Master of Sciences Thesis

Fault-Tolerant Design of a Pumping Kite Power Flight Control System

Felix Friedl, BSc.

February 2015

FH J HsKa stw TUD

FH | JOANNEUM
University of Applied Sciences

TU Delft
Delft University of Technology

Master of Science Thesis

Fault-Tolerant Design of a Pumping Kite Power Flight Control System

Submitted by:	Felix Friedl, BSc.	
Registration Number:	1210588009	
Academic Supervisors:	Dr.techn. Bernd Messnarz	FH Joanneum
	Dr.-Ing. Roland Schmehl	Delft University of Technology
Date of Submission:	February 09, 2015	

For obtaining the degree of
Master of Science in Engineering (MSc.)
at FH Joanneum University of Applied Sciences
Institute of Aviation - Aerospace Engineering

FH | JOANNEUM
University of Applied Sciences

The image on the cover page was created using a series of individual photographs taken by R. Schmehl on November 8th, 2014. It cannot be guaranteed that the depicted flight path matches the actual one.

Declaration
of
Academic Honesty

I hereby affirm in lieu of an oath that the present master thesis entitled

"Fault-Tolerant Design of a Pumping Kite Power Flight Control System"

has been written by myself without the use of any resources other than those indicated, quoted and referenced. Furthermore I declare that this thesis has not been submitted in any form for another degree or diploma at any university or other institution of tertiary education.

Delft, The Netherlands
February 09, 2015

Felix Friedl

Abstract

This master thesis results from the project *KitePower 2.0*, a cooperation of Delft University of Technology and Hochschule Karlsruhe University of Applied Sciences. The aim of this project is to increase the technology readiness level of a system developed in Delft for converting the kinetic energy of wind into electrical energy. To do so, a tethered airfoil is used to drive a generator on the ground by reeling the tether off a drum while flying the foil in crosswind patterns in the downwind area of the ground station. The introductory chapter of the document at hand takes a closer look at this concept and briefly discusses its advantages and disadvantages.
The subsequent part deals with the individual system components in detail. A special focus is put on certain software components such as the flight control system, the realtime simulation environment and its mathematical models.
After their introduction, the most important components are assessed regarding the effect on the entire system in case of their failure. For this purpose a Failure Mode and Effect Analysis as well as a Fault Tree Analysis are performed. On the basis of these methods the faults and failures of further interest for this thesis are defined.
In order to enable the system to recognize certain failures and initiate appropriate countermeasures a health supervisor is implemented and its abilities are discussed. Furthermore its two core classes are presented and their methods for fault detection are described. Another major part of this thesis is the development of the above mentioned automatic countermeasures. Chapter 5 presents these procedures and deals with their behavior in detail.
Finally using simulations the health supervisor's ability to detect failures as well as the countermeasure's effectiveness is discussed. The very last chapter draws a conclusion and gives an outlook to possible future developments.

Kurzfassung

Die vorliegende Masterarbeit entstand im Zuge des Projekts *KitePower 2.0*, einer Kooperation der Technischen Universität Delft und der Hochschule Karlsruhe. Ziel dieses Projektes ist es, den Technologiereifegrad eines in Delft entwickelten Systems zur Gewinnung elektrischer Energie aus Windkraft zu erhöhen. Dabei wird mittels eines bodengebundenen Fluggerätes, welches im Lee einer Bodenstation auf halbwind Kurs ein Seil von einer Winde abspult, ein Generator betrieben. Im einleitenden Teil der Arbeit wird dieses generelle Prinzip näher erläutert und kurz auf dessen Vor- und Nachteile eingegangen.
Der darauffolgende Abschnitt beschäftigt sich im Detail mit den einzelnen Systemkomponenten, wobei besonderes Augenmerk auf Softwarekomponenten wie die Flugregelung und die Echtzeit-Simulationsumgebung sowie dessen mathematische Modelle gelegt wird.
Im Anschluss werden die wichtigsten Komponenten bezüglich der Auswirkung auf das Gesamtsystem im Falle ihres Versagens untersucht. Zu diesem Zweck wird sowohl eine Failure Mode and Effect Analysis als auch eine Fault Tree Analysis durchgeführt. Auf Basis dieser beiden Methoden werden daraufhin jene Fehler definiert, die im Zuge der weiteren Arbeit von Bedeutung sind.
Um dem System zu ermöglichen, etwaige Fehlersituationen zu erkennen und flexibel darauf zu reagieren wird ein Health Supervisor implementiert und dessen Fähigkeiten werden klar abgegrenzt. Außerdem wird im Detail auf die zwei Hauptbestandteile des Health Supervisors eingegangen und die Methoden zum Erkennen von unerwünschten Zuständen werden erläutert.
Ein wesentlicher Bestandteil der vorlegenden Arbeit ist es auch Gegenmaßnahmen für bestimmte Fehlerszenarien zu entwickeln. In Kapitel 5 wird auf diese automatisierten Verfahren eingegangen und deren Verhalten im Detail beleuchtet.
Anhand von durchgeführten Simulationen werden schließlich sowohl die Fähigkeiten des Health Supervisors zum Erkennen von Fehlern, sowie die Wirksamkeit der eingeleiteten Gegenmaßnahmen diskutiert. Schlussendlich werden die wichtigsten Schritte noch einmal zusammengefasst und ein Ausblick auf mögliche zukünftige Entwicklungen gegeben.

Acknowledgements

Although only my name appears on the cover page of this thesis, many more people contributed to this work. Herewith I want to express my gratitude to the ones who made this possible and brought me to this point in my life.

First of all I want to acknowledge Dr.-Ing. Roland Schmehl for providing me with this unique research opportunity. Thank you for supporting my ideas and yet constructively guiding this project. My gratitude also goes to Dipl.-Ing. Dr.techn. Bernd Messnarz for his encouragement and supportive reasonability throughout my stay in Delft.
Furthermore I want to express my gratitude to my day-to-day supervisor Uwe Fechner, MSc. not only for his exemplary passion and determination for open source software but also for entrusting me with a lot of freedom in my work.

Great thanks shall also go to the entire KitePower 2.0 team and my fellow students from 6.08. I thankfully look back not only to your amicable support and several hours, not to mention the weekends, of stimulating discussions but also to the fun time we had improving our practical kite power skills.

Above all I want to express my deepest gratitude towards my immediate family. You have been a faithful and reliable source of emotional and financial support. All this has only been possible because of your loving encouragement.

