Delft University of Technology

Master thesis

Autonomous landing of Unmanned Aerial Vehicles

Eliminating tailsitter VTOL tip overs in high wind scenarios

Author:
S. T. Swart
4001699
Biomechanical Design

Supervisors:
Prof.dr.ir. J. Hellendoorn
S. Hulsman

November 26, 2024

Abstract

The need for automation of unmanned aerial vehicles (UAVs) rises with the increase of inexperienced operators. Extensive research has gone into automating and understanding multirotor UAVs and fixed wing UAVs. As vertical take off and landing (VTOL) UAVs are a relatively new category, there is ample opportunity for research in automation and understanding. This report focuses on automating the landing of a specific VTOL UAV: the Marlyn, created by ATMOS UAV.

Marlyn is a tailsitter, meaning that the centre of mass during the hover phase is relatively far away from the landing rods. This can lead to tip overs during landing. The centre of mass alone will cause a tip over moment at a pitch angle of $\theta \leq -32.5$ degrees. Experiments have been performed to find the relation between the incoming wind speed and the pitching moment generated by the wind. Adding the found pitching moment, the tip over limit angle decreases to $\theta \leq -47.5$ degrees. This corresponds to a wind speed of 8.4 m/s.

A surprising find was that not the wind nor the centre of mass was the biggest culprit for tip overs. The biggest factor was the reduction in controllability. Marlyn's bottom tail motor is in line with the landing rod that first touches the ground. This gives Marlyn no way to counter a forward tip over once it has been set in motion.

The firmware of Marlyn (PX4) comes with a complete simulation environment based on Gazebo. This simulation runs a simple aerodynamic model. The parameters for this model are based on estimations gained from XFLR. To determine and measure the difference between the simulation and reality, a sensitivity study is done. This study shows the influence of the parameters gained from XFLR on the lift and drag forces. After the sensitivity study, the values from XFLR are compared to the experimental data used to find the relation between the incoming wind speed and the pitching moment generated by the wind. The absolute values deviated significantly, but the relative differences between the values was more or less constant.

To counter the tip overs, three strategies are proposed. The first strategy revolves around detecting the moment the bottom landing rod touches the ground. By killing the motors, the wind will always prevent the forward tip over by blowing Marlyn upright. Using the inertial measurement unit inside the autopilot to determine the impact showed to be non feasible given the current design. The impact forces were not significant enough on all surfaces to reliably be used as a trigger to kill the motors. Mechanical switches have been shown in previous research to not work reliably in all conditions, mainly failing in muddy or sandy environments. Range finders and vision based distance sensors showed the most promising, if it can be guaranteed that they always point straight down.

The second strategy is based on generating a moment using the touch down impact to fling Marlyn upright. To generate this moment, a horizontal impact is generated by allowing Marlyn to drift with the wind in a controlled manner. The path generated for this could be done using artificial potential fields. These can be used to safely find the optimal path to the ground while staying within safety bounds. To validate the feasibility of this strategy, a simple inverted pendulum simulation is used. The Gazebo based simulation could not be used as is, since the control loops that try to stabilise Marlyn would try to counter the moment generated by the impact. The simple simulation has shown that there is merit in this strategy, but a lot of further research is required to validate that this also works in the real world.

The last strategy proposed tries to reduce the pitch angle during landing. This can be achieved via two methods that are discussed. The first rotates Marlyn so that the wings are not tangent but parallel to the wind. This method was discarded due to the actuators not being able to keep Marlyn in that unstable state for prolonged duration. The second method uses Marlyn's ability to rotate her wing motors independently. By rotating both into the wind, these motors can generate a force to counter the drag caused by the wind. This in turn leads to a smaller pitch angle required to remain stationary. The Gazebo based simulation is used to validate this method's feasibility. The wind speed at which the previously found tip over pitch angle was crossed was increased from 8.4 m/s to over 9 m/s. Due to instabilities in the simulation, wind speeds larger than 9 m/s could not be tested.

All strategies discussed have promising options. The most promising strategy to counter tip overs during

the landing of the VTOL tailsitter UAV Marlyn is: Rotating the wing motors into the incoming wind to reduce the required pitch angle to remain stationary. Further research is required to validate this strategy in real world scenarios.

List of Figures

List of Tables

Nomenclature

α	angle of attack
Δ	difference between the angle of the rows
δ	offset of the rows
ω	angular velocity
ϕ	roll angle of the aircraft
ψ	yaw angle of the aircraft
ρ	air density
τ	torque
θ	pitch angle of the aircraft
$\vec{\eta}$	rotations in Cartesian space with respect to the main axes
$\vec{\zeta}$	position in Cartesian space
$\vec{F}$	force
A	effective area
C_D	drag coefficient
c_g	centre of mass/centre of gravity
C_L	lift coefficient
C_m	moment coefficient
c_p	centre of pressure
d	distance
I	mass moment of inertia
M	moment around an axis
m	mass
S	surface area
T	thrust
v	velocity

Contents

Chapter 1

Introduction

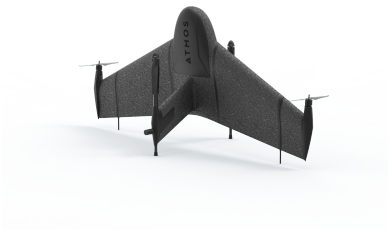
Drones (or UAVs, more on that later) are becoming more widespread in our society than ever. From filming to racing to delivery to military, drones are used for a plethora of use cases that were unthinkable twenty years ago. With this, the amount of drone operators grew as well. Ideally, each of these operators has had a training and is able to control the drone in every scenario imaginable. Sadly, this is not the case. Where governments try to address this by using regulations and licenses, engineers look towards automation. Automation can provide an easy and verifiable safe way to fly drones in more demanding scenarios like high wind, strong gusts or narrow corridors.

The level of automation can vary from model aircraft, where the operator directly controls the output to the control surfaces and the motor, to fully autonomous drones for delivery, where there is no operator at all. Each degree of added automation adds a level of complexity to the software that runs on the drone. And each degree of added automation also increases the requirement for more safety measures, but also the opportunity to provide them. For instance, adding a GNSS receiver to the drone can make position controlled flight possible. It can also provide limitations to the maximum velocity or, in case of remote control signal loss, to hold the position or even return to the take off location. To safely implement this, it also requires safeguards for when the GNSS solution is not accurate enough or when the sensor becomes disconnected.

One difficulty an engineer might face while trying to add a layer of automation is the immense amount of different types of drones. At the top level, drones can be split into three main categories, namely multirotors, fixed wing, and VTOL. Figure 1.1 gives an example of each type.

(a) multirotor [1]

(b) Fixed wing [2]



(c) VTOL [5]

Figure 1.1: Examples of the main drone categories.

Multirotor drones (Figure 1.1a) are the most well known drones used for filming and photography, and performing inspections. They can take off and land vertically, hover and can move in any direction at any time. A downside of them is that they need to actively generate all of their lift, limiting their size and flight duration.

Fixed wing drones (Figure 1.1b) resemble commercial aircrafts, having large wings to generate their lift. These drones can cross long distances efficiently while carrying relatively large and heavy payloads. Like commercial aircrafts, these are non-holonomic, as they always need to have a minimum forward speed to remain airborne.

The final category is the VTOL category (Figure 1.1c), which stands for Vertical Take Off and Landing.

These drones are also called hybrid drones as they combine the best of both the multirotor and fixed wing, being able to take off and land anywhere with very little space while also transitioning mid-air to fly efficiently.

(a) Standard [9]

(b) Tilt-rotor [8]

(c) Tailsitter [3]

Figure 1.2: Examples of the VTOL categories.

VTOL drones can be divided further into three categories, namely the standard category, tilt-rotor and tailsitter. Examples of these are shown in Figure 1.2. Standard VTOL drones look like someone glued wings on a multirotor. They have three (or more) rotors that generate lift during hover flight, and usually one or more motors pointing in the direction of fixed wing flight to provide the forward thrust.

Tilt-rotor drones differ from the standard category by having no dedicated fixed wing rotors. Instead, they tilt their rotors between fixed wing and hover flight. This reduces the amount of motors that need to be carried and also reduces drag generated by the standard's hover motors in fixed wing flight.

Tilt-rotors do add more complexity by adding servos for every rotor. Tailsitters reduce this added complexity by tilting the entire drone forward, instead of tilting only the rotors. This has the added benefit of being able to use the wings to generate lift in high wind scenarios, improving the hover efficiency. A major downside of this however, is that since in fixed wing flight the centre of mass is fixed, the location of the centre of mass in hover flight is relatively high. When in high wind scenarios the drone is landed, this can cause the drone to tip over, falling forward to the ground due to the high pitch angle required to counter the wind.

1.1 Marlyn

The drone of focus of this report is a Marlyn, made by ATMOS UAV (Figure 1.3). Marlyn is a VTOL tailsitter with rotating wing ends. These rotating wing ends (also known as rows) allow for greater yaw authority in high wind situations, as active thrust vectoring is used together with the normal control surfaces.

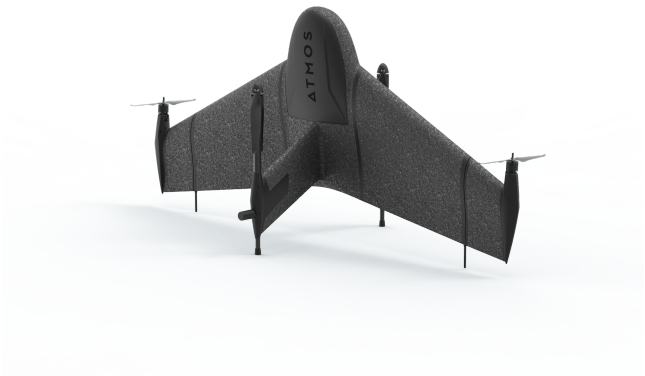


Figure 1.3: Marlyn drone by ATMOS UAV (ATMOS [5]).

1.1.1 Hardware

Marlyn has four motors, which are pair-wise optimized for different parts of the flight. The tail motors carry most of the weight during hover flight, while the wing motors provide all the thrust in fixed wing flight. In

order to reduce drag generated by windmilling propellers of the tail motors, the propellers fold away when the motors are off.

Sensor wise, Marlyn is equipped with two GNSS receivers, one in the nose and one in the body, providing stable positioning in the most used poses. She has a pitot tube, multiple barometers, IMUs, and compasses, providing redundancy in the pose estimation. The autopilot is a Cube Orange made by CubePilot running the open source autopilot stack PX4. She is powered by two (redundant) six cell LiPo batteries and can carry a camera payload in her nose that can easily be swapped.

1.1.2 Firmware

PX4 is an open source firmware stack for drones and unmanned vehicles, based on the Apache NuttX embedded operating system. NuttX is a unix emulator, providing basic functionality like peripheral drivers, scheduling, memory management, multi-threading and a command line interface. Drivers, modules, and system commands all run as separate threads, communicating via a library called Micro Object Request Broker, or μORB. This μORB allows for strict separation within the code base, keeping cross dependencies to a minimum.

All messages passing through μORB can be logged to an SD card using a logger module. The log files are written in the self-describing ULog format. In this log, the layout of the logged messages, as well as the data types of their fields, are encoded. This means the log can be read without requiring a predefined set of messages.

The PX4 firmware has basic manual, acrobatics and stabilized modes, as well as altitude hold, position hold and fully automatic flight modes. Next to this, a multitude of safety features and state estimators are available.

PX4 uses a control stack as seen in Figure 1.4. Due to the architecture of the control stack, it is very easy to change separate parts during run time without the need for a reboot or recompile. This allows for greater flexibility and testability, since all parts can be tested separately.

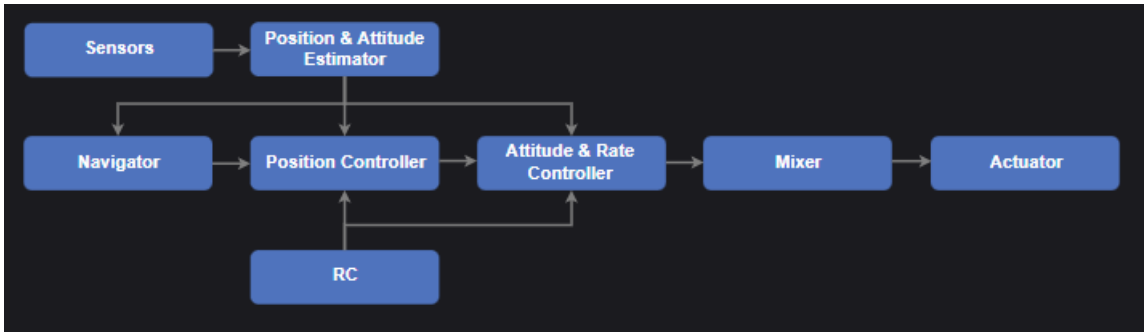


Figure 1.4: High level overview of the flight stack used by PX4 [19].

The position and attitude estimation is based on a Kalman filter, which is set up to estimate the current state of the aircraft and has the ability to reject sensors if they are not within expected bounds or their data does not match with the majority of the other sensors. An example of this would be that a gyro suddenly shows a positive rotation around the yaw axis while all magnetometers show a negative rotation around the same axis. The output of this Kalman filter is then used in the navigator, position controller, and attitude and rate controllers.

VTOL tailsitters are generally more efficient when hovering with the nose into the wind. This is because the inflowing wind causes the wings to generate lift. To take full advantage of this, PX4 includes a so-called weather vane mode. In this mode, all roll control efforts are also translated into a yaw control effort, which practically ensures that the nose is always pointing directly into the wind.

The mixer is responsible for translating body-axis control setpoints between -1 and 1 to motor RPMs and servo setpoints. These set points are then used by the actuator drivers to drive the actuators.

1.2 Dynamic Model

The state of a drone can be described by its pose. Using this description, modeling position based control becomes easy since the drone is already described in terms of its pose. let $\vec{\zeta} = (x \quad y \quad z)^T$ and $\vec{\eta} = (\phi \quad \theta \quad \psi)^T$. Together, $\vec{\zeta}$ and $\vec{\eta}$ describe the drone's position and orientation in 3D space. For orientation stability, it is

custom to use body-fixed orientations since it makes the calculations less complex. This means that a rotation from body-fixed to earth-fixed is required.

The coordinate rotation matrix from body-fixed frame to earth-fixed frame is given by

$$\boldsymbol{R}(\vec{\eta}) = \begin{pmatrix} C_\psi C_\theta & C_\psi S_\theta S_\phi - S_\psi C_\phi & C_\psi S_\theta C_\phi + S_\psi C_\phi \\ S_\psi C_\theta & S_\psi S_\theta S_\phi + C_\psi C_\phi & S_\psi S_\theta C_\phi - C_\psi S_\phi \\ -S_\theta & C_\theta S_\phi & C_\theta C_\phi \end{pmatrix} \tag{1.1}$$

where S_x and C_x are shorthand for $\sin(x)$ and $\cos(x)$, respectively. The rotation matrix from body-fixed angular velocities to earth-fixed is given by

$$\boldsymbol{T}(\vec{\eta}) = \begin{pmatrix} 1 & S_\phi T_\theta & C_\phi T_\theta \\ 0 & C_\phi & -S_\phi \\ 0 & \frac{S_\phi}{C_\theta} & \frac{C_\phi}{C_\theta} \end{pmatrix} \tag{1.2}$$

where T_x is shorthand for $\tan(x)$.

Combining (1.1) and (1.2), we get the following relation between the body-fixed pose derivatives and the earth-fixed pose derivatives.

$$\begin{pmatrix} \dot{\vec{\zeta}} \\ \dot{\vec{\eta}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{R}(\vec{\eta}) & 0_{3x3} \\ 0_{3x3} & \boldsymbol{T}(\vec{\eta}) \end{pmatrix} \begin{pmatrix} \vec{v} \\ \vec{\omega} \end{pmatrix} \tag{1.3}$$

To control a drone's orientation around its body axes, torques are required. They are given by (1.4).

$$\vec{\tau}_B = \begin{pmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{pmatrix} = \begin{pmatrix} l_{wing} k_{wing} \cos(\alpha - 4^\circ)(-\omega_2^2 + \omega_4^2) \\ l_{hover} k_{hover}(-\omega_1^2 + \omega_3^2) \\ l_{wing} k_{wing} \sin(\alpha - 4^\circ)(\omega_2^2 - \omega_4^2) \end{pmatrix} \tag{1.4}$$

where α is the angle of the rotating wing ends and $k = 2\rho A(\frac{K_v K_q}{K_t})^2$, where K_v and K_q are motor parameters and K_t is a linear mapping between torque and thrust. Note the 4° in τ_ϕ and τ_ψ, which come from the physical forward tilt of the wing motors.

