A Development of an Open-Source Wind Drone

Thomas Gehrmann

Masterarbeit in Physik
angefertigt im Physikalischen Institut

vorgelegt der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität
Bonn

November 2016

I hereby declare that this thesis was formulated by myself and that no sources or tools other than those cited were used.

Bonn, ...
Date Signature

1. Gutachter: Priv.-Doz. Dr. Philip Bechtle
2. Gutachter: Prof. Dr. Jochen Dingfelder

Acknowledgements

First of all, I warmly thank my supervisor Philip Bechtle for venturing this extraordinary project with me, for his wise guidance and his ability to set the autopilot back north, when the mood went straight south.

I thank Prof. Dr. K. Desch and all the coworkers at the research group for the warm welcome and positive atmosphere. In particular I want to thank Tobias Schiffer for supporting the outdoor experiments and providing the tether-unroll-mechanism.

Furthermore, I thank Prof. Dr. Jochen Dingfelder for evaluating my master thesis although it is far from his field of research.

Special thanks go to Christoph Sieg, for revealing the mystery from countless lines of code, for his just as many ideas and his craftsmanship to construct a drone which withstood more crashes than Quax, der Bruchpilot, yet still conquers the upper winds.

I also want to thank Udo Zillmann, the last member of the Airborne Wind Energy Project, for his financial and intellectual support.

I am deeply grateful that my friends offered both distraction and help during the long nights of the writing process. Not to forget the encouragement after disastrous flights with total loss or unintentional attacks on the kindergarten. By name I want to thank Sarmed Hussein for the tea and cake breaks, Mario Engel for taking the pictures of the wind drone, Patrick Ahlburg for always being around for a chat and daydreams in our "Club der Denker", Moritz Sümmermann for motivation, support, encouragement and many other things I cannot list because I want to finish this thesis, Svenja Treu for lending me Moritz and Anika Dreher for providing me with food and endless love.

At the end my deep gratitude goes to my parents who gave me the possibility to study and who have always supported me in all possible ways.

Contents

CHAPTER 1

Introduction

> Climate change is no longer some far-off problem; it is happening here, it is happening now.
>
> *(Barack Obama)*

Climate change by global warming is one of the biggest challenges humanity is facing today. The increase of the world's average temperature in the 20th century can not be explained by natural causes [1]. The reason for global warming lies in the anthropogenic increase of carbon dioxide concentration in the atmosphere. Fossil fuels have been identified as the main cause for this increase [2].

As depicted in Fig. 1.1, the world energy consumption is steadily increasing. Projections estimate that world energy consumption will grow from 161 PW h in 2012 to 239 PW h in 2040. Most of the energy is produced by fossil fuels. Coal, natural gas and liquid fuels account for 84 % of world energy consumption in 2012 and continue to provide 78 % in 2040 [3]. Flooding, aridity, stronger dispersion of diseases and acidification of the oceans are some of possible consequences, as the ecosystem is fragile.

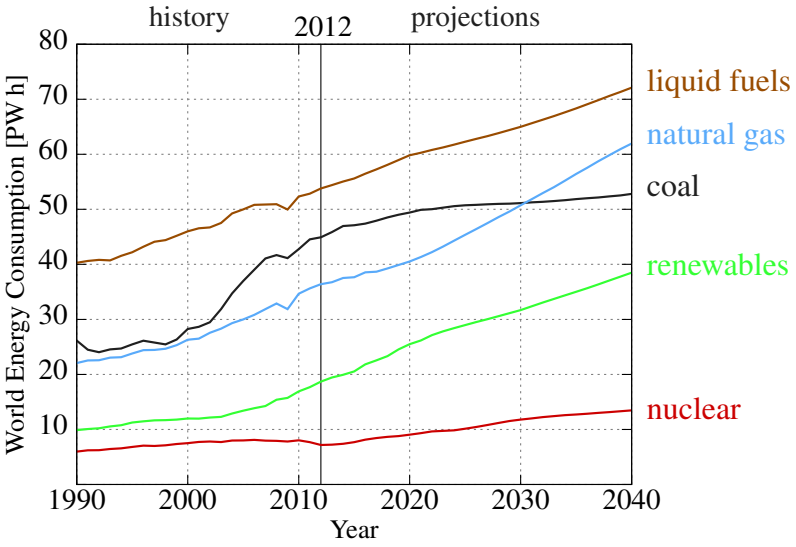


Figure 1.1: World energy consumption by source from 1990 - 2040. After 2012 the progress is projected by the U.S. Energy Information Administration. Projection reflects the effect of current policies, regulations and targets in major countries [3].

Renewable energy sources do not emit carbon dioxide. High-altitude wind power can supply 100 times todays world energy demand [4]. The biggest problem of renewable energy sources is that they are not dispatchable which makes them not suitable for base load, as they depend strongly on the environment. Wind speed varies during the year which results in a low capacity factor[1] of about 33 % [5]. However, high-altitude winds are faster and more consistent than winds in lower altitudes [6] which are tapped by conventional wind turbines. Reaching altitudes between 200 m and 1 000 m would increase the capacity factor significantly [7].

The height of conventional wind turbines is reaching its limit, as both construction costs rise exponentially and the material cannot withstand the high leverage forces. Airborne wind energy is an answer to tap winds in higher altitudes which cannot be harnessed by conventional wind power plants. It consists basically of an autonomously flying airborne device which is connected by a tether to the ground. Thus in comparison to the traditional wind turbines, the replacement of tower and fundament with a tether and a smart control algorithm saves material and enables the operation in much greater heights. In Chapter 2 the physics and the concept of airborne wind energy are explained in more detail, Chapter 3 introduces the nomenclature, specifies the required sensors and gives an example of optimal control.

Airborne wind energy is a promising approach with many advantages. Going without a tower and foundation, airborne wind energy systems need much less material consumption per unit of usable power output. Making upper winds accessible opens up many new sites for wind energy production, which reduces new investments in grid extensions. Faster winds generate more power since the wind power is proportional to the cube of the wind speed. Last but not least, high-altitude winds being steady means that airborne wind energy is available most of the time. Add it all up, airborne wind energy has the potential to replace fossil fuels and could be the energy miracle the environment is waiting for.

Many start-ups around the world are working on different concepts of airborne wind energy systems. No airborne wind energy system has been commercially realized with success yet. Each prototype costs in the order of 100 000 $, failure would quickly put the whole company in jeopardy. The aim of this thesis is to create an open source, low-cost test platform to experiment with different flight patterns and control algorithms. The algorithms and the software framework is illustrated in Chapter 4. The device is an autonomous flying glider plane held by a tether. Chapter 5 gives a detailed description of the modified glider plane, the hardware and the set-up, so that interested people can copy it and try some adventurous ideas. In Chapter 6 the results are presented and Chapter 7 concludes this thesis.

[1] The capacity factor of a power plant is the ratio of the actual energy produced in a given period, to its maximum potential, i.e. running full time at nameplate capacity

CHAPTER 2

Physical Foundations of Airborne Wind Energy

This chapter explains the functioning and discusses the fundamental physical limits of airborne wind energy. First a short recapitulation of the aerodynamics of a rigid wing is given in Sec. 2.1. Then, Sec. 2.2 covers the basic principle of airborne wind energy and describes several concepts of airborne wind energy systems.

2.1 Aerodynamics

Formulas and ideas in this section are taken from the book "Experimentalphysik 1" by Wolfgang Demtröder [8].

The aerodynamic force is the prime reason for both the tether tension which is used for power generation, and the flying ability of the aircraft. The aerodynamic force is a phenomenon that occurs in compressible fluids, i.e. gases. To derive the aerodynamic force, the following assumptions are made for simplification:

Perfect fluid Due to spatial different velocities of flow, frictional forces arise. Like in a perfect fluid with no viscosity, these forces are neglected here. For air streaming along a clean surface, this is a good approximation.

Incompressible flow The density of the fluid is at all points constant for an incompressible flow. While this is a good approximation for liquids, for gases this does not hold in general. However, for flow velocities below Mach[2] 0.3 the compressibility of air can be neglected [9].

Horizontal flow By assuming a horizontal flow of the fluid, changes of the potential energy due to altitude changes do not have to be considered.

Steady flow The velocity of the flow at any point does not change in time.

[2] The Mach number specifies the ratio of the flow velocity to the speed of sound. Mach 0.3 means a Mach number of 0.3 which is approximately $100\,\mathrm{m\,s^{-1}}$.

The aerodynamic force relies fundamentally on Bernoulli's equation (see B.1)

$$p + \frac{\varrho}{2}v^2 = p_0 = \text{const.}, \tag{2.1}$$

which states that the total pressure p_0 is constant. It is the sum of the static pressure p and the dynamic pressure $q = \frac{\varrho}{2}v^2$, where ϱ is the density of the fluid and v its velocity. A higher v increases the dynamic pressure, so the static pressure has to decrease.

Consider a fluid that flows through a narrowing pipe. The first half of the pipe has a cross-sectional area A_1, the second half a cross-sectional area A_2 which is smaller than A_1. By the continuity equation the velocity of the fluid increases by $\frac{A_1}{A_2}$ in the narrower part, which reduces the static pressure. This spatial velocity distribution can also be established by an object lying in the fluid, thus forcing the fluid to flow around the object. An example of this is air flowing around a wing, as shown in Fig. 2.1.

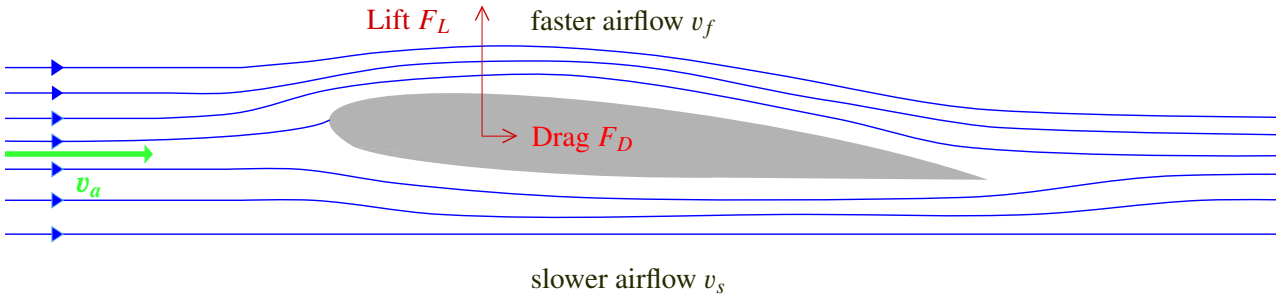


Figure 2.1: Steady flow of air along a wing.

Due to the asymmetric form of the wing, the airflow above the wing is faster than below the wing. Thus, a pressure difference Δp arises that leads to the lift force

$$F_L = \Delta p A = \frac{\varrho}{2}(v_f^2 - v_s^2)A = \frac{\varrho}{2}v_a^2(c_f^2 - c_s^2)A = \frac{\varrho}{2}v_a^2 C_L A. \tag{2.2}$$

The coefficient C_L is called lift factor and depends mainly on the form and the orientation of the wing. The wing area is given by A, the faster airflow velocity above the wing is v_f and the slower airflow below the wing is v_s with

$$v_f = v_a c_f \tag{2.3}$$
$$v_s = v_a c_s, \tag{2.4}$$

where $c_f > c_s$.

Similar to the lift, the drag force caused by the wing is

$$F_D = \frac{\varrho}{2}v_a^2 C_D A. \tag{2.5}$$

The drag points in the same direction as the apparent airspeed vector. The coefficient C_D is called drag factor and depends, similar to the lift factor, especially on the form and orientation of the wing. For modern wings, C_D is approximately 0.07 and C_L is around 1.

Summarized, aerodynamic forces F are given by

$$F = qAC, \tag{2.6}$$

with C a dimensionless coefficient.

The total aerodynamic force is divided into lift, drag and side force

$$F_S = qAC_S. \tag{2.7}$$

The side force is neglected, so the sum of the lift and drag results in the total aerodynamic force

$$F_a = \frac{\varrho}{2} v_a^2 A \sqrt{C_L^2 + C_D^2} = \frac{\varrho}{2} v_a^2 A C_R. \tag{2.8}$$

The overall coefficient for the total aerodynamic force C_R is close to the lift factor C_L since the drag factor is much smaller than the lift factor.

Aerodynamic moments M acting on the aircraft are similar defined as in Eq. (2.6), but to correct for dimension the wingspan l is included to give

$$M = qAlC. \tag{2.9}$$

The moments are divided into a roll, pitch and yaw moment, which act around the x-, y- and z-axis of the aircraft, respectively. The x-axis points from the center of gravity of the drone to the nose, the y-axis points starboard and the z-axis points downwards. The axes of the drone-fixed coordinate system are further described in Sec. 3.3.

Stability and Control Derivatives

The aerodynamic coefficients depend on one hand on the form of the wing, on the other hand on several flight and control parameters.

The main parameter that influences the lift factor C_L is the angle of attack α. It is the angle between the apparent airspeed vector and the x-axis of the drone-fixed coordinate system. Elevating the nose of the aircraft increases the angle of attack.

Figure 2.2 illustrates a typical dependency of C_L over α. The lift factor C_L is positive at $\alpha = 0°$ and increases approximately linearly with α within the working range (typically until $\alpha = 15°$ [9]). The stall angle α_{stall} produces the maximum lift factor. Above α_{stall} the lift factor decreases.

The gradient of the curve $C_{L,\alpha}$ until α_{stall} corresponds to the partial derivative

$$C_{L,\alpha} = \frac{\partial C_L}{\partial \alpha}. \tag{2.10}$$

Similarly, derivatives with respect to other parameters are taken. Derivatives with respect to flight condition parameters such as α are called *stability derivatives*. Derivatives with respect to deflection of control surfaces are called *control derivatives*.

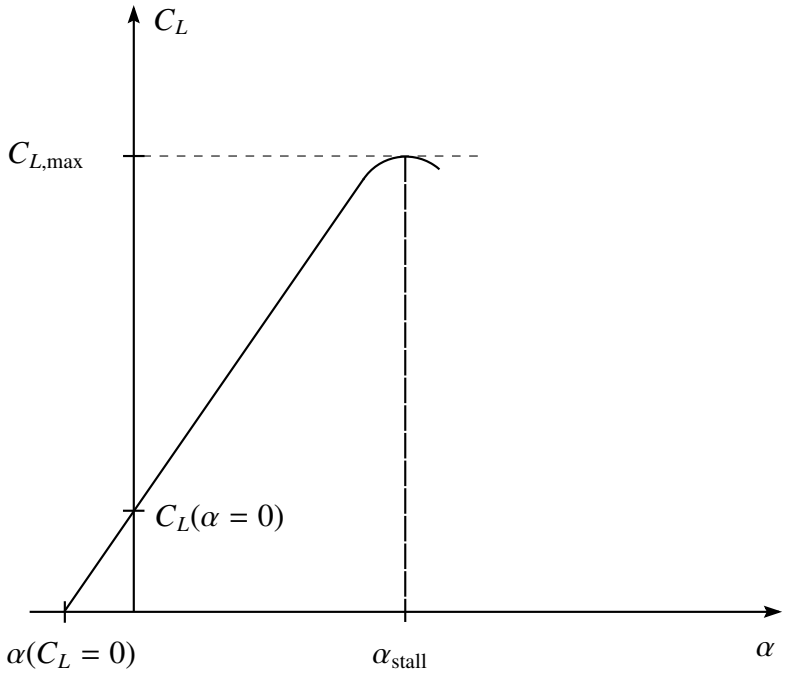


Figure 2.2: Characteristic curve of C_L [10].

This linearisation is often used to model the aerodynamics, as the coefficients can be easily computed by

$$C = C_0 + \frac{\partial C}{\partial \pi_1}\pi_1 + \frac{\partial C}{\partial \pi_2}\pi_2 + \cdots , \tag{2.11}$$

where π denominates different parameters.

The drag factor C_D can be modelled alternatively by [11]

$$C_D = C_{D,0} + k \cdot C_L^2, \tag{2.12}$$

where $C_{D,0}$ is the zero-lift drag and $k \cdot C_L^2$ is the lift-induced drag. This equation applies only in the working range since the drag increases even if the angle of attack α surpasses the stall angle α_{stall}.

The analysis of the angle of attack in Sec. 6.3 shows that α does not exceed 5° and Eg. (2.12) is a reasonable alternative to model the drag factor C_D.

The flight model described in Sec. 4.2 uses stability and control derivatives as well as Eq. (2.12) to simulate the dynamics of the aircraft.

2.2 Airborne Wind Energy

In his paper "Crosswind Kite Power" [12], Miles Loyd proposed two different concepts of airborne wind energy systems which are nowadays known as *drag mode* and *lift mode*. Both concepts are using crosswinds to increase the power output. By flying in crosswind direction, i.e. perpendicular to the wind direction, an air velocity much larger than the wind speed can be achieved. Since the total aerodynamic force is proportional to the square of the flow velocity, it induces a much larger tether tension than a wing flying in a static position.

The following gedankenexperiment, visualized in Fig. 2.3, illustrates which power can be extracted from the wind field. A tractor pulling a wing, which produces the aerodynamic force F_a, needs the power

$$P = \vec{v}_w \cdot \vec{F}_a = -v_w F_a \cos\gamma \tag{2.13}$$

to maintain its speed v_w. Turning this problem around, the power extracted from a wind field with velocity v_w where the tractor is at rest equals

$$P_{\text{wind}} = \vec{v}_w \cdot \vec{F}_a = v_w F_a \cos\gamma. \tag{2.14}$$

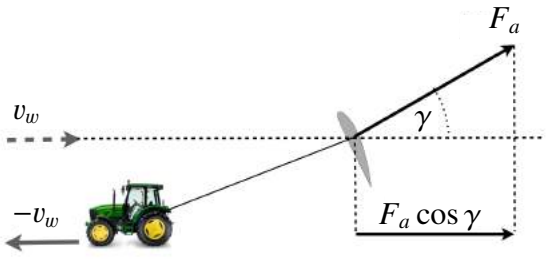


Figure 2.3: Gedankenexperiment to illustrate what power is extracted from the wind field [13].

The factor $\cos\gamma$ is called *cosine loss* which arises because the total aerodynamic force F_a is not in line with the wind direction. The main reason for γ is the elevation angle of the tether to achieve the intended altitude. Also, the aerodynamic force needs to compensate for the mass of the airborne device which increases γ further [13]. The power losses P_{loss} subtracted from the power extracted from the wind field P_{wind} yields the usable power that can be generated by an airborne wind energy system.

$$P_{\text{use}} = P_{\text{wind}} - P_{\text{loss}} \tag{2.15}$$

The power losses can be estimated by

$$P_{\text{loss}} \geq v_a F_D. \tag{2.16}$$

Inserting Eq. (2.14) and (2.16) into Eq. (2.15) yields

$$P_{\text{use}} \leq v_w F_a \cos\gamma - v_a F_D = \frac{\varrho}{2} v_a^2 A (v_w C_R \cos\gamma - v_a C_D). \tag{2.17}$$

Maximising the power output as a function of a the apparent airspeed v_a yields the maximum power

$$P_{\text{max}} = \frac{2}{27} \varrho v_w^3 A \frac{C_R^3}{C_D^2} \cos^3\gamma \tag{2.18}$$

that a flying device can harvest from the wind field for

$$v_a = \frac{2C_R}{3C_D} v_w \cos\gamma. \tag{2.19}$$

The optimal apparent airspeed (2.19) is given by $\frac{2}{3}$ of the maximal velocity that could be achieved by a tethered wing. Though, the maximal velocity $\frac{C_R}{C_D} v_w \cos\gamma$ would produce no power. This maximum

power production limit however does not include gravity and inertia of the mass of the system and assumes a constant C_L and C_D, and can be seen as an upper limit. The dynamics of the tether and its drag also reduce the performance.

Drag Mode

A way to produce power is to load the aircraft with little rotors driving a generator as implemented by the Californian start-up Makani [15] is shown in the bottom part of Fig. 2.4. These rotors, which are also used for launching and landing, put additional drag to the aircraft which explains the name of the mode. The circular flight pattern, that is demonstrated in the upper part of Fig. 2.4, assures an almost perpendicular motion of the aircraft to the wind direction to achieve a high cross-wind velocity. The power is generated on-board and has to be transported via the tether to the ground station. The tether consists of carbon fiber as structure and aluminium which serves as a conductor. The circle lies downwind on a sphere which is given by the constant length of the tether. These energy kites[3] reach altitudes between 80 m and 350 m. The 600 kW energy kite, including tether and ground station, weighs about 11 000 kg which is less than 10 % of a conventional wind turbine with a comparably rated power.

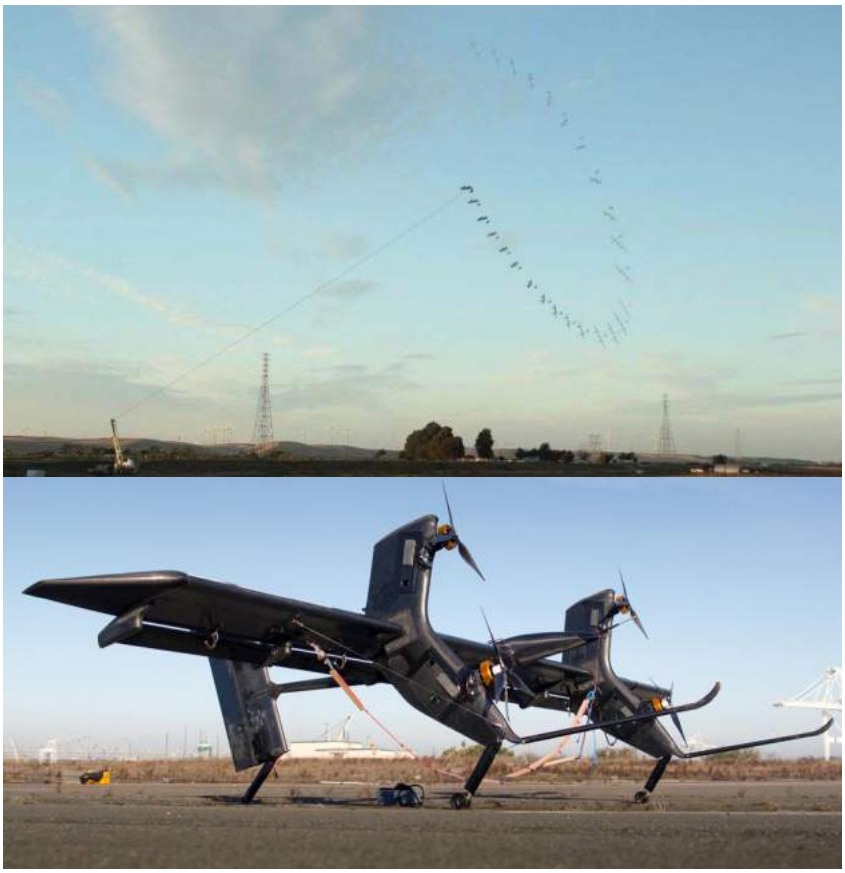


Figure 2.4: The upper picture is a time-laps shot of a test-flight of the wind drone in a circular flight pattern. The bottom picture shows the drone with the installed turbines [14].

To achieve the maximum power output, the additional drag $C_{D,\text{power}}$ of the rotors has to be

$$C_{D,\text{power}} = \frac{1}{2} C_D. \tag{2.20}$$

Flying at maximum velocity yields

$$v_a = \frac{C_R}{C_D + C_{D,\text{power}}} v_w \cos\gamma = \frac{2C_R}{3C_D} v_w \cos\gamma, \tag{2.21}$$

[3] The Makani terminology of the airborne device is energy kite.

which is the optimal apparent airspeed according to Eq. (2.19).

Lift Mode

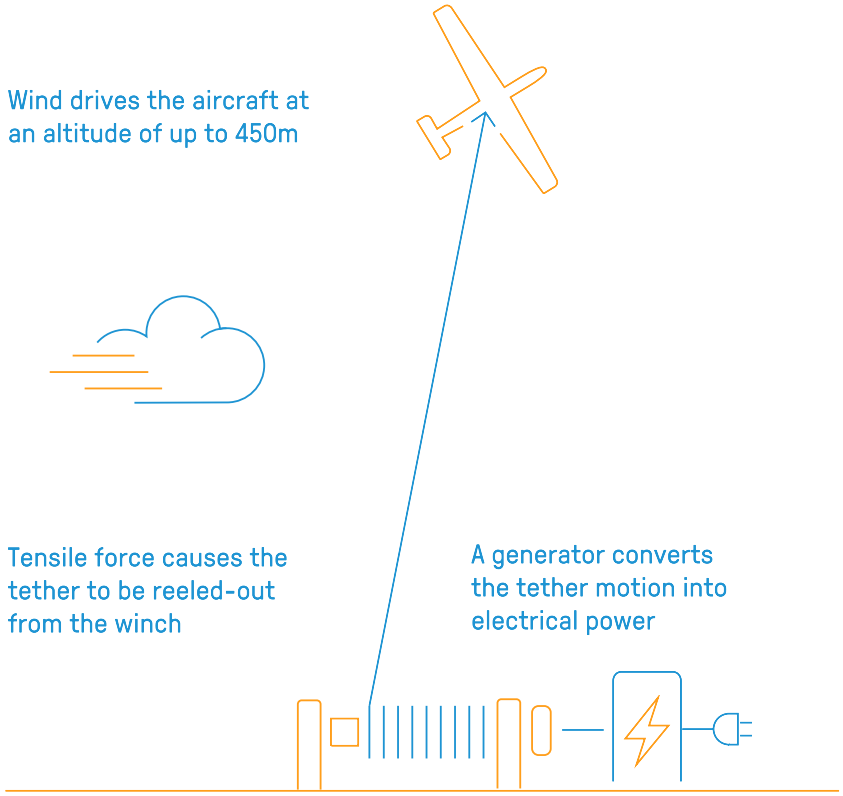


Figure 2.5: Schematic sketch of the lift mode from the start-up Ampyx Power [16].

Another way to produce power is to directly use the tether tension to unroll a drum which drives a generator. Since the tether length is limited, it has to be reeled in again. Therefore, the cycle is divided into two phases, a power production phase, where energy is generated, and a retraction phase, where energy is needed to reel in the tether. In the power production phase the power plane[4] flies a lying figure-8 pattern in crosswind direction at an altitude between 250 m and 450 m. As soon as the tether is completely unrolled, the aircraft glides towards the ground station to the initial height. Hence, the plane is flying periodic patterns between a minimum and maximum height. The power production is much larger than the power consumption due to the two different flight modes. Flying the figure-8 pattern produces a strong lift which results in a high tensile force, gliding back to the ground station reduces the lift and keeps the retraction phase short. The prototypes from Ampyx have wing spans from 5 m up to 40 m which corresponds to nominal power output of 50 kW to up to 2 MW, respectively.

The maximum power output can be gained by a reel-out velocity of the tether of

$$v_{\text{reel-out}} = \frac{1}{3} v_w, \tag{2.22}$$

so the power plane experiences a wind speed reduced by one-third. Flying at maximum velocity at this reduced wind speed $v_w' = \frac{2}{3} v_w$ results in the optimal airspeed

$$v_a = \frac{C_R}{C_D} v_w' \cos\gamma = \frac{2C_R}{3C_D} v_w \cos\gamma. \tag{2.23}$$

[4] The Ampyx terminology of the airborne device is power plane.

