occurs in the beginning of the simulation. This problem is discussed deeper in section 7.5.

Chapter 6

Results

This chapter presents results of the simulation model represented in Chapter 5.

More simulation results are also presented in Appendix B. All simulations presented in Appendix B are simulated in same weather conditions as simulations presented in this chapter. The only diffrence between simulations is the platform path.

Figure 6.1 shows a simulation of the landing approach in three dimensions. Parameters used in the represented model are listed in table 6.1.

Table 6.1: Parameters used during simulations

Parameter	Value
Sea State	4
Current x	4 m/s
Current y	4 m/s
Current z	4 m/s
Constant Wind x	4 m/s
Constant Wind y	4 m/s
Constant Wind z	4 m/s

Simulation shows that the UAV hovers at very low altitude for a long period, waiting for the landing signal given by the algorithm described in section 5.5.4. A peak, that occurs after the first landing attempt, can be noticed in figure 6.1. This peak happens after the UAV has violated boundaries of the largest imaginary cylinder (see section 5.11). The boundaries are violated because the horizontal controllers are not able to follow the desired path. It takes about $\frac{1}{3}$ of the simulation time to land the UAV, which is approximately 5 minutes. Considering the fact that the initial height is only 20 meters, five minutes is way to much time for performing a landing.

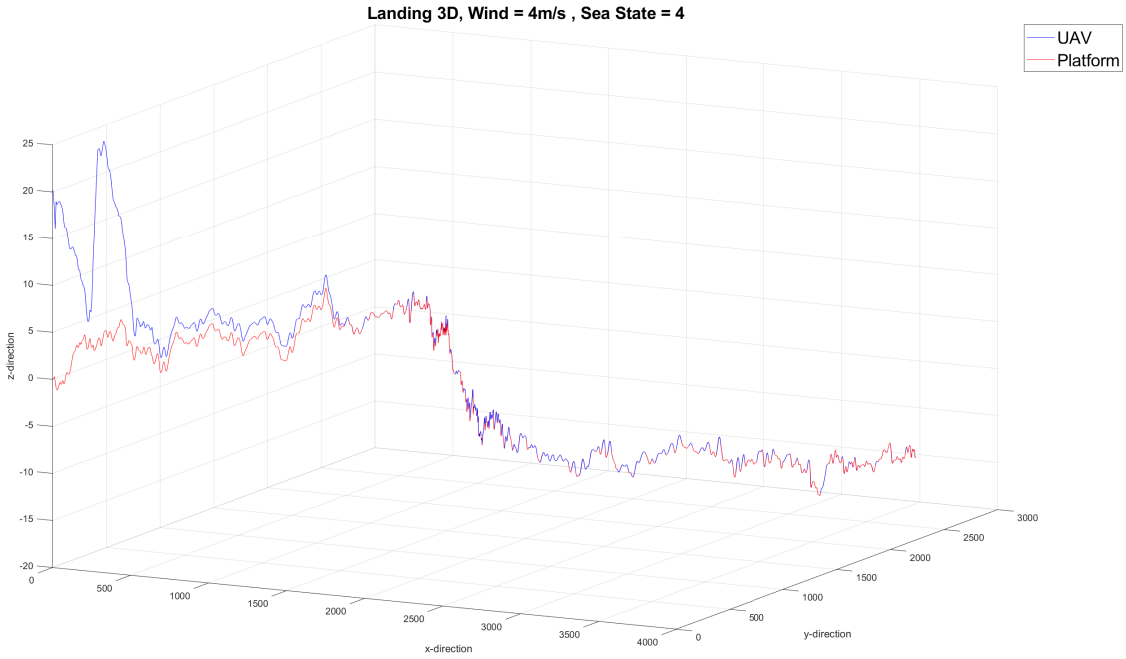


Figure 6.1: Landing shown in 3 dimensions, Wind = 4m/s, Sea State = 4

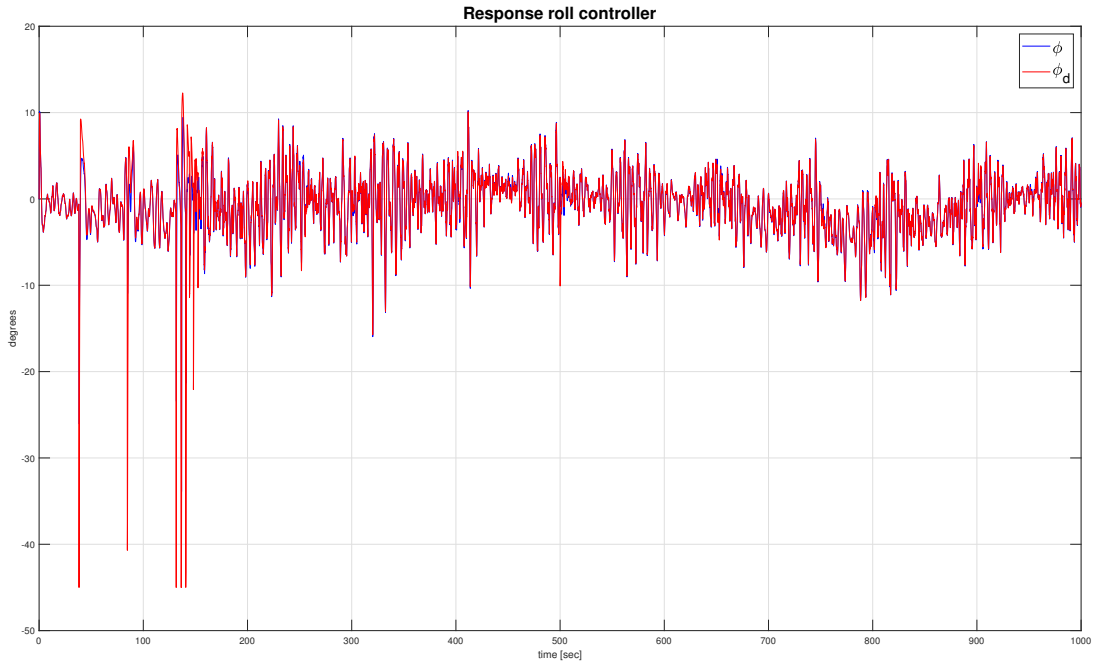


Figure 6.2: Roll response

Figure 6.2 shows the behaviour of the roll angle during the simulation shown in Figure 6.1. It is noticeable that the roll angle is oscillating with very high frequency. Still, it seems like the controllers are able to follow the desired roll position, which is satisfying.