Finally I want to thank Prof. Dr.-Ing. Rüdiger Haas and Prof. Dr.-Ing. Matthias Stripf from Steinbeis Institute and Karlsruhe University of Applied Sciences for funding parts of this work.

Contents

List of Figures

List of Tables

Nomenclature

Latin Symbols

B	Wing loading	$\mathrm{kg\,m^{-2}}$
d	Damping constant	$\mathrm{kg\,s^{-1}}$
$E_{\mathrm{K,\,w}}$	Kinetic energy of a given mass of air (wind)	J
F	Absolute value of a force	N
h	Height above ground	m
k	Spring constant	$\mathrm{kg\,s^{-2}}$
l_S	Length of a tether segment	m
l_T	Length of the tether	m
m_P	Particle mass	kg
m_{air}	Mass of the air	$\mathrm{m\,s^{-1}}$
n_S	Number of tether segments	–
P_{m}	Mechanical power	W
P_{w}	Power available in a given region of the atmosphere in the form of wind	W
r_d	Radius of the winch drum	m
U	Voltage supplied by batteries	V
u_{D}	Depower command	–
u_{S}	Steering command	–
v_r	Tether reel-out velocity	$\mathrm{m\,s^{-1}}$
WPD	Wind power density	$\mathrm{W\,m^{-2}}$
$\mathbf{D}$	Aerodynamic drag force	N
$\mathbf{F}$	Force vector	N
$\mathbf{G}$	Weight force vector	N
$\mathbf{g}$	Gravitational acceleration	$\mathrm{m\,s^{-2}}$
$\mathbf{K}$	Position of the kite projected onto unit sphere $\mathbb{S}^2$	m
$\mathbf{L}$	Aerodynamic lift force	N
$\mathbf{O}$	Tether exit point of swivel and origin of reference frame $(\cdot)_W$	m
$\mathbf{P}$	Kite position vector	m
$\mathbf{p}_{P,\,i}$	Position of particle i	m
$\mathbf{R}$	Residual vector	–

$\mathbf{v}_\mathrm{a}$	Apparent wind velocity (airspeed)	$\mathrm{m\,s^{-1}}$
$\mathbf{v}_\mathrm{w}$	Prevailing wind speed	$\mathrm{m\,s^{-1}}$
$\mathbf{v}_x$	Velocity vector with additional information x	$\mathrm{m\,s^{-1}}$
$\mathbf{Y}$	Particle system state vector	–
$\mathcal{C}_\alpha$	Cosine of α	–
$\mathcal{S}_\alpha$	Sine of α	–
$\mathbb{N}_S$	Direction towards small earth north pole	–
$\mathbb{P}^2$	Sphere with a radius of $\|(\mathbf{P})_W\|$ and its center at the tether exit point $\mathbf{O}$	–
$\mathbb{S}^2$	Unit sphere with its center at tether exit point $\mathbf{O}$ (small earth)	–
$\mathcal{T}_\mathbf{K}\mathbb{S}^2$	Tangent plane at point $\mathbf{K} \in \mathbb{S}^2$	–
${}_X\mathbb{T}_Y$	Transformation matrix from reference frame $Y \to X$	–

Greek Symbols

α	Angle of attack or empirical exponent in wind profile power law	$^\circ$ (deg) or –
γ	Flight path angle	rad
Ξ	System features	–
γ_W	Downwind direction ground	rad
η	Elevation angle	$^\circ$ (deg)
η_p	Pumping efficiency	–
θ	Pitch angle	rad
λ_{GB}	Ground station gear box ratio	–
ξ	Azimuth angle	$^\circ$ (deg)
ρ	Density of the air	$\mathrm{kg\,m^{-3}}$
ϕ	Roll angle	rad
χ	Course angle	rad
ψ	Yaw angle (heading)	rad
ω_d	Rotational velocity of the drum	$\mathrm{rad\,s^{-1}}$

Subscripts

$(\cdot)_{EG}$	Earth-groundstation reference frame
$(\cdot)_K$	Kite reference frame
$(\cdot)_S$	Local reference frame of tangent plane $\mathcal{T}_\mathbf{K}\mathbb{S}^2$ (small earth)
$(\cdot)_W$	Wind reference frame
$_0$	Initial condition
$_\mathcal{R}$	Radial component
$_\mathcal{T}$	Tangential component
$_{\mathrm{A,\ B,\ C,\ D}}$	Particles of four point kite model
$_{\mathrm{aero}}$	Aerodynamic

$_{des}$	Desired value (e.g. calculated by FCS)
$_{SW}$	Switch condition
$_{S}$, $_{\mathbb{S}^2}$	Referring to the small earth analogy
$_{W}$	Wind
$_{i}$	i-th term of a sequence
$_{K}$	Kite
$_{P}$	Particle
$_{S}$	Tether segment
$_{T}$	Tether
$_{KCU}$	Referring to kite control unit

Superscripts

*	Prediction

Abbreviations

AGL	Above ground level
AWE	Airborne Wind Energy
FCS	Flight control system
FMEA	Failure Mode and Effect Analysis
FTA	Fault Tree Analysis
GNSS	Global navigation satellite system
GUI	Graphical user interface
H/W	Hardware
IMU	Inertial measurement unit
KCU	Kite Control Unit
KPS	Kite Power System
LCOE	Levelized cost of energy
LEI	Leading Edge Inflatable
NASA	National Aeronautics and Space Administration (USA)
NED	North east down reference frame
S/W	Software
SISO	Single-input single-output
TUD	Delft University of Technology
WPD	Wind Power Density

Introduction

1

This first chapter points out the motivation for airborne wind energy and briefly introduces important terminology. In addition, it presents the framework in which this thesis evolved and it summarizes what to expect from the document at hand.

Energy is a critical resource for every society. Its availability and cost determine the society's economic wealth and prosperity for a large part. After fossil fuels were discovered as an easily accessible form of energy in the 19^{th} century the world's population increased almost sevenfold, while the average per capita income rose more than tenfold by the year 2000 [2].

However as mankind's consumption of energy keeps rising continuously and fossil fuel sources are limited, the ease of its accessibility has declined in recent decades. Not only the increase in price but also the negative environmental effects have lowered the public's acceptance of fossil fuels considerably. As a result the demand for sustainable energy has extensively grown. The global new investment in renewable power and fuels has increased by 27.3 % from 2009 to 214 10^9 USD in 2013 [29]. Next to solar and geothermal power, biomass and -fuel and many more, especially the kinetic energy of wind is of significant interest as a source of sustainable energy.