Other Airborne Wind Energy Systems

To tap winds in higher altitudes, a growing number of researchers and start-ups have come up with many different ideas. They vary between lighter-than-air systems and systems which depend on aerodynamic lift. Some groups are using soft kites, while others are convinced of rigid wing designs. Another classification is between systems which carry a turbine and generate electrical power on-board, and systems using the tether tension to produce power on the ground. Some of them are shortly described in this paragraph.

Not all concepts are designed to generate electricity. Like a sport kite being used to propel a surfer, an airborne wind energy system can be utilized to propel a ship or other loads. SkySails Marine [17] developed a towing soft kite system for the propulsion of ships which is illustrated in Fig. 2.6. The ground station is attached to the forecastle deck and launches the kite with a telescopic mast. The computer controlled traction kite flies a figure-8 pattern in 100 m to 300 m altitude and can generate up to 2 000 kW driving power [18]. This reduces fuel costs by 15 % on average per year. After use the kite flies to a static position above the ship and is retrieved and safely stowed by the ground station. SkySails Marine are the first who commercialized an airborne wind energy system, but had to declare insolvency in 2016 and had to be dissolved.

Figure 2.6: SkySails towing kite system in operation [17].

Figure 2.7: Buoyant Airborne Turbine [19].

Figure 2.7 demonstrates a lighter than air wind energy system developed by AltaerosEnergies [20]. The so called BAT (Buoyant Airborne Turbine) is similar to conventional wind turbines, just without the tower, which reduces a lot of material. The three blade turbine in a helium-filled shell is brought up to 600 m altitude which allows it to stay airborne even when no wind is blowing. Strong winds however cause a blowdown due to the increasing drag force. To alleviate the problems associated with blowdown, the BAT also produces aerodynamic lift [21]. The generated power is transferred by a tether to a portable ground station which gives the system a high mobility.

Sky WindPower [22] developed a copter system to harness wind at high altitudes as shown in Fig. 2.8. The flying electric generator is rated at 240 kW with rotor diameters of about 10 m. The rotors provide aerodynamic lift and generate electricity which is guided by a tether to the ground station.

Figure 2.8: Flying Electric Generator (FEG) from Sky WindPower [22].

Figure 2.9 shows a prototype form Enerkite [23] which is also a fixed-wing airborne wind energy system in lift mode like the system from Ampyx. But the control mechanism is located in the ground station to reduce weight of the wing. Two tethers are used to steer the wing. The largest prototype from EnerKite has a wing area of 125 m^2 and a rated power of 500 kW.

Figure 2.9: EnerKite lift mode airborne wind energy system [23].

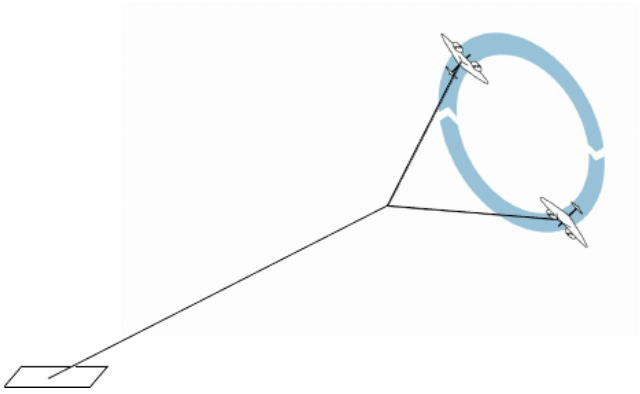


Figure 2.10: Multiple wing system to reduce tether drag [13].

The tether drag is a limiting factor especially of airborne wind energy systems flying in cross-wind direction. Future concepts are trying to minimize the tether drag by using a multiple wing system as illustrated in Fig. 2.10.

CHAPTER 3

Sensors and Coordinates

The wind drone is an autonomously flying unmanned aircraft. Hence, many sensors as described in Sec. 3.1 and a lot of real-time calculations are needed to stabilize the flight and ensure an optimal energy production. Nowadays, sensor technology and processing power are highly developed, which makes airborne wind energy possible, but due to defective measurements, a smart algorithm is needed to filter uncertainties out in real time. A filter and a simple example of optimal control is illustrated in Sec. 3.2. Sec. 3.3 introduces different coordinate systems necessary to compute the flight path and control parameters. This is followed by Sec. 3.4, describing the flight path of a lying figure-8 pattern.

3.1 Sensors

The sensors are the senses of the drone by collecting information about the environment. The functioning of the used sensors are shortly summarized in this section.

3.1.1 IMU

The inertial measurement unit (IMU) is a required device for an autopilot to measure the motion and attitude of the drone. It consists of an accelerometer, magnetometer and gyroscope.

Accelerometer

The accelerometer measures the proper acceleration in three dimensions. It basically consists of spring mass systems in the x-, y- and z-axis. The acceleration is measured by the deflection of the mass. By integrating the acceleration, the velocity and the position can be determined. The used accelerometer is a microelectromechanical system (MEMS). The mass and the spring are made of silicon which are only a few µm broad.

Magnetometer

The magnetometer is supported by the same structure as the accelerometer. It provides compass heading and determines surrounding influences simultaneously. In each axis, an electrical current is passed through a microscopic coil. The Earth's magnetic field induces a deflection due to the Lorentz force [24], so the x-, y- and z-component of the magnetic field can be measured.

Gyroscope

The gyroscope is typically a spinning wheel which conserves its orientation in an inertial frame of reference. As it is for the accelerometer and magnetometer, the gyroscope is based on a MEMS. It uses a vibrating structure for each axis to determine the orientation of the drone. A rotation perpendicular to the vibration direction causes a deflection due to the Coriolis effect which is dependent of the angular speed [25].

3.1.2 Barometer

The barometer determines the altitude of the aircraft. From the rate of change, the climb and sink rate can also be computed. The barometric MEMS sensor, relying on the piezoresistive effect, is measuring the atmospheric pressure by using resistors, whose resistance changes under stress. The resistors are embedded in a membrane which deforms under the influence of the exposed pressure [26]. With the barometric formula, which describes the dependency of the atmospheric pressure from the altitude, the altitude can be calculated.

3.1.3 Global Positioning System

A GPS module is another essential sensor. Using signals transmitted by satellites, it determines the position of the drone. From the rate of the position change, the velocity in three dimensions can be computed.

A satellite emits a signal with speed of light c which holds the point in time of the emission. A receiver compares the time stamp with its clock to compute the distance $s = c \cdot \Delta t$ to the satellite from the time difference Δt. By knowing the position of the satellite, the position of the receiver is fixed on a sphere. Signals from at least three satellites are needed to identify the location of the receiver[5]. Since the signal is sent with the speed of light, a precise measurement of time is crucial for an accurate definition of the position. Therefore, satellites have an atomic clock. However, the GPS receiver cannot be equipped with an atomic clock due to lack of space and reasons of cost. To overcome the inaccuracy of the inner clock of the receiver, a fourth satellite is added. Usually the emitted signals are disturbed. By using the signals from more than four satellite, the resulting error can be minimized [27]. To get a good estimate of the position for a stable flight, the GPS sensor needs to process the signals from more than eight satellites.

3.1.4 Airspeed Sensor

The digital airspeed sensor is an optional hardware to improve the flight stability especially in windy conditions. It consists of a Pitot-static tube which is connected via rubber tubing with a digital measurement unit. The Pitot tube is aligned along the flight direction. Exactly one of the airflow lines hits the tube perpendicularly as shown in Fig. 3.1. This point is called *stagnation point* and the flow velocity at this point is 0. As a result the pressure at the stagnation point is maximal and equals the total pressure p_0. The pressure at the holes on the side of the tube equals the static pressure p. According to Bernoulli's equation (2.1) the dynamic pressure $\frac{\varrho}{2} v_a^2$ is the difference of both. Knowing

[5] The intersection of three spheres results in two intersection points. One of them is far in outer space and can be excluded as the position of the GPS receiver.

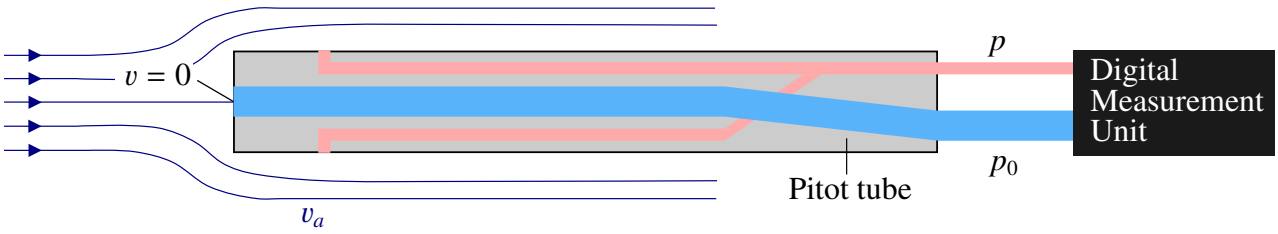


Figure 3.1: Digital airspeed sensor consisting of a Pitot tube and a digital measurement unit.

the density of air, the apparent airspeed can be calculated.

$$v_a = \sqrt{\frac{2}{\varrho}(p_0 - p)} \tag{3.1}$$

3.2 Optimal Control

An autonomously flying drone will not follow exactly a given path. Control inputs cannot be perfectly adjusted or unpredictable events can change the state of the drone significantly. This is why a good knowledge of the state as well as an optimal control algorithm are important to render possible a stable flight. Many sensors (as described in Sec. 3.1) are employed to measure the state of the drone like position, velocity, attitude and so on. The output however is disturbed due to statistical noise, and an accurate information on the state cannot be ensured.

The Kalman filter is an iterative algorithm that provides a precise estimate of the true state from noisy measurements. It assumes a discrete-time process in which the state evolves according to the linear stochastic state equation [28]

$$\vec{x}_k = \mathbf{F}_k \vec{x}_{k-1} + \mathbf{B}_k \vec{u}_k + \vec{w}_k. \tag{3.2}$$

The state transition model $\mathbf{F}_k$ describes the underlying dynamic of the state $\vec{x}_k$. The control vector $\vec{u}_k$ contains variables which influences the state, e.g. gravity and control surfaces of the wind drone. The control input model $\mathbf{B}_k$ specifies the forces and moments that result from the control vector and how they affect the state of the drone. Perturbations that influence the system are captured in the process noise vector $\vec{w}_k$. The measurement process $\vec{z}_k$[6] is modelled by

$$\vec{z}_k = \mathbf{H}_k \vec{x}_k + \vec{v}_k. \tag{3.3}$$

The observation model $\mathbf{H}_k$ relates the state space to the observed space, and $\vec{v}_k$ is a random variable describing measurement noise.

[6] The dimension of $\vec{x}_k$ and $\vec{z}_k$ can be different.

The measurement noise $\vec{v}_k$ as well as the process noise $\vec{w}_k$ are assumed to be given by normal distributions

$$\vec{v}_k \sim \mathcal{N}(0, \mathbf{R}_k), \tag{3.4}$$
$$\vec{w}_k \sim \mathcal{N}(0, \mathbf{Q}_k) \tag{3.5}$$

with vanishing means, the measurement noise covariance matrix $\mathbf{R}_k$, and the process noise covariance matrix $\mathbf{Q}_k$, respectively.

By knowing the covariance between different states and their errors, the Kalman filter is able to estimate states other than the one being measured, e.g. position measurements of the GPS sensor are able to estimate position, velocity, angles and gyro bias [29]. The Kalman filter is recursive and uses the previous calculated state $\hat{\vec{x}}_{k-1|k-1}$ and the estimate covariance matrix $\mathbf{P}_{k-1|k-1}$, which is a measure of the accuracy of the state estimation, to estimate the true state of the aircraft.

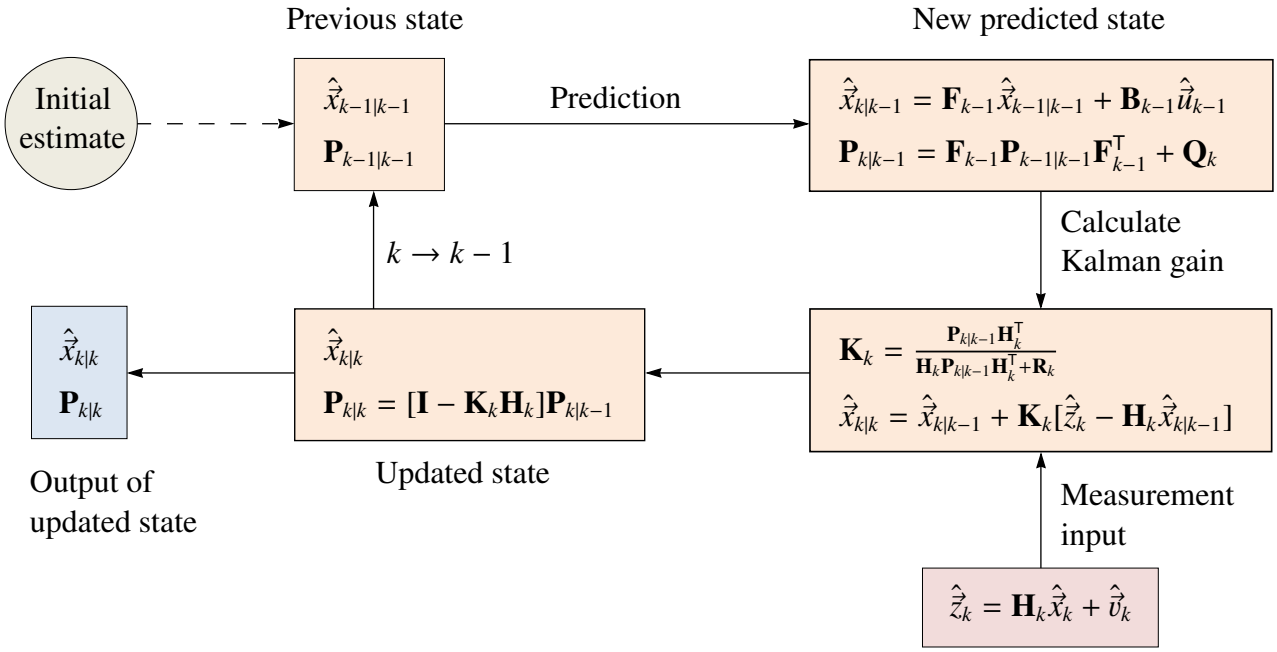


Figure 3.2: Flowchart of the Kalman filter [30].

The Kalman filter starts with an initial estimate of the state vector $\hat{\vec{x}}$ and the estimate covariance matrix $\mathbf{P}$. Afterwards the algorithm estimates the true state of the aircraft by iteration illustrated in Fig. 3.2.

Prediction An *a priori* prediction of the state $\hat{\vec{x}}_{k|k-1}$ and the estimate covariance $\mathbf{P}_{k|k-1}$ at time step k is made according to

$$\hat{\vec{x}}_{k|k-1} = \mathbf{F}_{k-1}\hat{\vec{x}}_{k-1|k-1} + \mathbf{B}_{k-1}\vec{u}_{k-1} \tag{3.6}$$
$$\mathbf{P}_{k|k-1} = \mathbf{F}_{k-1}\mathbf{P}_{k-1|k-1}\mathbf{F}_{k-1}^{\mathsf{T}} + \mathbf{Q}_k \tag{3.7}$$

in time step $k-1$.

Kalman Gain The Kalman gain

$$\mathbf{K}_k = \frac{\mathbf{P}_{k|k-1}\mathbf{H}_k^\mathsf{T}}{\mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^\mathsf{T} + \mathbf{R}_k} \tag{3.8}$$

is essentially given as the ratio of the uncertainty of the estimate and the sum of the the uncertainties of the estimate and the measurement[7]. It is a relative weight to determine how the measurement and prediction are combined. The Kalman gain goes to 1 for a precise measurement, whereas the Kalman gain goes to 0 for a noisy measurement.

Measurement Input A measurement according to Eq. (3.3) is performed and compared to the prediction in order to improve the estimate of the true state.

Update The measurement $\vec{z}_k$ is used to update the predicted state. The *a posteriori* estimate

$$\hat{\vec{x}}_{k|k} = \hat{\vec{x}}_{k|k-1} + \mathbf{K}_k[\vec{z}_k - \mathbf{H}_k\hat{\vec{x}}_{k|k-1}] \tag{3.9}$$

lies between the *a priori* estimate $\hat{\vec{x}}_{k|k-1}$ and the measurement $\vec{z}_k$. The measurement residual $\vec{z}_k - \mathbf{H}_k\hat{\vec{x}}_{k|k-1}$ is called innovation. A precise measurement gives a higher weight to the measurements, whereas very noisy sensor data gives a higher weight to the predicted state.

Furthermore the *a priori* estimate covariance is adjusted according to the following equation

$$\mathbf{P}_{k|k} = [\mathbf{I} - \mathbf{K}_k\mathbf{H}_k]\mathbf{P}_{k|k-1}. \tag{3.10}$$

Next Iteration The updated estimate is then used for controlling and navigation of the drone. The current state becomes the previous state and the algorithm starts the next iteration.

The Kalman filter gives an accurate estimate of the true state. It can also be used as an simple form of optimal control. To steer the drone, the control vector $\vec{u}_k$ is set by solving the following optimal control problem for every sampling time [31]

$$\underset{\vec{u}_k}{\text{minimize}} \qquad \left\|\vec{x}_{T_C} - \vec{x^r}_{T_C}\right\|_{P_C} + \int_0^{T_C} \left(\left\|\vec{x}_t - \vec{x^r}_t\right\|_{Q_C} + \left\|\vec{u}_t - \vec{u^r}_t\right\|_{R_C}\right) \mathrm{d}t \tag{3.11}$$

in which P_C, Q_C and R_C are positive symmetric weighting matrices, T_C is the prediction horizon, and $\vec{x^r}_t$, $\vec{u^r}_t$ are given references.

[7] The correct syntax of the Kalman gain is: $\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}_k^\mathsf{T}[\mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^\mathsf{T} + \mathbf{R}_k]^{-1}$. The denominator is the inverse of the resulting matrix.

3.3 Reference Frames

Vectors, e.g. forces or velocities, depend on the coordinate system in which they are defined. A coordinate system can be transformed in any other coordinate system by rotation. The rotation matrices around the axes are shown in Eq. (3.12).

$$
\begin{aligned}
M_x &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos x & \sin x \\ 0 & -\sin x & \cos x \end{pmatrix} \\
M_y &= \begin{pmatrix} \cos y & 0 & -\sin y \\ 0 & 1 & 0 \\ \sin y & 0 & \cos y \end{pmatrix} \\
M_z &= \begin{pmatrix} \cos z & \sin z & 0 \\ -\sin z & \cos z & 0 \\ 0 & 0 & 1 \end{pmatrix}
\end{aligned} \tag{3.12}
$$

The index stands for the axis around which the coordinate system is rotated. The total rotation matrix is obtained by multiplying the matrices from the left side. For example a rotation around the z-axis by z, followed by a rotation around the y-axis by y, followed by a rotation around the x-axis by x would lead to the rotation matrix

$$
M_{\text{total}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos x & \sin x \\ 0 & -\sin x & \cos x \end{pmatrix} \cdot \begin{pmatrix} \cos y & 0 & -\sin y \\ 0 & 1 & 0 \\ \sin y & 0 & \cos y \end{pmatrix} \cdot \begin{pmatrix} \cos z & \sin z & 0 \\ -\sin z & \cos z & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{3.13}
$$

This is called a *passive transformation* since the vector is left unchanged and the coordinate system is rotating. An *active transformation* rotates the vector and leaves the coordinate system as it is. The rotation happens according to the right-hand rule for the active transformation as well as for the passive transformation. The matrix for the active transformation is obtained by taking the inverse of the matrix of the passive transformation. For rotation matrices the inverse is the same as the transposed matrix.

$$
M^{-1} = M^{\mathsf{T}} \tag{3.14}
$$

An important coordinate system is the earth-fixed coordinate system. The x_e-axis shows to the north and the z_e-axis shows in the same direction as the gravity, so to the earth center. The y_e-axis is perpendicular to both, such that the three axes form a right-handed coordinate system. Thus, the y_e-axis shows to the east. This frame is also called north-east-down (NED) coordinate system [10]. The index e states that axes as well as vectors are described in the earth-fixed coordinate system. For example, the gravity vector in the earth-fixed frame is

$$
\vec{g}_e = \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix}_e. \tag{3.15}
$$

The drone-fixed coordinate system is shown in Fig. 3.3. The x_d points forward from the center of gravity to the nose, the y_d points starboard and the z_d respectively to comply the right-handed coordinate system.

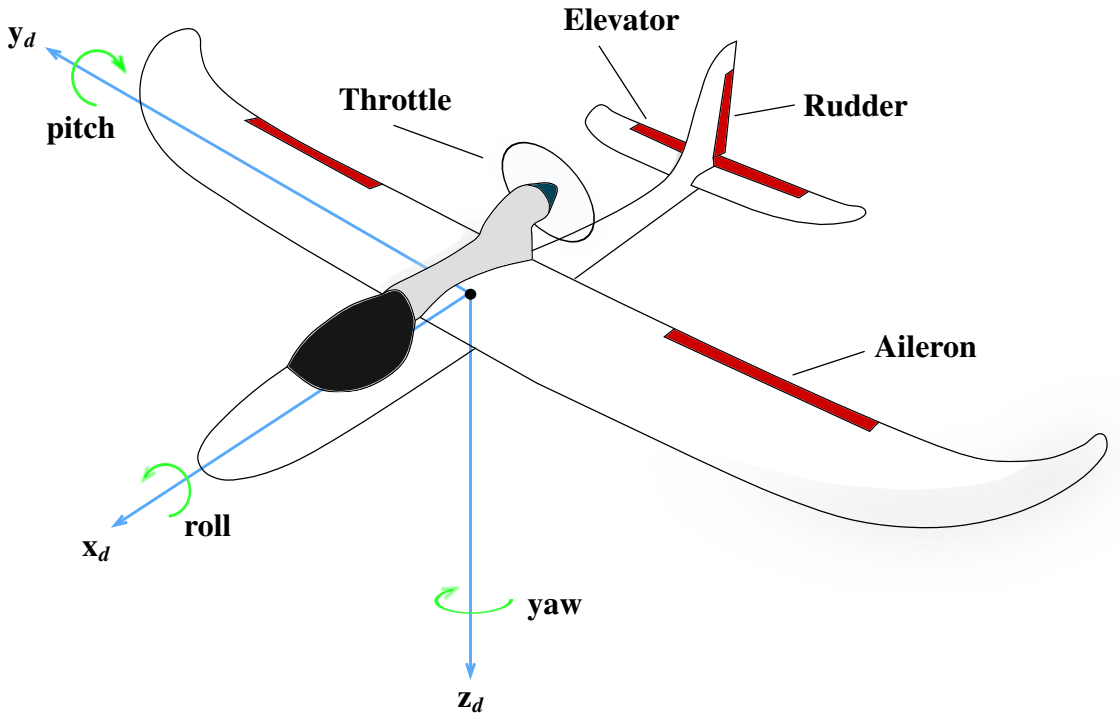


Figure 3.3: Illustration of drone-fixed frame and nomenclature of control surfaces.

The roll-pitch-yaw angle describe the orientation of the drone and accordingly of the drone-fixed coordinate system. They are Euler angles and define the rotation around the axes. The *yaw angle* Ψ describes the rotation around the z_d-axis and can be controlled with the rudder. The *pitch angle* Θ describes the rotation around the y_d-axis and is controlled by the elevator. The *roll angle* Φ defines the rotation around the x_d-axis and is controlled by the ailerons. The total rotation matrix to transform the earth-fixed to the drone-fixed coordinate system is, according to Eq. (3.13),

$$\begin{aligned} M_{de} &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\Phi & \sin\Phi \\ 0 & -\sin\Phi & \cos\Phi \end{pmatrix} \cdot \begin{pmatrix} \cos\Theta & 0 & -\sin\Theta \\ 0 & 1 & 0 \\ \sin\Theta & 0 & \cos\Theta \end{pmatrix} \cdot \begin{pmatrix} \cos\Psi & \sin\Psi & 0 \\ -\sin\Psi & \cos\Psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} \cos\Theta\cos\Psi & \cos\Theta\sin\Psi & -\sin\Theta \\ \sin\Phi\sin\Theta\cos\Psi - \cos\Phi\sin\Psi & \sin\Phi\sin\Theta\sin\Psi + \cos\Phi\cos\Psi & \sin\Phi\cos\Theta \\ \cos\Phi\sin\Theta\cos\Psi + \sin\Phi\sin\Psi & \cos\Phi\sin\Theta\sin\Psi - \sin\Phi\cos\Psi & \cos\Phi\cos\Theta \end{pmatrix}. \end{aligned} \tag{3.16}$$

The index *de* state that the rotation matrix transforms the earth-fixed coordinate system to the drone-fixed coordinate system. The gravity vector in the drone-fixed coordinate system is

$$\vec{g}_d = M_{de} \cdot \vec{g}_e = g \cdot \begin{pmatrix} -\sin\Theta \\ \sin\Phi\cos\Theta \\ \cos\Phi\cos\Theta \end{pmatrix}_d . \tag{3.17}$$

The rotation matrix that transforms the drone-fixed to the earth-fixed coordinate system is the inverse

of M_{de} and according to Eq. (3.14) the transposed matrix

$$M_{ed} = M_{de}^{-1} = M_{de}^{\mathsf{T}} \tag{3.18}$$

An also used coordinate system is the flight-path coordinate system (index k). The x_k-axis points in the same direction as the trajectory velocity vector $\vec{v}_k$. The y_k-axis lies in the earth horizontal plane and the z_k-axis is arranged such that the three axis form a right-handed coordinate system. To transform the earth-fixed coordinate system to the flight path coordinate system two angles are needed, the *flight-path azimuth angle* χ and the *angle of climb* γ.

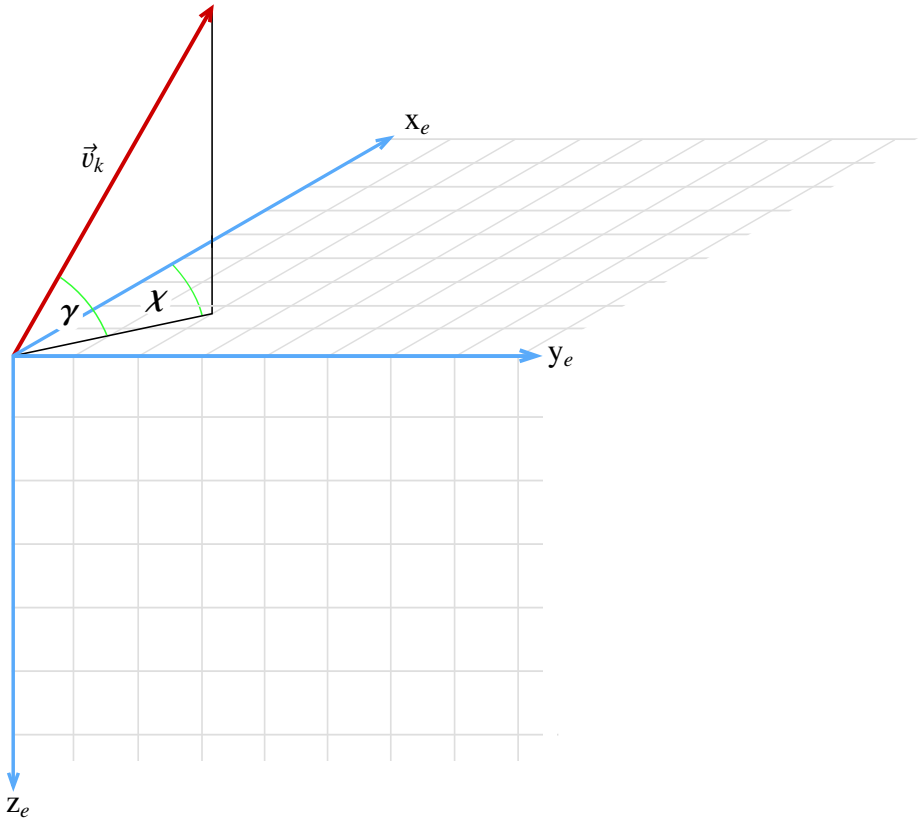


Figure 3.4: Flight-path azimuth angle and angle of climb.