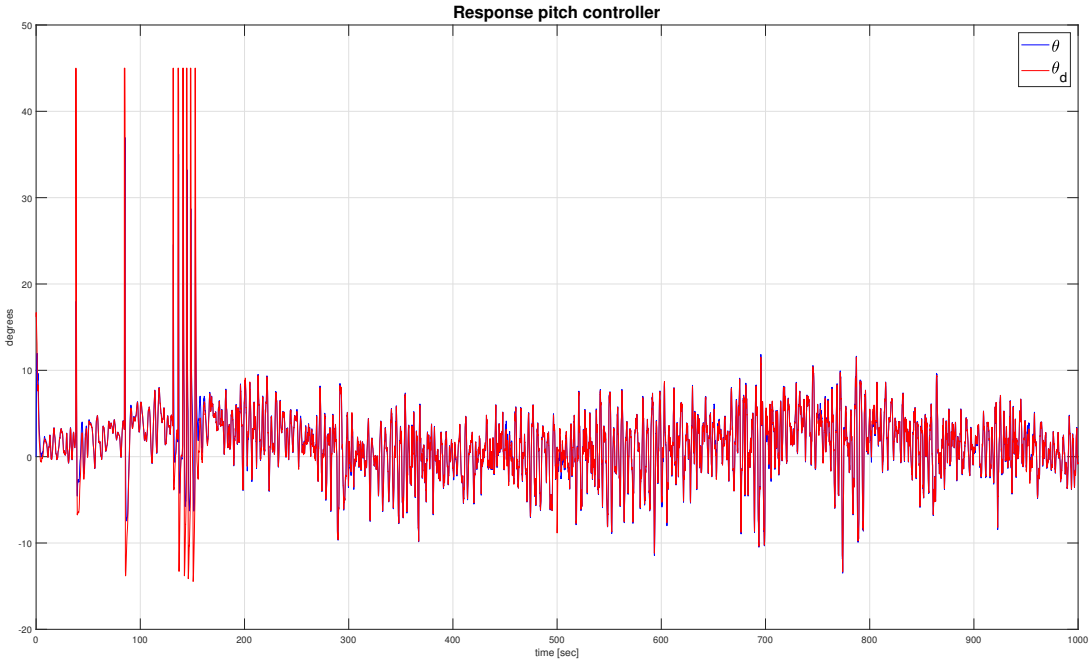


Figure 6.3: Pitch response

Figure 6.3 shows the behaviour of the pitch angle during the simulation shown in Figure 6.1. Like roll controllers, pitch controllers are also able to follow a reference varying with high frequency. The high frequency of the angles comes from the winds, that are present during simulation (see table 6.1).

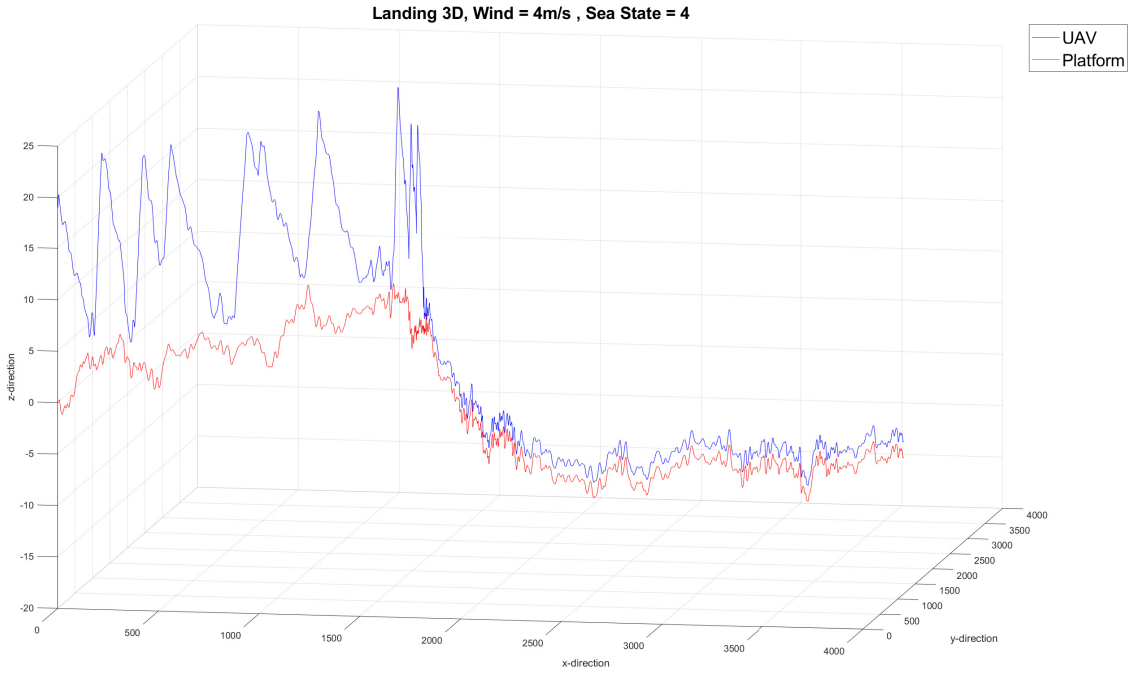


Figure 6.4: Landing shown in 3 dimensions, Wind = 4m/s, Sea State = 4

Another result of the simulation of the landing approach is presented i figure 6.4. The difference from figure 6.1 is that platform path is slightly changed. Still, the change is enough to affect the result. Several more peaks can be observed in figure 6.4 than figure 6.1. The reason is, as named earlier, violation of boundaries represented by the large cylinder described in section 5.5.4. Halfway in the simulation, there are no more peaks, but the UAV still does not perform a landing. The landing does not happen because no landing command is given form the algorithm described in section 5.5.4. The weakens of the current version of the algorithm is discussed deeper in chapter 7.