This is however not a new idea. The first documented construction converting the kinetic energy of wind dates back to around 1750 B.C. [19]. In the 17^{th} and 18^{th} century windmills were widely used as source for mechanical energy for e.g. grinding corn or pumping water. As the design of wind mills has evolved in recent decades, they are now mostly used for harvesting electrical energy. In 2013 the globally installed wind power capacity reached 318 10^6 W [30].

The potential of conventional wind mills is however limited. The principles of conservation of mass and momentum prevent an ideal wind turbine from extracting more than

around 60%[1] of the kinetic energy of the wind. Additionally, as the square-cube law makes larger structures increase in cost tremendously, modern wind mills have almost reached their economically feasible maximum size.

History has shown that the upcoming stagnation of an evolution demands for a revolution. Delft University of Technology (TUD) contributes to this revolution with extensive research in the field of airborne wind energy (AWE) systems.

1.1 Motivation for AWE and KPSs

The power P_W available in a given part of the atmosphere in the form of wind can be derived via the wind's kinetic energy $E_{\mathrm{K,\,w}}$ and is often expressed as wind power density (WPD):

$$E_{\mathrm{K,\,w}} = \frac{m_{air}\,\|\mathbf{v}_{\mathrm{w}}\|^2}{2} \quad \Rightarrow \quad P_{\mathrm{w}} = \frac{\rho_{air}\,A\,l\,\|\mathbf{v}_{\mathrm{w}}\|^2}{2\,t} \quad \Rightarrow \quad WPD = \frac{\rho_{air}\,\|\mathbf{v}_{\mathrm{w}}\|^3}{2} \tag{1.1}$$

As wind in higher altitudes is steadier and reaches significantly higher velocities than wind at ground level [3] the tower height of conventional wind mills greatly affects their power output and capacity factor (cf. equation (1.1)). Besides other limits, structural aspects restrict the tower's economically feasible height to a maximum of around 200 m [31]. AWE systems replace these expensive towers by light-weight tethers. This allows significantly higher regions and thus stronger and steadier winds to be reached. Additionally the use of tethers instead of towers makes AWE systems more suitable for offshore (especially deep-sea) applications in comparison to conventional wind mills and it vastly reduces the material involved. All in all AWE is believed to provide the future potential to a sustainable source of energy with a highly competitive levelized cost of energy (LCOE).

Back in 1980 Miles Loyd calculated, that a wing of 576 m^2 surface area can achieve an average power output of 6.7 10^6 W in crosswind operation at a wind speed of 10 ms^{-1} [25]. This idea has been picked up by a lot of institutions world wide. An overview of research and development activities in 2013 is shown in figure 1.1. Other than the team at *Altaeros Energies*, which uses a helium filled shell to stationarily lift a wind turbine to high altitudes, most projects employ tethered airfoils that dynamically fly in the downwind area of a ground station. Within the latter two main philosophies have evolved. Systems using airborne wind turbines, e.g. *Google Makani*, can be distinguished from systems that use one or more ground based generators. Obviously concepts that feature a generator on the ground have a big advantage of not having to lift heavy turbines into the air and the generated electrical power does not have to be transmitted from the airborne device to the ground. These systems however have to operate in so-called pumping cycles which comprise generation[2] and retraction phases. Thus energy cannot be converted continuously, but a certain amount of the energy gained needs to be fed back into the system in order to retract the airfoil as the maximum tether length is reached.

The amount of different airfoil concepts developed for generating a traction force from aerodynamic lift is numerous. In a nutshell they can be categorized as leading edge

[1] According to Betz's law for non-ducted wind turbines [5]

[2] Though nonsense from a physical point of view, this term has established and will be used also in this thesis.

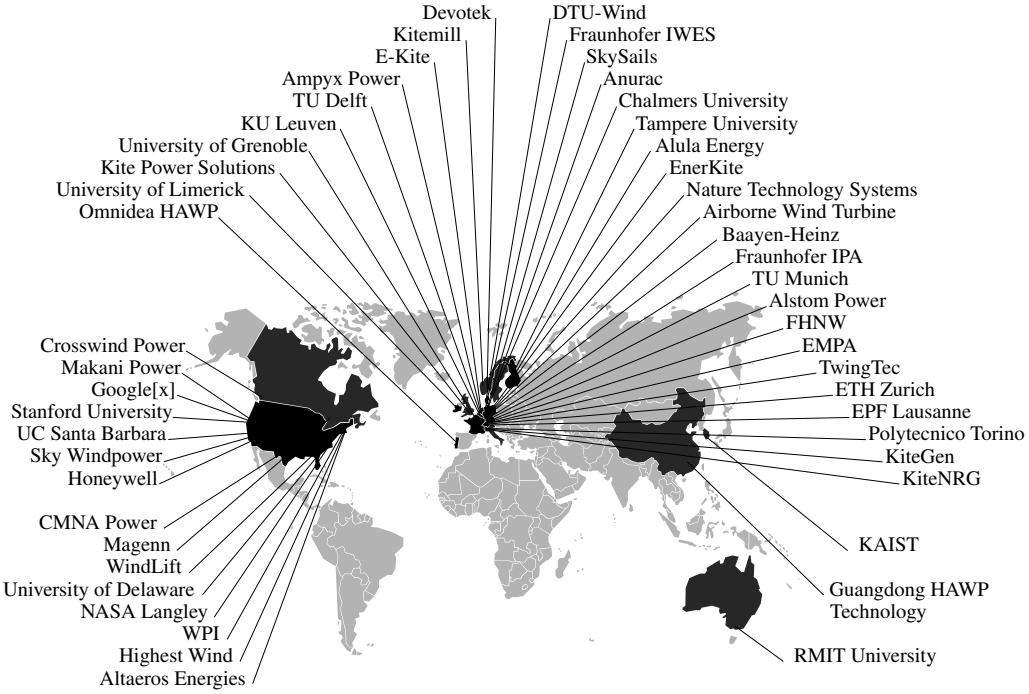


Figure 1.1: AWE research and development activities in 2013 by country and team [1]

inflatable (LEI), ram-air, semi-rigid and rigid wing kites. A more detailed description of the operating principle and the main components of the Kite Power System (KPS) developed by *Delft University of Technology* is given in chapter 2.