Both angles χ and γ can be obtained from the coordinates of the trajectory velocity vector in the earth-fixed coordinate system $\vec{v}_{k,e}$.

$$\vec{v}_{k,e} = \begin{pmatrix} u \\ v \\ w \end{pmatrix}_e = \begin{pmatrix} u_e \\ v_e \\ w_e \end{pmatrix} \tag{3.19}$$

$$\chi = \arctan \frac{v_e}{u_e} \tag{3.20}$$

$$\gamma = \arcsin \frac{-w_e}{|v_k|} = \arcsin \frac{-w_e}{\sqrt{u_e^2 + v_e^2 + w_e^2}} \tag{3.21}$$

As pictured in Fig. 3.4, the flight path coordinate system is obtained by rotating the earth-fixed coordinate system by χ around the z_e-axis and afterwards by γ around the rotated y_e-axis. The total

rotation matrix is

$$M_{ke} = \begin{pmatrix} \cos\gamma & 0 & -\sin\gamma \\ 0 & 1 & 0 \\ \sin\gamma & 0 & \cos\gamma \end{pmatrix} \cdot \begin{pmatrix} \cos\chi & \sin\chi & 0 \\ -\sin\chi & \cos\chi & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \cos\gamma\cos\chi & \cos\gamma\sin\chi & -\sin\gamma \\ -\sin\chi & \cos\chi & 0 \\ \sin\gamma\cos\chi & \sin\gamma\sin chi & \cos\gamma \end{pmatrix}. \tag{3.22}$$

The last coordinate system which is used in this thesis is the figure-8 pattern related coordinate system (index p). This coordinate system is explained in Sec. 3.4 where the figure-8 pattern is introduced.

3.4 Figure-8 Pattern

Since the drone is leashed by a tether, its flight path lies on a hemisphere. This chapter derives the geometry of the figure-8 pattern. The eight is divided into four segments, two crossing paths and two turning paths. Figure 3.5 shows a flat two dimensional figure-8 pattern to introduce some of the variables. The crossing paths are straight lines and the turning paths are arcs of a circle. The angle at center ζ defines the circular arc of the turning paths by

$$\zeta = 180° + 2\delta \tag{3.23}$$

with

$$\delta = \arccos\frac{d_c}{2r}. \tag{3.24}$$

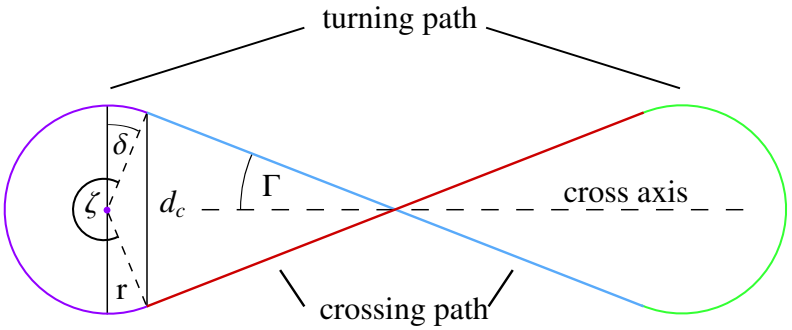


Figure 3.5: Two dimensional figure-8 pattern consisting of two crossing paths and two turning paths. The angle between a crossing path and the cross axis is the crossing angle Γ, the radius of the circular arc of the turning paths is r and the distance between the endpoints of both crossing paths is d_c.

Equation (3.23) holds only if the circular arc is greater than a semicircle, but as demonstrated in B.2, this always holds for the turning path of a figure-8 pattern lying on a hemisphere. Bringing this pattern on a hemisphere, the idea is to make the crossing paths *geodesic lines* and the turning paths small circle segments which connect the end points of the crossing paths.

Like the turning path, the crossing path is an arc of a circle, which can be created as an intersection of the hemisphere with a plane. If the plane contains the origin of the hemisphere, a great circle is obtained, otherwise a small circle[8] is obtained. The arc of a great circle is a geodesic line, so the crossing paths are made of great circles and the turning paths are made of small circles. Hence, four planes are needed to create the four path segments of the figure-8 pattern.

[8] Assuming the intersection is not empty. For example, longitudes are great circles, latitudes, except of the equator, are small circles.

The plane is described parametrically by the form

$$\vec{p} = d\vec{p}_0 + s\vec{u} + t\vec{v}, \tag{3.25}$$

where $\vec{p}$ points to all points on the plane, $d\vec{p}_0$ is the position vector which points to an arbitrary point on the plane and $\vec{u}$ and $\vec{v}$ are two linearly independent vectors spanning the plane. The parameters d, s and t are real numbers, in which d is constant and indicates the distance of the plane to the origin as the vectors $\vec{p}_0$, $\vec{u}$ and $\vec{v}$ are defined to be normalised and to be mutually perpendicular.

In the control algorithm (see Sec. 4.1) the vectors are also described in the coordinate system related to the plane, so a rotation matrix M_{pe} to transform the earth-fixed coordinate system to the plane related coordinate system has to be defined. Any orientation of the plane is defined by two angles. A plane which lies initially in the earth horizontal plane, i.e. in the x_e-y_e-plane, is rotated first around z_e by ψ and then around the rotated y_e by ϑ[9]. Afterwards the plane is shifted parallel by the distance d. The plane related coordinate system is defined such that the rotated plane forms the x_p-y_p-plane. The parametric representation of the plane is then

$$\vec{p}_p = -d\begin{pmatrix}0\\0\\1\end{pmatrix}_p + s\begin{pmatrix}1\\0\\0\end{pmatrix}_p + t\begin{pmatrix}0\\1\\0\end{pmatrix}_p, \tag{3.26}$$

and the rotation matrix is given by

$$\begin{aligned} M'_{pe} &= \begin{pmatrix}\cos\vartheta & 0 & -\sin\vartheta\\ 0 & 1 & 0\\ \sin\vartheta & 0 & \cos\vartheta\end{pmatrix}\cdot\begin{pmatrix}\cos\psi & \sin\psi & 0\\ -\sin\psi & \cos\psi & 0\\ 0 & 0 & 1\end{pmatrix}\\ &= \begin{pmatrix}\cos\vartheta\cos\psi & \cos\vartheta\sin\psi & -\sin\vartheta\\ -\sin\psi & \cos\psi & 0\\ \sin\vartheta\cos\psi & \sin\vartheta\sin\psi & \cos\vartheta\end{pmatrix}. \end{aligned} \tag{3.27}$$

The parametric representation of $\vec{p}_p$ described in the earth-fixed coordinate system is then

$$\vec{p}_e = M'_{ep}\cdot\vec{p}_p = -d\begin{pmatrix}\sin\vartheta\cos\psi\\ \sin\vartheta\sin\psi\\ \cos\vartheta\end{pmatrix}_e + s\begin{pmatrix}\cos\vartheta\cos\psi\\ \cos\vartheta\sin\psi\\ -\sin\vartheta\end{pmatrix}_e + t\begin{pmatrix}-\sin\psi\\ \cos\psi\\ 0\end{pmatrix}_e. \tag{3.28}$$

The origin of the plane related coordinate system is placed at the position vector $d\vec{p}_0$. In doing so the x_p-axis points always in the direction of the highest point of the intersection circle, if $0 < \vartheta \leq 90°$ and the origin coincides with the center of the intersection circle.

This is illustrated in Fig. 3.6. The origin of the earth-fixed coordinate system coincides with the center of the sphere. It is the anchor point of the tether. The sphere radius R is given by the tether length. The distance between the circle center and the sphere center is d. The radius r of the intersection circle can be calculated with Pythagoras' formula

$$r = \sqrt{R^2 - d^2}. \tag{3.29}$$

[9] $\psi \in [-180°, 180°]$ and $\vartheta \in [0°, 90°]$. This is sufficient to receive any orientation.

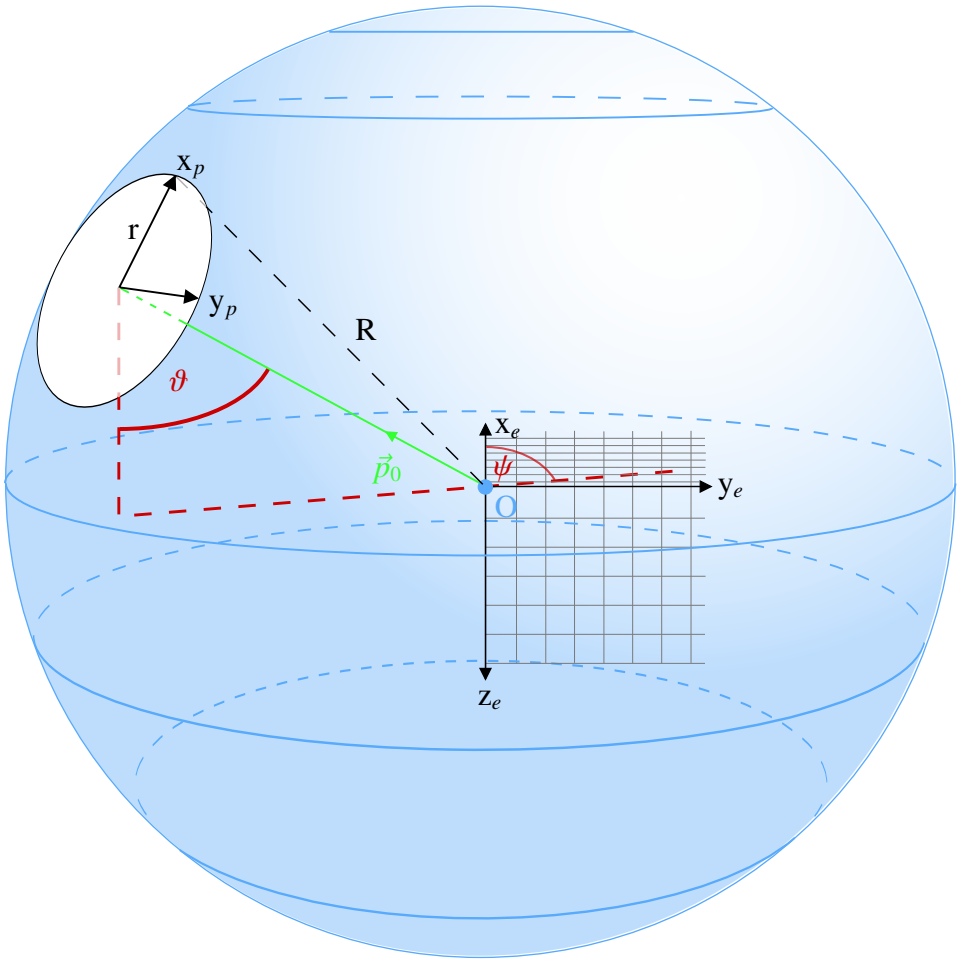


Figure 3.6: Sphere.

As mentioned above, the figure-8 pattern on the sphere is composed of the arcs of two great circle which produce the crossing paths and two small circles which generate the turning paths. The shape of the figure-8 pattern can be completely described by the arc angle A and crossing angle Γ of the geodesic lines. A determines the length of the geodesic line and Γ the width of the figure-8 pattern. Aligning the geodesic lines with these two angles, there is only one possible way to arrange the turning path such that the connection is smooth.

To produces a figure-8 pattern that lies initially on top of the sphere with its cross axis showing from west to east, i.e. being parallel to the y_e-axis, the rotation angles ψ and ϑ of the planes which create the geodesic lines have to be

$$\psi_1 = \Gamma, \qquad \vartheta_1 = 90^\circ \tag{3.30}$$

$$\psi_2 = -\Gamma, \qquad \vartheta_2 = 90^\circ. \tag{3.31}$$

For a given arc angle A the rotation angles ψ and ϑ of the planes, which produce the turning paths, as well as the distance of the plane from the origin can be calculated. In that case, the normal vector of the plane, i.e. $-\vec{p}_0$, lies in the y_e-z_e-plane. The angle η is the angle between this normal vector and

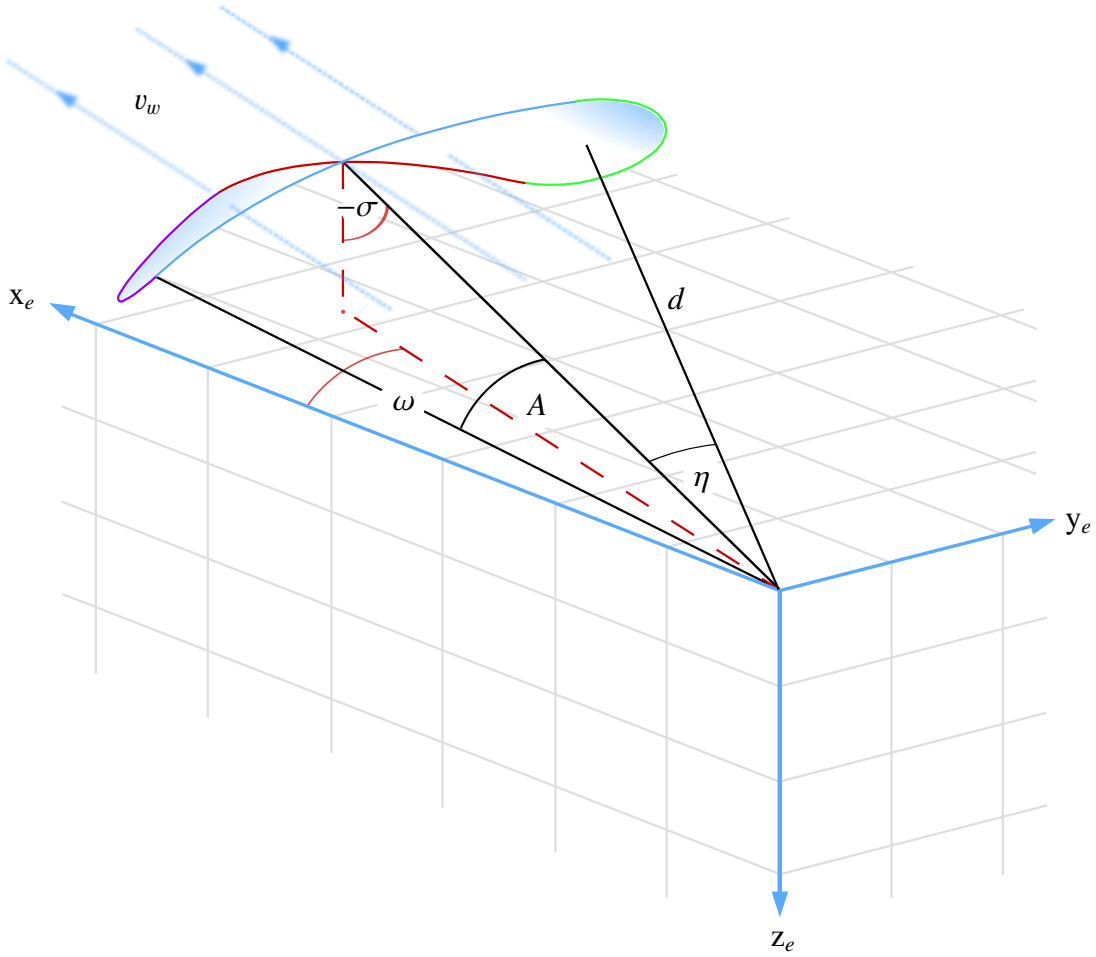


Figure 3.7: Illustration of the angles of a figure-8 pattern lying in wind direction on a hemisphere.

the connection line between the origin of the coordinate system and the crossing point of the two geodesic lines, which in this case is the $-z_e$-axis. The angle η can be calculated with the following idea:

Let $\vec{t}_{1,e}$ and $\vec{t}_{2,e}$ be the tangential vectors described in the earth-fixed coordinate system at the endpoint of each crossing path which will be connected by the turning path. By rotating the two geodesic lines by $-\eta$ around the x_e-axis the two tangential vectors are parallel to the earth horizontal plane, i.e. the z-component is 0. The calculation is done in B.3. The result is

$$\eta = \arctan\left(\frac{\tan A}{\cos \Gamma}\right). \tag{3.32}$$

To create a valid figure-8 pattern the range of A and Γ are restricted to

$$0^\circ < A < 90^\circ \tag{3.33}$$
$$0^\circ < \Gamma < 45^\circ. \tag{3.34}$$

With η fixed, both rotation angles ψ and ϑ of the plane which generate the turning paths are also fixed

to

$$\psi_1 = 90°, \qquad \vartheta_1 = \eta \tag{3.35}$$
$$\psi_2 = -90°, \qquad \vartheta_2 = \eta. \tag{3.36}$$

The distance can be calculated analogously. By rotating the two geodesic lines by $-\eta$ around the x_e-axis the vector from the origin to an endpoint of a crossing path lies then in the x_e-z_e-plane. The z-component of that vector is the distance of the plane from the origin. The calculation is performed in B.4. The result is

$$d = R \cdot (\sin\eta \cos\Gamma \sin A + \cos\eta \cos A). \tag{3.37}$$

Now the circle radius can be calculated according to Eq. (3.29). The last factor to complete the figure-8 is the angle at center ζ of the small circle. According to Eq. (3.23) the angle at center is

$$\zeta \overset{(3.24)}{=} 180° + 2\arccos\frac{d_c}{2r} \overset{(B.5)}{=} 180° + 2\arccos\frac{R\sin A\sin\Gamma}{r}. \tag{3.38}$$

The calculation of d_c is performed in B.5.

Orientation of the Figure-8 Pattern

The created figure-8 pattern should be aligned perpendicularly to the wind direction, i.e. the wind vector and the vector to the crossing point both lie in the same plane perpendicular to the Earth's horizontal plane. This can be done by a transformation matrix M_{pe} which is a rotation of the figure-8 lying on top of the hemisphere by ω around the z_e-axis and then by a rotation by σ around the rotated y_e-axis, as pictured in Fig. 3.7. The naive multiplication

$$M_\sigma \cdot M_\omega \cdot M'_{pe} \tag{3.39}$$

with

$$M_\omega = \begin{pmatrix} \cos\omega & \sin\omega & 0 \\ -\sin\omega & \cos\omega & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad M_\sigma = \begin{pmatrix} \cos\sigma & 0 & -\sin\omega \\ 0 & 1 & 0 \\ \sin\omega & 0 & \cos\sigma \end{pmatrix} \tag{3.40}$$

would not result in a rotation around the z_e-axis and the y_e-axis but to a rotation around the rotated z_e-axis and z_e-axis due to M'_{pe}. Thus, the total transformation matrix M_{pe} has to be defined differently.

The plane related coordinate system defines the planes which create the crossing and turning paths by intersection with the hemisphere. Performing an active transformation of the three axes given in Eq. (3.28) of each plane would rotate the plane in the desired position. The passive transformation matrix M_{pe} is thus obtained by taking the inverse of this product.

$$M_{pe} = \left(M_\omega^{-1} \cdot M_\sigma^{-1} \cdot M_{pe}'^{-1}.\right)^{-1} \tag{3.41}$$

It is specified in B.6.

CHAPTER 4

Software

Airborne wind energy systems are actively stabilized by sophisticated control algorithms rather than by massive static structures as conventional wind turbines. As such, it needs software for autonomous flight. An extensive autopilot code is available from `ArduPilot` [29]. The structure of the autopilot code and the algorithms to fly a lying figure-8 pattern are described in Sec. 4.1. To test the autopilot under different environmental conditions, a flight dynamics model is used. This model is specified together with the modifications of the original source code in Sec. 4.2. A ground control station is used to track flight data and communicate via telemetry with the wind drone during flight as described in Sec. 4.3.

4.1 ArduPilot Project

ArduPilot [29] is an open source project providing an autopilot software to fixed-wing aircraft (ArduPlane), various copters (ArduCopter) and rovers (ArduRover). The project contains all necessary software parts of the autopilot. This includes drivers of the sensors, a communication protocol, navigation and control algorithms. The latest[10] source code for fixed-wing aircraft ArduPlane 3.5 serves as a basis for the wind drone.

ArduPilot supports several autopilot boards. "The original APM1 autopilot board was based around the Arduino development environment" [29], which explains the name of the project. In this thesis the Pixhawk board is used since it is the latest[11] supported autopilot board and very powerful. More information about the Pixhawk can be found in Sec. 5.1.

4.1.1 Plane Parameters

ArduPlane contains more than two hundred parameters to configure the aircraft, which can be changed via a ground control station. Most of them are used to tune the aircraft, e.g. gains of control algorithms. Other parameters save values from calibration processes (see Sec. 5.3), e.g. offsets of accelerometers or gyroscopes. Some parameters are used to enable optional sensors like the airspeed sensor. A full

[10] As of February 2016
[11] As of October 2015

parameter list can be found in [29] with a short description of each parameter. In this thesis two parameters are added which correspond to ω and σ to define the orientation of the figure-8 pattern.

4.1.2 Basic Structure

Figure 4.1 shows the basic structure of the ArduPilot project. Most of the work in this thesis was done on the flight code itself. The vehicle specific flight code contains the entry point of the program. Similar to the Arduino development environment, the underlying code structure is composed of two functions: **void setup()** and **void loop()**. Only at the program start void setup() is called to initialise variables and classes. Afterwards void loop(), which determines the actual program sequence, is executed consecutively. Classes of sensors, controllers, the Kalman filter and so on are provided via

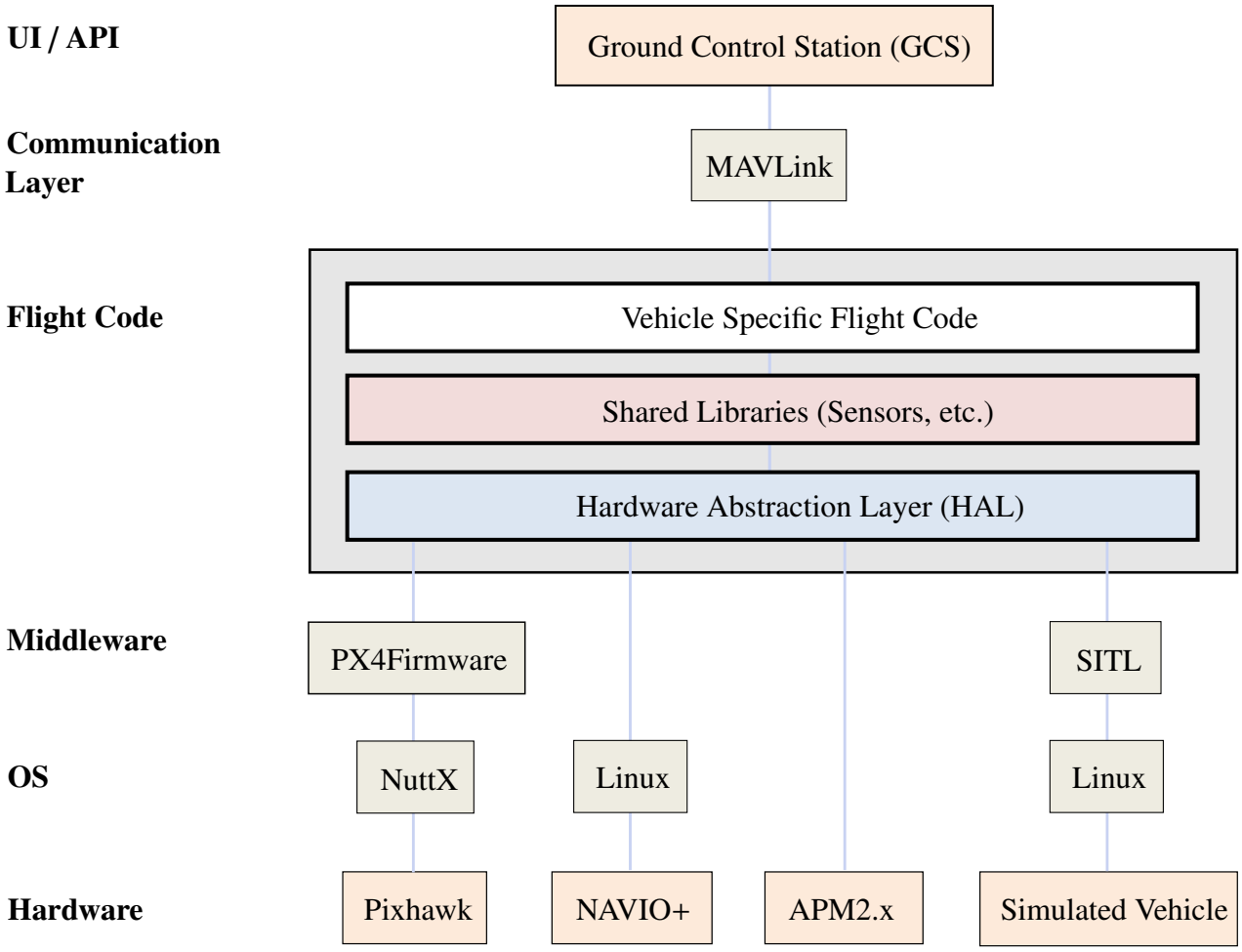


Figure 4.1: High level architecture [29].

shared libraries to the vehicle specific flight code. The hardware abstraction layer makes the flight code compatible to a variety of boards, e.g. Pixhawk, NAVIO+ or the APM board. The operating system on Pixhawk is NuttX [32] and the PX4Firmware provides driver for sensors. Using software in the loop (SITL), ArduPilot can also run on a simulated vehicle with no hardware. The micro air vehicle communication protocol, called MAVLink [33], is used for communication between the drone and a ground control station.

4.1.3 AP_Scheduler

The AP_Scheduler system is a set of about 50 tasks, which are processed one after the other. It is called by the void loop() function at 400 Hz. Some selected scheduler tasks (SCHED_TASK) are listed in Listing 4.1, which illustrate a basic guidance loop of the aircraft. Other tasks are used to read the radio control input of a transmitter, log flight data, communicate with the ground control station and so on.

```
const AP_Scheduler::Task Plane::scheduler_tasks[] = {
                                // Units:    Hz      us
    SCHED_TASK(ahrs_update,              400,    400),
    SCHED_TASK(stabilize,                400,    100),
    SCHED_TASK(set_servos,               400,    100),
    SCHED_TASK(navigate,                  10,    150),
    SCHED_TASK(read_airspeed,             10,    100),
    SCHED_TASK(update_alt,                10,    200),
    ...
};
```

Listing 4.1: Selected scheduler tasks of the AP_Scheduler

Each scheduler task contains the name of the function, the rate it is called in Hz and the maximum time the function is expected to take in µs.