Chapter 7

Discussion

Simulations of several landing approaches shows that the hexacopter is able to land on a moving platform on water. Still, there exist simulations that shows failed landing approaches in conditions that the hexacopter should be able to land in. The system is large, consisting many small details. Improving one tiny part of the system, can result into better simulation results.

7.1 Controllers

Figures 6.2 and 6.3 shows that the roll and pitch controllers are tuned properly. Roll and pitch angles are able to follow the reference, even it is oscillating with high frequency. Of course, there are limitations on how high the oscillation frequency can get before the desired signal cannot be tracked anymore (section 5.2).

The figures named above also shows that the reference signal is saturated, especially in the beginning. For a perfect modeled system, that should not happen, as the reference signal should not exceed the feasible range of the given angle. The problem of the reference signals are probably horizontal controllers, that are not tuned properly. Even though figures in section 5.4.3 shows almost perfect tracking, one has to be aware of that the boundaries are tight (Section 5.5.1), and that a wind gust can result into violating those, if not properly tuned.

Section 5.5.3 presents how the tuning parameters changes depended on altitude relative to the platform. The chosen approach does the tuning process more complicated as it requires two sets of parameters. It is questionable whether it is necessary to switch between two sets of parameters. If possible, tuning one set of parameters that can yield into both robust and aggressive behaviour will be a good replacement for the alternating sets, used in this project.

PIV control (section 5.4.3) looks like a good control design for horizontal controllers in this problem, as velocity of the platform is available all the time. Still other control designs has to be considered in order to get best possible result.

See section 7.5 for suggestions for further developing of horizontal controllers.

7.2 Wave prediction

A simple algorithm is developed to prevent a landing during platform movement upwards, caused by the waves. If not prevented, such landing can cause into destruction of the hexacopter. As mentioned in section 5.5.4, the main idea with the algorithm is to show that the problem is considered. It is not expected to work perfectly yet, but it would be one of the main parts of the system after further development.

Figure 6.4 shows the weakness of the current landing algorithm. The reason why the landing signal is never given in the simulation shown in figure 6.4 is that the highest waves occurs in the beginning of the process, the following waves never reaches the same height. As a result of that, the algorithm in section 4.2 never sends out the landing permission, and the hexacopter keeps on hovering at low altitude. The algorithm will be improved in future work.

7.3 Boundaries

Boundaries for the landing are described in section 5.5.1. One has to be aware of that dimensions of cylinders in figure 5.11 are just a educated guess, without any test. Most likely, the dimensions has to be adjusted in order to fit the problem. Still, the idea will most likely work, as proved in [7] and [3].

7.4 Modeling

The mathematical model seems to be close to realistic. There already exist models of hexacopter, such as [5] and [4], that are proven to work. This model does not deviate much from the listed ones, such that it is no reason to believe that there are some fatal errors in the model. Still, a problem is noticed when it comes to stabilizing the UAV during a constant wind. When a constant wind acts on the UAV body, and the goal is to hold constant position, one should expect that the UAV will tilt towards the wind direction in order to hold the position. That did not happened in the simulations. The problem probably lies in modeling errors of air forces. Due to the limited time frame of this project, the problem was not investigated deeper.

7.5 Future work

As this project is a preparation for masters thesis, there will be done improvements and further investigations of problems named in section 7.

Further development should begin by an investigation of the air forces and their implementation. An approach to this problem can be reading related literature and comparing models. Hopefully the error will be detected, and the mathematical model will get even more realistic.

Horizontal controllers definitely needs more investigation. First of all, a more systematic tuning approach should be conducted. A suggestion is to find tuning parameters by looking at the stability analysis. There are several approaches for tuning PID controllers listed up in [17] that can be deeper investigated in future development. It is hard to tell which tuning approach that will give the best result. Suggestion is to try different approaches, and compare results. After tuning, the necessity of two set of controllers should be considered.

Other control strategies than PID should also be investigated, especially when it comes to horizontal controllers. A suggestion is to investigate control strategies such as feedback linearization and backstepping, and compare results of those control strategies with the results of PID controllers. Stability of the total system should also be investigated, as finding stability regions can be necessary to decide weather conditions a landing can be performed in. This can be done using Lyapunov stability properties.