Equation (1.5) gives the relation between the thrust of the motors and the forces in the body Cartesian coordinate frame. Note here as well the 4°, which comes from the physical forward tilt of the wing motors.

$$\vec{T}_B = \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix} = \begin{pmatrix} k_{wing}(\sin(4^\circ + \alpha)(\omega_2^2 + \omega_4^2) \\ 0 \\ k_{wing} \cos(4^\circ + \alpha)(\omega_2^2 + \omega_4^2) + k_{hover}(\omega_1^2 + \omega_3^2) \end{pmatrix} \tag{1.5}$$

The Equations (1.4) and (1.5) lead to the principal equations of motion according to the Newton method. f_w are the disturbances caused by the wind. During normal hover operation, alpha will be equal and opposite, cancelling out most of the force in the T_x term. If this constraint is removed, this term could be used to offer an additional method of generating force in the body-fixed x-axis, which could reduce the pitch angle required to translate.

$$m(\dot{\vec{v}} + \vec{\omega} \times \vec{v}) = mg\boldsymbol{R}^T \vec{e}_z + \vec{T}_B + \vec{f}_w \tag{1.6}$$

Equation (1.7) gives the angular accelerations. Λ are the gyroscopic forces and will be neglected since the drone will not be spinning at a high enough angular velocity. τ_w is the torque on the drone caused by the wind. These forces cannot be neglected, as shown in Chapter 2. Marlyn will be assumed symmetrical in the body $z-x$ axis and the body $z-y$ axis, meaning τ_w will only have a component around the y axis.

$$\boldsymbol{I}\dot{\vec{\omega}} + \vec{\omega} \times (\boldsymbol{I}\vec{\omega}) + \Lambda = \vec{\tau}_B + \vec{\tau}_w \tag{1.7}$$

$$\dot{\vec{v}} = -\vec{\omega} \times \vec{v} + g\boldsymbol{R}^T \vec{e}_z + \frac{\vec{T}_B}{m} + \frac{\vec{T}_w}{m} \tag{1.8}$$

$$\dot{\vec{\omega}} = \boldsymbol{I}^{-1}(-\vec{\omega} \times (\boldsymbol{I}\vec{\omega})) + \boldsymbol{I}^{-1}\vec{\tau}_B + \boldsymbol{I}^{-1}\vec{\tau}_w \tag{1.9}$$

1.3 Problem Statement

Marlyn, being a tailsitter VTOL UAV, has a high centre of mass in hover configuration. Next to this, Marlyn needs to "lean" into the wind to remain stationary. This leads to the risk of tip overs during touch down, where instead of landing upright, Marlyn falls forwards on the ground. This leads to damage to the body, bottom hover propeller and motor, and potentially damage to the payload. This report aims to increase the understanding of how these tip overs occur, and find strategies to prevent them from happening.

1.4 Reading Guide

Chapter 2 will explore the reasons behind tip overs occurring. Chapter 3 handles the simulation environment that comes with the firmware where the dynamic model derived in Section 1.2 is implicitly implemented. This chapter will mainly focus on the influence of external disturbances from the wind and their influence on the aircraft by investigating the influence of the accuracy of the parameters in the simulation. Chapter 4 will discuss three strategies of potential solutions to the tip-over problem. Each specific potential solution is discussed in detail and their advantages and disadvantages are shown. Chapter 5 will summarise the findings, and finally Chapter 6 will provide recommendations for future work.

Chapter 2

Problem Analysis

Landing any UAV autonomously is no easy feat. Next to requiring the sensors to know where the UAV is relative to the ground, there are disturbances like objects, wind and other aircrafts that can heavily influence the path that has to be taken to the ground. For some UAVs, it does not end there. Tailsitter VTOL UAVs have, due to their shape, a large risk of tip over at or just after touch down.
Tailsitter VTOL UAVs are UAVs that take off like a multicopter and fly like a fixed-wing UAV. To transition between these modes, they rotate the entire airframe forwards. In fixed-wing mode, the position of the centre of mass has a given optimum for the airfoil. This optimum usually lies at around $\frac{1}{3}^{rd}$ of the main wing measured from the leading edge, or in the case of a delta wing, a bit further backwards. This means that the centre of mass in multicopter mode is relatively high off the ground, making it an inverted pendulum.
Next to this undesirable inverted pendulum, tailsitters have their wings perpendicular to the wind in multicopter mode, making them more susceptible to disturbances caused by the wind.

Tip overs can occur due to multiple reasons. Firstly, a tip over will occur if the centre of mass is outside of the support area and there is no counter moment balancing the UAV. This counter moment is usually generated by the propulsion systems and the wind. This is the most difficult case of tip over, since the motor that should prevent this motion cannot generate a moment around the point of rotation. The only option is to take off again so that the point of rotation changes.
Secondly, if this counter moment is too big, a tip over can occur in the opposite direction, regardless of the position of the centre of mass. This kind of tip over is the easiest to counteract, since there is usually a motor that can quickly provide a moment to prevent this.
Lastly, the impact of the landing can generate an impulse big enough to cause a tip over. This tip over can be in both directions, based on the direction of the impact.

This chapter will analyse the physics behind tip overs. To reduce the complexity of the analysis, the impact generated by the touch down is assumed to be negligible. This reduces the dynamic analysis to a static analysis with the pitching moments of the centre of mass, the wind, and the motors.

2.1 Centre of Mass Pitching Moment

The pitching moment generated by the centre of mass is directly related to the pitch angle. A simplification of the free body diagram can be found in Figure 2.1. In this figure, θ is the pitch angle, d_1 the distance from the centre of mass to the ground at $\theta = 0$ and d_2 the horizontal distance from the centre of mass (c_g) at $\theta = 0$ and the pivot point.
From Figure 2.1, the following equations can be derived.

$$M_{\theta,c_g} = -F_z \cdot d \tag{2.1}$$

In Equation (2.1), M_θ is the pitching moment, F_z is the gravitational force and d is the horizontal distance between the centre of mass and the pivot point. d can be calculated by (2.2).

$$d = d_2 \cdot \cos\theta - d_1 \cdot \sin\theta \tag{2.2}$$

From (2.2) follows that the pitching moment generated by the centre of mass without counteracting forces will cause a tip over if the inequality of Equation (2.3) is true.

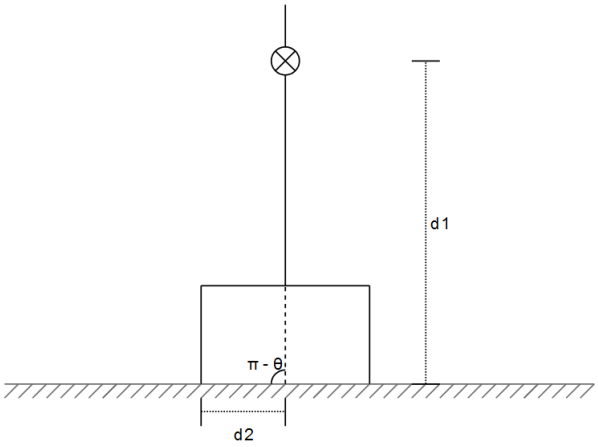


Figure 2.1: Free body diagram of Marlyn in the x-z plane.

$$\theta > \arctan \frac{d_2}{d_1} + \pi \cdot n \tag{2.3}$$

Since in the tip over situation, only pitch angles between 0 and 90 degrees are possible, n will be considered as 0. If the inequality of (2.3) is false, the centre of mass will generate a moment that counters tip overs.

Table 2.1: Properties of Marlyn.

Property	Value	Unit
m	5.87	kg
d_1	0.44	m
d_2	0.3	m

Using the dimensions given by the airframe of Marlyn (Table 2.1) and Equation (2.3), the maximum pitch angle where the centre of mass provides a stabilising moment is 32.5 degrees.

2.2 Wind Speed Pitching Moment

There is a relation between the horizontal velocity of a drone and its tilt angle. Having a tilt angle of 0 means that the drone has no horizontal thrust component, meaning it will not move by itself. By tilting the drone, this horizontal thrust component is increased, resulting in an acceleration in the same direction as the component (assuming the vertical thrust component stays the same).
Since the observations have to be the same regardless of the reference frame, the above can also be applied to a geostationary drone that is fighting the influence of the wind. This inflow of the wind on the angled body of the drone will also generate a moment which the drone's motors have to counteract to also not change its attitude. By flying at a constant horizontal velocity during conditions with no wind, a direct relation between the horizontal ground speed and the moment generated by the motors can be found. Since Marlyn will always rotate into the wind, this comes down to the relation between pitch angle and pitching moment. An experiment has been done during low to no wind conditions ($> 2m/s$) and by flying directly into the wind and with the wind, the wind influence can be eliminated.
The moment generated by the motors can be calculated by using a fit between the motor RPM and its generated thrust, corrected for the density altitude. These thrusts are then subtracted and multiplied by their arm.

$$M_{\theta,tailmotors,c_g} = T_{bottom} \cdot d_{bottom,c_g} - T_{top} \cdot d_{top,c_g} \tag{2.4}$$

Equation (2.4) shows the moment generated by the tail motors to stabilize the pitch angle. $T_{<motor\_name>}$ is the thrust generated by the motor and $d_{<motor\_name>,c_g}$ is the distance between the motor and the c_g. This

distance is equal for both motors.

$$M_{\theta,wingmotors,c_g} = (T_{left} + T_{right}) \cdot d_{wingmotor,c_g} \cdot \sin(\frac{4 \cdot \pi}{180}) \tag{2.5}$$

Next to this moment, the wing motors are rotated forward by 4°, giving them a thrust component that generates a moment. Equation (2.5) shows this moment expressed in wing propeller thrusts.
Finally, the airframe itself produces a moment due to its shape. This moment has to be compensated by the tail motors regardless of any wind. This leads to (2.7).

$$M_{\theta,c_g} = M_0 + M_{\theta,tailmotors,c_g} + M_{\theta,wingmotors,c_g} \tag{2.6}$$

$$M_{\theta,c_g} = M_0 + T_{bottom} \cdot d_{bottom,c_g} - T_{top} \cdot d_{top,c_g} + (T_{left} + T_{right}) \cdot d_{wingmotor,c_g} \cdot \sin(\frac{4 \cdot \pi}{180}) \tag{2.7}$$

The moment found in (2.7) has to balance the moment generated by the wind, since no angular acceleration is measured.
The results of this experiment can be found in Figure 2.2, where the dots are the median pitching moments during a constant horizontal velocity flight, corrected for the wind. The line is a third order polynomial fit though the data points, as described in (2.8). The actual box plots of the measurements can be found in Appendix A.

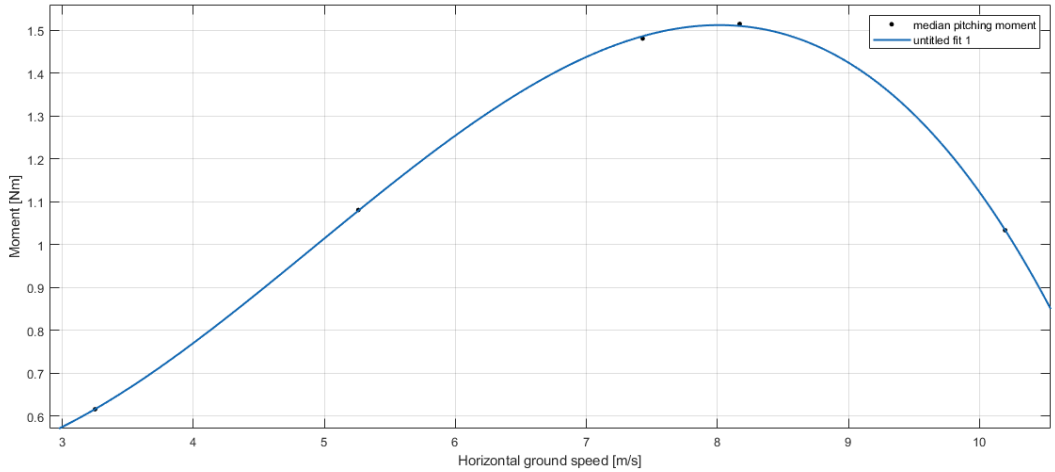


Figure 2.2: Measurements and fit of pitching moment relation to horizontal ground speed.

$$M_{\theta,wind,c_g} = -0.008763 \cdot v_{ground,h}^3 + 0.1294 \cdot v_{ground,h}^2 - 0.3864 \cdot v_{ground,h} + 0.8057 \tag{2.8}$$

The results of the experiment lays within the expected. The steep fall-off at the higher ground speeds can be explained by the self correcting moment that the airfoil has in forward flight configuration, meaning the body will generate some moment. This reduces the moment the motors need to generate. This self correcting moment will most likely be the main cause of the tip over, together with the centre of mass.

Assumptions

The calculations of the pitching moment rest on a couple of assumptions. The thrust of the motors is not directly measured but calculated by using fitted propeller data to go from RPM to thrust. This fit is only available for the static case with no inflow. Since the inflow influences the generated thrust, the absolute value of the thrust per motor will be slightly off. As the moment depends on the difference between the two thrusts and both thrusts are influenced by the inflow, this total influence on the moment is low.

Secondly, this data is mainly used to calculate the moment caused by the wind around the centre of mass the moment Marlyn touches down. In that situation, Marlyn is in ground effect. These measurements were taken at an altitude of about 25 meters, which is above the altitude where the ground effect would have an influence. This means that the moment at touch down can differ from the moment calculated above. To have the most accurate measurements, the test would have to be repeated at an altitude of at most 50 centimetres. With regards to the safety of the test executors, bystanders and the drone itself, it was chosen not to perform these tests and work with the slightly inaccurate measurements at greater altitude.

2.3 Changes During Touch Down

At touch-down, the centre of rotation changes to the contact point with the earth. This means that the bottom hover motor no longer has any contribution to the sum of all moments around the pivot point and that the centre of mass generates a moment based on the horizontal distance to the contact point (see Figure 2.3).

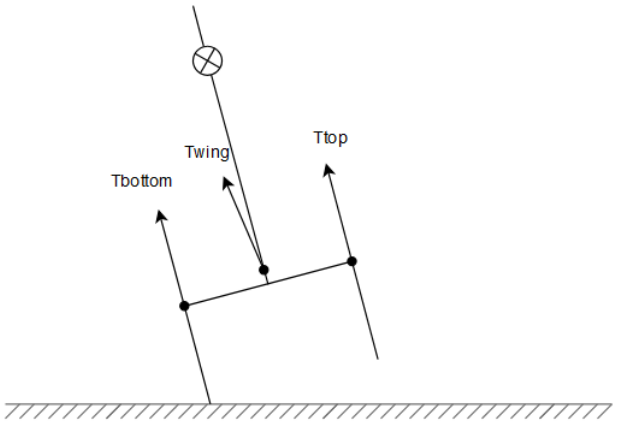


Figure 2.3: Free body diagram of Marlyn in the x-z plane during touch down.

The mass moment of inertia will change as well, but this can easily be calculated by using the Huygens–Steiner theorem.
This also means that the moment generated by the wind inflow will change. To transpose the moment to a different point of rotation, the point at which the force acts needs to be known (centre of pressure, c_p). Given the large angles of attack, it is assumed that there is no flow attachment and the airfoil is stalled. This c_p is the point where the wind force acts, and follows from the geometry of the airfoil.

$$T_h = T \cdot \sin(\theta) \tag{2.9}$$

$$d_{c_p,c_g} = \frac{M_{\theta,wind,c_g}}{T_h} \tag{2.10}$$

Equation (2.10) sets the value of d_{c_p,c_g} equal to the ratio between the total moment generated by the wind around the c_g and the horizontal thrust force generated. The horizontal thrust force can be derived from the total thrust force (2.9). In the most basic case, the total thrust force can easily be derived from the weight of the aircraft and the angle between the vertical axis and resultant thrust vector. Marlyn however, has an airfoil that can generate lift. This means that only part of the force required to remain airborne will be provided by the motors.
To calculate the actual thrust generated by each motor, the same look up table is used to go from RPM to thrust. The same assumption about the inflow in this look up table stands, meaning that this does influence the total thrust generated by the aircraft.

$$M_{\theta,wind,landingrod} = F_{drag,h} \cdot \big(d_1 \cdot \sin(\theta) + d_2 \cdot \cos(\theta) - d_{c_p,c_g} \cdot \cos(\theta)\big) \tag{2.11}$$

Equation (2.11) calculates the moment around the landing rod based on the drag force and the point where the force acts on the body. Additionally, the top tail propeller and the wing propellers now also contribute to a moment around the landing rod. Their contribution can be calculated with (2.12).

$$\begin{aligned} M_{\theta,propellers,landingrod} = &- T_{top} \cdot d_{top,landingrod} + (T_{left} + T_{right}) \cdot \sin(\frac{4 \cdot \pi}{180}) \cdot d_{wing,landingrod,x} \\ &- (T_{left} + T_{right}) \cdot \cos(\frac{4 \cdot \pi}{180}) \cdot d_{wing,landingrod,y} \end{aligned} \tag{2.12}$$

Equation (2.13) shows the total balance in pitching moments at the moment of touch down. As can be seen from this equation, since $M_{\theta,propellers,landingrod}$ is strictly negative, the only moment counteracting the propellers is the wind. This is in contrast with the free flight situation, where the bottom motor is also able to provide a counter moment.
Note that at this point, the impact forces are assumed to be negligible due to a sufficiently soft landing. Chapter 4 will show why this is in some scenarios a valid assumption, and will also add the forces for other scenarios to test a possible solution.

$$M_{\theta,total,landingrod} = M_{\theta,wind,landingrod} + M_{\theta,propellers,landingrod} \tag{2.13}$$

Figure 2.4 shows the results of the experiments as a moment around the landing rod that is touching the ground as a function of the wind speed. Figure 2.5 shows the same moment, but now as a function of the pitch angle. Based on these figures, the maximum pitch angle is $-47.5°$ and the corresponding maximum wind speed is $8m/s$.
This is significantly larger than the maximum pitch angle found in Section 2.1, which was $-32.5°$. This shows that the wind is helping to prevent the tip over, which makes sense. The main cause of the tip over is actually the thrust that motors are generating in an effort to stabilise the aircraft. Since the bottom motor is no longer able to provide a moment, and the moment generated by the wind decreases in higher winds due to the aircraft pitching forward more, balancing and compensating for gusts becomes almost impossible without prior knowledge of the wind.