The first function in AP_Scheduler is **void ahrs_update()**[12], which is called at the maximum frequency of 400 Hz and should not take more than 400 µs. It runs the Kalman filter algorithm to estimate the state of the aircraft.

void stabilize() calculates the demanded deflection of the control surfaces to steer the aircraft to the desired attitude. There is a separate control algorithm for the ailerons, the rudder and the elevator. The roll controller calculates the demanded aileron deflection, the pitch controller calculates the demanded elevator deflection and the yaw controller calculates the demanded rudder deflection.

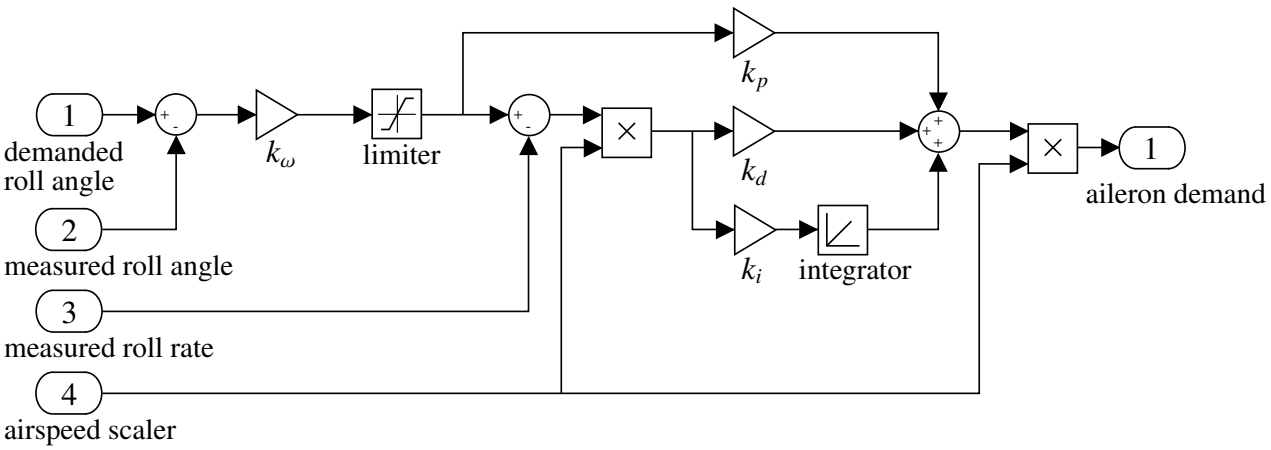


Figure 4.2: Block diagram of the roll controller [29].

[12] AHRS stands for Attitude Heading Reference System.

As an example, Fig. 4.2 shows the block diagram of the roll controller, a proportional-integral-derivative controller (PID controller), receiving data from four inputs. The *demanded roll angle* is determined by the navigation controller, which is called by **void navigate()**.

The measured roll angle and rate is estimated by the extended Kalman filter. The airspeed scaler scales the deflection of the control surfaces. It decreases for higher airspeeds, so the control surfaces are deflected more at low speeds and less in high speeds. The angle error (demanded roll angle - measured roll angle) multiplied by $k_\omega = \frac{1}{t_r}$ yields the demanded roll rate, as the roll time constant t_r controls the time in seconds from demanded to achieved roll angle. A small roll time constant leads to a faster response of the drone. It is set by the plane parameters as well as the proportional gain k_p, the damping gain k_d and the integrator gain k_i in order to tune the behaviour of the aircraft.

The computed aileron demand as well as the elevator and rudder demand, calculated by the pitch and yaw controller, respectively, are passed to **void set_servos()**, which actuates the servos to align the steering surfaces.

void update_alt() determines a demanded pitch angle and throttle output using a speed height controller.

4.1.4 Extended Kalman Filter

ArduPilot employs the extended Kalman filter to estimate 24 state variables:

- Attitude (Quaternions)
- Velocity (North, East, Down)
- Position (North, East, Down)
- Gyro bias offsets (Cartesian)
- Gyro scale factors (Cartesian)
- Z acceleration bias
- Earth magnetic field (North, East, Down)
- Body magnetic field (Cartesian)
- Wind velocity (North, East)

The extended Kalman filter is an extension to the normal Kalman filter where the state transition and observation model are not linear but are described by the non-linear stochastic difference equations [28]

$$\vec{x}_k = f(\vec{x}_{k-1}, \vec{u}_{k-1}, \vec{w}_{k-1}) \tag{4.1}$$

$$\vec{z}_k = h(\vec{x}_k, \vec{v}_k). \tag{4.2}$$

The principle of the algorithm described in Sec. 3.2 remains the same. First, the state transition is predicted according Eq. (4.1). Subsequently, the state estimate is updated by combining the prediction

with the current measurement.

To calculate the Kalman gain, which combines the predicted state with the current measurement input, the measurement noise covariance matrix R_k has to be known. Plane parameters for the Kalman filter define the measurement noise covariance matrix.

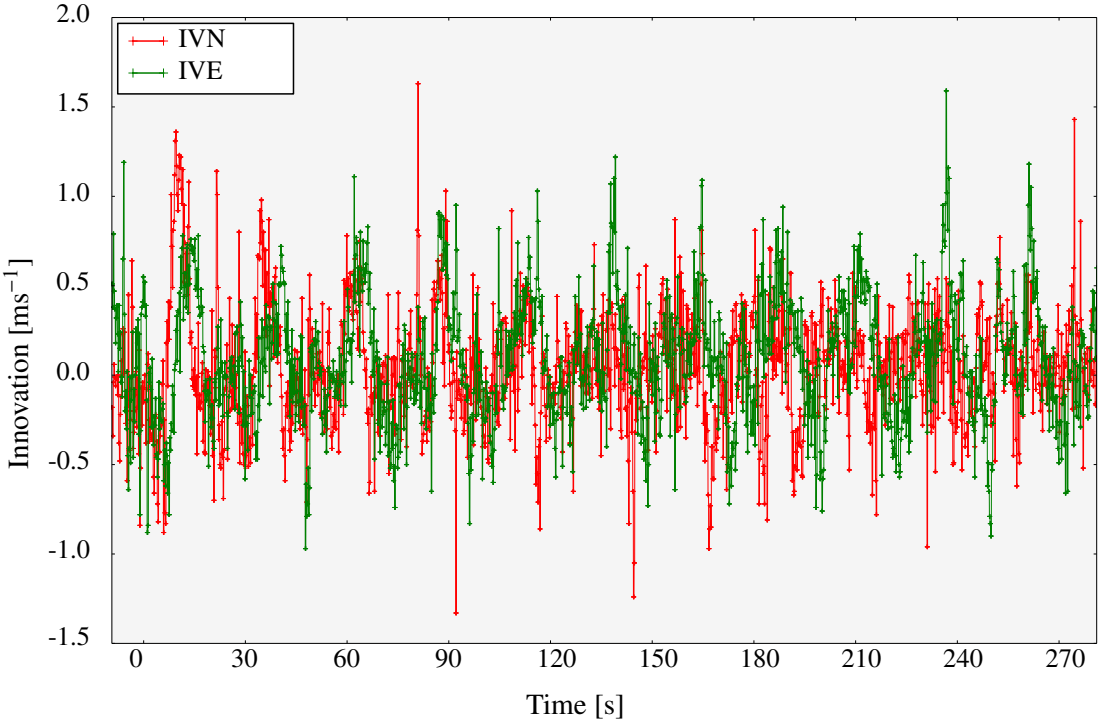


Figure 4.3: Innovation of north (IVN) and east (IVE) GPS velocity.

Figure 4.3 plots the innovations of North and East GPS velocity, which is the difference between the predicted and measured state. The data was taken while the drone was flying a circular path. Mean and standard deviation of IVN and IVE are listed in Tab. 4.1.

	Mean [m s^{-1}]	Standard deviation [m s^{-1}]
IVN	0.04	0.34
IVE	0.09	0.35

Table 4.1: Mean and standard deviation of IVN and IVE.

A good starting value for the lateral velocity noise[13] would be 0.35 m s^{-1}. For a better performance of the Kalman filter, the uncertainties for every sensor measurement can be set in the plane parameters by analysing logged flight data.

During flight the Kalman filter then adjusts R_k every time step by evaluating sensor data.

[13] There is only one plane parameter that captures the uncertainty of GPS velocity in both north and east direction.

4.1.5 Navigation Controller

The navigation controller is used for lateral navigation. It basically calculates the roll angle Φ that is needed to fly a curve with a certain radius at a constant altitude.

The lift vector of a plane is perpendicular to its wing area. For a roll angle Φ the lift vector is tilted by the same angle and can then be divided into a vertical and horizontal lift component. By controlling pitch and throttle, which is done in the speed height controller, the vertical lift equals the weight.

$$L \cos \Phi = mg. \tag{4.3}$$

In doing so the aircraft stays at the actual altitude. The horizontal component of the lift vector acts as a centripetal force

$$L \sin \Phi = \frac{mv^2}{r}, \tag{4.4}$$

where m is the mass of the drone, g is the gravitational field strength, v is the velocity of the aircraft with respect to the ground and r is the radius of the circular path. The roll angle Φ results from the quotient of Eq. (4.4) over Eq. (4.3)

$$\Phi = \arctan \frac{v^2}{gr} = \arctan \frac{a_{\text{lat}}}{g}, \tag{4.5}$$

with a_{lat} the lateral acceleration.

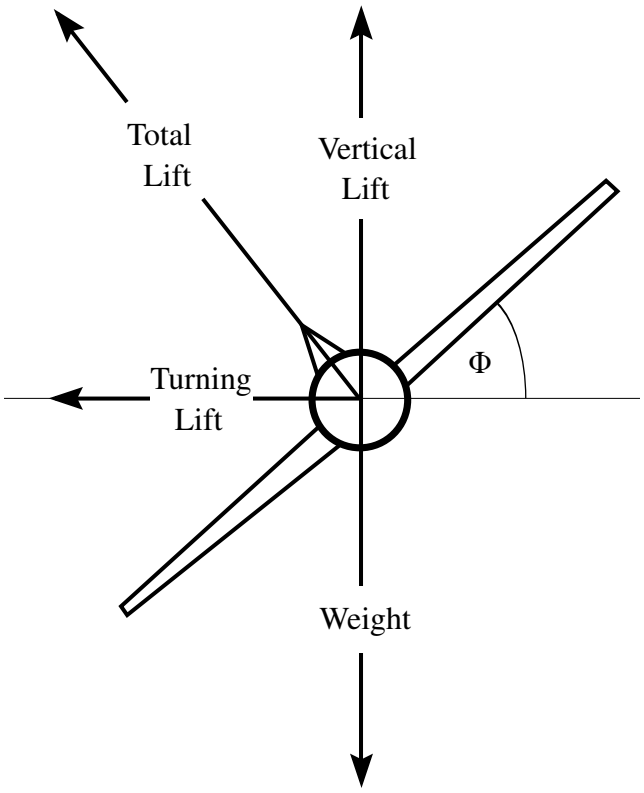


Figure 4.4: Aircraft in level turn [34]. Vertical lift equals weight.

To account for the pitch angle Θ, Eq. (4.5) is modified to (see B.7)

$$\Phi = \arctan \frac{a_{\text{lat}} \cos \Theta}{g}. \tag{4.6}$$

Due to the *directional stability* the aircraft orients towards the trajectory.

L1 Controller

In the code, navigation controllers are specified via an abstract parent class that declares virtual functions to assure that every navigation controller includes required functions, e.g. navigation to a waypoint. In the concrete controllers, the virtual functions are then defined. This facilitates the programming of a new navigation controller. For operation of the autopilot code a navigation controller is selected.

The navigation controller is called at 10 Hz by void navigate() in AP_Scheduler. In version ArduPlane 3.5 there is only one navigation controller available called *L1 controller*. The L1 controller includes two different navigation modes. One is the navigation to a waypoint, i.e. flying on a straight path to a given location, and the other is the loiter mode, i.e. circling around a location at a given radius and constant altitude.

For waypoint navigation a non-linear guidance logic according to the L1 control law is performed [35]. The principle of the L1 algorithm is illustrated in Fig. 4.5 for a straight path, the algorithm however can be used for any horizontal path. L_1 is a constant length and serves as a tuning parameter, a smaller L_1

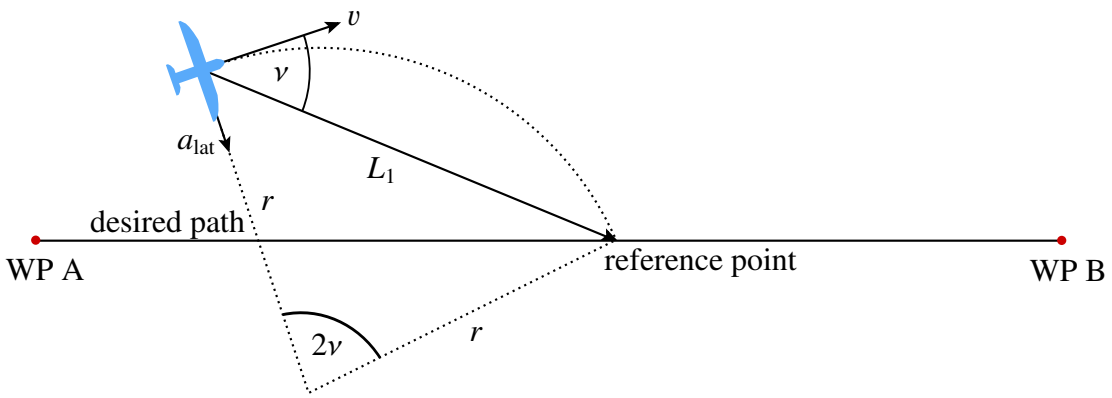


Figure 4.5: Diagram of the L1 control law [35].

increases the response of the aircraft. The angle between L_1 and the ground velocity of the drone v is ν.

To calculate the lateral acceleration a_{lat} in every loop first the reference point is calculated, it is the point on the desired path at the distance L_1 from the drone. Then the lateral acceleration a_{lat} is

calculated by

$$a_{\text{lat}} = 2\frac{v^2}{L_1}\sin\nu. \tag{4.7}$$

From the geometry it can be shown that the angle at center in Fig. 4.5 is 2ν. The relationship between L_1 and r is according to the law of cosine

$$\begin{aligned} L_1^2 &= r^2 + r^2 - 2\cdot r\cdot r\cos(2\nu) = 2r^2 - 2r^2(\cos^2\nu - \sin^2\nu) = 4r^2\sin^2\nu \\ \Rightarrow L_1 &= 2R\sin\nu. \end{aligned} \tag{4.8}$$

Inserting Eq. (4.8) into Eq. (4.7) yields

$$a_{\text{lat}} = 2\frac{v^2}{2r\sin\nu}\sin\nu = \frac{v^2}{r}. \tag{4.9}$$

Hence the aircraft flies on a circular path with radius r to the reference point, but before reaching it, a new reference point is calculated in the next cycle of the loop.

For the loiter mode a proportional-derivative (PD) control algorithm is implemented instead of the L_1 control algorithm. The lateral acceleration is computed by

$$a_{\text{lat}} = \frac{v^2}{r} + \omega_n^2 x + 2\zeta\omega_n\dot{x}, \tag{4.10}$$

where ζ is the damping ratio and ω_n is the natural frequency which can be set in plane parameters to tune the drone. $\frac{v^2}{r}$ is the *feed-forward* acceleration to fly a circle with radius r. The cross-track error d is the difference of the distance from the aircraft to the center of the circle and the radius of the circle, and the outbound velocity $\dot{d}$ is the component of the ground velocity radial to the circle.

In the last step a_{lat} is converted to the roll angle Φ according to Eq. (4.6).

WINDDRONE Mode

To program the lying figure-8 pattern, a flight mode in which the drone circles in a plane which can be tilted at any angle needs to be implemented. By switching between different circles according to Sec. 3.4 the figure-8 pattern is created. As a first approach this is done by extending the existing L_1 controller instead of establishing a new navigation controller with optimal control according to Eq. (3.11). Since no so-called optimal control algorithm is implemented in the ArduPilot code project yet, it cannot be guaranteed that the Pixhawk is able to run such an optimal control algorithm as it is computationally intensive.

For that a new flight mode called WINDDRONE is developed to calculate the demanded lateral acceleration to fly a tilted figure-8. By changing to this flight mode during flight it is initialised. The sphere radius R, the shape and orientation of the figure-8 are specified. From that, the circle radius r and the transformation matrix M_{pe} for each segment are computed.

The lateral acceleration depends on which point of the circle the drone is. The position is tracked by the navigation bearing angle α which is the angle between the x_p-axis and the projected position vector of the drone (see Fig. 4.6). The idea is that for a banked circle the lateral acceleration should

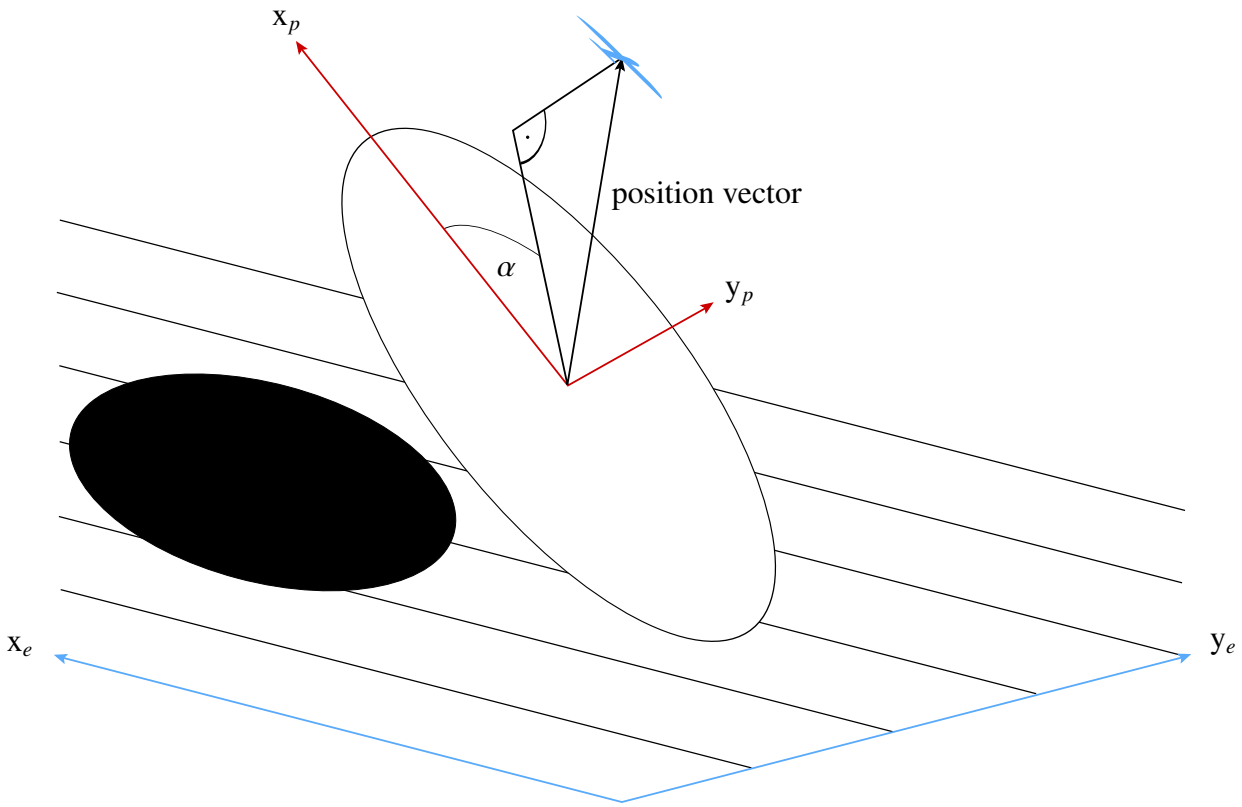


Figure 4.6: Drone flying along a circle which is not parallel to the Earth's horizontal plane.

become smaller by approaching the extrema of the circle, i.e. the lateral acceleration should be minimal at the highest and lowest point of the circle. In the extreme case that the circle is tilted by 90°, i.e. perpendicular to the Earth's horizontal plane, the lateral acceleration should be zero.

This can be achieved by taking the y_k-component of the acceleration vector described in the flight-path coordinate system (index k) as the feed-forward acceleration. The y_k-axis is always parallel to the earth-horizontal plane. The centripetal acceleration vector lies in the plane of the circle and points from the boarder to the center. In the extreme case where the circle is tilted by 90°, the centripetal acceleration vector lies always in the x_k-z_k-plane, so the y_k-component is zero.

For the case where the circle is parallel to the earth-horizontal plane, the y_k-axis points to the center of the circle[14], so the y_k-component equals the centripetal acceleration. The calculation of the centripetal acceleration described in the flight-path coordinate system is performed in B.8 for the complete case with $\omega \neq 0$ and $\sigma \neq 0$. The y_k-component here is

$$a_{\text{feed-forward}} = \frac{v^2}{r} \frac{\cos\vartheta}{\sqrt{\cos^2\alpha + \sin^2\alpha\cos^2\vartheta}}. \tag{4.11}$$

[14] The y_k-axis points to the center of the circle if the aircraft flies clockwise and it points away from the center if the aircraft flies counter-clockwise.

The feed-forward gain contains a singularity at $\vartheta = 90°$ and $\alpha = 90°$, but this point lies on the earth surface and is not reached by the drone. To control the flight path, the same PD controller from Eq. (4.10) is used, so the lateral acceleration is

$$a_{\text{lat}} = \frac{v^2}{r} \frac{\cos\vartheta}{\sqrt{\cos^2\alpha + \sin^2\alpha\cos^2\vartheta}} + \omega_n^2 x + 2\zeta\omega_n\dot{x}, \tag{4.12}$$

but x and $\dot{x}$ are defined differently. The horizontal deviation from the aircraft to the point of the circle where its projected position vector intersects is x, and $\dot{x}$ is the y_k-component of the trajectory velocity vector described in the flight-path frame $\vec{v}_{k,k}$.

To initialize the figure-8 pattern, the sphere radius R, the arc angle A, the crossing angle Γ and the orientation angles ω and σ have to be specified. The distance d, the circle radius r as well as the angles ϑ and ψ and the transformation matrix M_{pe} are calculated for each segment. To fly the figure-8 pattern, the acceleration command has to be adjusted to the circle which corresponds to the segment on which the drone is flying.

To determine on which segment the drone is the navigation bearing angle α is used. The circular arc of the turning paths is calculated in Eq. (3.23) and the circular arc of the crossing paths is $2AR$. So if the drone flies on a crossing path, the navigation bearing angle lies between $\alpha = -A$ and $\alpha = A$, and if the drone flies on a turning path, the navigation bearing angle lies between $\alpha = 270° + \delta$ and $\alpha = 90° - \delta$. If the drone reaches the end of a segment, it continues on the next segment and the specification of the new circle are passed to the navigation controller.

The formula for the feed-forward acceleration gets a bit more complicated since two more angles ω and σ are used to determine the orientation of the circles (see B.8). The final feed-forward acceleration is

$$a = \frac{v^2}{r} \frac{\cos\sigma\cos\vartheta - \sin\sigma\sin\vartheta\cos\psi}{\sqrt{(\cos\alpha\cos\sigma\sin\psi + \sin\alpha(\cos\sigma\cos\vartheta\cos\psi - \sin\sigma\sin\vartheta))^2 + (\cos\alpha\cos\psi - \sin\alpha\cos\vartheta\sin\psi)^2}}. \tag{4.13}$$

For every calculation of the lateral acceleration the altitude of the current destination on the path is transmitted to the speed height controller as a target altitude, as the navigation controller only controls lateral movement.

4.1.6 Speed Height Controller

The speed height controller is used for vertical navigation by calculating the demanded pitch angle and throttle. Like the navigation controller, the speed height controller offers an interface using abstract classes in *AP_SpdHgtControl.h* to facilitate the implementation of different speed height controllers.

The only available speed height controller in ArduPlane 3.5 is called *TECS* which stands for total energy control system. It receives a target altitude and a target airspeed[15] as input and calculates the

[15] The target airspeed is set by plane parameters to a reasonable value for stable flight, the target altitude is calculated during runtime by the autopilot.

demanded specific total energy (STE_{dem})[16]

$$STE_{\text{dem}} = \frac{1}{2}v_{a,\text{dem}}^2 + gh_{\text{dem}}, \tag{4.14}$$

which is the sum of the demanded specific kinetic energy $SKE_{\text{dem}} = \frac{1}{2}v_{a,\text{dem}}^2$ with the apparent airspeed v_a, and the demanded specific potential energy $SPE_{\text{dem}} = gh_{\text{dem}}$ with the gravitational acceleration g and height h. To maintain the total energy, TECS determines throttle demand which generates thrust to overcome drag.

Furthermore the specific energy balance SEB has to be correct. By controlling the pitch angle potential energy can be transferred to kinetic energy and vice-versa. Weighting factors ω_{SKE} and ω_{SPE} with $\omega_{SKE} + \omega_{SPE} = 2$ are used to specify how to react on height or speed error.

$$SEB_{\text{dem}} = SPE_{\text{dem}} \cdot \omega_{SPE} - SKE_{\text{dem}} \cdot \omega_{SKE}. \tag{4.15}$$

An $\omega_{SKE} = 2$ would ignore height errors completely, and the other way around an $\omega_{SPE} = 2$ would ignore speed errors completely. For special flight manoeuvres or special situations the weighting factors, set in the plane parameters, are overwritten by the autopilot on runtime. For example at take-off or if an underspeed condition[17] is detected, ω_{SKE} is set to 2.

The target airspeed is specified in the plane parameters. For the lying figure-8 pattern the target altitude is set to the height of the current point of the figure-8 pattern which is calculated in the navigation controller.

4.2 JSBSim

Using a simulation the operation of the new WINDDRONE flight mode can be reviewed. Furthermore the performance of the algorithm can be analysed by changing configuration parameters or the physical environment, e.g. wind speed or direction.

With JSBSim the dynamics of an aircraft can be simulated. The flight dynamics is formulated as four coupled, non-linear differential equations

- Position: $\dot{\vec{s}} = f_s(\vec{V}, \vec{\Phi})$
- Rotation Angle: $\dot{\vec{\Phi}} = f_\Phi(\vec{\Omega}, \vec{\Phi})$
- Trajectory Velocity: $\dot{\vec{V}} = f_V(\vec{F}, \vec{V}, \vec{\Omega}, \vec{\Phi})$
- Rotation Speed: $\dot{\vec{\Omega}} = f_\Omega(\vec{M}, \vec{\Omega})$,

where $\vec{M}$ is the moment vector and $\vec{F}$ is the force vector acting on the aircraft. The differential equations implemented in the source code of JSBSim are referenced to the book "Aircraft Control

[16] Specific energy means energy per unit mass.
[17] An underspeed condition occurs, if the apparent airspeed is only 15 % higher than the airspeed where a stall happens.

and Simulation" by Brian Stevens and Frank Lewis [36].

This set of differential equations calculates position, rotation angle, trajectory velocity and rotation speed, which results from moments and forces captured in $\vec{M}$ and $\vec{F}$.

To calculate forces and moments acting on the drone, JSBSim provides a physical model of the atmosphere and of the aircraft. The atmosphere model determines temperature, pressure, air density, wind, etc., and the aircraft model defines its weight, inertia and characteristic measurements, e.g. wingspan and wing area.

Furthermore the aerodynamics are specified in the aircraft model using stability and control derivatives. As described in Sec. 2.1, the aerodynamic coefficients can be modelled as the sum of stability and control derivatives (see Eq. (2.11)). As the coefficients depend on many parameters only those with the most influence are included to a simplified model.