Algorithm described in section 5.5.4 has to be developed further, as it is one of the most important parts of the system. It would be interesting to find a correlation between waves and wind, and try to estimate waves. [24] shows an approach of estimation waves using Kalman filter, that will be interesting to investigate deeper.

Boundaries from section 5.11 has to be adjusted, but this is hard to do before testing in real enviorements.

Finally, points that has to be developed further are listed under:

1. Investigate modeling of air forces
2. Tune horizontal controllers by using a systematic tuning approach
3. Investigate other control strategies
4. Do a stability analysis of the whole system
5. Improve landing permission algorithm from section 5.5.4
6. Investigate possibility of developing a wave estimator
7. Adjust boundaries in figure 5.11

Chapter 8

Conclusion

The main goal with this thesis was to build a solid ground of a simulation environment for further development in the masters thesis.

Low-level mathematical models of forces and torques acting on the vehicle are implemented and merged together to a simulation environment. Literature with many citations are used as support in creating models, which is an indication that the simulation environment is close to realistic. On the other hand, it is hard to tell before the tests are performed on the real system.

Control algorithms has been developed and tuned. Based on the simulation results (chapter 6) the attitude controllers seems to be well functioning and properly tuned. The horizontal controllers gives partial satisfying results. Simulations shows sudden deviations from the reference signals. In addition, the reference signal produced by the horizontal controllers sometimes exceed the feasible range. Therefore, a more through examination of the horizontal controllers has to be done (see chapter 7).

The state machine, described in section 5.5.2, and boundaries, described in section 5.5.1, has given satisfying results in all simulations. The results indicates that these algorithms can be implemented to the real system. Of course, boundaries has to be adjusted, as mentioned in chapter 7.

Overall, the model builds a solid ground for further development. Unfortunately, the current control algorithms can not be used on a real system, as there are to many failed simulations. Still, after some future work is done (chapter 7) there is certainly a opportunity for using the control algorithms on a real system.

Bibliography

[1] Specs hexacopter. https://www.quadh2o.com/hexh2o/hexh2o-kit/.

[2] Christian Månsson and Daniel Stenberg. Model-based design development and control of a wind resistant multirotor uav. 2014.

[3] Joel Hermansson, Andreas Gising, Martin Skoglund, and Thomas Schön. *Autonomous landing of an unmanned aerial vehicle*. Linköping University Electronic Press, 2010.

[4] Mostafa Moussid, Adil Sayouti, and Hicham Medromi. Dynamic modeling and control of a hexarotor using linear and nonlinear methods. *International Journal of Applied Information Systems*, 9(5), 2015.

[5] Johan Fogelberg. Navigation and autonomous control of a hexacopter in indoor environments. 2013.

[6] John Oyekan, Bowen Lu, Bo Li, Dongbing Gu, and Huosheng Hu. *A Behavior Based Control System for Surveillance UAVs*, pages 209–228. 08 2010.

[7] Vegard Line. Autonomous landing of a multirotor uav on a platform in motion. Master's thesis, NTNU, 2018.

[8] Andrea Alaimo, V Artale, Gabriele Barbaraci, C Milazzo, Calogero Orlando, and Angela Ricciardello. Lqr-pid control applied to hexacopter flight. 9:47–56, 01 2016.

[9] Andreas Vikane Hystad and Joakim Brobakk Lehn. Model, design and control of a quadcopter. *Mestrado em cibernética e robótica, Norwegian University of Science and Technology, Trondheim*, 2015.

[10] Alexandre Borowczyk, Duc-Tien Nguyen, André Phu-Van Nguyen, Dang Quang Nguyen, David Saussié, and Jerome Le Ny. Autonomous landing of a multirotor micro air vehicle on a high velocity ground vehicle. *IFAC-PapersOnLine*, 50(1):10488–10494, 2017.

[11] Lorenzo Marconi, Alberto Isidori, and Andrea Serrani. Autonomous vertical landing on an oscillating platform: an internal-model based approach. *Automatica*, 38(1):21–32, 2002.

[12] Quad h20. https://www.quadh2o.com/.

[13] E800. https://www.dji.com/e800/info.

[14] Ardupilot. http://ardupilot.org/about.

[15] Dune. https://lsts.fe.up.pt/toolchain/dune.