1.2 Terminology

In order to contribute to a common understanding, the following section shall give a short introduction to the terminology that has evolved for KPSs as well as for fault-tolerant applications.

1.2.1 Kite terminology

Using the ground station as a reference, **upwind** or **windward** is the direction the wind comes from whereas **downwind** or **lee side** is the direction the wind blows towards. The kite can be flown in a controlled manner in the downwind area of the ground station only. This quarter spheric region with the ground station in the sphere's center as shown in figure 1.2 is commonly denoted as the **wind window**. The angle between the downwind direction and the kite's perpendicular projection onto the earth's surface is called **azimuth** (ξ), whereas the vertical angle between ground and kite is termed **elevation** (η).

The highest point of the wind window is the **zenith**. It is part of the **edge** of the wind window that can theoretically only be reached by a kite with an infinite glide ratio. Positions along the edge of the wind window range from **9** o'clock at the most left point facing downwind via **12** o'clock (i.e. zenith) to **3** o'clock all the way to the right. Along this semicircular arc the pulling force of the kite reaches its lowest value. Continuously flying

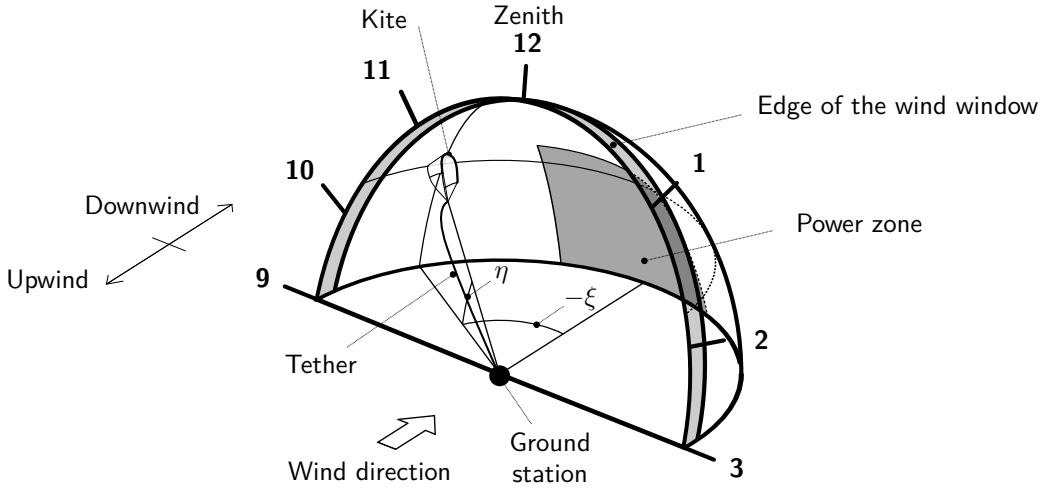


Figure 1.2: Wind window and basic terminology in the kite's flight envelope

the kite towards 12 o'clock is termed **parking**. This is a so-called **fail safe state** since the kite produces low traction force and is very stationary at its highest possible position requiring only low steering inputs. As the kite is flown to a more downwind position in the wind window it reaches the so-called **power zone** where it experiences the highest angle of attack (α) and thus the highest pulling force. In order to keep the kite flying close to the power zone it can be flown in **figure-of-eight** or circular patterns in **crosswind** operation. A figure-of-eight can either be performed in up- or downloop manner. While flying a downloop as shown in figure 2.2 results in a more constant pulling force, it is less safe than an uploop since the kite's leading edge temporarily points directly towards the ground. While continuously flying these patterns the tether is reeled off the drum. This flight phase is referred to as generation, reel-out or **traction phase** and is followed by the reel-in or **retraction phase** in order to complete one entire **cycle**.

Note that throughout this thesis positions, velocities and accelerations are consistently given in Eulerian specification with respect to a certain reference frame. Angles are always[3] defined by right hand rotation around a certain axis of a given reference frame. All coordinate systems are right handed. The notation of a vector $(\mathbf{v}_B^A)_C$ comprises its physical direction A and additional information B. When used in the context of a reference frame C, the containing reference frame is denoted outside the brackets. In general vectors are noted bold, in lowercase, and straight while matrices and points are bold, uppercase, and straight. Scalars may be either lower or uppercase and are written in italics.

1.2.2 Fault terminology

Fault management systems are applied in many different technological areas. This has led to a variety in accepted terms and definitions. Terminology for fault-tolerant purposes used within this thesis is based on the findings of *IFAC Technical Committee Safeprocess* [4] which has made an effort to define commonly applicable definitions. The following list contains important terms relevant for this thesis and their definitions.

[3] Exept for the azimuth angle ξ which is defined positive to the right facing downwind in order to match the definition of the earth's longitude

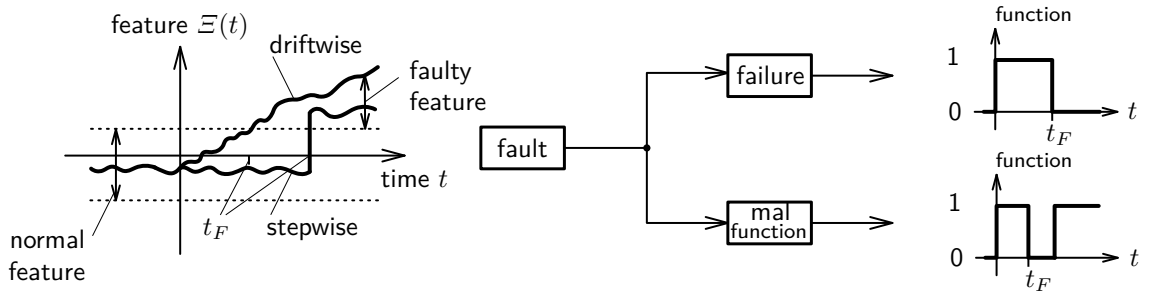


Figure 1.3: Development of failure and malfunction events from a fault which causes a stepwise or driftwise change of a feature [20]

Feature Any known physical quantity of the system such as input variables or measured parameters.

Fault An unpermitted deviation of at least one characteristic property (feature) of the system form the acceptable, usual or standard condition.

Failure A permanent interruption of a system's ability to perform a required function under specified operating conditions.