Forces and moments are calculated according to Eq. (2.6) and Eq. (2.9), respectively.

The default aircraft model in JSBSim is the Rascal 110 model plane, which is also used for testing the autopilot. The implementation of a model of the wind drone, however desirable, is beyond the scope of this thesis.

As an example, the calculation of the lift force in JSBSim of the Rascal 110 model is demonstrated. The lift force is calculated according to Eq. (2.2). JSBSim uses two parameters, angle of attack α and deflection angle of the elevator η, to model the lift factor C_L, resulting in the modelled lift force

$$F_L = \frac{\varrho}{2} v_a^2 A \cdot \left(\frac{\partial C_L}{\partial \alpha} \alpha + \frac{\partial C_L}{\partial \eta} \eta \right). \tag{4.16}$$

The value for $\frac{\partial C_L}{\partial \eta}$ in the Rascal 110 model plane is 0.2. For $\frac{\partial C_L}{\partial \alpha}$ JSBSim uses a piecewise linear function to model the characteristic curve in Fig. 2.2 with a derivative of 5 if $\alpha < 23$ rad and -1.9 otherwise.

With linearised stability and control derivatives JSBSim can model moments and forces which result from control inputs, i.e. deflection of control surfaces, and flight conditions to update the position, rotating angle, trajectory velocity and rotation speed in discrete time steps.

To simulate the tether on which the aircraft is leashed, a force acting on the center of gravity pointing to the start location is added as an external reaction to the model. The force acts not before a specified distance between aircraft and starting location, which is chosen to be a bit less than the hemisphere radius.

4.3 Ground Control Station

A ground control station is a software application that communicates with the drone via telemetry. It is used for configuration and calibration of the drone as well as displaying real-time flight data. Furthermore, a mission can be planned which is a list of flight commands like take-off, navigation to a waypoint and so on. Within the mission, the commands are performed, subsequently.

Mandatory hardware configuration such as the calibration of the accelerometer (see Sec. 5.3) are executed using the ground control station. In addition, plane parameters can be set before and during flight.

CHAPTER 5

Hardware

This chapter covers the entire construction and assembly of the hardware for the drone. A glider plane was purchased, which had to be modified to endure the large forces acting on the plane while in tethered flight (Sec. 5.2). To enable autonomous flight, the plane was fitted with various sensors and modules (Sec. 5.1). These have to be calibrated to ensure correct operation of the autopilot (Sec. 5.3).

5.1 Pixhawk

The Pixhawk is developed as an open-hardware project [37]. It runs a real time operating system called NuttX, which executes the compiled ArduPilot code. The software package PX4Firmware provides drivers for sensors and other peripherals which are connected to the Pixhawk via DF13 connectors. The Pixhawk already includes required sensors like an accelerometer, a gyroscope, a

Figure 5.1: Pixhawk.

magnetometer and a barometer. Additionally, a GPS/compass module and a digital airspeed sensor are attached to the I2C serial port of the Pixhawk. To improve the estimate of the true state of the drone, additional sensors like a range finder or optical flow sensor can be connected, but are not used here.

5.2 Building the Wind Drone

This section describes the transformation of a glider plane to a wind drone. The Easy Star II glider plane from Multiplex is used as it has multiple favourable features. It furnishes a large internal compartment to provide enough space for all the electronics and sensors, and a rear engine for minimal interference with the sensitive sensors. Besides, it is an often used and widely available model.

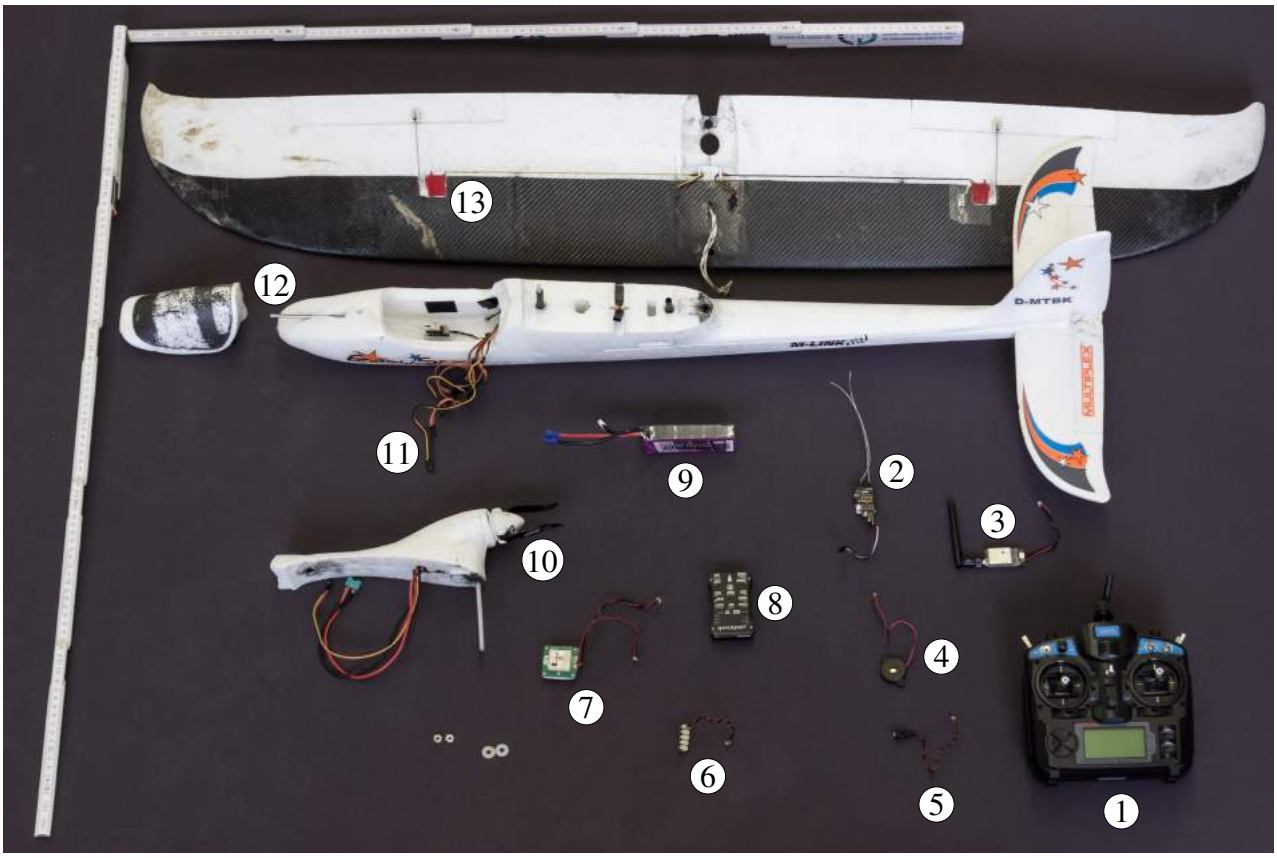


Figure 5.2: Unmounted wind drone with all hardware parts:

1 Transmitter for remote radio control
2 Receiver to receive signals from the transmitter
3 Telemetry used for communication between ground control station and drone
4 Buzzer makes sounds to inform about status
5 Safety switch to prevent for accidental arming
6 I2C splitter provides three additional ports for digital peripherals
7 GPS/Compass module provides positioning and heading data during flight
8 Pixhawk
9 Battery provides power
10 Propeller provides thrust
11 Connection cables to servos
12 Airspeed sensor measures apparent airspeed
13 Servos to steer the ailerons; servos for rudder and elevator are inside the fuselage

In Fig. 5.2, the unmounted wind drone is displayed with all hardware parts. The production of

airborne wind energy involves large forces acting on the device. Although the aim of this project was not to build a power source, still a large load is affecting the wind drone. Therefore resilient materials are needed, and the styrofoam is reinforced by carbon tubes and fabric at multiple locations.

The original wing consists of a left and a right part, connected by a long plastic tube, which leads through the wing. This plastic tube is replaced by a carbon tube as it adds more stability to the wing.

A string is lashed up around the middle of the carbon tube to fasten the tether. To disperse the resulting force acting on the tube, a short carbon tube with a larger diameter is put over the long carbon tube[18] as shown in Fig. 5.3.

Figure 5.3: Two nested carbon tubes.

After inserting the nested carbon tube into the wing parts, both halves are glued together and extra styrofoam is attached on a recess with foam glue, as marked in Fig. 5.4.

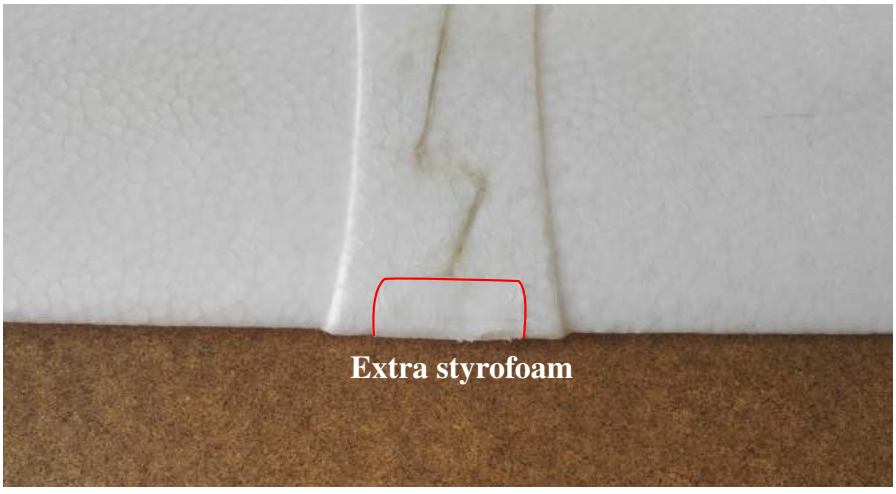


Figure 5.4: Glued wing parts with extra styropor in the recess.

[18] The long carbon tube has a length of 59 cm and a diameter of 8 mm; the short carbon tube has a length of 6 cm and a diameter of 10 mm

Edges are reduced using sandpaper, so that later carbon fibre fabric can be attached to the wing. Furthermore, the whole surface is roughened with sandpaper where the carbon fibre fabric is placed. Thereby, epoxy resin, which fixes the carbon fibre fabric to the wing, can deliver better adhesion.

As the wing experiences a much higher load in the tethered flight than in free flight, it is reinforced with carbon fibre fabric. Before the carbon fibre fabric is affixed to the wing, four carbon racks are placed perpendicular to the carbon tube as demonstrated in Fig. 5.5. For this, four slits are cut in the wing, and a hollow in each rack is rasped such that they fit on the carbon tube. Epoxy resin and foam glue is used to fix the racks.

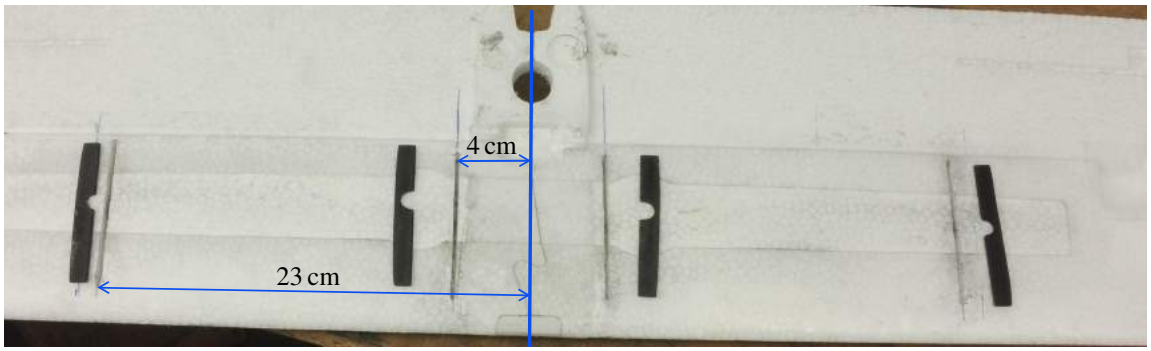


Figure 5.5: Four racks with hollow and rounded top edge.

Finally the carbon fibre fabric is attached to the wing with epoxy resin and hardener. After the curing process a small piece of the carbon fibre fabric in the middle of the wing above the carbon tube is cut out so that a string can be lashed up around the carbon tube, as depicted in Fig. 5.6.
In the original condition the wing was plugged from the right and the left side in the body. Since the reinforced wing now consists of one part, the roof has to be cut from the fuselage. By doing so, the wing can be plugged on the fuselage from above, as demonstrated in Fig. 5.7.
For this purpose carbon tubes are installed as a guidance. Short carbon tubes with a diameter of 10 mm are stuck in the front and back of the wing. Carbon tubes with a smaller diameter of 8 mm are fixed to the fuselage.
The roof is tightened down to the fuselage with two thread rods and nuts. Carbon tubes serve also as an encasement for the nylon rods.

After modifying the glider plane, the Pixhawk, sensors and other peripherals are incorporated. The embedding of the GPS/compass module and the airspeed sensor is pictured in Fig. 5.9. Both are glued on a separate piece of styrofoam which is fixed with hook and loop fastener in the compartment.

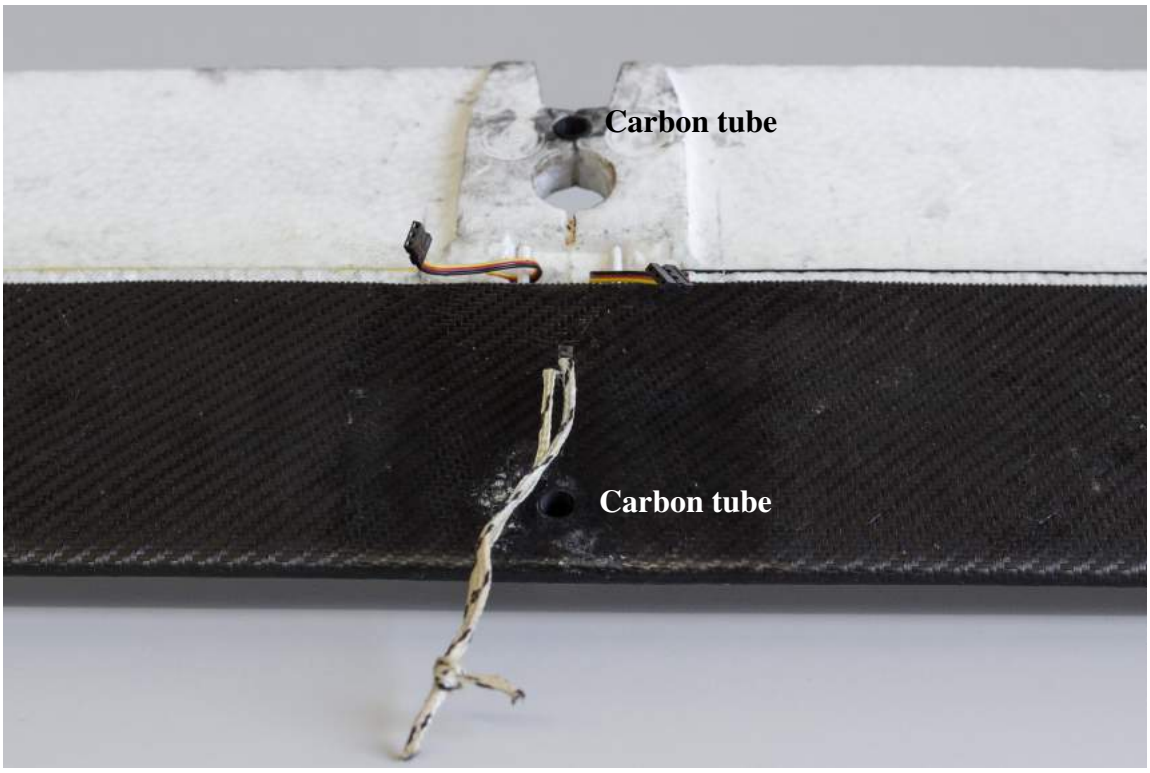


Figure 5.6: Reinforced wing and string attached to the carbon tube inside the wing.

(a) (b)

Figure 5.7: (a) Carbon tubes are installed in the fuselage as wing guidance. The left carbon tube is also used as an encasement for a thread rod. (b) Wing plugged on the fuselage.

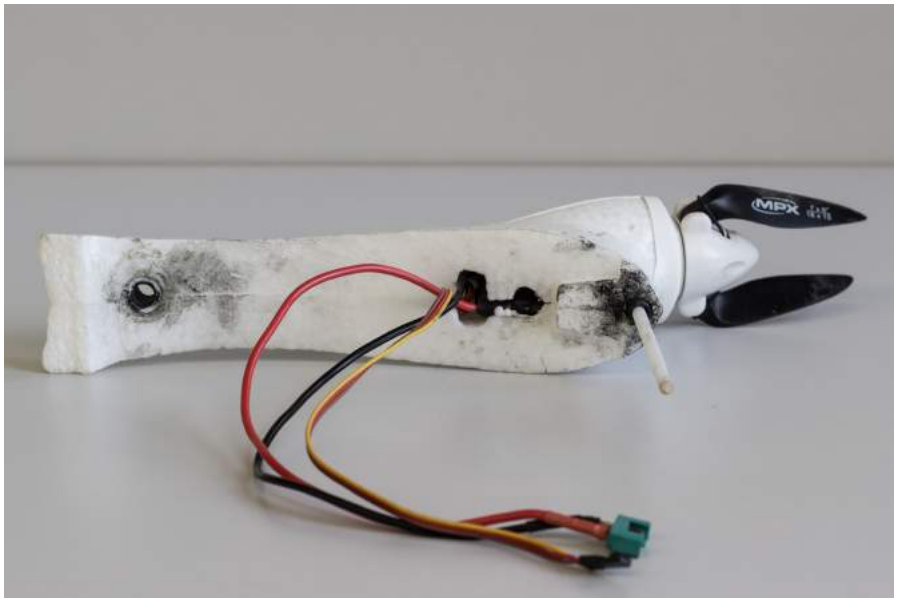
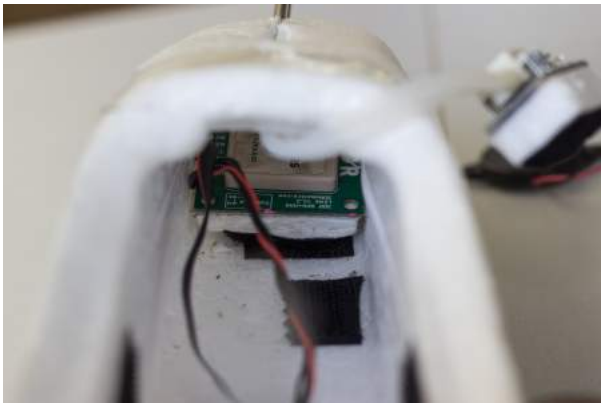
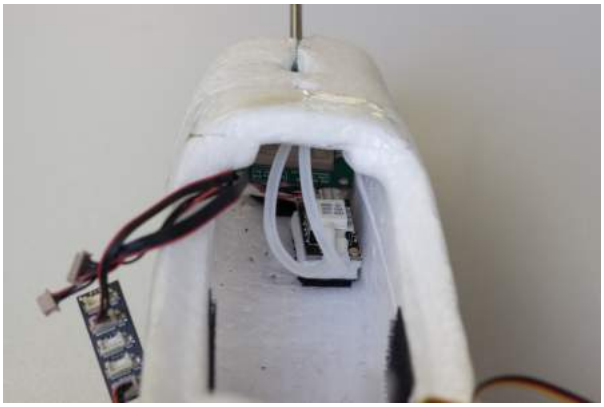


Figure 5.8: Roof.



(a)



(b)

Figure 5.9: Incorporated sensors fixed with hook and loop fastener. (a) GPS/compass module (b) Airspeed sensor.

The GPS/compass module is placed in the nose of the aircraft as far as possible from the motor to reduce interference. The airspeed sensor is placed directly behind the GPS/compass module. It is connected via rubber tubing to the Pitot tube, which measures the differential pressure to determine the apparent airspeed.
The Pitot tube is pushed through the foam in the cockpit, so it points into the air stream. Since the flight on a tether is susceptible to crash-landing nose first, the tube is not fixed at the front as shown in Fig. 5.10. The battery and Pixhawk are inserted side by side under the cockpit cover behind the airspeed sensor and are fixed with a hook and looper fastener.

Figure 5.10: Pitot tube fixed on the nose of the drone.

Figure 5.11: Complete Wind Drone.

Figure 5.11 pictures the complete wind drone. The total weight including all sensors, battery, etc. is about 950 g. The increase in weight due to the reinforcement of the wing amounts for 130 g.

5.3 Configuration and Calibration

After the construction, the wind drone has to be configured and sensors have to be calibrated for a stable operation of the autopilot. The configuration procedure is very well documented by the

ArduPilot project [29]. Nonetheless, a short roundup will be given here and some outcomes of the calibration process are presented.

5.3.1 Mandatory Hardware Configuration

In this section mandatory calibration and configuration processes are described, which are performed on ground. These processes are selected and initialised by a ground control station.

Accelerometer Calibration

The ground control station guides the user through the accelerometer calibration procedure by requesting to place the aircraft in different positions. In detail these are level, right side, left side, nose down, nose up and its back.

Compass Calibration

The compass calibration is performed by rotating the drone slowly around all axes to calculate offsets caused by magnetic distortions. First the aircraft is oriented towards north and rotated around its y_d-axis about 360°, then it is turned by 90° around its z_d-axis and afterwards rotated around the x_d-axis until level.

Radio Control Calibration

The control sticks and toggle switches of the radio control transmitter are moved through their full range. The maximum and minimum position are recorded and saved.

Flight Mode Configuration

ArduPlane provides many flight modes that can be selected with the radio control transmitter during flight. The flight mode switch of the transmitter can choose between 6 flight modes which are set via a ground control station.

MANUAL One of the six flight modes has to be the MANUAL mode. This is the regular flight mode with no assistance of the autopilot. The aircraft is only controlled by the user with the radio control transmitter.

FBWA The fly by wire (FBWA) mode is the best mode for inexperienced pilots. Like in the MANUAL mode, the control surfaces and throttle are controlled by the user. However, the aircraft is stabilized by the autopilot, e.g. by unhanding the transmitter, the drone will steer to a level position. Furthermore roll and pitch angle are limited, which is specified by the plane parameters. For example, if the aileron stick is hold hard right, the drone will bank at the maximum roll angle and fly a curve instead of turning on its own axis.

AUTOTUNE This mode is used to tune the roll and pitch controller. The drone is assisted by the autopilot like in the FBWA mode.

RTL By switching to the return to launch (RTL) mode the drone starts circling above its home location[19] at an constant height. Radius of the circle and height can be set in the plane parameters.

AUTO To follow a mission set by the ground control station, the AUTO mode is applied.

In this thesis, the following new mode was implemented.

WINDDRONE This mode flies a lying figure-8 pattern. The orientation of the figure-8 can be set in flight by the plane parameters via a ground control station.

A description of the other flight modes can be found in the ArduPilot documentation [29].

5.3.2 Roll and Pitch Controller Tuning

The control gains of the roll and pitch controller determine the response of the aircraft to roll and pitch commands. Tuning these parameters leads to a stable flight and effective navigation of the drone. A simple way to do this is to use the AUTOTUNE mode. In the AUTOTUNE mode the aircraft flies in the same way as in the FBWA mode, but the algorithm learns how the drone responds to attitude changes. So the user has to prompt many extreme control inputs until reasonable tuning values are achieved which are saved under plane parameters.

5.3.3 Navigation and Speed Height Controller Tuning

The navigation controller makes use of an proportional-derivative (PD) controller to compute the demanded lateral acceleration. The damping ratio ζ and the natural frequency ω_n determine the response of the drone to deviations from the demanded flight path. Proper values for both parameters can be obtained by observing the reaction of the drone to immediate changes of the demanded flight path. If the drone converges very slowly to the new flight path, ω_n can be increased and ζ decreased. Otherwise, if the flight of the drone becomes unstable by oscillating strongly around the new flight path, ω_n has to be decreased and ζ to be increased.

Tuning of the speed height controller is skipped as flying the lying figure-8 pattern takes up extreme adjustments. Tuning parameters like the maximum and minimum pitch angle as well as maximum climb and sink rate are set to large values shortly before switching to the WINDDRONE mode. Values for the maximum and minimum pitch angle are set to $80°$ and $-80°$, respectively. The maximum climb and sink rate are set to $20\,\mathrm{m\,s^{-1}}$ and $-20\,\mathrm{m\,s^{-1}}$, respectively.

Maintaining these values for other flight modes is not recommended, as for example, a stable automatic take-off becomes nearly impossible. For this reason, the tuning parameters are kept by their default values for normal flight mode and are only set to extreme values just before switching to the WINDDRONE mode.

5.3.4 Airspeed Sensor Calibration

The airspeed sensor calibration is performed by flying a circular path for several minutes. The calibration process is depicted in Fig. 5.12, which plots the apparent airspeed (green) and ground

[19] The home position is locked when the aircraft is armed.

speed (red) of the drone. The data is logged as soon as the drone is armed and ends when it is disarmed. The background color indicates the selected flight mode.

The safest mode to arm the drone is in the MANUAL mode, as the control lies completely by the user. Before flying the drone, the flight mode is switched to FBWA to check the response of the control surfaces. As this mode stabilises the attitude of the aircraft, the control surfaces should react to position changes, e.g. elevating the nose should deflect the elevator downwards to induce a pitch moment to return the aircraft to level position.

To fly the circular path for the calibration procedure, a mission is prepared with a ground control station. The mission consists of only two flight commands. The first command is to take-off up to 5 m altitude, the second command is to switch to RTL mode with an target altitude of 70 m and target radius of 70 m.

After circling around for several minutes, the flight mode is switched to FBWA to easily land the drone. The throttle is turned off and the aircraft is piloted to the ground.

The calibration procedure is performed during the AUTO mode where the drone is flying a circular path. After the take-off command, the speed height controller maintains a constant airspeed of $17\,\mathrm{m\,s^{-1}}$. For a constant wind speed and wind direction, the ground speed varies around the airspeed. If the drone flies against the wind direction, the ground speed is lower at the airspeed by the amount of the wind speed. If the drone flies in direction of the wind speed, the ground speed increases by the amount of the wind speed compared to the airspeed.

The wind speed can be calculated from the difference of the maximum or minimum ground speed to the airspeed. By consulting the heading of the aircraft at the maximum and minimum ground speed, the direction of the wind can be calculated, too. Fortunately, the Kalman filter also estimates the wind speed and direction as shown in A.1.
For a calibrated airspeed sensor the airspeed should be in the middle of the sinusoidal curve of the ground speed. As can be seen in Fig. 5.12, at the beginning of the calibration procedure, the airspeed sensor measures an airspeed that is higher than the true airspeed but during the process the measurement converges to the true estimate.

Before flying, the airspeed measurement is very noisy. This is not a calibration problem. As the airspeed sensor measures the dynamic pressure according to Eg. (3.1), the airspeed measurement varies with the square root of the pressure, leading to a lot of variance with small pressure changes near zero.