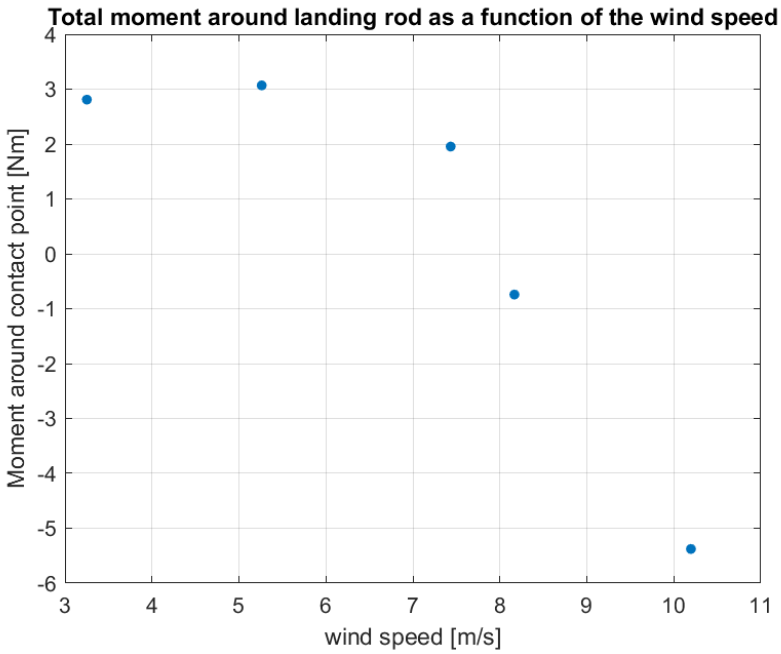


Figure 2.4: Calculated total moment around landing rod at touch down as a function of the wind speed.

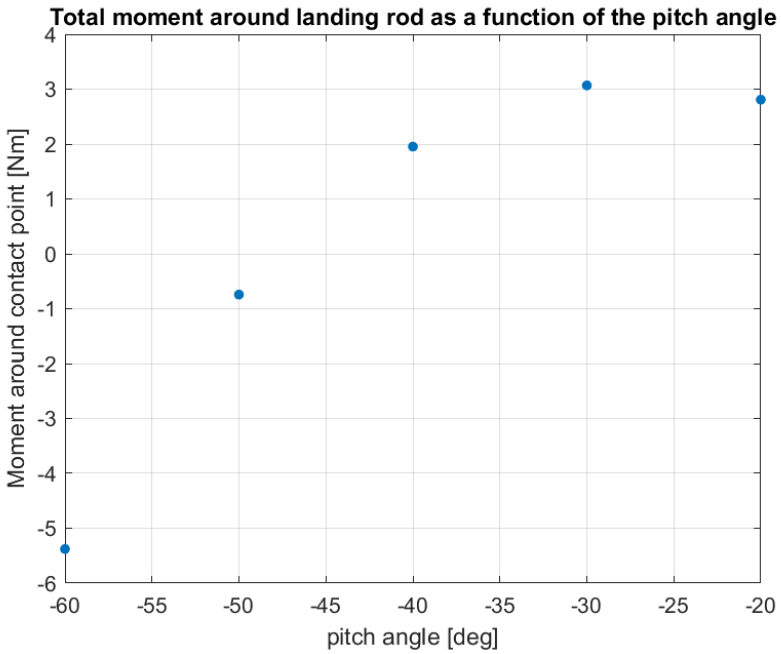


Figure 2.5: Calculated total moment around landing rod at touch down as a function of the pitch angle.

The found maximum wind speed is significantly lower than the $14m/s$ that is the goal. How is it then that Marlyn is able to be landed in wind speeds of up to $14m/s$?
Part of this can be explained by the wind not having a set speed. It fluctuates, has gusts and the relative lulls in wind speed can be used to land the aircraft safely. Other tricks are to land with a non zero horizontal velocity backwards. This adds an impulse at touch down, which can cause the aircraft to rotate upright, even if the static moment balance would indicate otherwise.
The next chapter will explore methods of applying these "tricks" in a controller, allowing for fully automatic landing in high wind conditions.

Chapter 3

Simulation Environment

To validate any solution strategies safely, a simulation is required. PX4 comes with a full simulation suite based on Gazebo, where it is possible to define a custom aircraft and run the firmware on Linux. In this simulation suite, a lift and drag plugin is provided which calculates the lift forces, drag forces, and the pitching moment. The lift and drag forces are applied at the centre of pressure (c_p), the pitching moment is applied at the centre of mass (c_g). Both of these parameters are given to the simulation as fixed values and are determined based on the physical properties of the air frame.

It is assumed that the air frame consists of two airfoils, the main wing and the tail. The main wing generates lift and drag. It is assumed that this airfoil generates all lift for the entire air frame.

The tail does not generate any lift but does contribute to drag and has a moment which causes the air frame to have the tendency to align itself into the wind during forward flight.

The parameters used in the simulation environment to calculate the lift forces, drag forces and pitching moment are estimated using a software tool called XFLR, which can be used to analyse the aerodynamic properties of airfoils. It is fairly accurate up to α_{stall}, but tends to deviate with larger αs. Therefore, $C_{L,\alpha_{stall}}$ is validated using experimental data. Tables 3.1, 3.2, 3.3, and 3.4 show the values found by XFLR. In these tables, S is the surface area of the airfoil, α_{stall} is the angle of attack at which the airfoil stalls, C_L is the lift coefficient, C_D is the drag coefficient, and C_M is the pitching moment coefficient. Note the negative value of $C_{L,\alpha_{stall}}$ in Table 3.2, indicating that the lift generated decreases after stall.

Table 3.1: Values for the generic simulation parameters.

Parameter	Value
S	0.608
α_{stall}	0.38

Table 3.2: Values for the simulation parameters of C_L as found by XFLR.

Parameter	Value
$C_{L,finiteplate}$	1.0
$C_{L,\alpha}$	3.7242
$C_{L,\alpha_{stall}}$	-1.5

Table 3.3: Values for the simulation parameters of C_D as found by XFLR.

Parameter	Value
$C_{D,finiteplate}$	1.0
$C_{D,0}$	0.05
$C_{D,\alpha}$	0.5
$C_{D,\alpha_{stall}}$	2.0

Table 3.4: Values for the simulation parameters of C_m as found by XFLR.

Parameter	Value
$C_{m,0}$	0.04712
$C_{m,\alpha}$	-0.26422
$C_{m,\alpha_{stall}}$	-1

At the end of each section of this chapter, a sensitivity study is done to get a feeling for the contribution of a single value from the simulation on the total lift, drag or pitching moment calculated. These sensitivity studies can then be used to improve the accuracy of the simulation environment in future research. The values found by XFLR are used as a baseline for the sensitivity studies.

The angle of attack (α) is defined as the angle between the horizontal plane of the airfoil and the incoming airflow. Since the airfoil is attached to the air frame, the horizontal plane rotates with the pitch angle of the air frame. Therefore, to calculate α, the airfoil is first rotated with the pitch angle of the air frame, before the angle between the horizontal plane and the incoming airflow is calculated. Since our air foil is not horizontal when the angle between the air frame's vertical axis and the incoming airflow is zero, a so-called α_0 is added to the α of the airfoil.

During normal hover in low wind conditions and no translations, Marlyn's airfoil is almost perpendicular to the airflow. It will therefore generate a marginal amount of lift, if any at all. When hovering in high wind conditions or with fast translations, Marlyn will tilt forwards, decreasing α and will start to generate lift. This has a large effect on the steady state pitch angle of the aircraft, since less vertical thrust of the motors is required to keep the altitude stable. This results in Marlyn pitching forward further than if no lift was generated, reducing the drag caused by the airfoil. This is the main mechanism behind Marlyn's more efficient hovering in high wind conditions than normal multirotors or more traditional VTOL drones.

3.1 Lift

To calculate the lift generated by an airfoil, the simulation uses the well known equation that is given by (3.1), where F_L is the generated lift force, C_L is the lift coefficient, ρ the air density, v the airspeed of the airfoil, and S the surface area of the airfoil.

$$F_L = C_L \cdot \frac{1}{2} \cdot \rho \cdot v^2 \cdot S \tag{3.1}$$

The equation to calculate C_L is dependent on the current α and the α at which the airfoil stalls (α_{stall}). Stall is the phenomenon where an increase of α results not in an increase of lift, but a sudden reduction of it due to separation of the airflow from the airfoil. It is something most fixed-wing pilots try to avoid as much as possible. Below α_{stall}, (3.2) is valid. Above α_{stall}, a different equation is used (3.3). In (3.3), the second line is empirically validated by Hoburg and Tedrake [12].

The value for $C_{L,finiteplate}$ is arbitrarily chosen to represent the losses due to the airfoil having finite dimensions.

$$C_L = C_{L,\alpha} \cdot \alpha \tag{3.2}$$

$$C_L = \min \begin{cases} C_{L,\alpha} \cdot \alpha_{stall} + C_{L,\alpha_{stall}}(\alpha - \alpha_{stall}) \\ C_{L,finiteplate} \cdot 2 \cdot \sin(\alpha) \cdot \cos(\alpha) \end{cases} \tag{3.3}$$

3.1.1 Sensitivity Analysis

In the lift model, there are some parameters that can be set externally based on the physical properties of the airfoil. These are $C_{L,finiteplate}$, $C_{L,\alpha}$, and $C_{L,\alpha_{stall}}$. To find how they influence the total lift generated by the airfoil, a sensitivity study is done. The results of this study are a good indication of the influence of each parameter on the error between the simulation and the real world.

As can be seen in (3.1), F_L has a linear relation with C_L, so any relation of the parameters with C_L will by definition also be the same as their relation with F_L. This simplifies the sensitivity study by allowing to discard the other terms.

The baseline values of this sensitivity study are given by Table 3.2.

$C_{L,finiteplate}$

$C_{L,finiteplate}$ is used to scale the flat plate model to account for losses due to the dimensions of the airfoil being finite. Its relation to C_L is harder to deduce from the formulas alone, since it only appears in (3.3) inside of the *min* function. To get an indication of its relation to C_L and F_L, a series of plots have been made that can be seen in Figure 3.1.

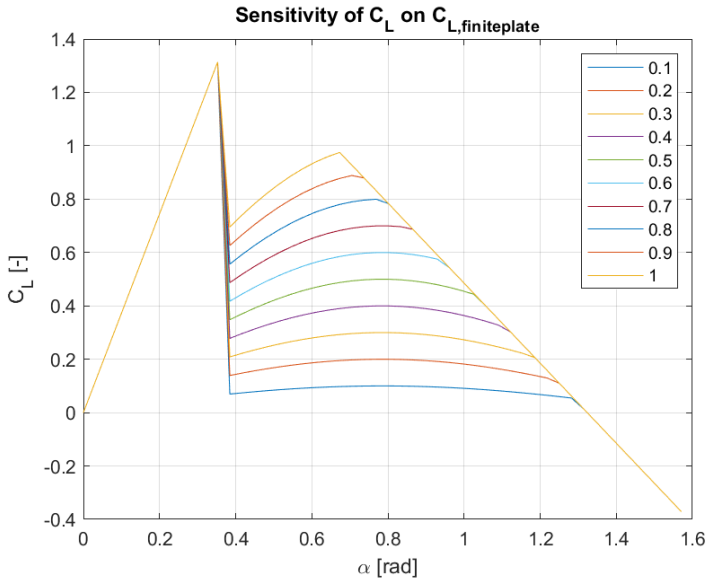


Figure 3.1: Sensitivity of C_L on $C_{L,finiteplate}$.

The values of $C_{L,finiteplate}$ are chosen to represent a range where the influence is clearly visible while not considering values that have no meaning (i.e. $C_{L,finiteplate} = 0$, or $C_{L,finiteplate} > 1$).

As can clearly be seen in Figure 3.1, $C_{L,finiteplate}$'s influence is limited to the region bounded by $\alpha > \alpha_{stall}$ and $C_{L,\alpha_{stall}}(\alpha - \alpha_{stall})$, where its value determines the value of C_L directly. This means that the influence of $C_{L,finiteplate}$ on C_L is very large within the bounded range and it is thus very important to find a realistic value if the error of the simulation with respect to reality is to be minimized.

$C_{L,\alpha}$

$C_{L,\alpha}$ is the coefficient of the influence α has on C_L. It appears in Equations (3.2) and (3.3) in the top part. It does not appear in the bottom part of (3.3). This indicates that its influence will most likely be across all regions of α, until the bottom part of (3.3) becomes lower than the top part. With very low values of $C_{L,\alpha}$, it could even be the case that the flat plate model is never lower.

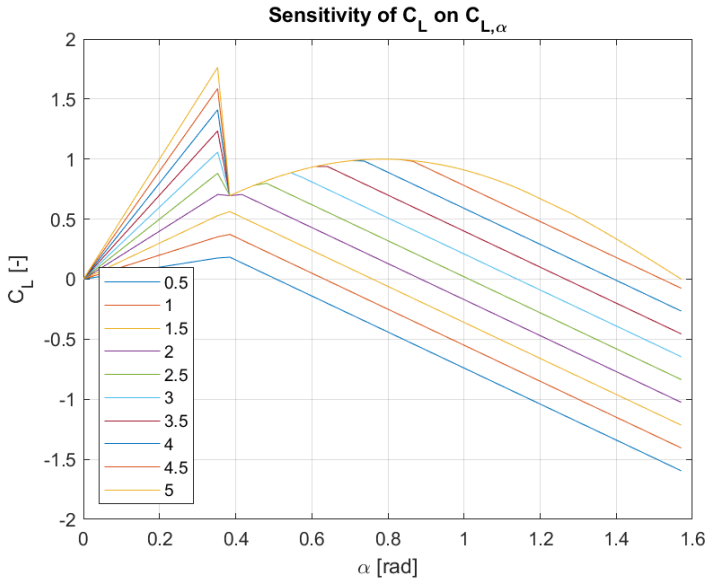


Figure 3.2: Sensitivity of C_L on $C_{L,\alpha}$.

The values of $C_{L,\alpha}$ are chosen to represent a range where the influence is clearly visible while not considering values that have no meaning (i.e. $C_{L,\alpha} = 0$ or values that will not be visible due to constraints on C_L).

Figure 3.2 shows the values of C_L over α with varying $C_{L,\alpha}$. It can clearly be seen that $C_{L,\alpha}$ influences the value of C_L strongly over the entire range of α until it is bounded by $C_{L,\alpha_{stall}}(\alpha - \alpha_{stall})$, which only occurs for $C_{L,\alpha} > 2$. This means that getting an accurate value is very important. Luckily, this value can be fairly accurately be estimated by XFLR.

$C_{L,\alpha_{stall}}$

$C_{L,\alpha_{stall}}$ is the influence of α has on C_L after stall. Its main goal is to model the loss of lift generated by the separation of flow that is present when the airfoil is stalled. Given its goal, it only appears in the top part of Equation (3.3). This indicates that its influence on C_L will be visible only with $\alpha > \alpha_{stall}$. The results of this sensitivity analysis can be found in Figure 3.3.

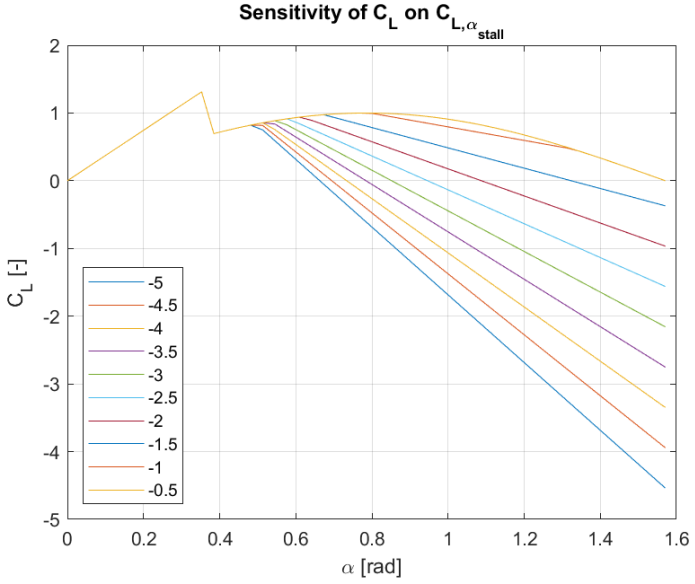


Figure 3.3: Sensitivity of C_L on $C_{L,\alpha_{stall}}$.