Malfunction An intermittent irregularity in the fulfillment of a system's desired function.

Error A deviation between a measured or computed value and the true, specified or theoretically correct value, or (within control theory context) the deviation between a given, desired and the actual value.

Disturbance An unknown and uncontrolled input acting on a system.

Protection Means by which a potentially dangerous behavior of the system is suppressed if possible, or means by which the consequences of a dangerous behavior are avoided.

Monitoring A continuous real-time task of determining the conditions of a physical system, by recording information, recognizing and indicating anomalies in the behavior.

Supervision Monitoring a physical system and taking appropriate actions to maintain the operation in the case of faults.

Note that a fault is a state within the system, whereas failures and malfunctions are events. Both failures and malfunctions result from one or more faults, which can develop abruptly or driftwise (cf. figure 1.3). Faults can be classified to various different types like manufacturing faults, assembly faults, normal operation faults (e.g. wear), incorrect operation fault (e.g. overload), software fault and many more. They are present regardless

of whether the system is operating or not and may, but do not necessarily, lead to failures or malfunctions. These however terminate the system's ability to perform a required function permanently or temporarily.

1.3 KitePower 2.0

The research documented in this thesis is carried out within the department Wind Energy at the Faculty for Aerospace Engineering at TUD and is part of a project called KitePower 2.0. The framework for this project was created in the beginning of 2014 when TUD and Karlsruhe University of Applied Sciences started a privately funded cooperation aiming at pushing AWE to the next level. The target of this development project is to achieve 24 hours continuous automatic operation of a pumping KPS. To meet this ambitious goal the complete design is being revised in the first project phase lasting for one year. The budget allows a team of six persons to redesign the main components for meeting demanding requirements and to reach a substantially higher technology readiness level. The project team in Delft continues its work using the existing 20 $10^3\,W$ ground station focusing on airborne system components, while the team in Karlsruhe develops a new 32 $10^3\,W$ ground station.

1.4 Scope of this thesis

The aim of this thesis is to present a project proposed within the KitePower 2.0 framework of TUD's Wind Energy department in March 2014. The objective of this project is investigating means to augment the flight control system of the current KPS by an algorithm that detects potentially hazardous situations and reconfigures the control system adequately in order to ensure safe operation. To reach this goal, three key questions were raised:

1. Which internal faults, failures and malfunctions or external threats are likely to occur and how can they be classified?
2. How can the system autonomously detect faults, failures, malfunctions or external threats?
3. How shall the system reconfigure its control strategies to ensure safe operation when a hazardous situation is encountered?

Chapters 3 to 5 of the document at hand provide solutions to these central problems. As described in chapter 6 the algorithms developed are tested in a real-time simulation environment briefly introduced in section 2.5.
Towards the end of this project the commercial interest in the kite power system grew unexpectedly fast. This was on one hand highly appreciated by the entire team but on the other hand led to a shift of priorities that did not allow to test the code developed during flight before this thesis was submitted. This circumstance may be seen as a pity, it is however not unsatisfying as in-flight validating of the software was, from the beginning on, only considered as nice-to-have add-on if time allowed.

System description

2

The following chapter outlines the operating principle and the main components of the KPS developed at TUD in recent years. Special attention is drawn to the general software architecture, the flight control system as well as the simulation software.

The overall setup of the system developed at TUD comprises a kite connected to a winch on the ground via a single tether. The winch is part of the ground station, which also features an electric motor, battery modules and the control center.

As illustrated in figure 2.1 the electric motor provides both the ability to reel the tether onto the winch's drum (i.e. retraction phase) as well as reeling off the tether (i.e. traction or generation phase) while thus acting as generator. During generation phase the kite is flown in a so-called *figure-of-eight* pattern in the downwind area of the ground station. It therefore reaches high apparent velocities and thus a high aerodynamic load F_{aero}. An exemplary downloop figure-of-eight maneuver is shown as continuous shot in figure 2.2. In order to further increase the aerodynamic load during the generation phase

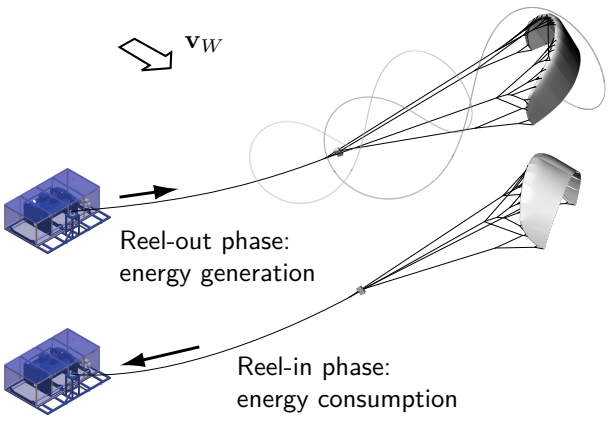


Figure 2.1: Operating principle of a pumping KPS [31]

Figure 2.2: Kite flying figure-of-eight maneuver in the power zone during traction phase [11]

the kite's angle of incidence is set to a relatively high value while reeling out, which is often denoted as *powering the kite*. For a $25\,\mathrm{m}^2$ kite with an aspect ratio of around 5 this total aerodynamic force is in the order of magnitude of up to $6\ 10^3\,\mathrm{N}$ for wind speeds of around $7\,\mathrm{ms}^{-1}$. Taking into account gravitational effects and drag of the tether, the force F_T actually rotating the drum has to be distinguished from the aerodynamic force of the kite itself. Multiplied with the reel-out velocity the tether's pulling force yields the mechanical energy generated during traction phase:

$$P_m = F_T \cdot v_r \tag{2.1}$$

As soon as the maximum tether length is reached, the kite is prepared for the retraction phase by decreasing its angle of incidence and flying it towards the wind window's zenith (cf. figure 1.2). As the tether is winded onto the drum, a certain part of the energy gained has to be fed back into the system. The amount of energy needed is mostly determined by the kite's depower capability and can be assessed via the pumping efficiency η_p calculated using the net mechanical energy $E_{m,o} - E_{m,i}$ and the mechanical energy $E_{m,o}$ gained during traction phase [17]:

$$\eta_p = \frac{E_{m,o} - E_{m,i}}{E_{m,o}} \tag{2.2}$$

Figure 2.3 illustrates this behavior using data acquired at an average wind speed of $8.5\,\mathrm{ms}^{-1}$ using a Genetrix Hydra $14\,\mathrm{m}^2$ kite on June 23, 2012.