[16] Pm spectra. https://www.dune-project.org/.

[17] Thor I Fossen. *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons, 2011.

[18] Drag coefficient. https://www.engineeringtoolbox.com/drag-coefficient-d_627.html.

[19] Randal W Beard and Timothy W McLain. *Small unmanned aircraft: Theory and practice*. Princeton university press, 2012.

[20] Mss toolbox. http://www.marinecontrol.org/.

[21] Inertiua calculation. http://hyperphysics.phy-astr.gsu.edu/hbase/mi.html.

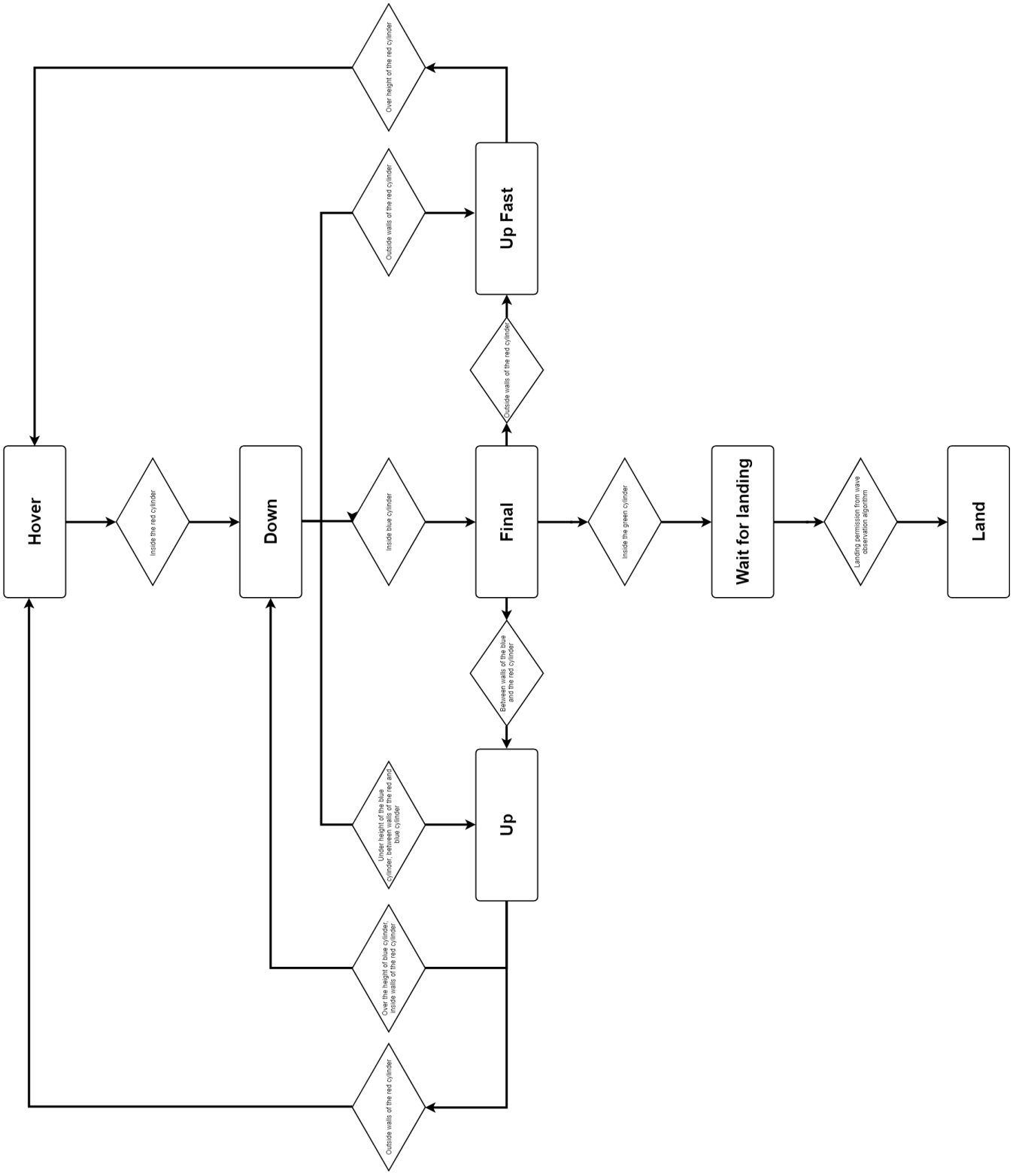
[22] Dji matrice. https://www.dji.com/matrice600-pro/info.