As seen in the sensitivity analyses that have been above before, $C_{L,\alpha_{stall}}$ acts as a bound for very high values of α. It is then no surprise that lowering the value of $C_{L,\alpha_{stall}}$ causes the bounds to become steeper.

Conclusion

The parameters that are used to calculate C_L work on distinct areas of the $\frac{C_L}{\alpha}$ curve. This means that there are clear tuning knobs available if in the simulation environment a part of the curve deviates from reality.

3.2 Drag

The simulation calculates drag experienced by an airfoil based on the well known formula written below, where F_D is the generated drag force, C_D is the drag coefficient, A the effective area of the airfoil, ρ the air density, and v the airspeed of the airfoil.

$$F_D = C_D \cdot \frac{1}{2} \cdot \rho \cdot A \cdot v^2 \tag{3.4}$$

The formula to calculate C_D is dependent on the current α and the α at which the airfoil stalls (α_{stall}). Below α_{stall}, Equation (3.5) is valid. Above α_{stall}, a different equation is used (3.6). Note that in (3.6), a flat plate model is integrated to bound the C_D at higher αs. In (3.6), the second line is empirically validated by Hoburg and Tedrake [12].

The values for $C_{D,0}$, $C_{D,\alpha}$, $C_{D,\alpha_{stall}}$, and α_{stall} are gained using XFLR. As with $C_{L,\alpha_{stall}}$, $C_{D,\alpha_{stall}}$ is validated using experimental data. $C_{D,finiteplate}$ is chosen arbitrarily to account for the losses of the airfoil having finite dimensions.

$$C_D = C_{D,0} + C_{D,\alpha} \cdot \alpha \tag{3.5}$$

$$C_D = \min \begin{cases} C_{D,0} + C_{D,\alpha} \cdot \alpha_{stall} + C_{D,\alpha_{stall}}(\alpha - \alpha_{stall}) \\ C_{D,0} + C_{D,finiteplate} \cdot 2 \cdot \sin(\alpha)^2 \end{cases} \tag{3.6}$$

3.2.1 Sensitivity Analysis

In the drag model, there are some parameters that can be set externally based on the physical properties of the airfoil. These are $C_{D,finiteplate}$, $C_{D,0}$, $C_{D,\alpha}$, and $C_{D,\alpha_{stall}}$. To find how they influence the total drag generated by the airfoil, a sensitivity study is done. The results of this study is a good indication of the influence of each parameter on the error between the simulation and the real world.

F_D has a linear relation with C_D, so any relation of the parameters with C_D will by definition also be the same as their relation with F_D.

Unless being the subject of the sensitivity study, the values of the parameters are set to the values found in Table 3.3.

$C_{D,0}$

$C_{D,0}$ is the drag generated by the airfoil when $\alpha = 0$. The way it is used in (3.5) and (3.6) indicates that the relation between $C_{D,0}$ and C_D is linear. This means that the relation between $C_{D,0}$ and F_D is also linear and acts as an offset over the entire graph.

$C_{D,finiteplate}$

$C_{D,finiteplate}$ is used to scale the flat plate model to account for losses due to the dimensions of the airfoil being finite. Its relation to C_D is harder to deduce from the formulas alone, since it only appears in (3.6) inside of the *min* function. To get an indication of its relation to C_D and F_D, a series of plots have been made that can be seen in Figure 3.4.

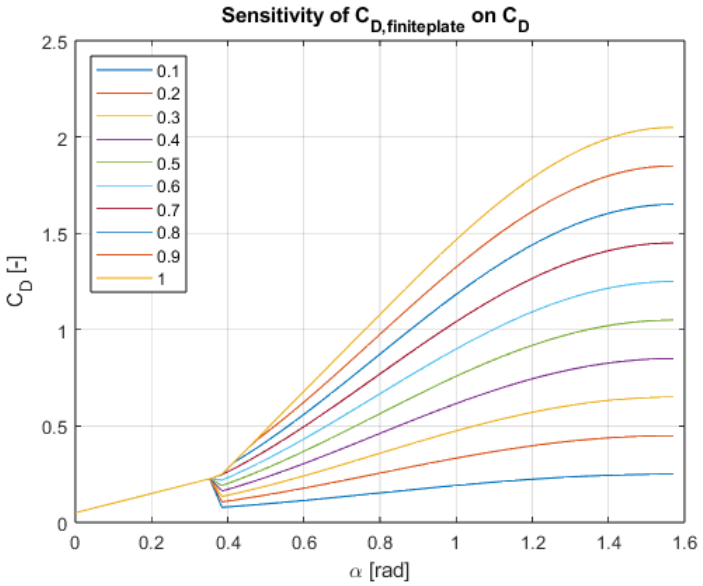


Figure 3.4: Sensitivity of C_D on $C_{D,finiteplate}$.

The values of $C_{D,finiteplate}$ are chosen to represent a range where the influence is clearly visible while not considering values that have no meaning (i.e. $C_{D,finiteplate} \leq 0$).

As expected, $C_{D,finiteplate}$ has no influence on C_D before stall occurs. After this point, it completely dominates C_D as it is lower than the top part of (3.6) if $C_{D,finiteplate}$ is lower than 0.8. At 0.8 and above, the influence of the top part of (3.6) can be seen for $\alpha < 0.45$ when $C_{D,finiteplate} = 1$, with the limit α lowering with lower $C_{D,finiteplate}$.

Figure 3.4 shows that the influence of $C_{D,finiteplate}$ on C_D is very large after α_{stall} and it is thus very important to find a realistic value if the error of the simulation with reality is to be minimized.

$C_{D,\alpha}$

$C_{D,\alpha}$ is the coefficient of the influence α has on C_D. It appears in (3.5) and (3.6) in the top part. It does not appear in the bottom part of (3.6). This indicates that its influence will most likely be in the lower regions of α, until the bottom part of (3.6) becomes lower than the top part.

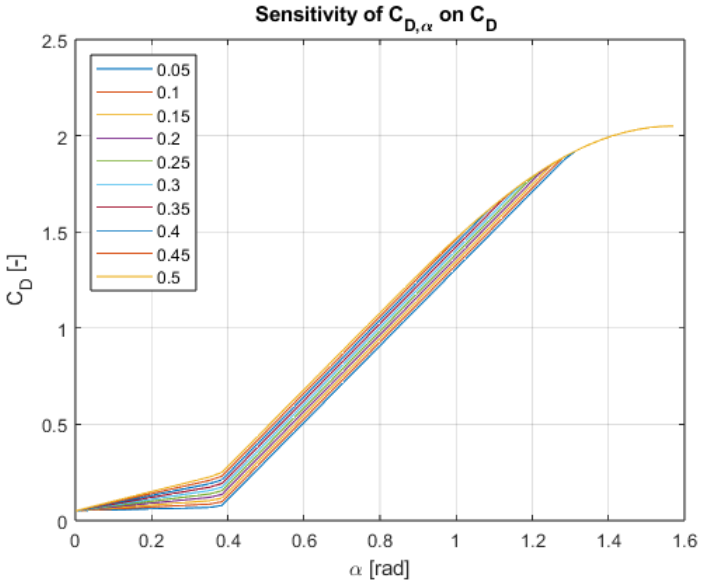


Figure 3.5: Sensitivity of C_D on $C_{D,alpha}$.

Figure 3.5 shows C_D for a varying value of $C_{D,\alpha}$. As predicted, its influence is only visible during pre-stall α and post-stall α until the flat plate model gives a lower C_D.

The values of $C_{D,\alpha}$ are chosen to represent a range where the influence is clearly visible while not considering values that have no meaning (i.e. $C_{D,\alpha} = 0$ or values that will not be visible due to constraints on C_D).

The influence of $C_{D,\alpha}$ on C_D is a lot less pronounced than the influence of $C_{D,finiteplate}$ on C_D. This means that the importance of getting an accurate value is less than that of $C_{D,finiteplate}$. However, since the sensitivity of C_D on $C_{D,\alpha}$ is not 0, there still is some merit in getting an accurate value.

$C_{D,\alpha_{stall}}$

$C_{D,\alpha_{stall}}$ is the influence α has on C_D after stall. Its main goal is to model the effect of the induced drag generated by the turbulent flow that is present when the airfoil is stalled. Given its goal, it only appears in the top part of (3.6). This indicates that its influence on C_D will be visible only with $\alpha > \alpha_{stall}$. The results of this sensitivity analysis can be found in Figure 3.6.

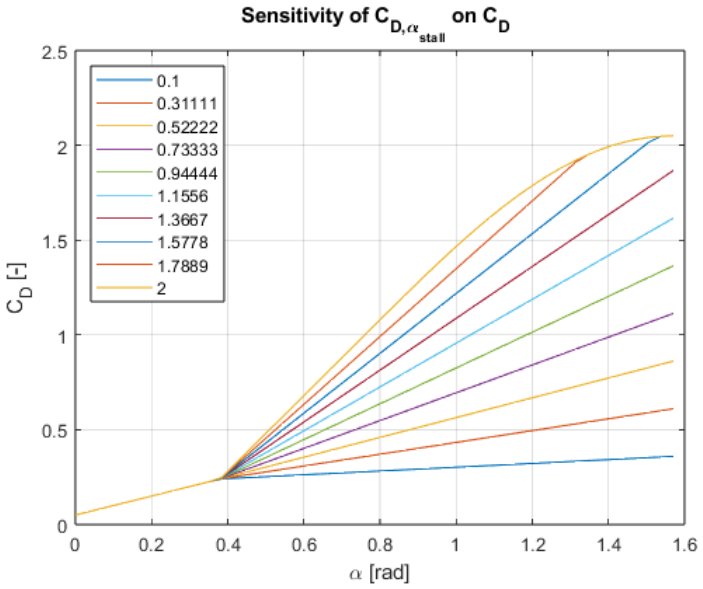


Figure 3.6: Sensitivity of C_D on $C_{D,\alpha_{stall}}$.

The values of $C_{D,\alpha_{stall}}$ are chosen to represent a range where the influence is clearly visible while not considering values that have no meaning (i.e. $C_{D,\alpha_{stall}} = 0$ or values that will not be visible due to constraints on C_D).

As expected, C_D is only sensitive to $C_{D,\alpha_{stall}}$ when $\alpha > \alpha_{stall}$. Any value below 2 provides a lower bound for C_D during stall. This can also be deduced from (3.6). The influence of $C_{D,\alpha_{stall}}$ on C_D is quite strong, making it important to get an accurate value.

Conclusion

The parameters that are used to calculate C_D work on distinct areas of the C_D/α curve. This means that there are clear tuning knobs available if in the simulation environment a part of the curve deviates from reality. The contribution of each of the parameters differs greatly. This is due to the nature of the upper limit provided by the $\sin^2$, which puts a relatively tight bound on the range of lower α values.

3.3 Pitching Moment

Finally, the simulation environment calculates the pitching moment of the airfoil. This is done based on the well known formula below, where M_θ is the pitching moment, ρ the air density, A the effective area, v the airspeed, and C_m the pitching moment coefficient.

$$M_\theta = C_m \cdot \frac{1}{2} \cdot \rho \cdot A \cdot v^2 \tag{3.7}$$

$$C_m = C_{m,0} + C_{m,\alpha} \cdot \alpha \tag{3.8}$$

$$C_m = C_{m,0} + C_{m,\alpha} \cdot \alpha_{stall} + C_{m,\alpha_{stall}}(\alpha - \alpha_{stall}) \tag{3.9}$$

3.3.1 Sensitivity Analysis

In the pitching moment model, there are some parameters that can be set externally based on the physical properties of the airfoil. These are $C_{m,0}$, $C_{m,\alpha}$, and $C_{m,\alpha_{stall}}$. To find how they influence the total pitching moment generated by the airfoil, a sensitivity study is done. The results of this study is a good indication of the influence of each parameter on the error between the simulation and the real world.

M_p has a linear relation with C_m, so any relation of the parameters with C_m will by definition also be the same as their relation with M_θ.

Unless being the subject of the sensitivity study, the values of the parameters are set to the values found in Table 3.4.

$C_{m,0}$

$C_{m,0}$ is the pitching moment generated by the airfoil when $\alpha = 0$. The way it is used in Equations (3.8) and (3.9) indicates that the relation between $C_{m,0}$ and C_m is linear. This means that the relation between $C_{m,0}$ and M_θ is also linear.

$C_{m,\alpha}$

$C_{m,\alpha}$ is the coefficient of the influence α has on C_m. It appears in (3.8) and (3.9).

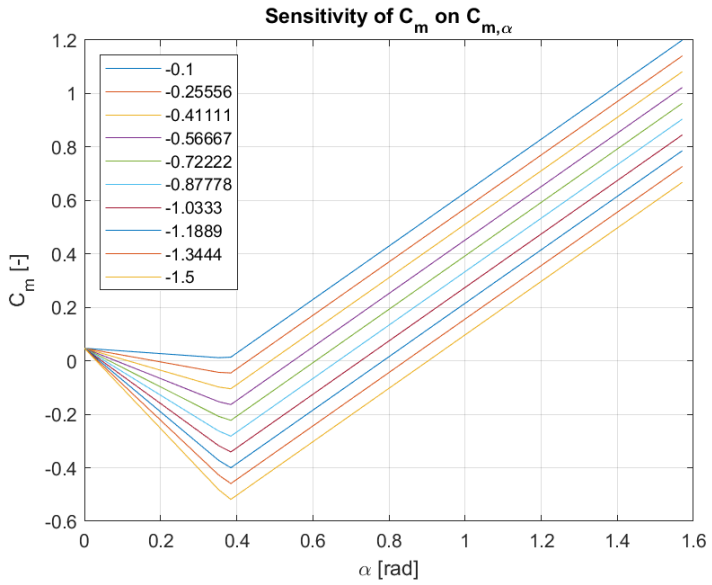


Figure 3.7: Sensitivity of C_m on $C_{m,\alpha_{stall}}$.

Figure 3.7 shows C_m for a varying value of $C_{m,\alpha}$.

The values of $C_{m,\alpha}$ are chosen to represent a range where the influence is clearly visible while not considering values that have no meaning (i.e. $C_{m,\alpha} = 0$ or values that will not be visible due to constraints on C_m).

The influence of $C_{m,\alpha}$ on C_m is quite strong. This means that the importance of getting an accurate value is important.

$C_{m,\alpha_{stall}}$

$C_{m,\alpha_{stall}}$ is the influence of α has on C_m after stall. This indicates that its influence on C_m will be visible only with $\alpha > \alpha_{stall}$. The results of this sensitivity analysis can be found in Figure 3.8.

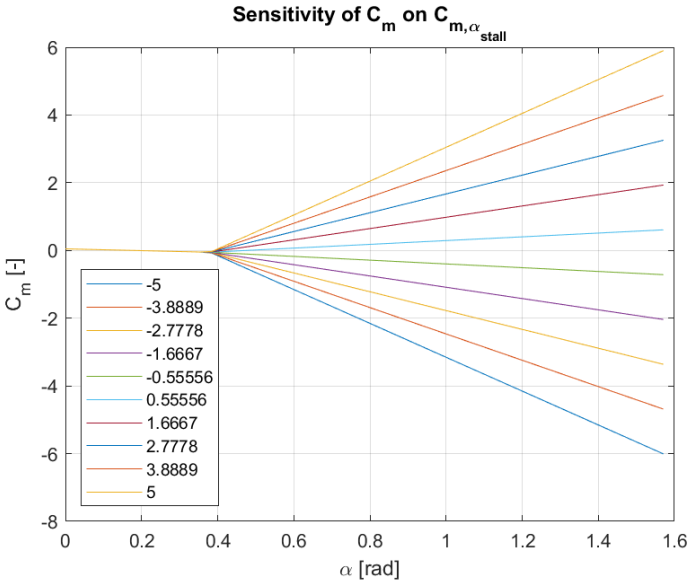


Figure 3.8: Sensitivity of C_m on $C_{m,\alpha_{stall}}$.

The values of $C_{m,\alpha_{stall}}$ are chosen to represent a range where the influence is clearly visible while not considering values that have no meaning (i.e. $C_{m,\alpha_{stall}} = 0$ or values that will not be visible due to constraints on C_m).

As expected, C_m is only sensitive to $C_{m,\alpha_{stall}}$ when $\alpha > \alpha_{stall}$. Any value below 2 provides a lower bound for C_m during stall. This can also be deduced from Equation (3.9). The influence of $C_{m,\alpha_{stall}}$ on C_m is quite strong, making it important to get an accurate value.