2.1 Main components

Profound knowledge of the system components and their operating principle is of great importance when designing a fault-tolerant control system. The sections below shall give a brief description of the key hardware components as well as a more detailed introduction to the software used.

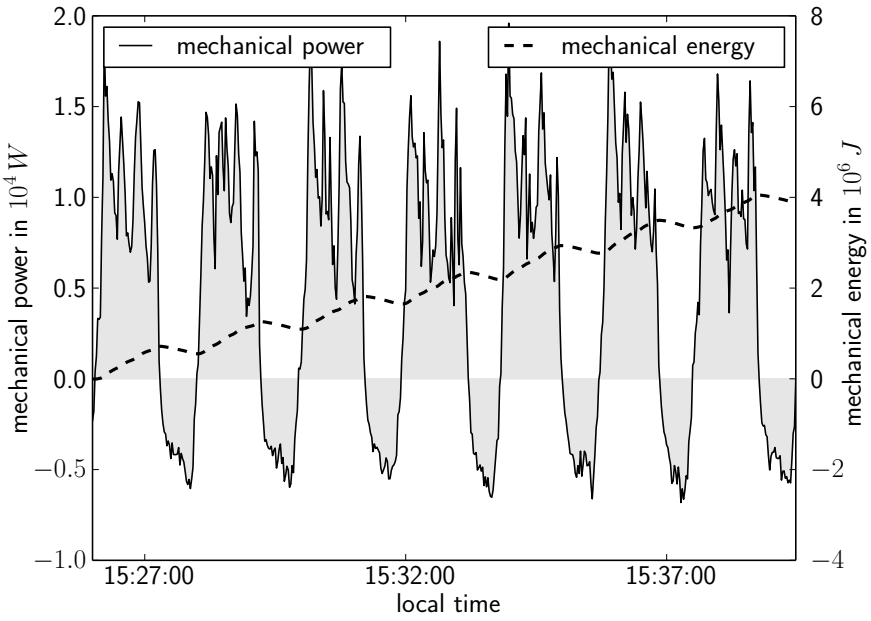


Figure 2.3: Mechanical power and energy during seven cycles; data was acquired at an average wind speed of $8.5\,\mathrm{ms}^{-1}$ using a Genetrix Hydra $14\,\mathrm{m}^2$ kite on June 23, 2012

Figure 2.4: LEI DelftV3 kite with bridle system, KCU and tether
LEI DelftV3 kite with bridle system, KCU and tether (Photo: R. Schmehl)

2.1.1 Kite

For generating traction force, currently the custom made LEI kite *DelftV3* is used. Although most likely not viable for commercial applications, flexible wings are very much suited for the research conducted at TUD. Their light-weight structure allows usage particularly in low wind conditions and the small packing volume makes transportation easy. Compared to rigid wings, flexible kites are rather forgiving concerning crashes since they can sustain impacts and cause less harm to persons and objects on the ground. Such a system can therefore be deployed easily and quickly also by inexperienced student teams. Additionally the complex behavior of flexible structures moving through flow fields provides a broad range of research topics to be investigated at TUD. However the considerably lower glide ratio as well as the poor lifetime make flexible kites less favorable than rigid kites as an economic product.

Wing

The DelftV3's wing itself (cf. figure 2.4) consists of a sail, inflatable beams in span and chord direction and rigid chordwise reinforcements. The design was derived from supported leading edge LEI kites which are elsewise, in smaller sizes, used for kite surfing and is subject to modifications as it is refined regularly. Neglecting the pressurized air and including the bridle system it has a mass of 10.6 kg and a projected surface area of $25\,\mathrm{m}^2$ allowing it to carry loads up to $8\ 10^3\,\mathrm{N}$[1]. Since the materials do not have reasonable structural rigidity on their own, internal pressure is essential to attain the desired shape. The significant deformation under high loads is to be taken into account when designing such a kite and is being investigated in various ongoing projects.
As illustrated in figure 2.5 the wing is supported by several bridle attachment points along the leading edge and at the wing tips. While the main load is transferred via the front tube suspension part of the bridle system, the lines connected to the wing tips are utilized for steering and powering the kite. By varying the steering line length asymmetrically the wing is deformed, aerodynamic forces acting on the wing tips are not balanced anymore and a yawing moment is induced. In addition the side experiencing a higher angle of attack creates greater aerodynamic forces. Since the leading edge bridles do not allow rolling motion, solely the drag force contributes to the introduced yaw moment. By pulling both steering lines symmetrically the wing's angle of incidence is changed which is referred to as powering or depowering of the wing.

Bridle

The bridle system's task is transferring the aerodynamic load distributed over the wing to the tether. As mentioned above, it consists of the leading edge bridle part (i.e. front tube suspension) as well as the bridle part connected to the wing tips. While the steering lines are attached to tapes that run into the KCU as shown in figure 2.5 the leading edge bridles connect to the main tether directly, passing the KCU. As the wing's shape is to a large degree determined by the bridle geometry, it has to adapt to different flight conditions by itself. This ability is provided by pulleys that automatically adjust to the apparent forces within the bridle system. Though very interesting, this is not of specific concern in this

[1] This is an estimated value that has not yet been verified in tests.

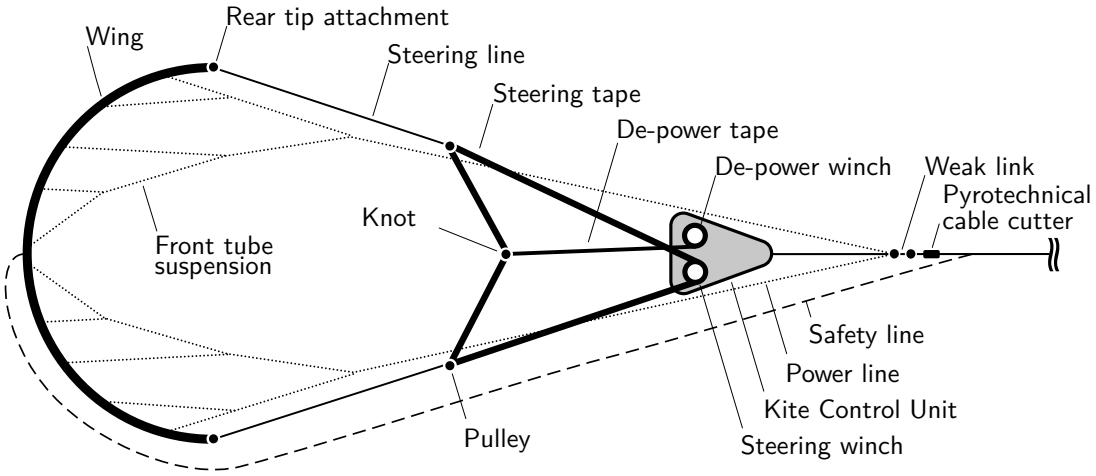


Figure 2.5: Schematic depiction of kite, bridle and KCU [28]