[23] Beufort scale. http://www.tranoy.net/stavanger/weather/beauforts.htm.

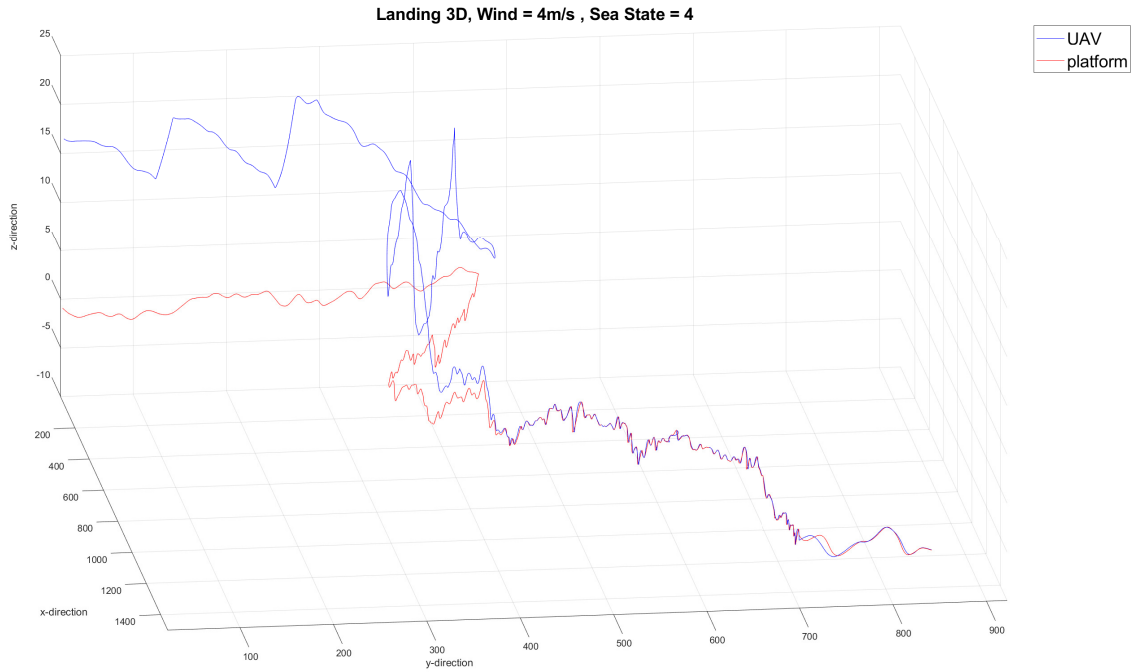
[24] Cecilia Linroth. Statistical analysis of wave heights using kalman filtering methods, 2014.