Conclusion

The parameters that are used to calculate C_m work on distinct areas of the C_m/α curve. This means that there are clear tuning knobs available if in the simulation environment a part of the curve deviates from reality. The contribution of each of the parameters differs greatly. This is due to the nature of the upper limit provided by the $\sin^2$, which puts a relatively tight bound on the range of lower α values.

3.4 Validation

The experimental data captured for Chapter 2 can be used to get an estimate of the error between the theoretical equations from the simulation and the real world. The easiest method of validation is to compare the pitch angle from experimental data with the theoretical pitch angle. The theoretical pitch angle follows from the simple balance of vertical forces given by Equation (3.10). Note the added $\frac{1}{\cos\alpha}$ to F_D, which is used to calculate the vertical component of the drag.

$$F_z = F_L \cdot \cos(\alpha) + F_D \cdot \frac{\sin(\alpha)}{\cos(\alpha)} \tag{3.10}$$

Substituting (3.1) and (3.4) into (3.10) and reforming it to calculate v yields Equation (3.11).

$$v = \sqrt{\frac{2 \cdot F_z \cdot \cos(\alpha)}{\rho \cdot S \cdot (C_L \cdot \cos(\alpha) + C_D \cdot \sin(\alpha))}} \tag{3.11}$$

With (3.11) and the definitions for C_L and C_D given by (3.2), (3.3), (3.5), and (3.6) respectively, it is possible to calculate the theoretical relation between the horizontal velocity and α. Figure 3.9 shows this relation, together with the experimentally found points.

Figure 3.9 shows that the experimental data has the same pattern as the theoretical fit, albeit with an offset. This offset can be explained by the accuracy (or lack thereof) of the values for the variables studied in the previous sections. This offset results in an error in the simulation where the pitch angle that corresponds to the wind speed is lower than in reality.

The impact of this offset might look quite severe. However, the main range of interest starts at $\theta \geq 47.5deg$,

which, since both graphs show an equivalent pattern, can be obtained by increasing the wind speed by this difference.

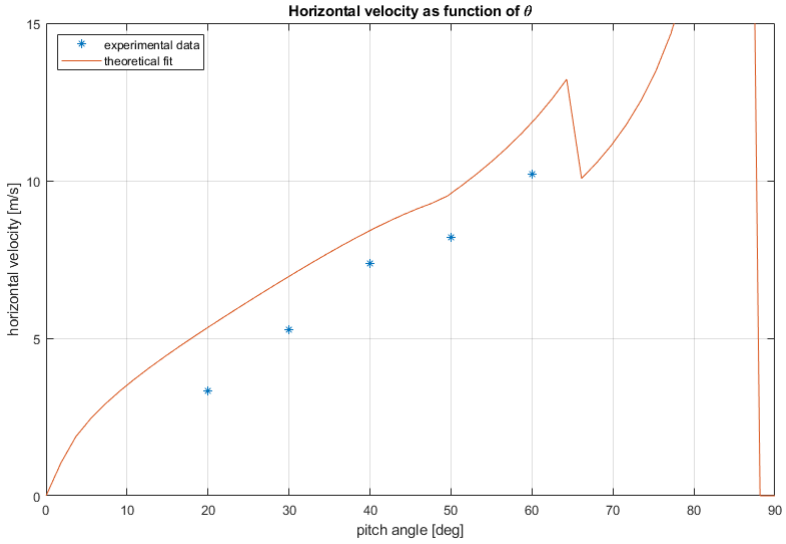


Figure 3.9: Horizontal velocity as a function of θ.

Chapter 4

Strategies

The innate maximum wind speed at which Marlyn can be landed without the risk of a tip-over ($8.4m/s$) is significantly smaller than the goal ($14m/s$), as shown in Chapter 2. That chapter also showed that the wind is actively trying to accelerate the aircraft backwards around the landing rod that is touching the ground. The motors are the main cause of the tip over, because they cannot fully control the aircraft. A small overshoot in the pitch control or a gust of wind suddenly stopping cannot be corrected for in time, resulting in a tip-over when the pitch angle is sufficiently high.
This chapter will discuss multiple possible solutions to the tip over problem, namely touch down detection, dynamic landing and pitch angle reduction. These possible solutions will be explored and compared with the side note that the solution should work at any location, not just the take-off location.

4.1 Touch Down Detection

In Section 2.3, it was found that not the wind, but the top and wing motors generating an unopposed moment causes tip-overs. An obvious solution would then be to stop the motors in the event of touch-down. Immediately stopping the motors at touch-down removes the moment generated by these motors, resulting in a moment backwards generated by the wind. If the stopping of the motors is done gradually, where the controllability is guaranteed, it would seem like an easy solution.

The main problem to solve here is to properly detect the touch-down event. If the event is detected too early, Marlyn could fall to the ground. If the event is detected too late, a tip-over could already have occurred. In the case of a false positive during flight, the outcome could lead to a disaster.

Multiple solutions for this problem will be proposed, ranging from using onboard sensors like the accelerometer to detect impact, or estimating the altitude with a combination of a barometer and a GNSS receiver, to using time of flight sensors to measure the distance to the ground, to using a mechanical switch on the part that touches the ground the first.

4.1.1 Altitude Estimation

Barometers and GNSS receivers can be combined to get the altitude, but they generate varying results. The barometers and GNSS receivers inside Marlyn will be fused with the IMU to provide a better altitude estimation. In practice, Marlyn always lands with the absolute value of the altitude being smaller than $30cm$.
This is good enough to be able to ramp down one motor, making tip-overs a lot rarer. The main downside of this solution is that it requires the world to be perfectly flat if Marlyn is to land on any other location than her take-off location. This constraint directly disqualifies this option, since one of the constraints is that the UAV should be able to land at any location where the ground is sufficiently flat.

4.1.2 IMU

Accelerometer data can be used to detect the impact accelerations that are generated at touch-down, if they are sufficiently large or sufficiently distinct from normally occurring vibrations. To determine if this solution can be used, first a baseline of the normal vibrations has to be known. Then a large amount of landing impacts need to be measured on all possible landing surfaces. These different data sets can then be compared to see if touch-down impact acceleration can be detected with sufficient certainty.
This does not require a high level of accuracy in low wind conditions. However, touch-down impacts in high

winds require a high level of accuracy and false positives should never occur.
To get a baseline of Marlyn's normal vibrations, logs of multiple flights in varying wind conditions have been analyzed. From these logs, a subset is taken where Marlyn is in hover and are cut off a second before Marlyn is disarmed manually at touch-down. This should give enough margin to never have a touch-down in the data, due to the current landing protocol of the drone.

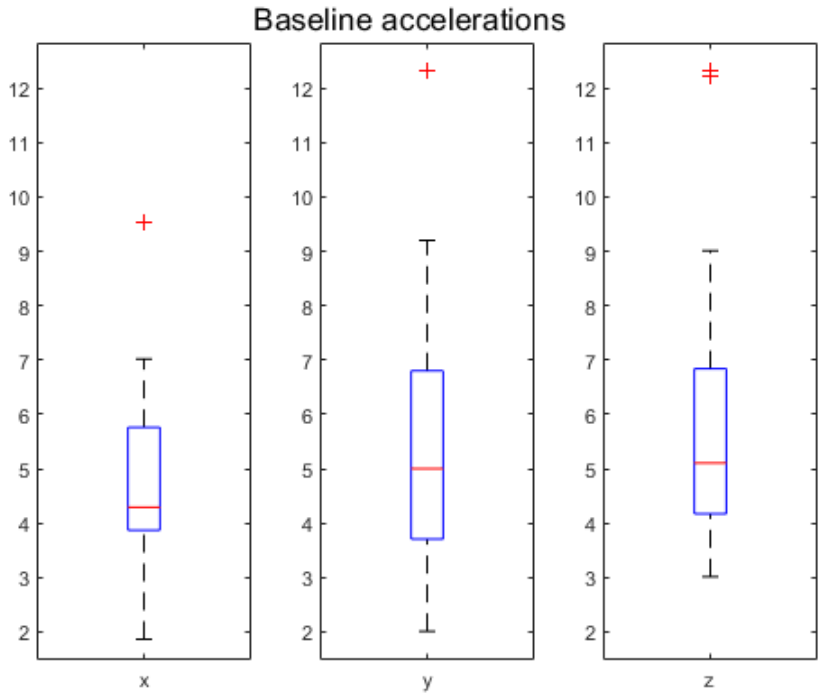


Figure 4.1: Vibrations during flight.

Figure 4.1 shows a box plot of the maximum normal vibrations during a normal flight.
Then, the logs of 20 landings on asphalt are analyzed, determining the impact acceleration by taking the maximum acceleration in the two seconds around the manual disarm. Since Marlyn has a carbon fiber frame and asphalt is very stiff, it is expected that the touch-down impact can be seen as a single peak above the baseline.

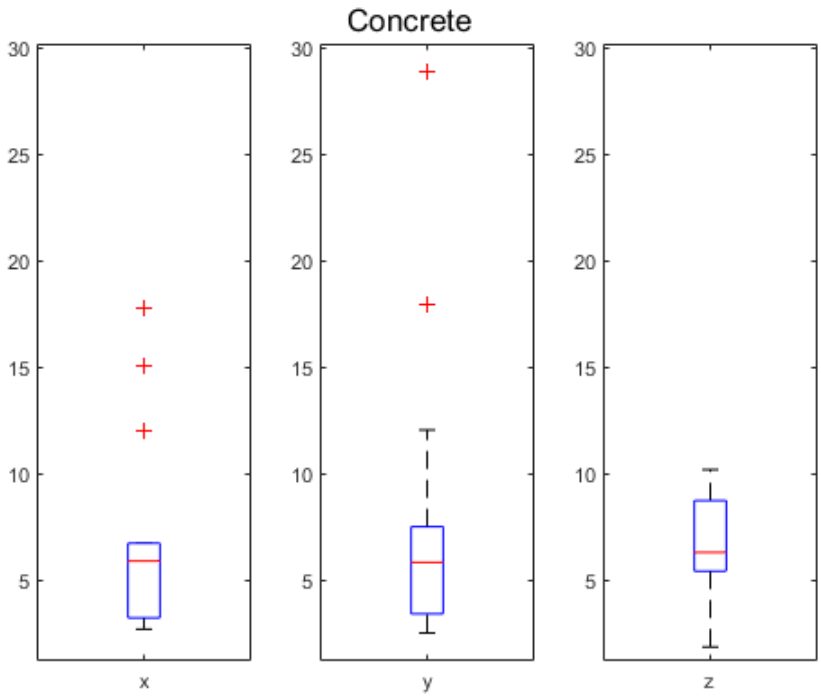


Figure 4.2: Vibrations during landing on concrete.

Figure 4.2 shows a box plot of the maximum acceleration in the two seconds around the manual disarm when landing on a hard surface like asphalt. It can be seen that these values are, apart from some outliers, not necessarily larger than the normal vibrations.

From the boxplots above, it cannot clearly be determined which one is describing the landing impact and which one is the baseline. Therefore, this cannot be used to reliably turn off the motors without risking the motors to be turned off mid flight.

4.1.3 Mechanical Switches

If it is impossible to use internal sensors to detect the landing, then maybe external sensors that directly sense the environment can be the solution.
The simplest method of detecting whether or not the aircraft is touching the ground is by using a switch or pressure sensor that triggers if enough load is on one of the landing rods. Van Ravels [21] has shown the advantages and disadvantages of such a system.

Van Ravels [21] shows that ground contact switches as implemented in the previous versions of Marlyn were not reliable in all environments. Especially muddy and/or sandy environments showed to significantly decrease the reliability of these switches. An alternative that was researched by Van Ravels [21] are pressure sensors higher up in the landing feet. These however, were very dependent on the angle at which the force was applied to the landing rod.

4.1.4 Range Finders

It is also possible to use range finders or time of flight sensors to determine the moment of touch-down. These sensors will have to be able to be pivoted since the body of Marlyn hangs in the wind. There are two types of range finders, light based or acoustics based. They both work on the same principle, namely they send a signal and measure the time it takes to return. Acoustic range finders usually measure the distance to the vegetation, where light based sensors have the precision to distinguish between vegetation and the ground. This last thing has been used by Anthony et al. [4] to measure crop heights. This lack of precision of the acoustics based sensors

makes them not suitable for this use case, since a drone cannot land on top of tall grass or a tree. Light based range finders do have the accuracy and penetration ability that are required for this use case.

4.1.5 Vision

Janousek and Marcon [13] have used a highly accurate flight controller in combination with a visual sensor and multiple IR diodes to land multiple UAVs. Optical flow has also been used to land UAVs with high accuracy on moving platforms (Herissé et al. [11], Kim et al. [15]). The advantage over optical flow and other vision systems over using a marker is that the UAV can land on any target location and does not rely on the presence of markers. Optical flow also has the advantage of being less computation intensive.
As stated in the section above, a mechanism is required to make sure the vision system is always pointing somewhat in a direction where it can see the ground directly below Marlyn. This can be done using lenses or servos, where lenses has the main benefit of not adding more moving parts.

4.1.6 Feasibility

Multiple methods of finding the right timing to turn off the motors have been discussed. Research and experiments have shown that using the altitude estimation, the IMU to measure the impact of landing, mechanical switches, and range finders are not feasible to be used reliably in all environments. Vision based solutions do show promising results if the difficulty of always pointing it straight down can be overcome.

4.2 Dynamic Landing

As any pilot is able to tell, no landing is the same. There is always a need to adjust the path based on external disturbances, sometimes severe enough to warrant a go-around. Different wind conditions require different approaches. Then why is it that this is hardly ever considered with VTOL drones? They mainly rely on the attitude control loops to keep the aircraft stable during a generic descend. This section explores methods of exploiting the control that is available during the approach to counter the influence of the wind on the landing.
Dynamic path planning is currently a hot topic in robotics. How can we make a robot navigate an environment as optimally as possible while the environment is prone to change at any moment. This has successfully been done for UAVs with multiple different additional optimization strategies (Lee, Walker, and Cohen [16], Wei, Žefran, and DeCarlo [22] and Chen et al. [6]). For most of these strategies, it is required to have an accurate estimation of the altitude of the aircraft with respect to the ground. This can be quite tricky to measure, as seen in the previous section. Due to this, a certain amount of robustness is required against an inaccurate altitude estimation, which will be discussed for every potential solution discussed below.

4.2.1 Artificial Potential Field

An interesting solution to dynamic path planning is using artificial potential fields (APFs). These APFs can be thought of as creating a 3D landscape where the no-fly zones, obstacles and other disturbances are hills. The steepness and height of the hill is determined by how repellent that location is to the drone. The drone itself can be thought of being a marble, rolling through the valleys of this landscape. Figure 4.3 shows a visual representation of this.

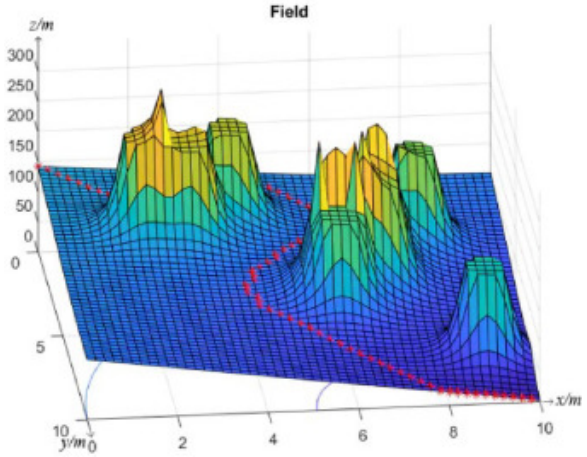


Figure 4.3: Visual representation of an AFP with a path planned through it (Wu et al. [23]).

This approach has the main benefit that there always will be an algebraic solution for the current commanded heading of the drone. Next to this, the calculations themselves are relatively simple and can easily be done on a microcontroller in real-time. A major downside is that drones can get stuck in local minima (also called dead zones or dead points). Chen et al. [6] has shown that there are ways around these dead zones by adding an optimal control law that is able to look ahead and therefor scout the dead zones ahead of time.

Marlyn has a control law in place called weather vane, which translates roll control actions into yaw control actions (see Section 1.1.2). This forces Marlyn to always be pointing her nose into the wind, which has an advantage when it comes to APFs, since it reduces the 3D problem to a 2D problem.

The APF used for landing Marlyn in the simplest case would be a valley going from the drone's position straight down. Adding the influence of the wind would not change the valley, since the pitch of the UAV is unconstrained by the APF. If a constraint to the pitch at touch-down is added, the valley changes shape. At this point, the UAV is no longer able to maintain its position against the wind, so it starts to drift. This means that there will be a horizontal acceleration at touch-down. This is a trade-off between constraining the pitch angle to prevent tip-overs and constraining the horizontal acceleration at touch-down to prevent tip-overs the other way.

This pitch constraint can be translated into an airspeed constraint by using equation 3.11. The same equation can also be used to get an accurate estimation of the airspeed at any moment during the landing approach. This gives all required inputs to shape the APF for any airspeed and any pitch constraint during touch down.

4.2.2 Dynamic Landing Manoeuvres

Dynamically unstable touch down

Next to dynamically planning a path to take the wind into account, adding a whip-like motion at the moment just before touchdown would also suffice the constraints. This would have to be done with sufficient altitude since the landing gears need room to be flicked.