context and thus not considered any closer.
In addition to the above mentioned bridle system, the DelftV3 kite features a so-called 5^{th} or safety line. As shown in figure 2.5 it connects the leading edge directly with the tether and is not under tension during normal operation. It is only used in case of an emergency and allows the system to land in tethered glide mode. In order to trigger this flight condition the main tether has to be cut just above its connection to the 5^{th} line. This is done either mechanically by an integrated weak link or in a pyrotechnical manner following a command issued by the system operator. If this is done, KCU and bridles swing down underneath the wing and stabilize the flight by their weight, similar to a paraglider pilot. The kite then starts gliding with a fairly low air speed as the surface load is very low and can be pulled towards the ground station.

In addition to the bridles described the DelftV3 features depower lines connected to the trailing edge as visible in figure 2.4. These lines provide a more even load distribution in span direction when flying in a powered state.

2.1.2 Kite control unit

The cable robot suspended between bridles and main tether as shown in figure 2.4 is referred to as kite control unit (KCU). Next to steering and depower motors, gearboxes and a depower break, it houses drums for steering and depower tapes. Though disadvantageous from an aerodynamic point of view, the use of tapes for the lower part of the steering mechanism is favored over conventional lines. Tapes can be reeled easier and require less space on the drum. The maximum no-load reel speed for both motors is $0.4\,\mathrm{ms}^{-1}$.
The KCU also features means for communication with the ground station. For redundancy purposes three different wireless links are used. The main link is established via a $5\ 10^6\,\mathrm{Hz}$ dipolar directional antenna. It is based on the User Datagram Protocol and uses Google Protocol Buffers for serialization. The main link is backed up by a slower $2.4\ 10^6\,\mathrm{Hz}$ serial link. For both of these links the ground control center has to be functional. In order to mitigate this dependency a remote control can directly connect to the KCU via a $2.4\ 10^6\,\mathrm{Hz}$ link.
The on-board voltage of 11.6 V is provided by batteries which deliver power for a maximum

flight duration of approximately two hours. Ongoing projects investigate the implementation of airborne wind turbines for autonomous power supply of the KCU. Intelligence inside the KCU is provided by two computers. A Micromint Electrum board is used for not so time critical tasks like communication while motor control is done by a faster ASTI board.

2.1.3 Tether

The tether currently in use is a 4 mm Gleistein DynaOne® rope with a total length of 10^3 m and a weight of 0.8 kg per 100 m. As it does not transmit electrical power or any kind of signals to the kite, its only purpose is transferring traction force. According to the manufacturer the rope has a maximum braking load of 13 10^3 N and a special coating to enhance its lifetime under alternating bending stresses. The tether is one of the major safety critical system components. Since it is not redundant, it is designed according to a safe-life philosophy and has to be exchanged after it has experienced a certain amount of cycles or as a given age is reached.

2.1.4 Ground station

The ground station's main functionality is converting the traction force delivered by the tether into electrical energy. It does so by letting the tether reel off from a drum and thereby rotate an electrical machine via a gearbox. This machine has a nominal power of 18 10^3 W and can act as a generator during reel-out phase as well as a motor for retracting the airborne components during reel-in phase. The electrical energy generated or used is supplied to or taken from a battery module that has a capacity of 20 10^3 Wh. Excessive energy that cannot be stored in the batteries is burned within the dump load module using power resistors or can be provided to external modules such as the control center via a 240 V AC power output.
In order to make sure that the main tether is fed evenly onto the drum, the ground station houses a spindle motor which rotates a worm drive. By doing so the drum is constantly positioned in a way that the tether reaches the swivel in a straight manner. The ground station's nominal force is 4 10^3 N and it features a maximum reeling speed of 8 ms^{-1}. [32]

2.1.5 Control center

All components on the ground used to control the system are bundled together to a work station for two persons. This is referred to as control center and comprised industrial computers, monitors, joysticks and all other I/O devices necessary. It is powered by the ground station's battery module. The control center and the software used is described in detail in chapter 2.4.

2.2 Sensors

Especially for the purpose of automatic operation several different sensors distributed over all the system components are required. The most important ones shall be mentioned here, as they provide data useful for detecting faults.

A sensor measuring wind speed and direction is mounted on a six meter high beam located several meters in the upwind direction of the ground station. It transfers data to the control center wirelessly. Directly at the swivel elevation and azimuth angle of the tether leaving the ground station as well as the traction force are measured. Amongst others the sensors on airborne components include a GPS receiver, an inertial measurement unit (IMU) and potentiometers and temperature sensors on the steering and depower motor. As the flexible kite deforms under load, sensors mounted on the wing (e.g. IMU on a strut) show significant errors, even if they them-self work perfectly fine. The KCU's battery voltage is also measured and recorded.

2.3 Reference frames

As a full description of all reference frames affecting this project is not necessary for understanding the document at hand, only the so-called small-earth-analogy and some basic dependencies shall be explained at this point.

The small-earth-analogy was introduced at TUD as former automatic flight controllers were being developed. It allows using well established aviation theory. The following lines present this concept and are accompanied by figure 2.6 for better understanding. From this point on the term *small* or the subscripts $_S$ or $_{\mathbb{S}^2}$ shall be used when referring to a small earth analogy.
Imagine standing on a unit sphere $\mathbb{S}^2$ with its center at the tether exit point **O** at the ground station's swivel, one could interpret the kite as a freely flying object in the sphere's airspace with a gravitational force in the magnitude the tether force pointing towards the center of $\mathbb{S}^2$. The kite's elevation and azimuth angles can then be considered as small latitude and small longitude. Correlating with the earth's north pole the unit sphere's zenith can be denoted small north pole $\mathbb{N}_S$. The kite's attitude can be described relative to a reference frame based on a tangent plane on $\mathbb{S}^2$ using Euler angles. This correlates to the so-called north-east-down (NED) reference frame commonly employed for aerospace applications. For control purposes the kite's position can be projected onto the small earth's surface. Then the autopilot can for example be used to minimize the tracking error of the kite's path over small ground and a given desired track as it is often done in flight control systems. Neglecting the small height (i.e. tether length) the complexity of the autopilot's task can be reduced to a SISO problem. The actual gravity plays a minor roll as it is one order of magnitudes smaller than the small gravity.