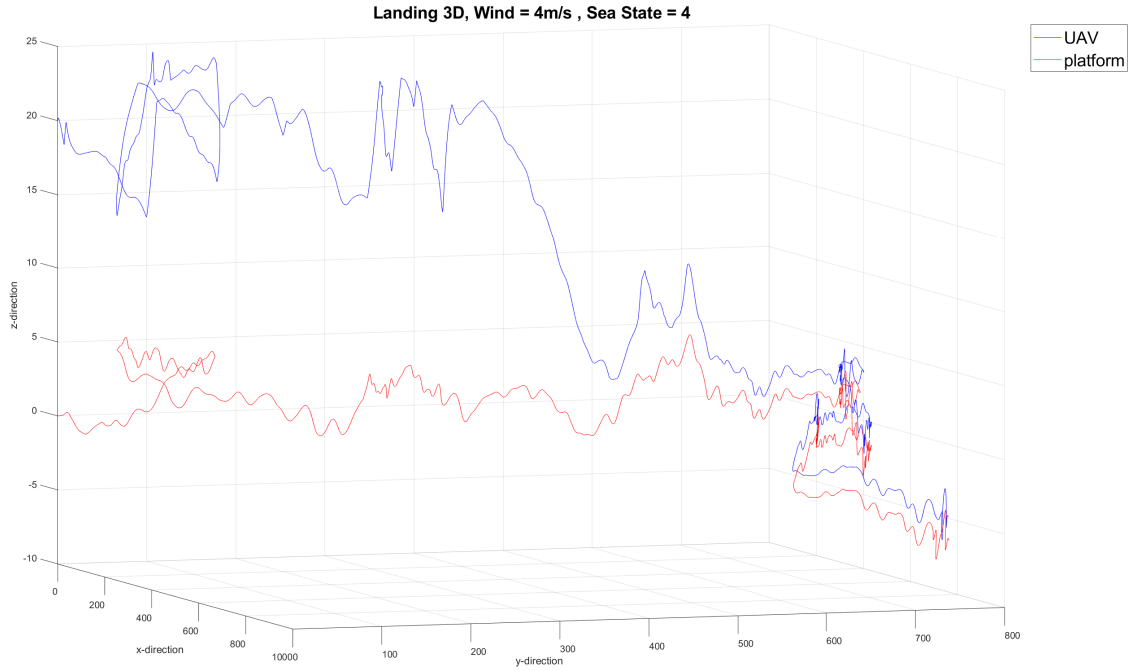
Appendices

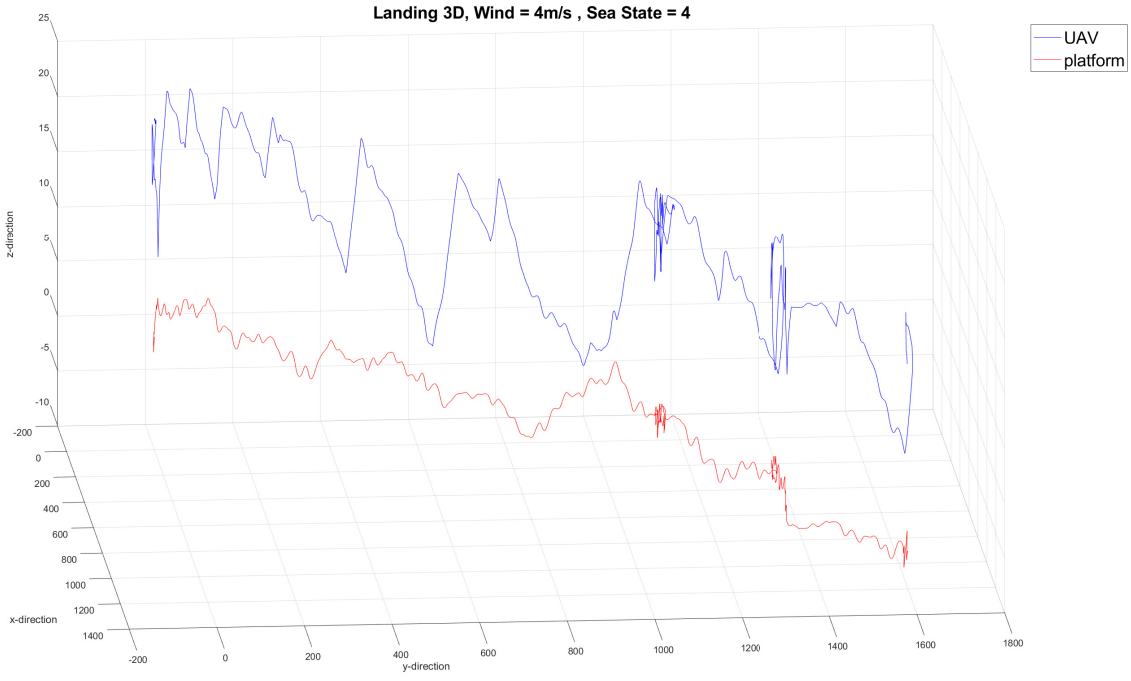
A Flow chart - state machine



B Simulation results




Landing 3D, Wind = 4m/s , Sea State = 4
UAV
platform
z-direction
x-direction
y-direction


Landing 3D, Wind = 4m/s , Sea State = 4
UAV
platform
z-direction
x-direction
y-direction