4.2.3 Feasibility

The feasibility of this type of solutions can be mathematically determined. For this, the impact forces need to be added to Equation (2.13). These forces are calculated by Equation (4.1). Δt is assumed to be $0.05s$ due to having a stiff structure and some EPS foam in between the point of impact and the centre of mass.

A major assumption for this solution is that the controllers that normally control the angular rates of the aircraft, do not counter the angular velocity generated by the impact.

$$\vec{F}_{impact} = \frac{m \cdot \vec{v}}{\Delta t} \tag{4.1}$$

$$M_{\theta,impact} = \vec{F}_{impact} \cdot \vec{d}_{CoM,landingrod} \tag{4.2}$$

By adding the impact forces to (2.13), which then becomes (4.3), the feasibility can easily be mathematically

be determined by calculating the envelope in which this solution works, and then checking the envelope against safety and operational concerns. Only the impact force in the horizontal plane will be discussed.

$$M_{\theta,total,landingrod} = M_{\theta,wind,landingrod} + M_{\theta,propellers,landingrod} + M_{\theta,impact} + M_{\theta,c_g} \tag{4.3}$$

The impact forces work on the point of contact during touch down, and will thus need to be translated to the centre of mass in order to show their contribution around the point of contact. This is done by inverting the forces and transposing them onto the centre of mass.

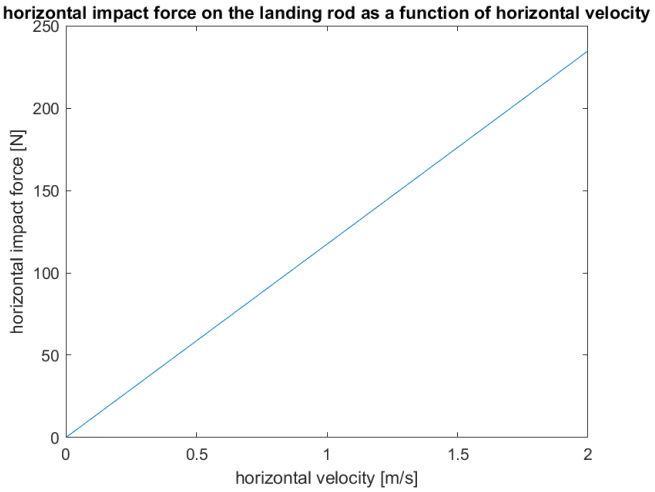


Figure 4.4: Horizontal impact forces on the landing rod, based on horizontal velocity.

Figure 4.4 shows the horizontal component of (4.1) as a function of the horizontal velocity. As expected by looking at (4.1), the figure is a linear plot with the maximum impact force at 10 m/s. Note that the forces shown are in the inverse direction of the horizontal velocity. This way they can be transposed onto the centre of mass, with the signs correct in the global coordinate frame.

Validating the feasibility is done using a simple dynamic model of a pendulum. Equation (4.4) shows the main equation used in the simulation. During the first $0.05s$ of the simulation, the impact moment given by (4.2) is added. For simplicity, the aerodynamical forces are considered constant and not changing with θ. This has the added benefit of only requiring to simulating the change in moments, as the equilibrium between motor-generated moment and wind-generated moment is assumed to always be exactly met.

$$\ddot{\theta} = \frac{m \cdot g \cdot d_{c_g,landingrod} \cdot sin(\theta)}{I} \tag{4.4}$$

The pivot point of the pendulum is not directly under the centre of mass when $\theta = 0$ (see Figure (2.1)). Instead, when $\theta > 0$, the pivot point changes to the top landing rod. This is incorporated in the simulation by switching the pivot point when $\theta > 0$. This causes a hard switch of the direction of the moment caused by the centre of mass.

The simulation is run for wind speeds varying from 0 to $10m/s$. For each wind speed, the drift speed is set to a value from 0 to $2m/s$. The code of the simulation can be found in Appendix C. Figure 4.5 shows the results of the simulation.

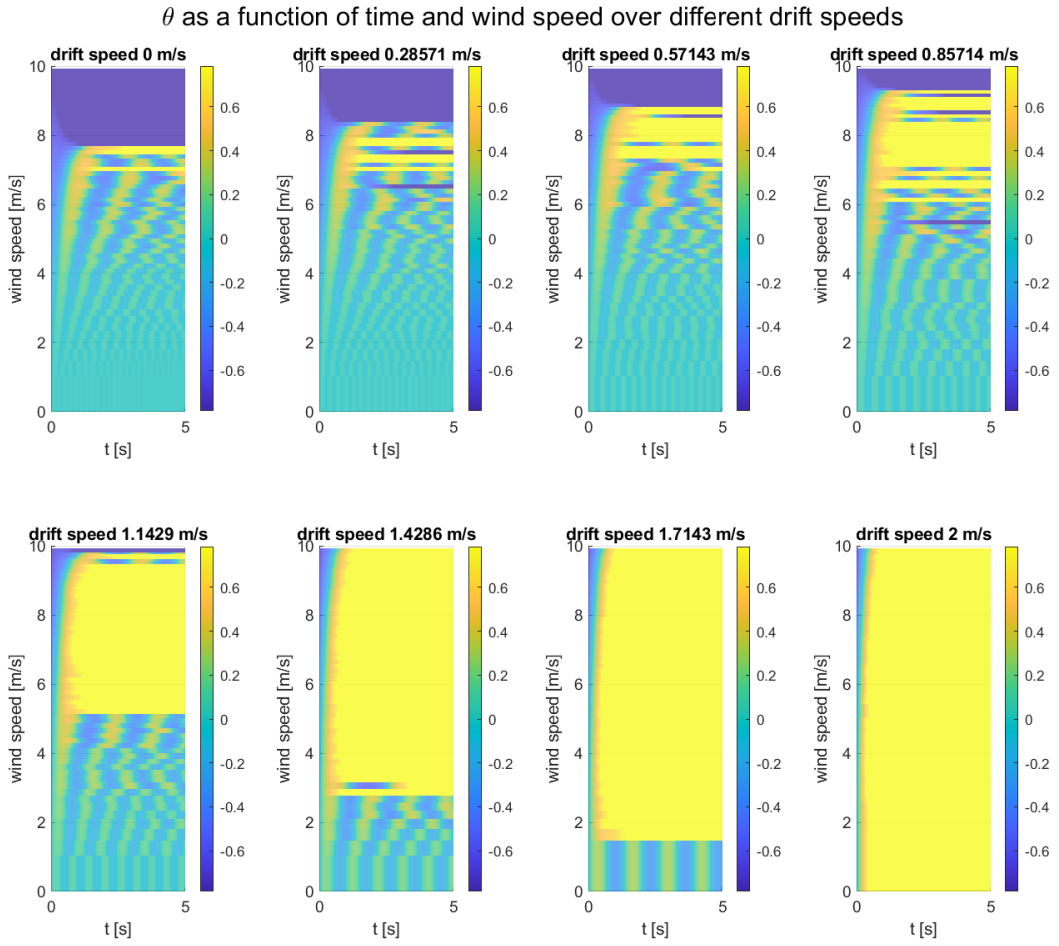


Figure 4.5: Results from a dynamic simulation comparing different drift speed scenarios.

The simulation results show that, given the assumptions, this solution is feasible. A drift speed of $1.1429 m/s$ is enough to generate a large enough moment to land without a tip over in wind speeds of up to $10 m/s$. Above $1.4 m/s$, due to limitations in the aerodynamical model used in the simulation, Marlyn always falls backwards. More research is required to validate this solution with less assumptions and outside of the simulation.

4.3 Pitch Angle Reduction

Marlyn has unique geometry which allows for rotation of the wing motors (see Section 1.1.1). This rotation allows for high authority in body-yaw movements in high winds. Next to this, the wing propellers are more efficient when they have inflow. This leads to the conclusion that rotating the wing motors into the wind in high wind scenarios would be beneficial, allowing for more efficient thrust generation and preventing Marlyn from pitching forward. This would in theory allow for safer landings, since the pitch angle deflection would be lower in higher winds.

4.3.1 Reducing Wind-prone Surface Area

A simple solution for reducing the pitch angle would be to rotate Marlyn perpendicular to the wind. While this reduces the efficiency of hovering significantly, it also significantly reduces the area influenced by the wind, and increases the angle at which tip-overs occur. To achieve the best of both worlds, the largest part of the descent can be done with the body into the wind. At a safe altitude, Marlyn could rotate perpendicular to the wind and then perform the landing.

This would also reduce the pitch angle at which Marlyn hovers, meaning the added mechanical or optical complexity would not be required to get a valid measurement close enough to the landing location if a time of flight sensor was added.
The main issue with this is that while facing the wind, Marlyn is dynamically stable. But when facing perpendicular to the wind, Marlyn is only statically stable, where any disturbance tries to push her back to the dynamically stable state. Controlling Marlyn this way would require an enormous control effort, something the wing motors are not able to provide.

4.3.2 Dynamic Control Allocation

Dynamic control allocation has been used in multiple cases successfully to compensate for failures and to optimally distribute the control task over the available actuators (Luo et al. [17], Härkegård [10], Tjønnås and Johansen [20], Mousaei et al. [18], De Wagter et al. [7]). Multiple methods for control allocation have been discussed by Johansen and Fossen [14]. By using these techniques, the required torque to stabilize or move the aircraft can be delivered by different actuators based on the state of the aircraft. This shifts the problem from a control problem to a control allocation problem.

As shown by De Wagter et al. [7], this can include passive lift devices like airfoils, making sure the total effect can be accounted for. These effects include any adverse effects like the airfoil's generated pitching moment and induced drag.

Due to the architecture of the firmware (Figure 1.4 in Section 1.1.2), it is relatively easy to take any required input for this dynamic control allocator by simply subscribing to the desired μORB topics.
To simplify things, instead of looking at it like a dynamic control allocation problem, it can be thought of a mixer problem. The mixer is the part of the control stack as shown in Figure 1.4 in Section 1.1.2 that determines which actuator is going to perform which control action. It is a bit like dynamic control allocation, but then statically defined.

The mixer is a simple matrix that converts the setpoints for pitch, roll, yaw and throttle into motor and control surface specific setpoints between -1 and 1 by multiplication. Equation (4.5) gives the basic form, where y is the output setpoints, A is the mixer matrix and $c = [\theta_c, \phi_c, \psi_c, T]$ the body axes setpoints given by the controllers.

$$\vec{y} = A\vec{c} \tag{4.5}$$

The commanded pitch angle that is mixed into the row offset, will be capped to the horizon ($\frac{\pi}{4} \leq \delta \leq 0$) to prevent any loss of efficiency and to prevent the row motors to actively pull the drone down. Figure 4.6 shows two options for transforming the commanded pitch angle into the row offset.

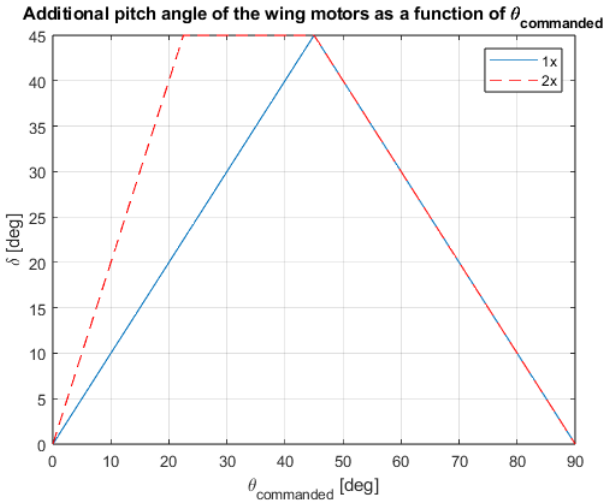


Figure 4.6: Additional motor pitch angle at different wind speeds.

By adding this offset, the angle at which the system applies the roll and yaw controlling forces is changed. Experiments have shown that this very quickly makes the system unstable. To account for this, the same

rotation is applied to the roll and yaw commands. This rotation is a simple 2D rotation of the two axis using the well known rotation matrix, given by (4.6).

$$\begin{bmatrix} \phi_c' \\ \psi_c' \end{bmatrix} = \begin{bmatrix} \cos\delta & -\sin\delta \\ \sin\delta & \cos\delta \end{bmatrix} \begin{bmatrix} \phi_c \\ \psi_c \end{bmatrix} \tag{4.6}$$

Marlyn's row propellers are less efficient without any inflow. For this reason, the maximum throttle of those motors is reduced in hover. This means they are mainly used for control and the bulk of the weight is carried by the tail motors, which are optimized for low to no inflow conditions. By tilting the row motors into the wind, a situation is created where there is inflow. This makes it no longer necessary to reduce the maximum throttle of the row motors. Scaling them up while facing the wind allows for an even larger benefit. By applying a scaling that starts at the reduced throttle at 0 row pitch angle and reaches maximum at 90 degrees total tilt, safe landing can be done in even higher winds.

A main advantage from this approach is that it is not required to know the exact altitude of the aircraft with respect to the ground. By angling the wing ends at a sufficiently high altitude (i.e. 20 meters or more), a large margin can be taken to account for terrain elevation differences. During this descend, the aircraft can be stabilized, posing no risk for any dynamic manoeuvres that need to be timed perfectly.

4.3.3 Feasibility

The experiments are about taking off, flying to an altitude of 20 meters and then sending a land command. The pitch angle during touchdown is measured and compared to the baseline case, together with the accelerations in the body x axis. The pitch angle should be below 47.5 degrees, while the accelerations should also be low enough to not cause a tipover the other way.
Since this solution is all about reducing the pitch angle in steady state, the experiments are about flying up to 20 meters in increasing wind and then measuring the average pitch angle over 10 seconds. This is done for the default configuration (which can both be used to validate the formulas in the simulator have been implemented correctly, as well as serving as a baseline to test the dynamic control allocation solution against).
Figure 4.7 shows the baseline case, without any modifications. In Chapter 2, the maximum pitch angle was determined to be 47.5°, or -0.83 rad. The baseline results show that indeed, between 8 and 9 m/s, that line is crossed.
Figure 4.8 shows the implemented dynamic control allocation solution. It can clearly be seen that the pitch angle is smaller for every situation but the very low winds. The line at -0.83 rad is not crossed at 9 m/s. Due to simulation constraints that reflect the real world (the drone is blown away before it can take off), wind speeds above 9 m/s cannot be tested.

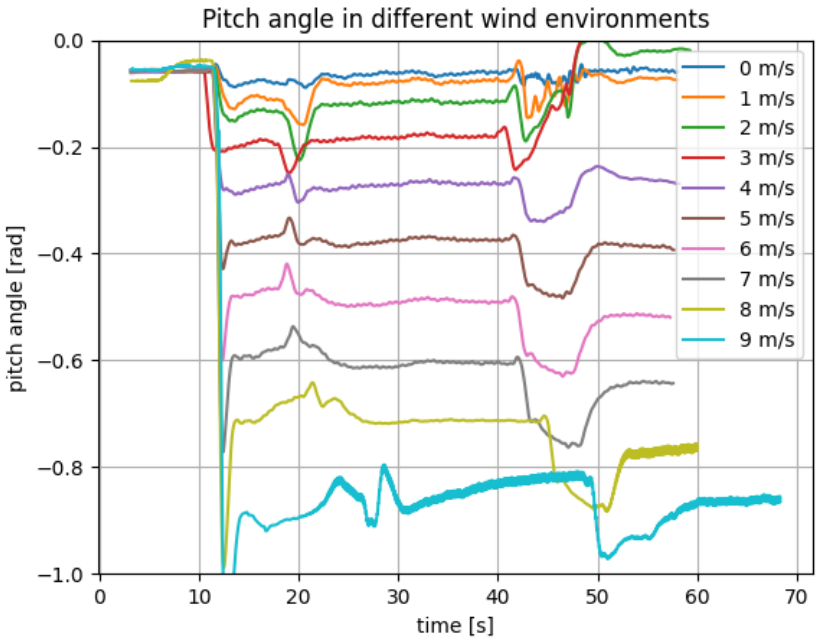


Figure 4.7: Pitch angle in time at different wind speeds.