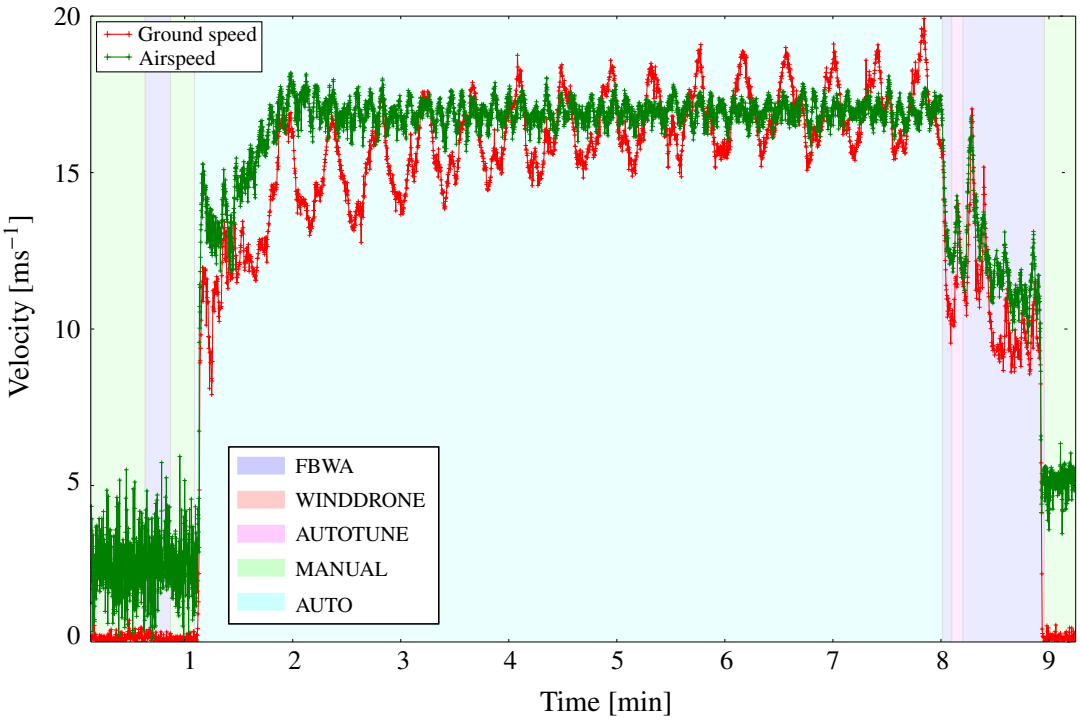


Figure 5.12: Airspeed sensor calibration

CHAPTER 6

Results

The flight algorithm is first evaluated in a simulation using JSBSim and the simple model of the tether force (see Sec. 4.2) before it is tested on the wind drone. Sec. 6.1, Sec. 6.2 and Sec. 6.3 discuss different results of the simulation phase.

After a successful operation of the algorithm on the simulated aircraft, the autopilot code is loaded to the Pixhawk. Sec. 6.4 presents the data collected with the wind drone by flying the lying figure-8 pattern with and without a tether.

6.1 Flight Pattern

A Rascal 110 model plane is the default aircraft in JSBSim and used to test the autopilot code. As it is larger and heavier than the constructed wind drone, the sphere radius has to be set to 400 m as the Rascal is not able to fly stable at a smaller radius.

The shape of the figure-8 pattern is defined in the code with the arc angle A and the crossing angle Γ. A value of 20° is chosen for both angles as it results in the intended shape.

The wind direction is set from the north for analyses of the simulation. To align the figure-8, ω is set to 0°, so that the cross axis of the figure-8 stands perpendicular to the wind direction. The height of the figure-8 is controlled by the tilting angle σ.

To dive into the figure-8 pattern, the drone first circles on the surface of the hemisphere at the same altitude as the upper initial point of a crossing path. Therefore, a mission consisting of a take-off command followed by a RTL command with the specified radius and altitude is set. As soon as the wind drone approaches the initial point of the crossing path, the flight mode is switched via the radio control transmitter to WINDDRONE and the wind drone dives into the figure-8 pattern.

This procedure specifies the direction of the motion by which the figure-8 pattern is flown, namely, the crossing paths are always flown downwards.

For the landing procedure another mission is set to land the aircraft close to the take-off position. After flying some cycles along the figure-8 pattern, the mode is switched again with the radio control transmitter to AUTO at the endpoint of the crossing path to execute the landing mission.

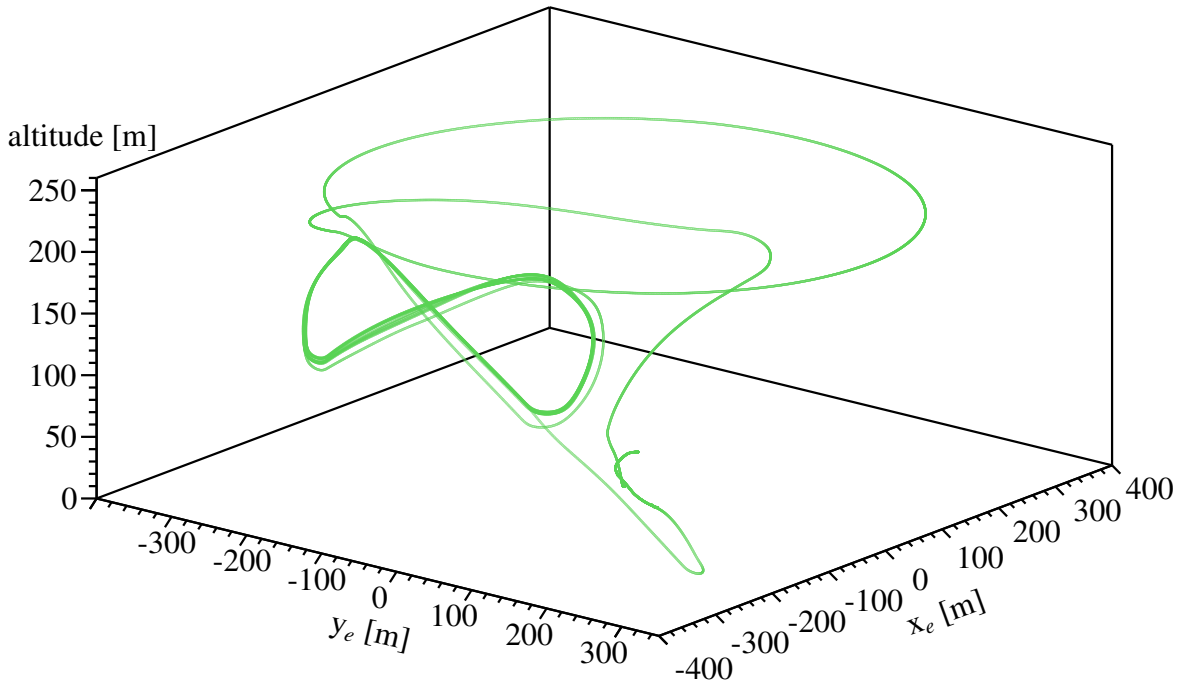


Figure 6.1: Full simulated flight path.

The full simulated flight path is illustrated in Fig. 6.1 for $\omega = 0°$ and $\sigma = 60°$. Both ω and σ are plane parameters set by a ground control station.

To improve the full flight path, the orientation of the figure-8 could be determined by the algorithm, instead of being manually set. The autopilot can use the wind direction, estimated by the Kalman fiter while the drone is flying on the circular path, to set ω. By creating a command in the mission, where the aircraft changes by itself the flight mode to WINDDRONE as it reaches the initial point of the crossing path of the figure-8, a full autonomous flight can be achieved.

6.1.1 Flight Path Deviation

In order to investigate how accurate the autopilot works, the deviation of the flight path from the demanded path along the figure-8 pattern is analysed. The deviation is the difference between the actual position and the desired position. Furthermore, the average of the flight path deviation along the figure-8 is examined for different wind speeds and tilting angles σ.

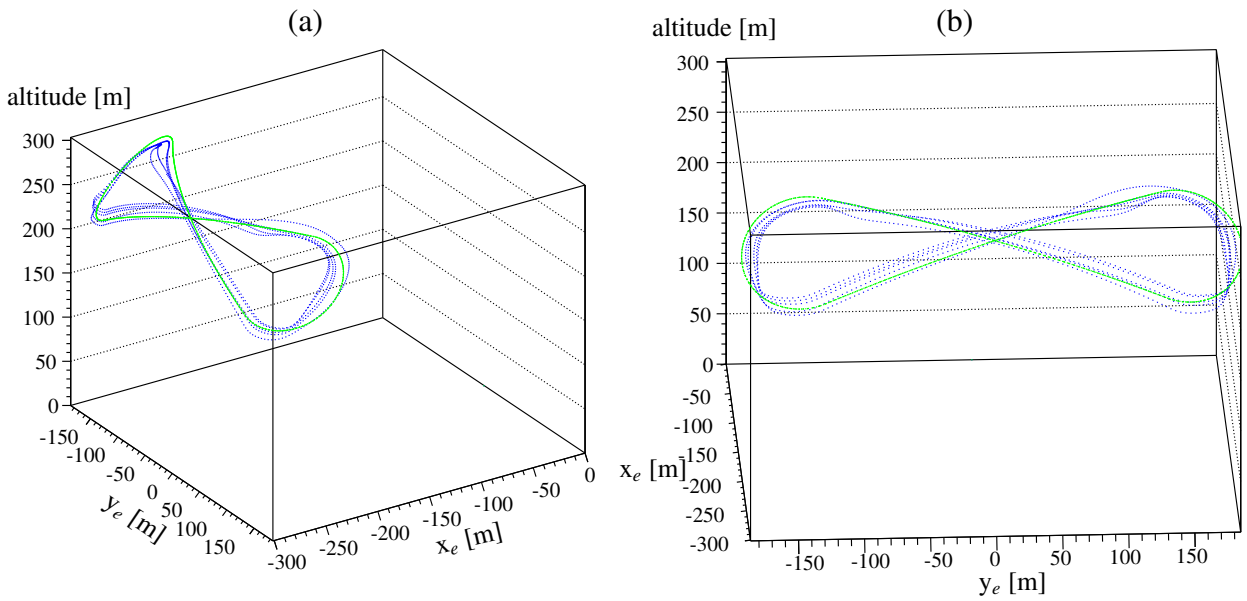


Figure 6.2: Comparison between actual (blue) and demanded (green) flight path plotted from two perspectives. Wind is coming from north at $10\,\mathrm{m\,s^{-1}}$ and $\sigma = 45°$.

Figure 6.2 compares the actual with the demanded flight path from two perspectives. The average deviation along the figure-8 pattern amounts to 8.32 m. The average deviation is similar in each segment of the lying figure-8 pattern.

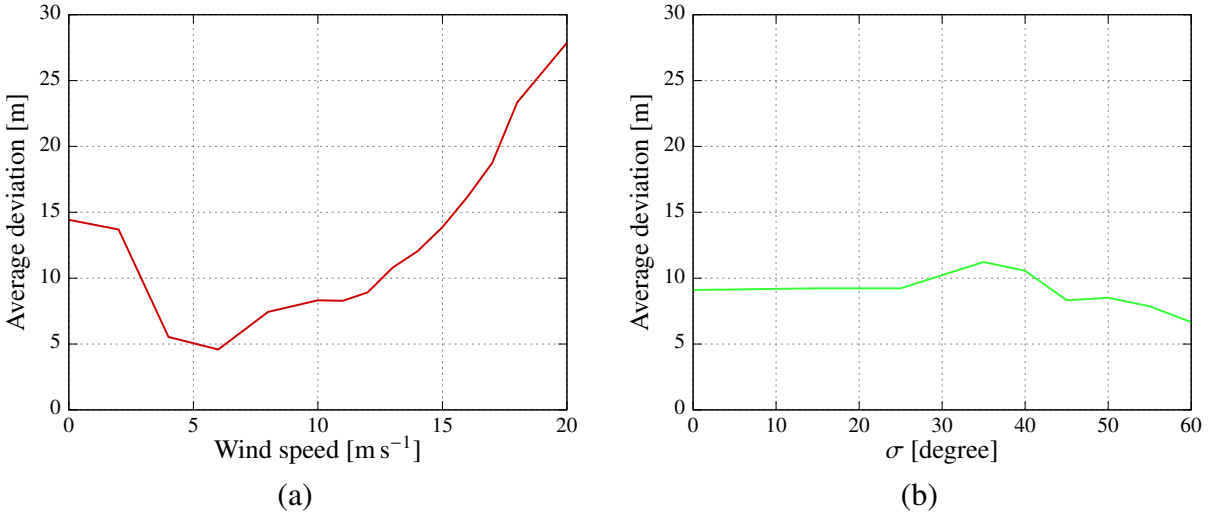


Figure 6.3: Average flight path deviation as a function of (a) wind speed for $\sigma = 45°$ and (b) σ for $10\,\mathrm{m\,s^{-1}}$ wind speed.

The average flight path deviation is plotted as a function of the wind speed and the tilting angle σ, which is depicted in Fig. 6.3. The average flight path deviation generally increases with higher wind speeds, as the disturbance becomes larger. For small wind speed below $5\,\mathrm{m\,s^{-1}}$ the average flight path deviation also increases to up to 15 m. The minimum flight path deviation with 4.6 m is reached at a wind speed of $6\,\mathrm{m\,s^{-1}}$.

A similar pattern is also observed for different values of σ. For example, a minimum flight path deviation of only 1.6 m is reached at a wind speed of 10 m s^{-1} for $\sigma = 60°$, see A.2. The average flight path deviation does not vary much for different σ at constant wind speed. This is independent of the wind speed.

This allows the conclusion that a moderate wind speed adds the most stability to fly a lying figure-8 pattern.

6.1.2 Drone Attitude

Figure 6.4 illustrates the attitude of the drone during its flight along the figure-8 pattern. The plot gives another insight into the functioning of the navigation as well as the speed height controller, as the attitude is determined by these controllers. To further investigate the attitude of the drone in flight,

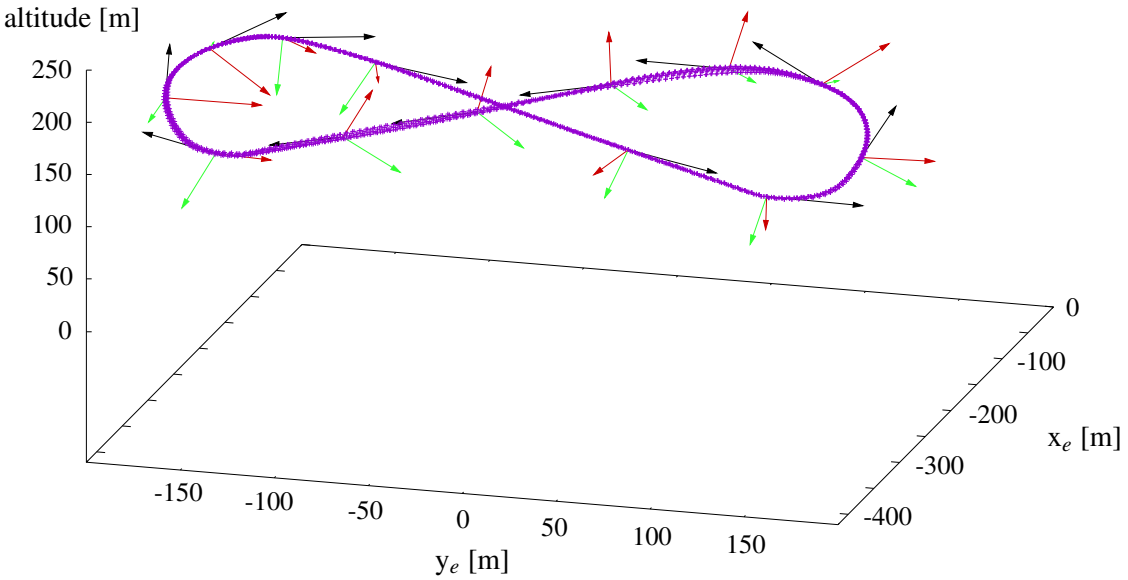


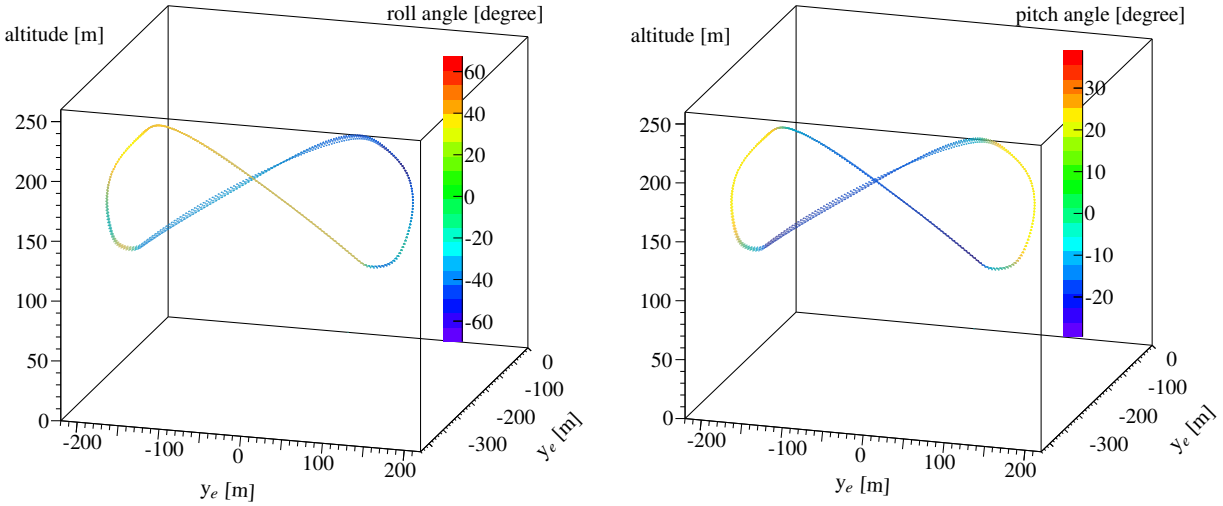
Figure 6.4: Attitude along figure-8. The drone-fixed coordinate system is illustrated for one cycle every two seconds; one point on the crossing is omitted for clarity. x_d in black, y_d in red, z_d in green. wind speed of 10 m s^{-1} and $\sigma = 60°$ are set.

the roll as well as the pitch angle are plotted along the figure-8 in Fig. 6.5 for the same parameters as in Fig. 6.4.

In part (b) the direction of motion can again be seen. On the crossing paths the pitch angle is negative and on the turning paths positive.

In part (a) the roll angle can be seen to switch between extreme values very fast at the transition from the crossing path to the turning path. For example, at the bottom left corner of the figure-8, the roll angle switches from about $-30°$ to about $35°$ and back to $-20°$ in a short distance.

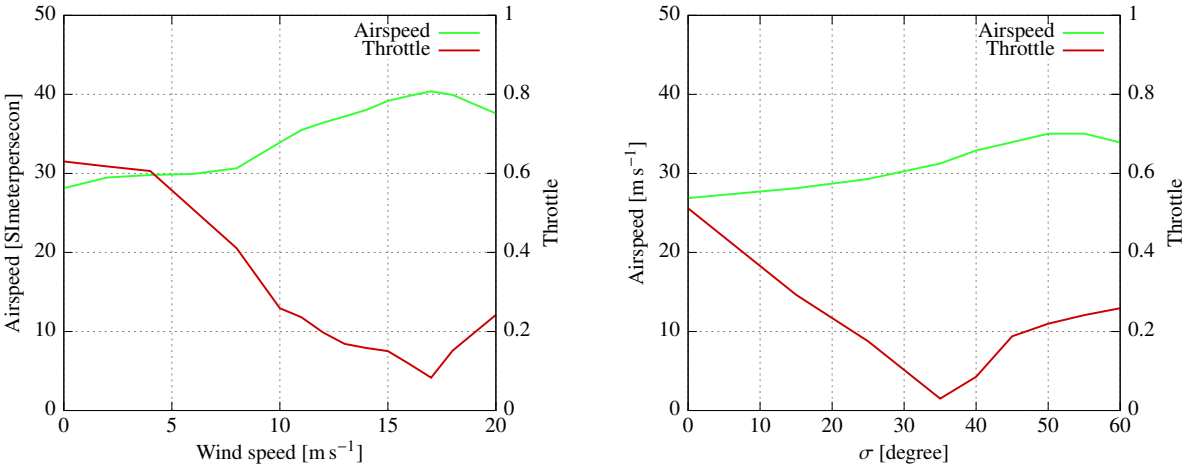
This results from the navigation algorithm determining the lateral acceleration demand, which jumps from one segment to another. This could be avoided by including the tether tension in the control algorithm to fly upward bends.



(a) (b)

Figure 6.5: (a) Roll angle and (b) pitch angle along figure-8 pattern for $\sigma = 60°$ and $10\,\mathrm{m\,s^{-1}}$ wind speed.

6.2 Airspeed and Throttle



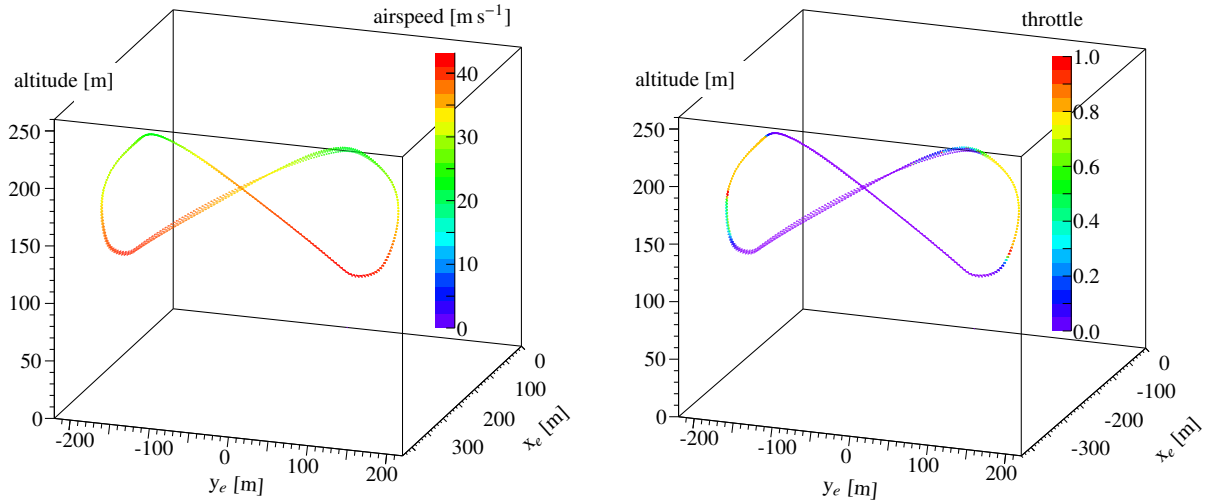
(a) (b)

Figure 6.6: Airspeed and throttle on flight path (a) as a function of wind speed at $\sigma = 60°$, (b) as a function of σ at wind speed of $10\,\mathrm{m\,s^{-1}}$. Throttle value of 1 means full throttle, throttle value of 0 means no throttle.

Looking at part (a) of Fig. 6.6 it can be seen that airspeed is increasing with the wind speed while throttle is decreasing, but only to a wind speed of $17\,\mathrm{m\,s^{-1}}$. As can be seen in Sec. 6.1, the average

deviation is high for high wind speeds, so the controller has to adjust for the deviation.

Part (b) shows the relationship of airspeed and throttle to different tilting angles σ at $10\,\mathrm{m\,s^{-1}}$ wind speed. Airspeed is increasing while throttle decreases for larger σ until 35° is reached, as the lying figure-8 pattern lies more in the wind. For stronger tilting angles, throttle starts increasing, as the drone gets problems to fly upwards on the turning path.



(a) (b)

Figure 6.7: (a) Airspeed and (b) throttle along figure-8 pattern for $\sigma = 60°$ and $10\,\mathrm{m\,s^{-1}}$ wind speed.

Figure 6.7 demonstrates airspeed and throttle along the path. As can be seen in part (b) up to full throttle (throttle = 1) is needed to fly on the turning path which increases the average throttle over the full flight path, while the throttle along the crossing path is near zero.

Part (a) illustrates the airspeed along the path. As it can be seen, airspeed increases strongly by flying downwards on the crossing paths to more than $40\,\mathrm{m\,s^{-1}}$ maximum airspeed at the lowest point of the figure-8.

Wind Direction

If the cross axis of the figure-8 pattern is perpendicular to the wind speed, the average airspeed along the crossing path from East to West and from West to East are the same. For a given side angle of the wind speed, they vary and one could estimate the wind direction to adjust the orientation of the figure on flight.

The results in Tab. 6.1 shows that the average airspeed on one crossing path is higher than on the other, a kind of asymmetry. To investigate this further, the drone flies level with a demanded roll angle of 0° for the whole flight at zero wind speed. This means the drone should fly straight forward, but the drone has the tendency to fly to the right which means that an additional lateral acceleration acts on the aircraft[20].

[20] Data is plotted in Fig. A.8.

This can explain why the average airspeed along the crossing paths differs even if the cross axis of the figure-8 pattern is perpendicular to the wind direction. The source of this additional acceleration is unknown.

	Airspeed [m s^{-1}]		Throttle	
ω [degree]	West to East	East to West	West to East	East to West
-15	36.16	31.87	0.003	0.164
-5	35.81	31.74	0.003	0.130
0	35.13	33.87	0.005	0.044
5	34.62	34.43	0.008	0.018
15	33.15	35.31	0.102	0.004

Table 6.1: Airspeed and throttle on Crossing path for different angles between the cross axis of the figure-8 and the wind direction.

Table 6.1 shows also the airspeed and throttle along the crossing paths if the wind direction is not perpendicular to the cross axis of the figure-8 pattern. The wind still comes from the north, but the orientation of the figure-8 is changed by ω. Figure 6.8 plots the airspeed along the figure-8 pattern with $\omega = 15°$.

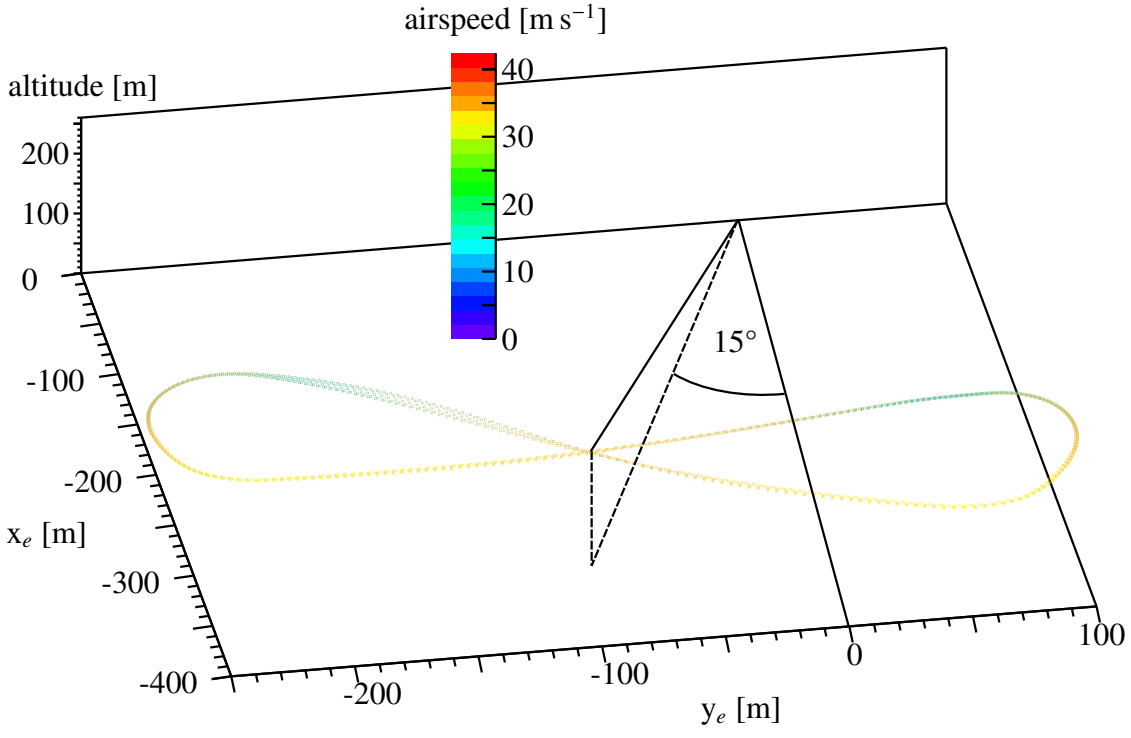


Figure 6.8: Airspeed along figure-8 Pattern with wind direction not perpendicular to cross axis.