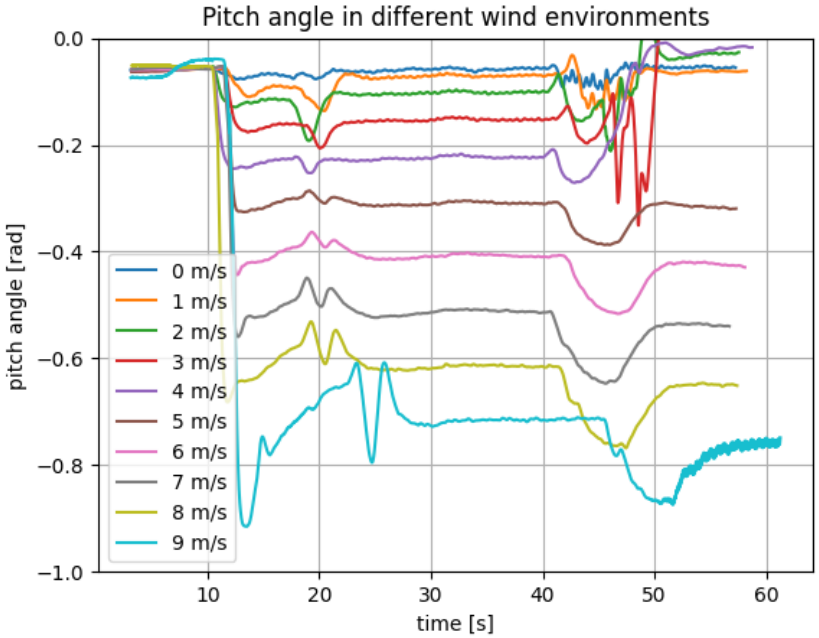


Figure 4.8: Pitch angle in time at different wind speeds with the additional offset mixed in.

Chapter 5

Conclusion

To increase the level of safe automation during the landing of VTOL tailsitters, the problems aroud tip overs after touch down have been studied. The most influential cause of these tip overs is not the centre of mass that is outside of the support area of the airframe, but the change in centre of rotation, which in turn disables one of the motors to apply a correcting moment. If a gust of wind is experienced during touch down and the gust dies down before the motors are turned off, the bottom tail motor can not counter the moment generated by the top tail motor and the wing motors, resulting in a tip over. Experiments have been done to determine the pitching moment and drag forces that are caused by the airframe in varying wind speeds. With the current airframe of Marlyn, tip overs after touch down can occur at wind speeds of 8 m/s or higher.

To increase the wind speeds Marlyn can safely land in without the risk of a tip over, multiple solution strategies have been discussed and explored for feasibility. The first solution discussed is detection of the touch down. By disabling all motors at the moment of touch down, the wind would blow Marlyn upright. This solution requires a reliable touch down detection. Using the IMU to detect the impact of touch down proved to be unreliable, especially on softer surfaces like grass. Mechanical switches in the tail relied heavily on the absence of dirt, dust and other debris that could inhibit the workings of the switches. The angle of the touch down also played a large role in the reliability of these mechanical switches.

Instead of detecting the touch down, alternatives were proposed to reliably, accurately and precisely estimate the altitude. A combination of the barometer and GNSS receivers was dismissed due to it only working on the take off location without requiring an accurate digital surface model of the flight area. Range finders and using vision sensors proved to be promising, and a lot of research has already been done on that field. Implementing them in a tailsitter where the sensor needs to point towards the ground to get a valid reading requires moving parts or lenses, which can get interfered with by the environment the same way they can interfere with the mechanical switches.

The second solution discussed is using a dynamic landing path that allows for Marlyn to drift with the wind just before touch down, lowering the pitch angle during touch down, or even generating a pitching moment backwards. Simulations have shows that this is an effective strategy, with a couple of caveats. The moment generated by the horizontal impact is strong enough to prevent the tip over. However, this moment will most likely be countered by the motors before it is able to come to full effect. The vertical impact is also not taken into account, which provides a moment that leads to more frontal tip overs. This combined with the lack of a counter moment once the motors have compensated for the impact moment, could lead to more tip overs instead of less. More research is required to validate this solution direction.

The last solution strategy uses the unique design of Marlyn's airframe, namely the capability of rotating the wing motors independently. By pointing the wing motors into the wind, part of the force required to counter the drag forces will be generated by these motors, reducing the required pitch angle Marlyn needs to have in order to counter the drag forces. An added benefit is that the wing propellers are more efficient if they have inflow, meaning the drivetrain itself will also be more efficient. For the experiments, the commanded pitch angle was mixed into the angle of the wing motors, capped at the horizon. This showed a reduction of steady state pitch angle in wind speeds of 9 m/s of 0.1 rad, or 5°.

Chapter 6

Recommendations

In Section 3.4, it is shown that the relation between θ and the wind speed in the simulation environment does not exactly match with the experimental data given the tuning parameters discussed in Chapter 3. This raises the question whether the F_L and F_D curves and parameters used are close enough to the real world. A recommendation for future work would be to validate the curves for F_L and F_D experimentally and thus possibly improve the simulation.

The analysis done in Section 4.2.3 is a simplified analysis, where it is assumed that the angular rate controller does not counter the horizontal impact, the additional pitching moment generated by a change in θ is not included, and that there is no vertical impact. In future work, this simulation can be expanded by removing the assumptions by implementing those effects in the simulation, or by using the simulation environment discussed in Chapter 3.

Section 4.3.3 discusses the method and simulated results of the dynamic control allocation solution. In this simulation, the efficiency increase of the wing propellers with inflow is not taken into account. This increase is significant enough to warrant further research, as it might improve the results gained even further. Additionally, an option exists to increase the throttle of the wing motors once the additional angle has saturated.

Bibliography

[1] URL: https://www.dji.com/nl/mini-4-pro?site=brandsite&from=landing_page.

[2] URL: https://www.af.mil/About-Us/Fact-Sheets/Display/Article/104469/mq-1b-predator/.

[3] URL: https://docs.px4.io/main/en/frames_vtol/tailsitter.html.

[4] David Anthony et al. "On crop height estimation with UAVs". In: *IEEE International Conference on Intelligent Robots and Systems* Iros (2014), pp. 4805–4812. ISSN: 21530866. DOI: 10.1109/IROS.2014.6943245.

[5] ATMOS. *Marlyn: Mapping And Surveying Fixed Wing VTOL Drone*. 2021. URL: https://www.atmosuav.com/product/marlyn.

[6] Yong Bo Chen et al. "UAV path planning using artificial potential field method updated by optimal control theory". In: *International Journal of Systems Science* 47.6 (2016), pp. 1407–1420. ISSN: 14645319. DOI: 10.1080/00207721.2014.929191. URL: http://dx.doi.org/10.1080/00207721.2014.929191.

[7] C. De Wagter et al. "Multi-Lifting-Device UAV Autonomous Flight at Any Transition Percentage". In: *EuroGNC 2013* (2013), pp. 1–15.

[8] James Drew. *New search for VTOL uavs May resurrect Bell tiltrotor*. Dec. 2019. URL: https://www.flightglobal.com/new-search-for-vtol-uavs-may-resurrect-bell-tiltrotor/119404.article.

[9] Elisha. *How VTOL technology changes the UAS field*. May 2023. URL: https://avnongroup.com/how-vtol-technology-changes-the-uas-field/.

[10] Ola Härkegård. "Dynamic control allocation using constrained quadratic programming". In: *Journal of Guidance, Control, and Dynamics* 27.6 (2004), pp. 1028–1034. ISSN: 15333884. DOI: 10.2514/1.11607.

[11] Bruno Herissé et al. "Landing a VTOL unmanned aerial vehicle on a moving platform using optical flow". In: *IEEE Transactions on Robotics* 28.1 (2012), pp. 77–89. ISSN: 15523098. DOI: 10.1109/TRO.2011.2163435.

[12] Warren Hoburg and Russ Tedrake. "System identification of post stall aerodynamics for UAV perching". In: *AIAA Infotech@ Aerospace Conference and AIAA Unmanned... Unlimited Conference*. 2009, p. 1930.

[13] Jiri Janousek and Petr Marcon. "Precision landing options in unmanned aerial vehicles". In: *2018 International Interdisciplinary PhD Workshop, IIPhDW 2018* (2018), pp. 58–60. DOI: 10.1109/IIPHDW.2018.8388325.

[14] Tor A. Johansen and Thor I. Fossen. "Control allocation - A survey". In: *Automatica* 49.5 (2013), pp. 1087–1103. ISSN: 00051098. DOI: 10.1016/j.automatica.2013.01.035.

[15] Jeongwoon Kim et al. "Outdoor autonomous landing on a moving platform for quadrotors using an omnidirectional camera". In: *2014 International Conference on Unmanned Aircraft Systems, ICUAS 2014 - Conference Proceedings* (2014), pp. 1243–1252. DOI: 10.1109/ICUAS.2014.6842381.

[16] Jeong-Won Lee, Bruce Walker, and Kelly Cohen. "Path Planning of Unmanned Aerial Vehicles in a Dynamic Environment". In: (Mar. 2011). DOI: 10.2514/6.2011-1654.

[17] Yu Luo et al. "Model predictive dynamic control allocation with actuator dynamics". In: *Proceedings of the American Control Conference* 2 (2004), pp. 1695–1700. ISSN: 07431619. DOI: 10.23919/acc.2004.1386823.

[18] Mohammadreza Mousaei et al. "Design, Modeling and Control for a Tilt-rotor VTOL UAV in the Presence of Actuator Failure". In: *IEEE International Conference on Intelligent Robots and Systems* 2022-October (2022), pp. 4310–4317. ISSN: 21530866. DOI: 10.1109/IROS47612.2022.9981806. arXiv: 2205.05533.

[19] PX4. *PX4 Architectural Overview*. 2023. URL: https://dev.px4.io/v1.11_noredirect/en/concept/architecture.html.

[20] Johannes Tjønnås and Tor A. Johansen. "Adaptive control allocation". In: *Automatica* 44.11 (2008), pp. 2754–2765. ISSN: 00051098. DOI: 10.1016/j.automatica.2008.03.031.

[21] Casper Van Ravels. *Project touch down*. 2020.

[22] Shangming Wei, Miloš Žefran, and Raymond A. DeCarlo. "Optimal control of robotic systems with logical constraints: Application to UAV path planning". In: *Proceedings - IEEE International Conference on Robotics and Automation* (2008), pp. 176–181. ISSN: 10504729. DOI: 10.1109/ROBOT.2008.4543205.

[23] Zhengtian Wu et al. "Robot path planning based on artificial potential field with deterministic annealing". In: *ISA Transactions* 138 (2023), pp. 74–87. ISSN: 00190578. DOI: 10.1016/j.isatra.2023.02.018. URL: https://doi.org/10.1016/j.isatra.2023.02.018.

Appendix A

Boxplots of pitch moment experiments

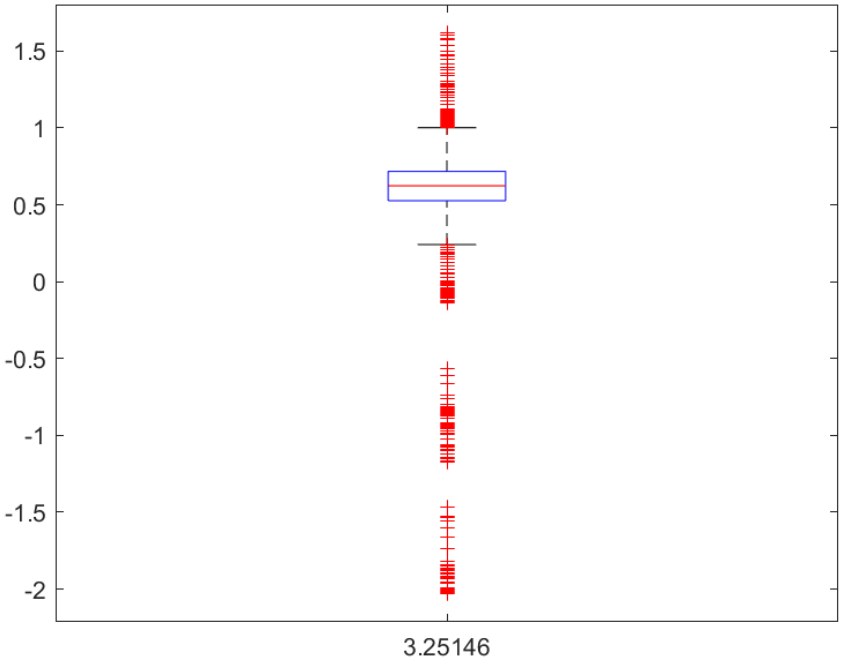


Figure A.1: Pitching moment against wind speed.

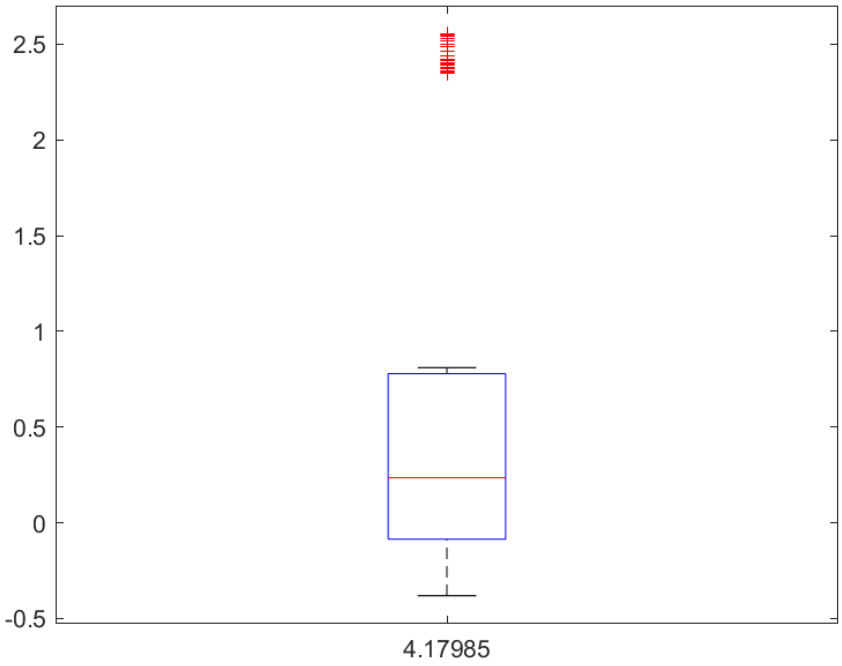


Figure A.2: Pitching moment against wind speed.

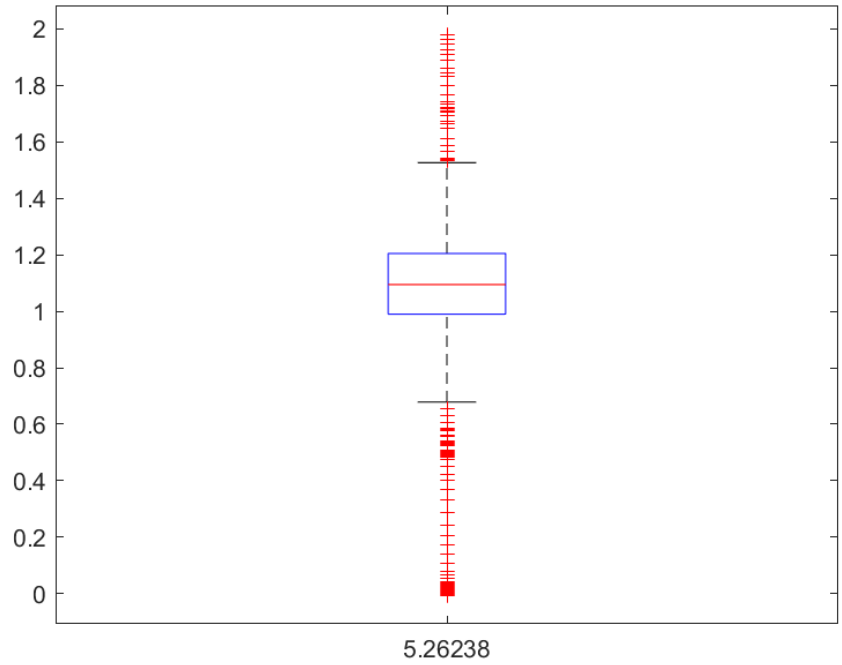


Figure A.3: Pitching moment against wind speed.

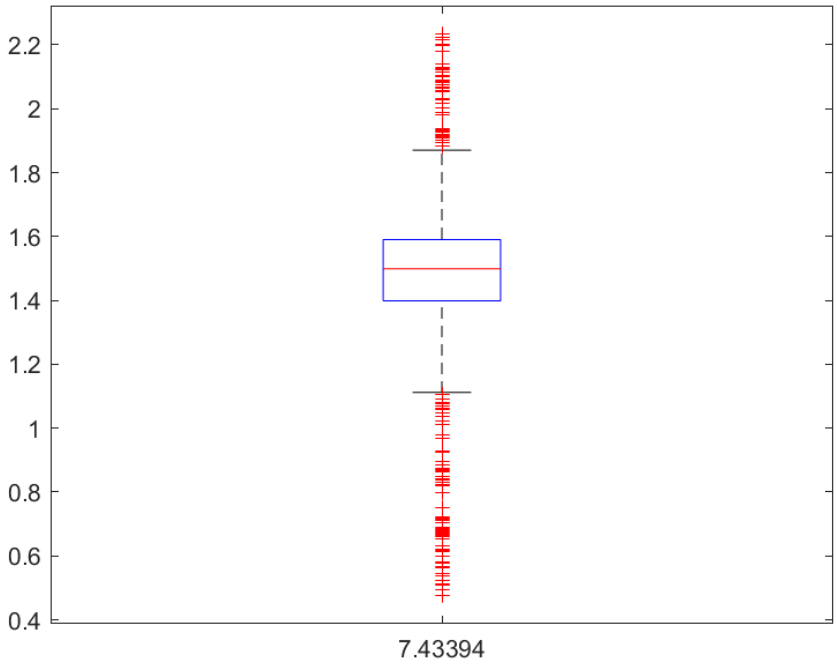


Figure A.4: Pitching moment against wind speed.

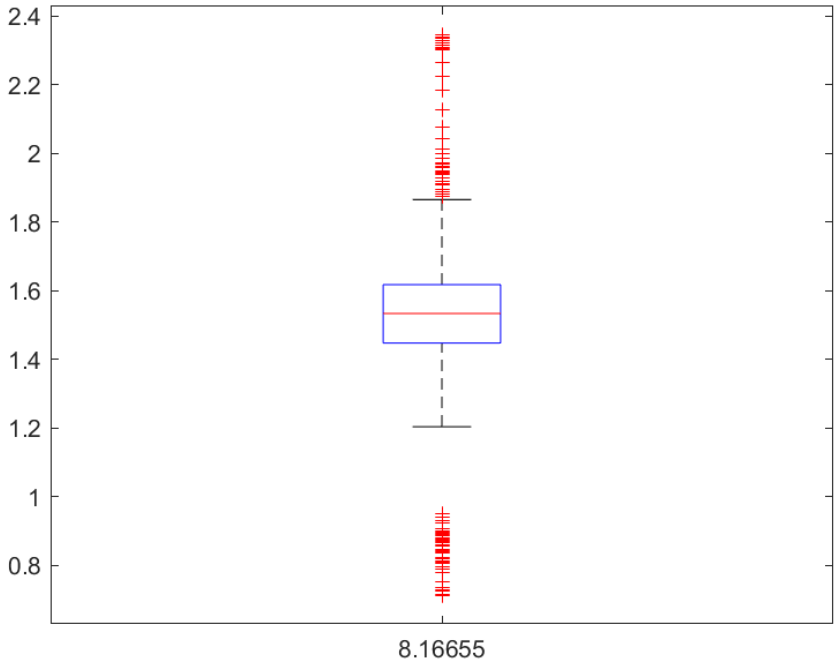


Figure A.5: Pitching moment against wind speed.

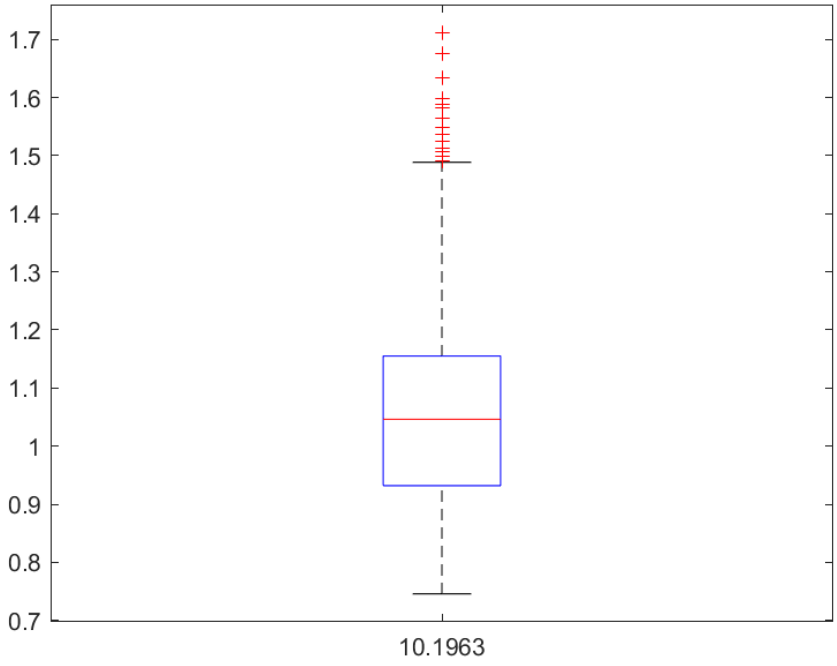


Figure A.6: Pitching moment against wind speed.

Appendix B

Matlab code for sensitivity study

C_L

```
clc; clear;
alpha = linspace(0,pi/2,50);
alpha_stall = 0.38;

%% finite plate
CL_alpha = 3.7242;
CL_alpha_stall = -1.5;

CL_finiteplate = linspace(0.1,1,10);

CL = zeros(10,50);

for i = 1:numel(CL_finiteplate)
    CL_normal = alpha(alpha<=alpha_stall).*CL_alpha;
    CL_stall = min(alpha_stall.*CL_alpha + CL_alpha_stall*(alpha(alpha>
       alpha_stall) - alpha_stall),...
    CL_finiteplate(i) .* 2 .* sin(alpha(alpha>alpha_stall)) .* cos(alpha(
       alpha>alpha_stall)));

    CL(i,:) = [CL_normal, CL_stall];
end

fig = figure(1);
plot(alpha, CL);
legend(string(CL_finiteplate),'Location','NorthEast');
grid on;
ylabel('C_L [-]')
xlabel('\alpha [rad]')
title('Sensitivity of C_{L,finiteplate} on C_L ')
saveas(fig, 'sensitivityCLFinitePlate.png')

%% Cd_alpha
CL_alpha = linspace(0.5,5,10);
CL_alpha_stall = -1.5;

CL_finiteplate = 1;

CL = zeros(10,50);

for i = 1:numel(CL_alpha)
    CL_normal = alpha(alpha<=alpha_stall).*CL_alpha(i);
```

```
    CL_stall = min(alpha_stall.*CL_alpha(i) + CL_alpha_stall*(alpha(alpha>
        alpha_stall) - alpha_stall),...
    CL_finiteplate .* 2 .* sin(alpha(alpha>alpha_stall)) .* cos(alpha(alpha>
        alpha_stall)));

    CL(i,:) = [CL_normal, CL_stall];
end

fig = figure(2);
plot(alpha, CL);
legend(string(CL_alpha),'Location','SouthWest');
grid on;
ylabel('C_L [-]')
xlabel('\alpha [rad]')
title('Sensitivity of C_{L,\alpha} on C_L ')
saveas(fig, 'sensitivityCLalpha.png')

%% Cd_alpha_stall
CL_alpha = 3.7242;
CL_alpha_stall = linspace(-5,-0.5,10);

CL_finiteplate = 1;

CL = zeros(10,50);

for i = 1:numel(CL_alpha_stall)
    CL_normal = alpha(alpha<=alpha_stall).*CL_alpha;
    CL_stall = min(alpha_stall.*CL_alpha + CL_alpha_stall(i)*(alpha(alpha>
        alpha_stall) - alpha_stall),...
    CL_finiteplate .* 2 .* sin(alpha(alpha>alpha_stall)) .* cos(alpha(alpha>
        alpha_stall)));

    CL(i,:) = [CL_normal, CL_stall];
end

fig = figure(3);
plot(alpha, CL);
legend(string(CL_alpha_stall),'Location','SouthWest');
grid on;
ylabel('C_L [-]')
xlabel('\alpha [rad]')
title('Sensitivity of C_{L,\alpha_{stall}} on C_L ')
saveas(fig, 'sensitivityCLalphastall.png')
```

C_D

```
clc; clear;
alpha = linspace(0,pi/2,50);
alpha_stall = 0.38;

%% finite plate
Cd_0 = 0.05;
Cd_alpha = 0.5;
Cd_alpha_stall = 2.0;

Cd_finiteplate = linspace(0.1,1,10);

```

```
Cd = zeros(10,50);

for i = 1:numel(Cd_finiteplate)
    Cd_normal = Cd_0 + alpha(alpha<=alpha_stall).*Cd_alpha;
    Cd_stall = Cd_0 + min(alpha_stall.*Cd_alpha + Cd_alpha_stall*(alpha(
       alpha>alpha_stall) - alpha_stall),...
    Cd_finiteplate(i) .* 2 .* sin(alpha(alpha>alpha_stall)).^2);

    Cd(i,:) = [Cd_normal, Cd_stall];
end

figure(1);
plot(alpha, Cd);
legend(string(Cd_finiteplate),'Location','NorthWest');
grid on;
ylabel('C_D [-]')
xlabel('\alpha [rad]')
title('Sensitivity of C_{D,finiteplate} on C_D ')

%% Cd_alpha
Cd_0 = 0.05;
Cd_alpha = linspace(0.05,0.5,10);
Cd_alpha_stall = 2.0;

Cd_finiteplate = 1;

Cd = zeros(10,50);

for i = 1:numel(Cd_alpha)
    Cd_normal = Cd_0 + alpha(alpha<=alpha_stall).*Cd_alpha(i);
    Cd_stall = Cd_0 + min(alpha_stall.*Cd_alpha(i) + Cd_alpha_stall*(alpha(
       alpha>alpha_stall) - alpha_stall),...
    Cd_finiteplate .* 2 .* sin(alpha(alpha>alpha_stall)).^2);

    Cd(i,:) = [Cd_normal, Cd_stall];
end

figure(2);
plot(alpha, Cd);
legend(string(Cd_alpha),'Location','NorthWest');
grid on;
ylabel('C_D [-]')
xlabel('\alpha [rad]')
title('Sensitivity of C_{D,\alpha} on C_D ')

%% Cd_alpha_stall
Cd_0 = 0.05;
Cd_alpha = 0.5;
Cd_alpha_stall = linspace(0.1,2,10);

Cd_finiteplate = 1;

Cd = zeros(10,50);

for i = 1:numel(Cd_alpha_stall)
    Cd_normal = Cd_0 + alpha(alpha<=alpha_stall).*Cd_alpha;
    Cd_stall = Cd_0 + min(alpha_stall.*Cd_alpha + Cd_alpha_stall(i)*(alpha(
       alpha>alpha_stall) - alpha_stall),...
```

```
    Cd_finiteplate .* 2 .* sin(alpha(alpha>alpha_stall)).^2);

    Cd(i,:) = [Cd_normal, Cd_stall];
end

figure(3);
plot(alpha, Cd);
legend(string(Cd_alpha_stall),'Location','NorthWest');
grid on;
ylabel('C_D [-]')
xlabel('\alpha [rad]')
title('Sensitivity of C_{D,\alpha_{stall}} on C_D ')
```

C_m

```
clc; clear;
alpha = linspace(0,pi/2,50);
alpha_stall = 0.38;
Cm_0 = 0.04712;

%% Cd_alpha
Cm_alpha = linspace(-0.1,-1.5,10);
Cm_alpha_stall = -1;

Cm = zeros(10,50);

for i = 1:numel(Cm_alpha)
    CL_normal = Cm_0 + alpha(alpha<=alpha_stall).*Cm_alpha(i);
    CL_stall = Cm_0 + alpha_stall.*Cm_alpha(i) + Cm_alpha_stall*(alpha(alpha
       >alpha_stall) - alpha_stall);

    Cm(i,:) = [CL_normal, CL_stall];
end

fig = figure(1);
plot(alpha, Cm);
legend(string(Cm_alpha),'Location','SouthWest');
grid on;
ylabel('C_m [-]')
xlabel('\alpha [rad]')
title('Sensitivity of C_{m,\alpha} on C_m ')
saveas(fig, 'sensitivityCmalpha.png')

%% Cd_alpha_stall
Cm_alpha = -0.26422;
Cm_alpha_stall = linspace(-5,-0.5,10);

Cm = zeros(10,50);

for i = 1:numel(Cm_alpha_stall)
    CL_normal = Cm_0 + alpha(alpha<=alpha_stall).*Cm_alpha;
    CL_stall = Cm_0 + alpha_stall.*Cm_alpha + Cm_alpha_stall(i)*(alpha(alpha
       >alpha_stall) - alpha_stall);

    Cm(i,:) = [CL_normal, CL_stall];
end

```

```
fig = figure(2);
plot(alpha, Cm);
legend(string(Cm_alpha_stall),'Location','SouthWest');
grid on;
ylabel('C_m [-]')
xlabel('\alpha [rad]')
title('Sensitivity of C_{m,\alpha_{stall}} on C_m ')
saveas(fig, 'sensitivityCmalphastall.png')
```

Appendix C

Matlab code for dynamic pendulum simulation

```
clear; clc; close all;
%%
alpha = linspace(deg2rad(10),deg2rad(90),101);
horizontal_velocity_initial = linspace(0,10,11);
vertical_velocity_initial = linspace(-1,-5,5);

alpha_stall = 0.38;

CL_alpha = 3.7242;
CL_alpha_stall = -1.5;
CL_finiteplate = 1;

Cd_0 = 0.05;
Cd_alpha = 0.5;
Cd_alpha_stall = 2.0;
Cd_finiteplate = 1;

S = 0.608;
F_z = 6 * 9.81;
rho = 1.225;

%% Calculate Cd and Cl
CL_normal = alpha(alpha<=alpha_stall).*CL_alpha;
CL_stall = min(alpha_stall.*CL_alpha + CL_alpha_stall*(alpha(alpha>
    alpha_stall) - alpha_stall),...
CL_finiteplate .* 2 .* sin(alpha(alpha>alpha_stall)) .* cos(alpha(alpha>
    alpha_stall)));

CL = [CL_normal, CL_stall];

Cd_normal = Cd_0 + alpha(alpha<=alpha_stall).*Cd_alpha;
Cd_stall = Cd_0 + min(alpha_stall.*Cd_alpha + Cd_alpha_stall*(alpha(alpha>
    alpha_stall) - alpha_stall),...
Cd_finiteplate .* 2 .* sin(alpha(alpha>alpha_stall)).^2);

Cd = [Cd_normal, Cd_stall];

%% Calculate wind speed corresponding to given alpha
v_from_alpha = sqrt((2*F_z.*cos(alpha))./(rho*S*(CL.*cos(alpha)+Cd.*sin(
    alpha))));

```

```
%% Calculate and plot impact forces
dt = 0.05;
m = 5.868922;                              % kg
figure;
plot(linspace(0,2,5), m*(linspace(0,2,5))/dt)
xlabel("horizontal velocity [m/s]")
ylabel("horizontal impact force [N]")
title("horizontal impact force on the landing rod as a function of
   horizontal velocity")
saveas(gcf, 'horizontal_impact_forces.png')

%% Simulation
t = linspace(0,5,5001);
wind_speed = v_from_alpha;
alpha = alpha(wind_speed <= 10);
wind_speed = wind_speed(wind_speed <= 10);
drift_speed = linspace(0,2,8);
results = zeros(size(wind_speed,2), size(drift_speed, 2), size(t,2));

for i = 1:numel(wind_speed)
    v = wind_speed(i);
    for j = 1:numel(drift_speed)

        % Calculate perceived wind speed
        d = drift_speed(j);
        perceived_wind_speed = v-d;
        if perceived_wind_speed < min(wind_speed)
            perceived_wind_speed = min(wind_speed);
        end

        % Find starting theta. starting theta_dot is assumed 0.
        theta0 = -deg2rad(90) + interp1(wind_speed, alpha,
           perceived_wind_speed);

        % Run simulation
        [t,y] = ode45(@(t,y) func(t,y,d), t, [theta0; 0]);
        results(i,j,:) = y(:,1);
    end
end

%% Plot results
figure;
for i = 1:numel(drift_speed)
    subplot(2,4,i)
    surf(t,wind_speed', squeeze(results(:,i,:)), 'FaceAlpha', 0.75)
    caxis([-pi/4,pi/4])
    view(2)
    shading flat
    title(['drift speed ' num2str(drift_speed(i)) ' m/s'])
    xlabel('t [s]')
    ylabel('wind speed [m/s]')
    zlabel('\theta [rad]')
    colorbar
end
sgtitle('\theta as a function of time and wind speed over different drift
   speeds')
set(gcf, 'Position', [100,100,1000,800])
saveas(gcf, 'driftsolutions.png')
```

```
93
94  %% Simulation function
95  function dydt = func(time,state,drift_speed)
96      dt_impact = 0.05;
97      m = 5.868922;                           % kg
98      g = 9.81;                               % m/s^2
99      height_in_hover = 0.64;                 % m
100     distance_top_cog = 0.19811;             % m
101     distance_hover_motors_cog = 0.3;        % m
102
103     height_cog = height_in_hover - distance_top_cog;
104     distance_contact_cog = sqrt(height_cog^2+distance_hover_motors_cog^2);
105     I = m * distance_contact_cog^2;
106
107     theta = state(1);
108
109     % Calculate moment caused by centre of mass
110     % and correct for switching of contact point
111     d = -1 * sign(theta) * (distance_hover_motors_cog * cos(theta)) ...
112         + height_cog * sin(theta);
113
114     moment = m * g * d;
115
116     % Add horizontal impact moment
117     moment = moment + (time<dt_impact) * (m .* drift_speed ./ dt_impact .*
        ...
118         (height_cog * cos(theta) + distance_hover_motors_cog * sin(theta)));
119
120     dydt = [state(2); moment/I];
121
122     % constrain to horizon
123     if theta < -pi/4
124         dydt = [0;0];
125     end
126
127     if theta > pi/4
128         dydt = [0;0];
129     end
130 end
```