For positive values of ω, the airspeed increases on the crossing path from east to west while the airspeed decreases on the crossing path from west to east, compared to the airspeed at $\omega = 0°$. At the same time the throttle goes down when airspeed is increasing and vice versa.

6.3 Angle of Attack α

The linearisation of the aerodynamic coefficients is only a good approximation for a specific range of values for the flight condition parameters. The most influential flight condition parameter is the angle of attack α. The maximum value of its working range is about 15° (see Sec. 2.1).

Table 6.2 shows that for various tilting angles σ and wind speeds the average angle of attack along the figure-8 pattern is well below the maximum of the working range.

Moreover, the average angle of attack does not differ notably between the segments for given wind speed and σ[21].

Wind speed \ σ	0°	25°	50°	60°
0 m s^{-1}	3.55°	4.48°	1.48°	1.76°
6 m s^{-1}	3.22°	3.11°	1.60°	1.80°
10 m s^{-1}	2.66°	2.62°	2.06°	1.61°
16 m s^{-1}	2.31°	2.55°	2.24°	2.14°

Table 6.2: Average angle of attack for different wind speeds and tilting angles σ.

The angle of attack can be used to optimise the tether tension by flying the lying figure-8 pattern as the lift and drag factor depend on it. An angle of attack that maximises the lift to drag ratio increases the power output of the airborne wind energy system.

Thus a possible future improvement of the flight algorithm would be to optimise the angle of attack for maximal tether tension and thus energy production.

[21] Data not shown.

6.4 Real Data

In a first attempt the real wind drone flies without a tether to monitor, if it is able to perform the extreme control inputs to fly a lying figure-8 pattern. The sphere radius is set to 150 m and σ to 45°. The wind direction on this trial is from the east, so ω is set to −90°.

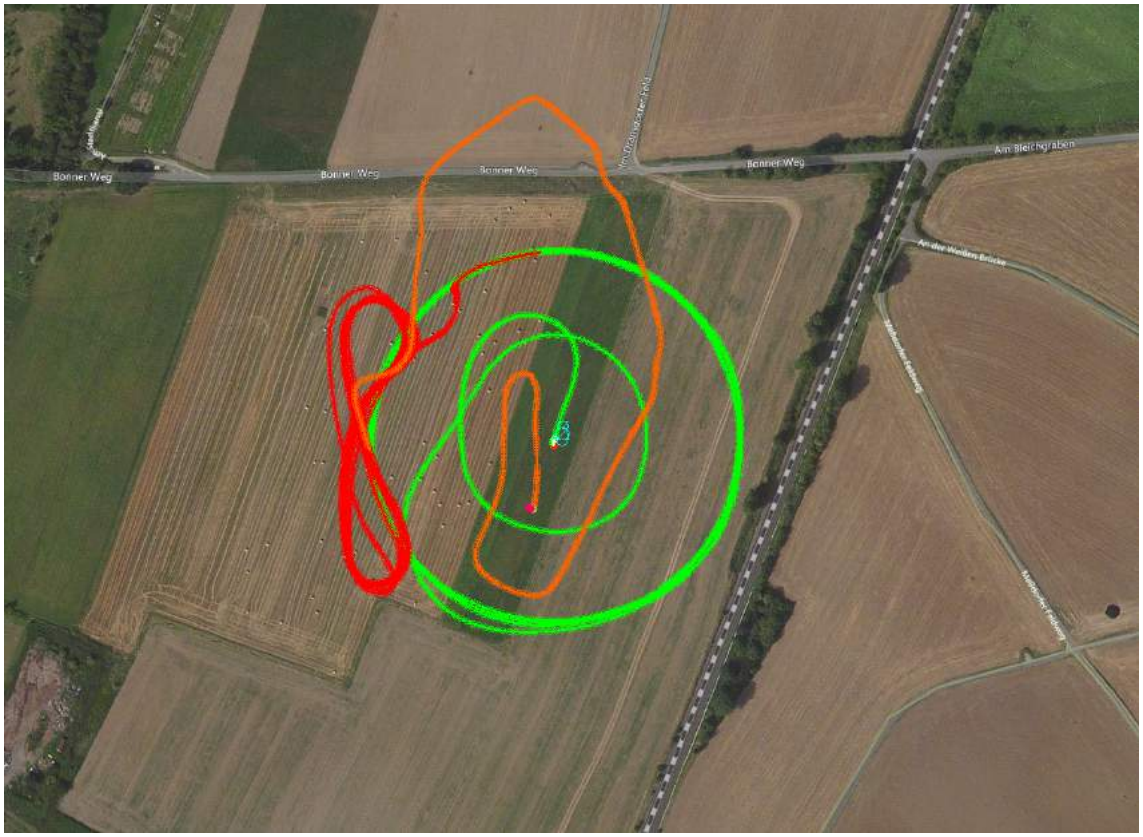


Figure 6.9: Real flight without tether for $\omega = -90°$ and $\sigma = 45°$.

Figure 6.9 shows the actual flight path of the drone. The different colors of the flight path indicate different flight modes. For the first stage (green) a mission is planned consisting of a take-off command followed by a counter-clockwise circling command. The radius is set to 100 m and the altitude of the circle is set to 112 m as the height of the upper starting point of the crossing path is also 112 m.

The WINDDRONE mode, representing the second stage of the flight path, is marked in red. The switching into the WINDDRONE mode is executed too early, nonetheless the drone flies stably the figure-8 pattern. In order to land the drone, the mode is switched to FBWA (orange).

After this successful flight of the lying figure-8 pattern, the same procedure is carried out with the drone flying on a tether. The full flight path with the tether is pictured in Fig. 6.10.

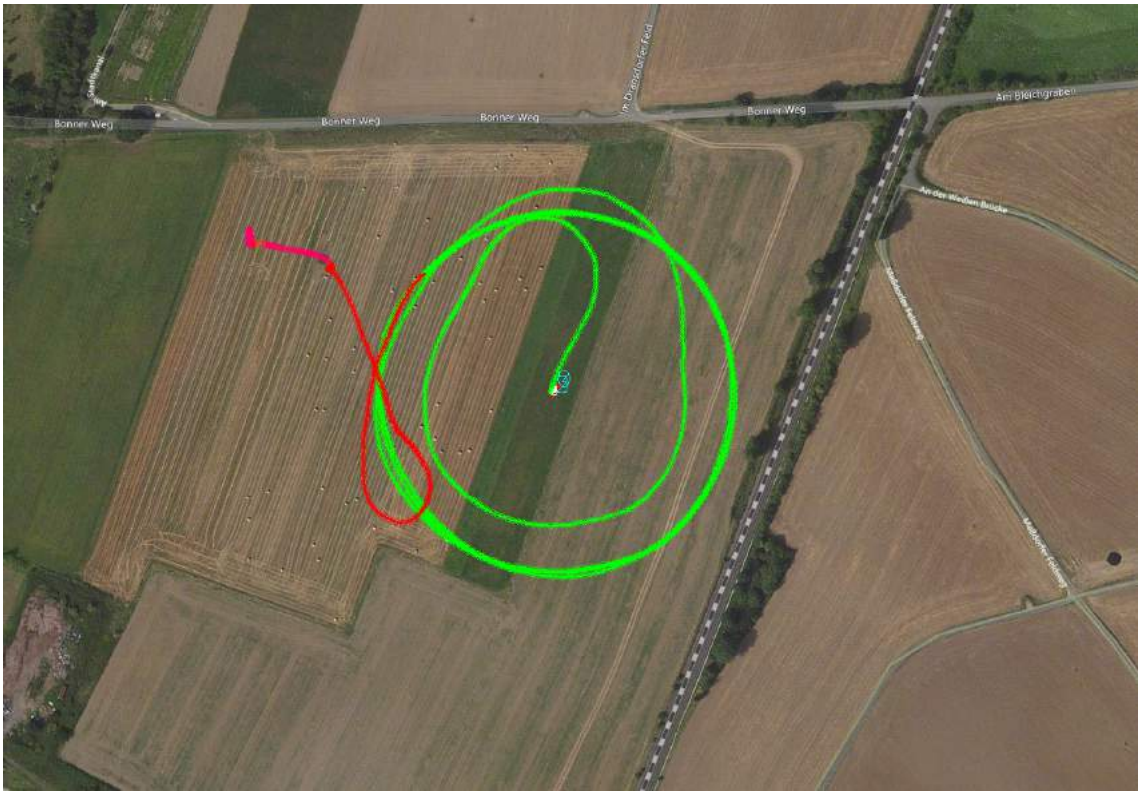


Figure 6.10: Real flight with tether for $\omega = -90°$ and $\sigma = 45°$.

Again the first stage of the flight path is performed in the AUTO mode (green) followed by the WINDDRONE mode (red). As can be seen the drone enters the figure-8 pattern in the desired way and follows the pattern correctly until the second turning path where control over the drone was lost. Even switching to different flight modes could not prevent the drone from crashing.

The analysis of the simulation shows that the airspeed is maximal at the transition from the crossing path to the turning path. Furthermore the demanded roll angle determined by the navigation controller jumps up at this point. The resulting force acting on the drone could destroy the tail, as it is broken after the crash.

The wind drone is meant to be a test-platform, therefore this crash can still be seen as a success. As the drone completes three of four segments of the figure-8 pattern, probably a material failure occurs rather than an error in the software.

CHAPTER 7

Conclusion and Outlook

A model glider plane was successfully transformed into an autonomously flying wind drone that could safely be operated without a tether. The flight by the tether showed a promising approach which will be an auspicious project for the future.

In the scope of this thesis, a commercially available model glider plane was reinforced with carbon fibre, resulting an increase of stability of the styrofoam based structure. Furthermore, the plane was equipped with various sensors to transfer real-time measurements to the autopilot. In order to implement defined flight paths, algorithms, that include physical models and forces, were developed using an open-source autopilot code as a basis.

The autopilot was tested using a flight dynamics model to simulate the motion of the wind drone. Furthermore the performance of the wind drone was analysed under different environmental influences. Considering the results of the simulation, the autopilot was tested on the wind drone in the real world.

The aim of this thesis was to create an open-source test platform which can be copied and improved by the airborne wind energy community. Analyses of the simulation and the real flight are showing that there are a lot of possibilities to upgrade the software as well as the hardware.

A control algorithm which is divided in a navigation along the tether and perpendicular to the tether instead of a lateral and vertical navigation could enhance the performance of the autopilot. With a more powerful autopilot board an optimal control algorithm using the Kalman filter also for control could be implemented.

As the tether drag is a limiting factor of the energy production of airborne wind energy systems, a more detailed simulation of the tether could provide a deeper understanding of its influence on the whole system.

To study the power generated by the wind drone, the algorithm could be expanded by an power cycle path, where the drone pulls the tether to unroll a drum on the ground. A force sensor could determine the tether tension which can then also be used in the control algorithm. Finally, an unwinding and rewinding mechanism could provide further studies about the power generating applicability of the open-source wind drone.

Bibliography

[1] S. Rahmstorf and H. J. Schellnhuber, *Der Klimawandel*, 7th ed., 2012.

[2] A. P. Stott et al.,
Attribution of twentieth century temperature change to natural and anthropogenic causes,
Climate Dynamics **17**.1 (2001), ISSN: 1432-0894,
URL: http://dx.doi.org/10.1007/PL00007924.

[3] U.S. Energy Information Administraion, *International Energy Outlook 2016*,
URL: http://www.eia.gov/forecasts/ieo/pdf/0484(2016).pdf.

[4] K. Marvel, B. Kravitz and K. Caldeira, *Geophysical limits to global wind power*,
Nature Clim. Change **3**.2 (2013), URL: http://dx.doi.org/10.1038/nclimate1683.

[5] U.S. Energy Information Administraion, *Electric Power Monthly*,
URL: https://www.eia.gov/electricity/monthly/pdf/epm.pdf.

[6] C. L. Archer and K. Caldeira, *Global Assessment of High-Altitude Wind Power*,
Energies **2**.2 (2009), ISSN: 1996-1073, URL: http://www.mdpi.com/1996-1073/2/2/307.

[7] EWC: European Weather Consult, URL: http://www.weather-consult.com.

[8] W. Demtröder, *Experimetalphysik 1*, 7th ed., 2015.

[9] J. F. Douglas et al., *Fluid mechanics*, 5th ed., 2005.

[10] R. Brockhaus, W. Alles and R. Luckner, *Flugregelung*, 3rd ed., 2011.

[11] J. Buchholz, *Regelungstechnik und Flugregelung*, 2nd ed., 2010, URL:
http://www.grin.com/de/e-book/82818/regelungstechnik-und-flugregler.

[12] M. L. Loyd, *Crosswind Kite Power*, Journal of Energy **4**.3 (1980),
URL: http://arc.aiaa.org/doi/10.2514/3.48021.

[13] M. Diehl, "Airborne Wind Energy: Basic Concepts and Physical Foundations",
Airborne Wind Energy, ed. by U. Ahrens, M. Diehl and R. Schmehl,
Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, ISBN: 978-3-642-39965-7,
URL: http://dx.doi.org/10.1007/978-3-642-39965-7_1.

[14] D. Vander Lind,
"Analysis and Flight Test Validation of High Performance AirborneWind Turbines",
Airborne Wind Energy, ed. by U. Ahrens, M. Diehl and R. Schmehl,
Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, ISBN: 978-3-642-39965-7,
URL: http://dx.doi.org/10.1007/978-3-642-39965-7_28.

[15] Makani, URL: https://www.google.com/makani.

[16] Ampyx power, URL: https://www.ampyxpower.com.

[17] SkySails, URL: http://www.skysails.info/.

[18] F. Fritz,
"Application of an Automated Kite System for Ship Propulsion and Power Generation",
Airborne Wind Energy, ed. by U. Ahrens, M. Diehl and R. Schmehl,
Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, ISBN: 978-3-642-39965-7,
URL: http://dx.doi.org/10.1007/978-3-642-39965-7_20.

[19] Futura Science, URL: http://www.futura-sciences.com/.

[20] Altaeros energies, URL: http://www.altaerosenergies.com.

[21] C. Vermillion, B. Glass and A. Rein, "Lighter-Than-Air Wind Energy Systems",
Airborne Wind Energy, ed. by U. Ahrens, M. Diehl and R. Schmehl,
Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, ISBN: 978-3-642-39965-7,
URL: http://dx.doi.org/10.1007/978-3-642-39965-7_30.

[22] Sky WindPower, URL: http://www.skywindpower.com/.

[23] EnerKite Flugwindkraftanlagen, URL: http://www.enerkite.de/.

[24] A. L. Herrera-May et al.,
A resonant magnetic field microsensor with high quality factor at atmospheric pressure,
Journal of Micromechanics and Microengineering **19**.1 (2009),
URL: http://stacks.iop.org/0960-1317/19/i=1/a=015016.

[25] R. P. G. Collinson, "Inertial Sensors and Attitude Derivation",
Introduction to Avionics Systems, Dordrecht: Springer Netherlands, 2011,
ISBN: 978-94-007-0708-5, URL: http://dx.doi.org/10.1007/978-94-007-0708-5_5.

[26] F. Chollet and L. Haobing, *A (not so) short introduction to MEMS*, 5.2, 2015,
ISBN: 978-2-9542015-0-4, URL: http://memscyclopedia.org/introMEMS.html.

[27] J. Roth, *Mobile Computing, Grundlagen, Technik, Konzepte*, 2nd ed., dpunkt.verlag, 2005,
ISBN: 3-89864-366-2.

[28] G. Welch and G. Bishop, "An introduction to the Kalman filter. Department of Computer Science, University of North Carolina", Chapel Hill, NC, unpublished manuscript, 2006.

[29] ArduPilot Community, *ArduPilot Autopilot Suite*,
URL: http://ardupilot.org/ardupilot/.

[30] M. van Biezen, *Special Topics - The Kalman Filter*, 2016,
URL: http://www.ilectureonline.com/.

[31] M. Zanon, S. Gros and M. Diehl,
"Model Predictive Control of Rigid-Airfoil Airborne Wind Energy Systems",
Airborne Wind Energy, ed. by U. Ahrens, M. Diehl and R. Schmehl,
Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, ISBN: 978-3-642-39965-7,
URL: http://dx.doi.org/10.1007/978-3-642-39965-7_12.

[32] NuttX, *NuttX Real-time Operating System*,
URL: http://www.nuttx.org/Documentation/NuttX.html.

[33] MAVLink, *Micro Air Vehicle Communication Protocol*, URL: http://mavlink.org.

[34] Aerospaceweb, *Bank Angle and G's*,
URL: http://www.aerospaceweb.org/question/performance/q0146.shtml.

[35] S. Park, J. Deyst and J. P. How, "A new nonlinear guidance logic for trajectory tracking", *AIAA guidance, navigation, and control conference and exhibit*, 2004.

[36] L. S. Brian and L. L. Frank, *Aircraft Control and Simulation*, John Wiley & Sons, Inc., Hoboken, New Jersey (2003).

[37] PX4 Autopilot project, *PX4 Autopilot*, URL: https://pixhawk.org/.

APPENDIX A

Additional Data

A.1 Wind Speed Estimate

The Kalman filter can estimate the wind speed and direction by flying a circular pattern. Figure A.1 illustrates the calibration process. From minute 1 to minute 8 the drone follows a circular pattern and the estimate of the wind speed becomes more accurate.

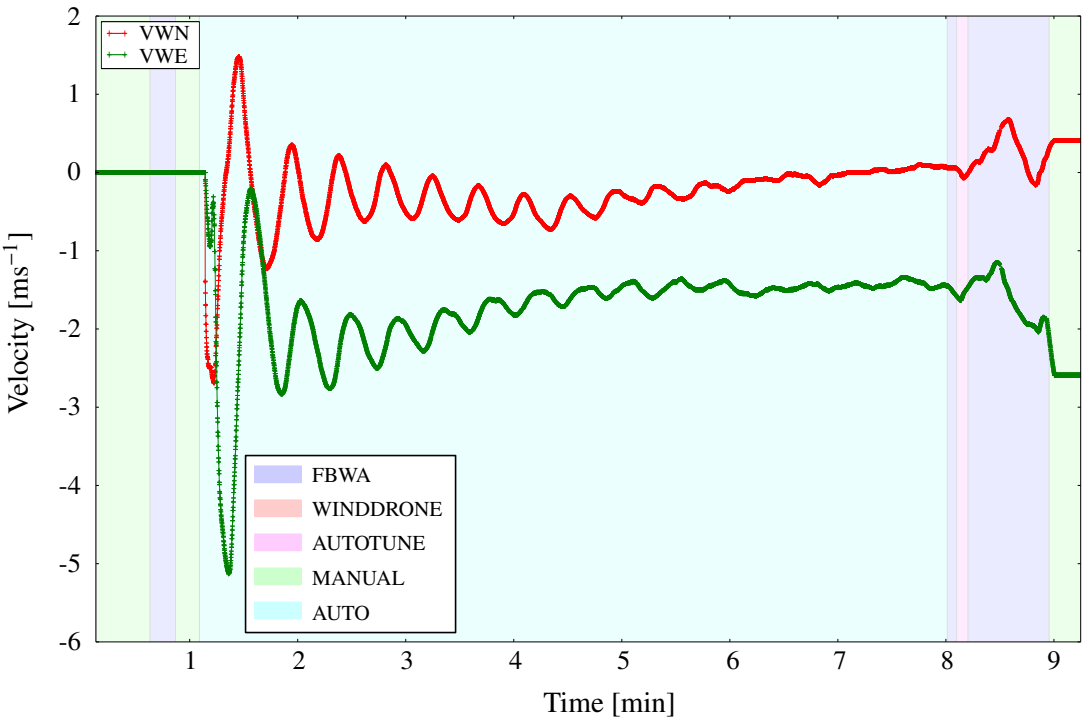


Figure A.1: Wind estimate. The Kalman filter estimates the north component (VWN) and east component (VWE) of the wind speed.

A.2 Flight Path Deviation

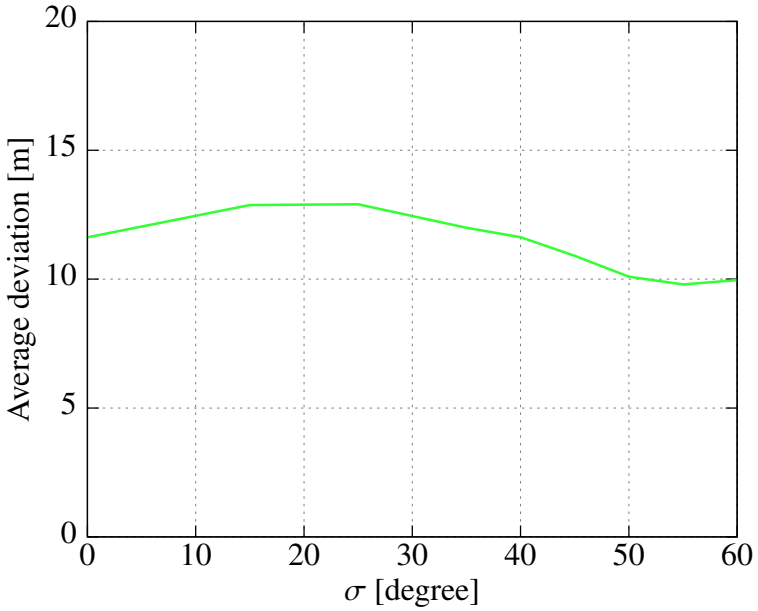


Figure A.2: Average deviation vs sigma at wind speed 4 m s^{-1}.

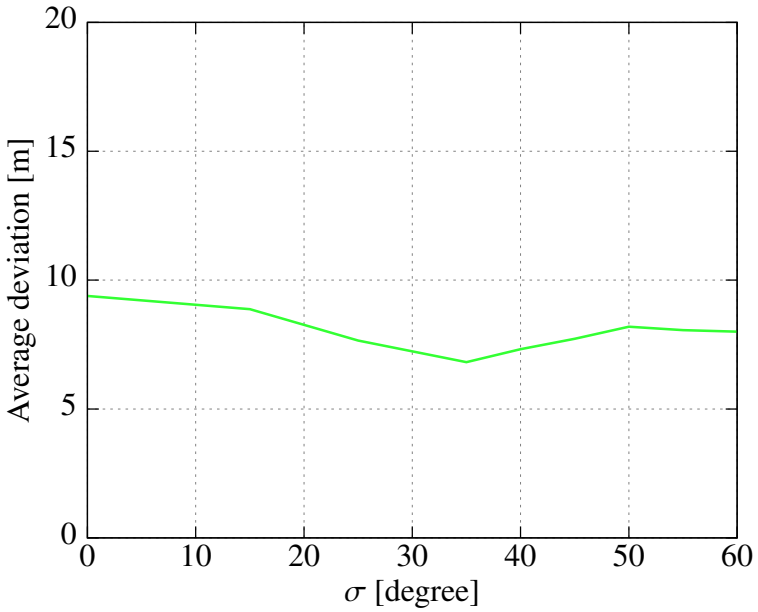


Figure A.3: Average deviation vs sigma at wind speed of 16 m s^{-1}.

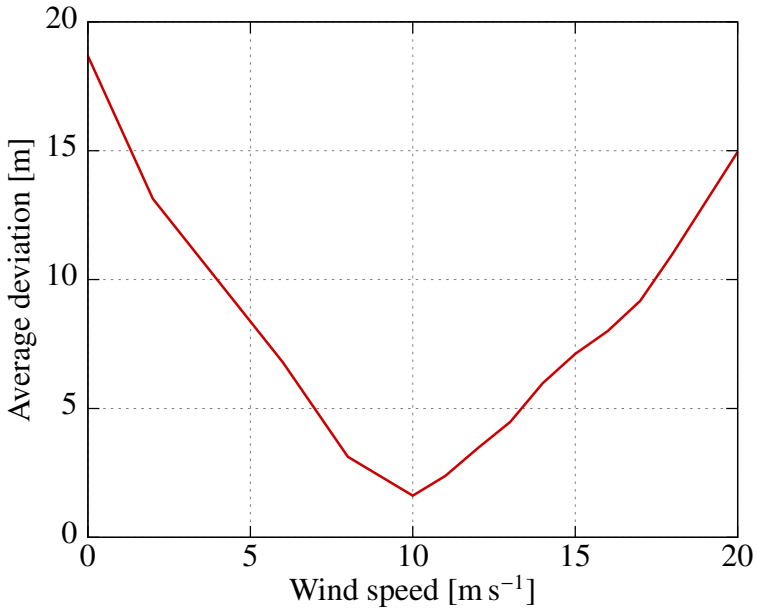


Figure A.4: Average deviation vs sigma at $\sigma = 60°$.

A.3 Airspeed and Throttle

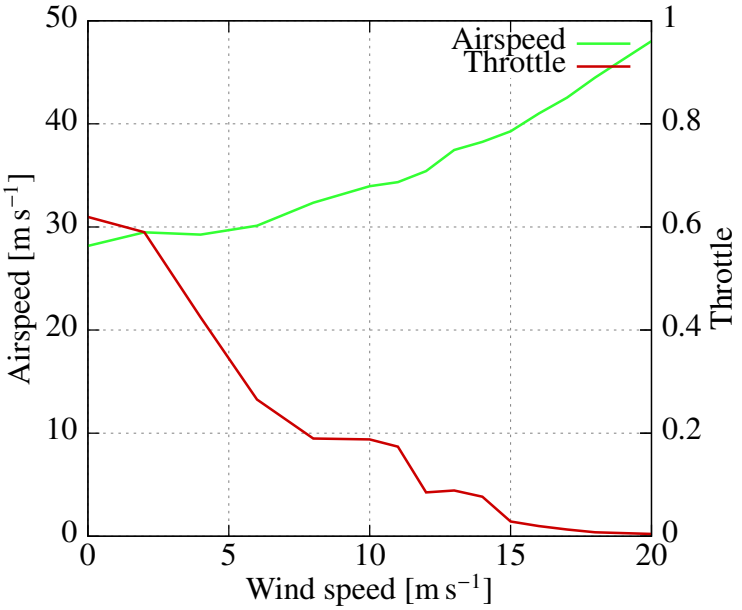


Figure A.5: Airspeed and throttle vs wind speed at $\sigma = 45\,\mathrm{m\,s^{-1}}$.

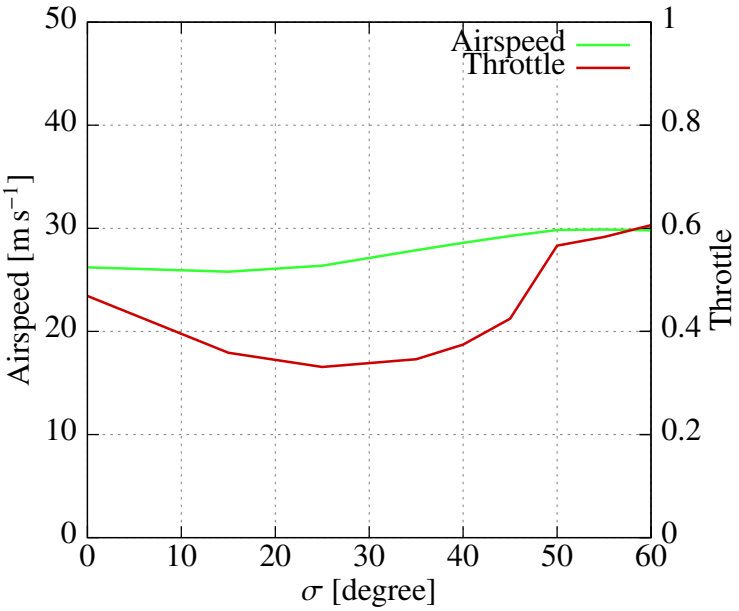


Figure A.6: Airspeed and throttle vs sigma at wind speed of 4 m s^{-1}.

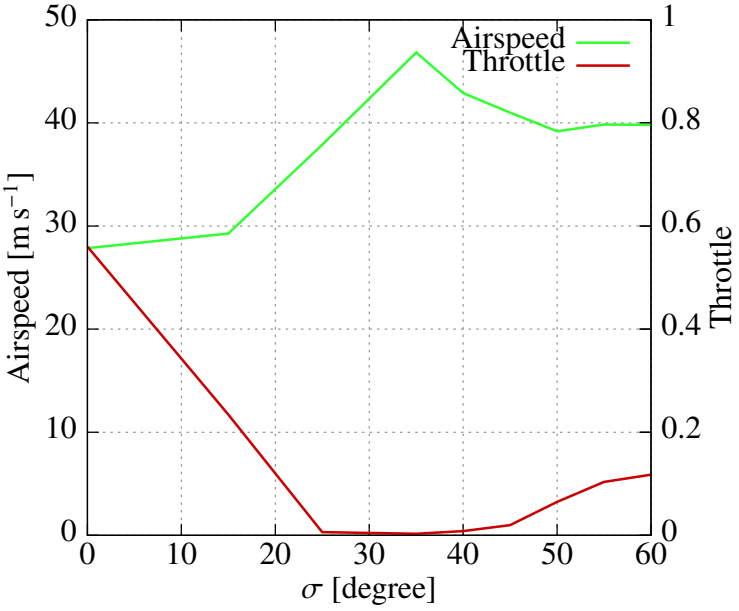


Figure A.7: Airspeed and throttle vs sigma at wind speed of 16 m s^{-1}.

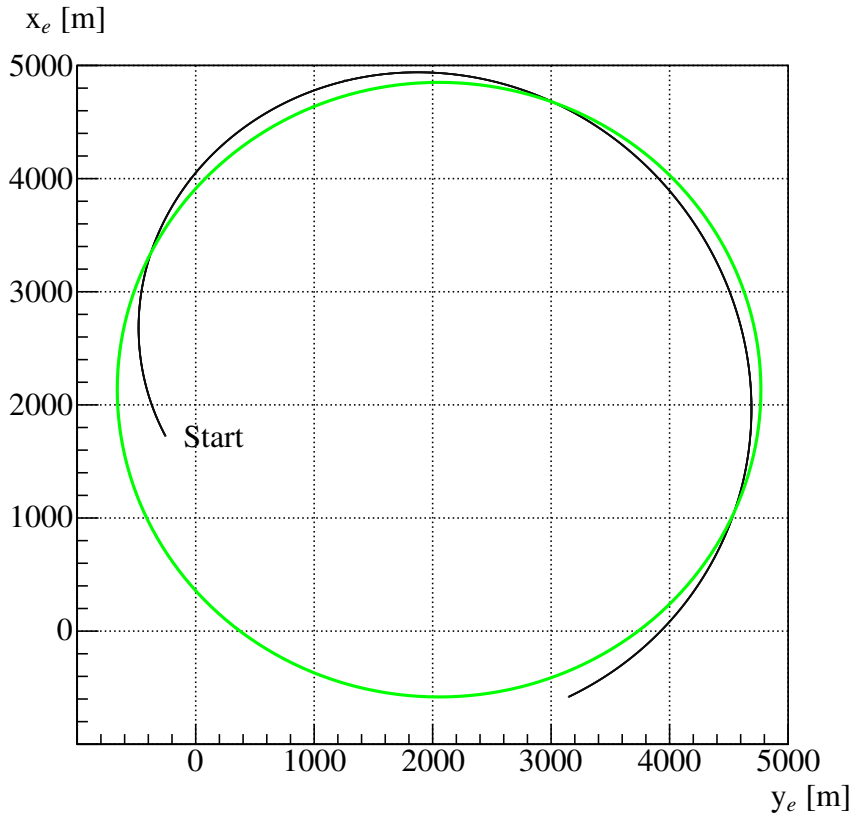


Figure A.8: Level flight with zero roll angle (black). A circle is fitted to the flight path to estimate the radius of the flight path (green). The radius is about 2 700 m.

A.4 Real Flight with Tether

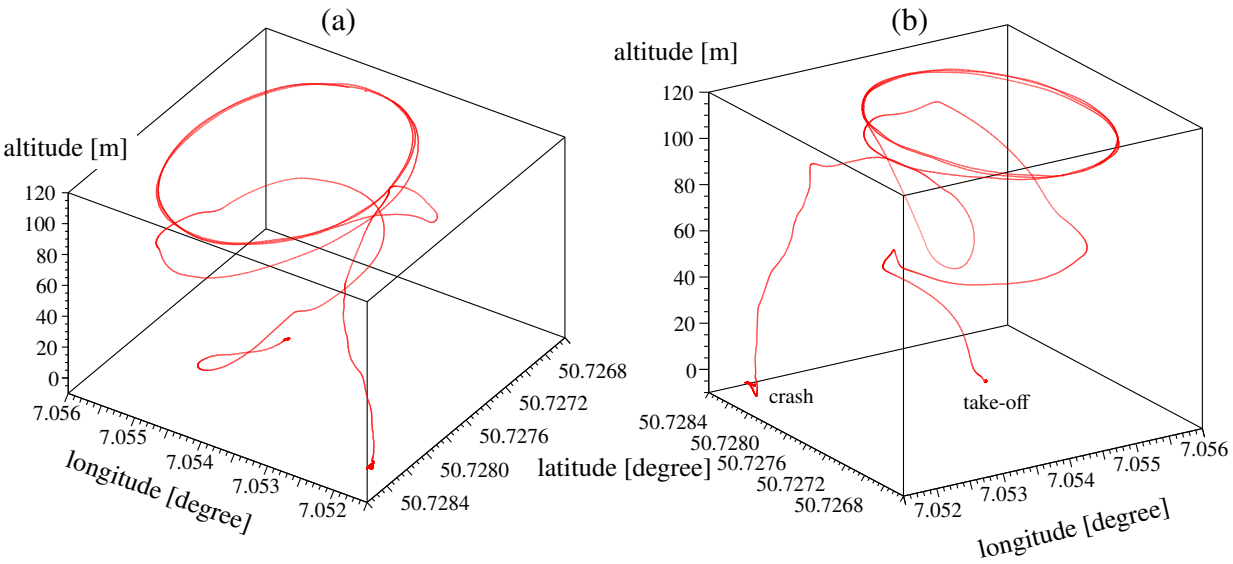


Figure A.9: Real flight with tether, $\sigma = 45°$

APPENDIX B

Derivations

B.1 Bernoulli's Equation

The derivation is taken from [8].
To move the liquid volume ΔV against the pressure p the work

$$\Delta W_1 = F_1 \Delta x_1 = p_1 A_1 \Delta x_1 = p_1 \Delta V_1 \tag{B.1}$$

is needed in the first half of the pipe. And equivalently in the second half with the smaller cross-sectional area the work

$$\Delta W_2 = p_2 \Delta V_2 \tag{B.2}$$

is needed. The kinetic energy of the fluid is given by the following equation:

$$E_{\text{kin}} = \frac{1}{2}\Delta m v^2 = \frac{\varrho}{2} v^2 \Delta V \tag{B.3}$$

For a perfect fluid with no friction the sum of potential and kinetic energy has to be constant.

$$p_1 \Delta V_1 + \frac{\varrho}{2} v_1^2 \Delta V_1 = p_2 \Delta V_2 + \frac{\varrho}{2} v_2^2 \Delta V_2 \tag{B.4}$$

Furthermore applies for incompressible flow $\Delta V_1 = \Delta V_2$. Hence, the Bernoulli equation follows:

$$p_1 + \frac{\varrho}{2} v_1^2 = p_2 + \frac{\varrho}{2} v_2^2 \tag{B.5}$$

B.2 Angle at Center ζ

This section derives the claim that the turning paths are greater than a semicircle, i.e. angle at center $\zeta \geq 180°$. To create a figure-8 pattern lying on a hemisphere, the arc angle A of the crossing paths must be smaller than 90°. Otherwise a fraction of the turning paths would not be on the upper hemisphere, i.e. it would be under the earth's surface. Furthermore the crossing angle Γ should be smaller than 45°.

Without loss of generality, consider a figure-8 pattern created according to Eq. (3.30), (3.31), (3.35) and (3.36), i.e. a figure-8 pattern lying on top of the sphere. If the tangential vectors of the endpoint of

the crossing paths are pointing away from the cross axis, the turning paths are greater than a semicircle.

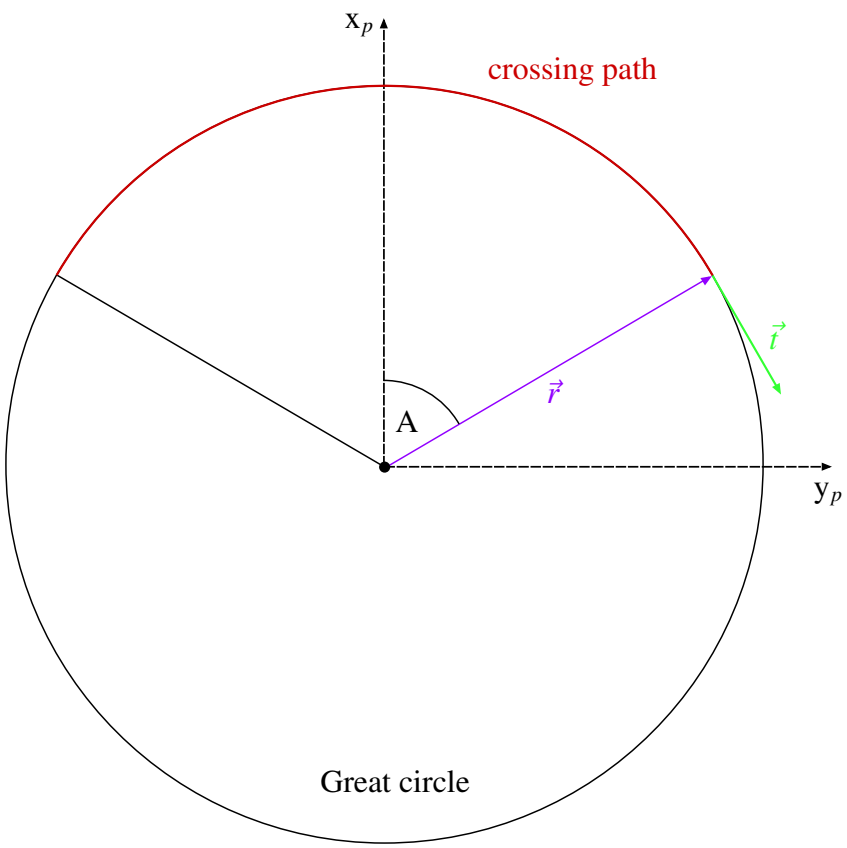


Figure B.1: Tangential vector of an endpoint of a crossing path.

Let's look at the upper right tangential vector. The cross axis of the figure-8 pattern is parallel to the y_e-axis. If the x_e-component of the tangential vector is positive, it points away from the cross axis. Due to the symmetry, the x_e-component of the bottom right tangential vector is negative, i.e. is also pointing away from the cross axis.

The upper right tangential vector in the plane related coordinate system vector yields as shown in Fig. B.1:

$$\vec{t}_p = \begin{pmatrix} -\sin A \\ \cos A \\ 0 \end{pmatrix} \tag{B.6}$$

By multiplying $\vec{t}_p$ with M'_{ep} it is transformed into the earth-fixed coordinate system.

$$\begin{aligned} \vec{t}_e &= M'_{ep}(\psi = -\Gamma, \vartheta = 90^\circ) \cdot \vec{t}_p \\ &= \begin{pmatrix} 0 & \sin\Gamma & \cos\Gamma \\ 0 & \cos\Gamma & -\sin\Gamma \\ -1 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} -\sin A \\ \cos A \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} \sin\Gamma\cos A \\ \cos\Gamma\cos A \\ \sin A \end{pmatrix} \end{aligned} \tag{B.7}$$

As it can be seen in Eq. (B.7), the x_e-component is always positive for the specified ranges of A and Γ.

B.3 Calculation of η

Without loss of generality, consider again a figure-8 pattern which is lying on top of the hemisphere with its cross axis being parallel to the y_e-axis. To create a smooth figure-8 pattern, the tangential vectors of the endpoints of the crossing paths have to lie on the plane which creates the small circle segments.

Due to symmetry reasons, only the upper right tangential vector described in the earth-fixed coordinate system in Eq. (B.7) is considered. By rotating it about η around the x_e-axis the tangential vector becomes parallel to the x_e-y_e-plane. In this manner, η can be calculated:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\eta & \sin\eta \\ 0 & -\sin\eta & \cos\eta \end{pmatrix} \cdot \begin{pmatrix} \sin\Gamma\cos A \\ \cos\Gamma\cos A \\ \sin A \end{pmatrix} = \begin{pmatrix} \sin\Gamma\cos A \\ \cos\eta\cos\Gamma\cos A + \sin\eta\sin A \\ -\sin\eta\cos\Gamma\cos A + \cos\eta\sin A \end{pmatrix} \tag{B.8}$$

Setting the z_e-component to zero yields η:

$$0 = -\sin\eta\cos\Gamma\cos A + \cos\eta\sin A \tag{B.9}$$

$$\Rightarrow \tan\eta = \frac{\sin A}{\cos A\cos\Gamma} \tag{B.10}$$

B.4 Calculation of d

The plane, which creates the turning paths, is a small circle, i.e. $d \neq 0$. It can be calculated with the following idea:

Consider the vector $\vec{r}$ which points from the origin to the upper right endpoint of the crossing path.

$$\vec{r}_p = R\begin{pmatrix} \cos A \\ \sin A \\ 0 \end{pmatrix} \tag{B.11}$$

$$\begin{aligned} \vec{r}_e &= R\begin{pmatrix} 0 & \sin\Gamma & \cos\Gamma \\ 0 & \cos\Gamma & -\sin\Gamma \\ -1 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} \cos A \\ \sin A \\ 0 \end{pmatrix} \\ &= R\begin{pmatrix} \sin\Gamma\sin A \\ \cos\Gamma\sin A \\ -\cos A \end{pmatrix} \end{aligned} \tag{B.12}$$

By rotating $\vec{r}$ about η around x_e-axis the vector then lies in the x_e-z_e-plane. The absolute value of the z_e-component equals the distance d from the plane to the origin.

$$\vec{r}_{e,\mathrm{rot}} = R\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\eta & \sin\eta \\ 0 & -\sin\eta & \cos\eta \end{pmatrix} \cdot \begin{pmatrix} \sin\Gamma\sin A \\ \cos\Gamma\sin A \\ -\cos A \end{pmatrix} = R\begin{pmatrix} \sin\Gamma\sin A \\ \cos\eta\cos\Gamma\sin A - \sin\eta\cos A \\ -\sin\eta\cos\Gamma\sin A - \cos\eta\cos A \end{pmatrix}. \tag{B.13}$$

Therefore the distance from the origin of the hemisphere to the plane which produces the turning paths is given by

$$d = R(\sin\eta\cos\Gamma\sin A + \cos\eta\cos A). \tag{B.14}$$

B.5 Calculation of d_c

The distance d_c between the endpoints of both crossing path can be calculated very easily. The vector $\vec{r}$ to the upper right endpoint is given by Eq. (B.12), the vector to the bottom right endpoint is obtained by replacing $-\Gamma$ with Γ

$$\Rightarrow \vec{r}_e = R\begin{pmatrix} -\sin\Gamma\sin A \\ \cos\eta\cos\Gamma\sin A - \sin\eta\cos A \\ -\sin\eta\cos\Gamma\sin A - \cos\eta\cos A \end{pmatrix}. \tag{B.15}$$

Subtracting one vector from the other one yields a vector pointing from one endpoint to the other endpoint of the crossing path. The absolute value of this vector results in

$$d_c = 2R\sin\Gamma\sin A. \tag{B.16}$$

B.6 Total Transformation Matrix M_{pe}

To create a lying figure-8 pattern which is orientated to the wind direction, first a figure-8 pattern on top of the hemisphere with its cross axis being parallel to the y_e-axis is created. Then it is aligned to the wind direction, i.e. the cross axis being perpendicular to the wind direction, by a rotation about ω around the z_e-axis. Finally, the figure-8 pattern is tilted by a rotation about σ around the rotated y_e-axis.

The idea to obtain the total transformation matrix M_{pe} is to perform an active transformation of the planes which define the figure-8 pattern on top of the hemisphere. The active transformation matrix is given by

$$M_{\text{active}} = \left(M_\sigma \cdot M_\omega\right)^{-1} = M_\omega^{-1} \cdot M_\sigma^{-1} \tag{B.17}$$

The axes of the plane are given by Eq. (3.28) which are, written in matrix form, the same as M'_{ep}. The product $M_{\text{active}} \cdot M'_{ep}$ yields the axes of the plane in the end position. The inverse of this product is the passive transformation matrix M_{pe}. The result is shown on the next page in landscape format because the matrix becomes quite large. The matrix can be simplified a bit since the angles ψ and ϑ are known for each segment of the figure-8 pattern (see Eq. (3.30), (3.31), (3.35) and (3.36)).

$$M_{pe} = \begin{pmatrix} \cos\omega\cos\sigma\cos\vartheta\cos\psi - \sin\omega\cos\vartheta\sin\psi - \cos\omega\sin\sigma\sin\vartheta & \sin\omega\cos\sigma\cos\vartheta\cos\psi + \cos\omega\cos\vartheta\sin\psi - \sin\omega\sin\sigma\sin\vartheta & -\sin\sigma\cos\vartheta\cos\psi - \cos\sigma\sin\vartheta \\ -\cos\omega\cos\sigma\sin\psi - \sin\omega\cos\psi & -\sin\omega\cos\sigma\sin\psi + \cos\omega\cos\psi & \sin\sigma\sin\psi \\ \cos\omega\cos\sigma\sin\vartheta\cos\psi - \sin\omega\sin\vartheta\sin\psi + \cos\omega\sin\sigma\cos\vartheta & \sin\omega\cos\sigma\sin\vartheta\cos\psi + \cos\omega\sin\vartheta\sin\psi + \sin\omega\sin\sigma\cos\vartheta & -\sin\sigma\sin\vartheta\cos\psi + \cos\sigma\cos\vartheta \end{pmatrix} \tag{B.18}$$

The simplified matrices for each segment are

$\psi = \Gamma$, $\vartheta = 90°$

$$M_{pe,\mathrm{cross1}} = \begin{pmatrix} -\cos\omega\sin\sigma & -\sin\omega\sin\sigma & -\cos\sigma \\ -\cos\omega\cos\sigma\sin\Gamma - \sin\omega\cos\Gamma & -\sin\omega\cos\sigma\sin\Gamma + \cos\omega\cos\Gamma & \sin\sigma\sin\Gamma \\ \cos\omega\cos\sigma\cos\Gamma - \sin\omega\sin\Gamma & \sin\omega\cos\sigma\cos\Gamma + \cos\omega\sin\Gamma & -\sin\sigma\cos\Gamma \end{pmatrix} \tag{B.19}$$

$\psi = -\Gamma$, $\vartheta = 90°$

$$M_{pe,\mathrm{cross2}} = \begin{pmatrix} -\cos\omega\sin\sigma & -\sin\omega\sin\sigma & -\cos\sigma \\ \cos\omega\cos\sigma\sin\Gamma - \sin\omega\cos\Gamma & \sin\omega\cos\sigma\sin\Gamma + \cos\omega\cos\Gamma & -\sin\sigma\sin\Gamma \\ \cos\omega\cos\sigma\cos\Gamma + \sin\omega\sin\Gamma & \sin\omega\cos\sigma\cos\Gamma - \cos\omega\sin\Gamma & -\sin\sigma\cos\Gamma \end{pmatrix} \tag{B.20}$$

$\psi = 90°$, $\vartheta = \eta$

$$M_{pe,\mathrm{turn1}} = \begin{pmatrix} -\sin\omega\cos\eta - \cos\omega\sin\sigma\sin\eta & \cos\omega\cos\eta - \sin\omega\sin\sigma\sin\eta & -\cos\sigma\sin\eta \\ -\cos\omega\cos\sigma & -\sin\omega\cos\sigma & \sin\sigma \\ -\sin\omega\sin\eta + \cos\omega\sin\sigma\cos\eta & \cos\omega\sin\eta + \sin\omega\sin\sigma\cos\eta & \cos\sigma\cos\eta \end{pmatrix} \tag{B.21}$$

$\psi = -90°$, $\vartheta = \eta$

$$M_{pe,\mathrm{turn2}} = \begin{pmatrix} \sin\omega\cos\eta - \cos\omega\sin\sigma\sin\eta & -\cos\omega\cos\eta - \sin\omega\sin\sigma\sin\eta & -\cos\sigma\sin\eta \\ \cos\omega\cos\sigma & \sin\omega\cos\sigma & -\sin\sigma \\ \sin\omega\sin\eta + \cos\omega\sin\sigma\cos\eta & -\cos\omega\sin\eta + \sin\omega\sin\sigma\cos\eta & \cos\sigma\cos\eta \end{pmatrix} \tag{B.22}$$

B.7 Calculation of Roll Angle Φ

Without loss of generality assume a yaw angle $\Psi = 0$. The transformation matrix M_{ed} then yields according to Eq. (3.16)

$$M_{ed} = \begin{pmatrix} \cos\Theta & \sin\Phi\sin\Theta & \cos\Phi\sin\Theta \\ 0 & \cos\Phi & -\sin\Phi \\ -\sin\Theta & \sin\Phi\cos\Theta & \cos\Phi\cos\Theta \end{pmatrix}. \tag{B.23}$$

The total lift vector described in the earth-fixed coordinate system is

$$\vec{L}_e = M_{ed}\begin{pmatrix} 0 \\ 0 \\ -L \end{pmatrix}_d = L\begin{pmatrix} -\cos\Phi\sin\Theta \\ \sin\Phi \\ -\cos\Phi\cos\Theta \end{pmatrix}_e. \tag{B.24}$$

The vertical component of the lift has to be equal weight

$$L\cos\Phi\cos\Theta = mg \tag{B.25}$$

and the lateral component equals the centripetal force

$$L\sin\Phi = \frac{a_{\text{lat}}}{r}, \tag{B.26}$$

which results in

$$\Phi = \arctan\frac{a_{\text{lat}}\cos\Theta}{g} \tag{B.27}$$

B.8 Centripetal Acceleration

The centripetal acceleration points from the boarder of the circle to its center. The magnitude is $\frac{v^2}{r}$, with the circle radius r and the tangential velocity v. The easiest way to describe the centripetal acceleration vector $\vec{a}$ is in the plane related coordinate system:

$$\vec{a}_p = -\frac{v^2}{r}\begin{pmatrix} \cos\alpha \\ \sin\alpha \\ 0 \end{pmatrix}_p. \tag{B.28}$$

To describe the centripetal acceleration vector in the flight-path coordinate system, the following transformation has to be done

$$\vec{a}_k = M_{ke}M_{ep}\vec{a}_p. \tag{B.29}$$

The first step is to transform $\vec{a}_p$ to $\vec{a}_e$.

$$\vec{a}_e = M_{ep}\vec{a}_p = -\frac{v^2}{r}\begin{pmatrix} M_{ep}^{11}\cos\alpha + M_{ep}^{12}\sin\alpha \\ M_{ep}^{21}\cos\alpha + M_{ep}^{22}\sin\alpha \\ M_{ep}^{31}\cos\alpha + M_{ep}^{32}\sin\alpha \end{pmatrix}. \tag{B.30}$$

The upper index indicates the matrix element. The first number stands for the row and the second number for the column. Transforming $\vec{a}_e$ to $\vec{a}_k$ yields

$$\vec{a}_k = M_{ke}\vec{a}_e = -\frac{v^2}{r}\begin{pmatrix} \dots \\ -\sin\chi(M_{ep}^{11}\cos\alpha + M_{ep}^{12}\sin\alpha) + \cos\chi\cdot(M_{ep}^{21}\cos\alpha + M_{ep}^{22}\sin\alpha) \\ \dots \end{pmatrix}. \tag{B.31}$$

The navigation control algorithm uses only the y-component, so the others are not calculated. To calculate χ the trajectory must be known. The unit trajectory velocity is the tangential vector on the circle which can be easily described in the plane related coordinate system.

$$\vec{t}_p = \begin{pmatrix} -\sin\alpha \\ \cos\alpha \\ 0 \end{pmatrix}_p. \tag{B.32}$$

In the earth-fixed coordinate system the unit trajectory vector yields in compact notation

$$\vec{t}_e = M_{ep}\vec{t}_p = \begin{pmatrix} -M_{ep}^{11}\sin\alpha + M_{ep}^{12}\cos\alpha \\ -M_{ep}^{21}\sin\alpha + M_{ep}^{22}\cos\alpha \\ -M_{ep}^{31}\sin\alpha + M_{ep}^{32}\cos\alpha \end{pmatrix}. \tag{B.33}$$

χ can be calculated with Eq. (3.20)

$$\chi = \arctan\frac{-M_{ep}^{21}\sin\alpha + M_{ep}^{22}\cos\alpha}{-M_{ep}^{11}\sin\alpha + M_{ep}^{12}\cos\alpha} = \arctan Z. \tag{B.34}$$

Inserting this into Eq. (B.31) yields

$$\vec{a}_k = M_{ke}\vec{a}_e = -\frac{v^2}{r}\begin{pmatrix} \dots \\ -\frac{Z}{\sqrt{Z^2+1}}(M_{ep}^{11}\cos\alpha + M_{ep}^{12}\sin\alpha) + \frac{1}{\sqrt{Z^2+1}}\cdot(M_{ep}^{21}\cos\alpha + M_{ep}^{22}\sin\alpha) \\ \dots \end{pmatrix}. \tag{B.35}$$

By inserting the matrix elements from M_{ep} which is calculated in Eq. (B.18), the result in Eq. (4.13) is reconstructed.

List of Figures

List of Tables