2.4 Software architecture

The main advantage of using an airborne KCU instead of ground based control solutions as for example described in [15] or [26] is that only one tether is required. This greatly reduces the aerodynamic drag but introduces the need for a distributed software architecture in order to enable automatic flight. The following section describes this distributed design and takes a closer look at the current flight control system.

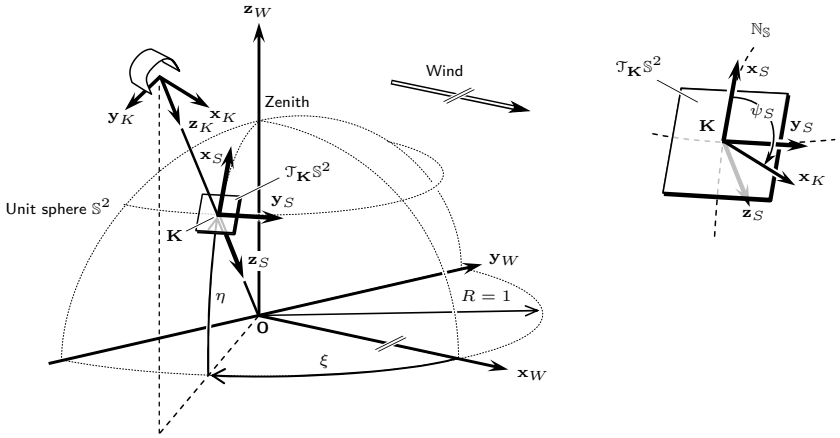


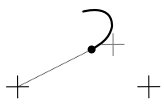
Figure 2.6: Illustration of the small-earth-analogy with reference frames $(\cdot)_W$, $(\cdot)_K$ and $(\cdot)_S$; note that the special case of $\theta_S = \phi_S = 0$ in depicted [24]

Table 2.1: Maximum tolerable time budget of the major system components [16]

Component	maximum delay	Remarks
GNSS sensor and IMU	20 10^{-3} s	from kite
Wired link	5 10^{-3} s	to KCU
Wireless link	15 10^{-3} s	to ground station
Kite state estimator	5 10^{-3} s	
Flight path controller	20 10^{-3} s	
Wireless link	15 10^{-3} s	to KCU
Motor controller	20 10^{-3} s	

2.4.1 Distributed design

As the different components of the KPS control system are physically distributed to separate parts, modular software architecture is necessary. As of now five computers (two within the KCU and three at the ground control center) are used. Accurate timing of these components and their communication infrastructure are of great importance. As described in [16] non stable control behavior was observed during flight tests when the latency between a measurement and the corresponding reaction of an actor exceeded 100 ms. To fulfill these requirements Linux tuned for low latency as suggested in [9] was chosen as main operating system. The time budget allowed for each component in order to stay within the maximum tolerable latency is based on its technical specifications and shown in table 2.1. Except from the winch control, which has firm real-time requirements, communication between the distributed software components is realized via the transport layer ZeroMQ. This message library is easy to apply, supports the use of various programming languages and its *publish - subscribe* pattern is well suited for distributed designs. In combination with the flexible and straight forward serialization library Google Protocol Buffers the required time budget is met.



2.4.2 Flight control system

The flight control system (FCS) is currently under development and thus subject to frequent changes. The following section refers to an intermediate version that was used to develop the health supervision software.
As the time scales of different control tasks within a KPS strongly differ from each other, they can be distinguished into three categories. Similar to common FCSs, which consist of guidance or navigation (i.e. outer loop) as well as attitude controllers (i.e. inner loop), the KPS's control structure consists of a flight path planner (i.e. outer loop), a flight path controller (i.e. intermediate loop) and a course controller (i.e. inner loop) as proposed in [34]. The following sections describe the individual parts in detail.
The presented system has evolved from earlier control systems described in [35]. Although as well very promising, other approaches that have been taken at TUD, as for example tracking of a fixed path on the small earth [23], are out of the scope of this project and shall not be considered at this point.

Flight path planner

The KPS's flight path planner can be compared to an aircraft's navigation loop. It does not perform any steering itself, but it chooses desired future states and makes use of the inner loops to find means to actually reach those. For KPSs one could say, that the flight path planner exercises control on pumping cycle level. When flying one entire cycle the system experiences five individual flight phases referred to as system states. Once the switch criteria for a certain flight phase have been reached, the flight path planner's job is to issue an update on the desired state, thus switching to the next flight phase. As shown in figure 2.8 the system can only reside within one states at a time and switching conditions are clearly defined. One entire pumping cycle with its individual flight phases is depicted in figure 2.7 based on data that has been recorded during a flight test on June 23, 2012. Note that the data recorded was gained with an earlier version of the flight control software, when the system state `ssWaitUntilPointedAtZenith` was used instead of `ssWaitForHighElevation`.

When switching to the next system state the flight path planner chooses one or more new target points $\mathbf{P}_{\mathrm{des},n} \in \mathbb{S}^2$ on the unit sphere, it adapts the desired depower setting and it issues a certain set force to the winch controller. An incomplete overview of this behavior if given in figure 2.8. Bear in mind that the actual values are constantly subject to changes as the flight trajectory is being optimized and different wings are tested.

Flight path controller

The flight path controller is only active during system states with more than one target point. The current version of the flight path planner for example issues two target points for flying figures of eight during reel-out. The flight path controller's job is to issue only one of those points at a time and to switch to the next one appropriately. In order to fly downloop figures-of-eight during the system state `ssReelOut`, for example, it uses four sub-states which are listed together with their switch conditions in table 2.2. This behavior is explained in the following sentences and illustrated in figure 2.9. While residing within the sub-states `FLY_LEFT` or `FLY_RIGHT` the course controller introduced in the next section

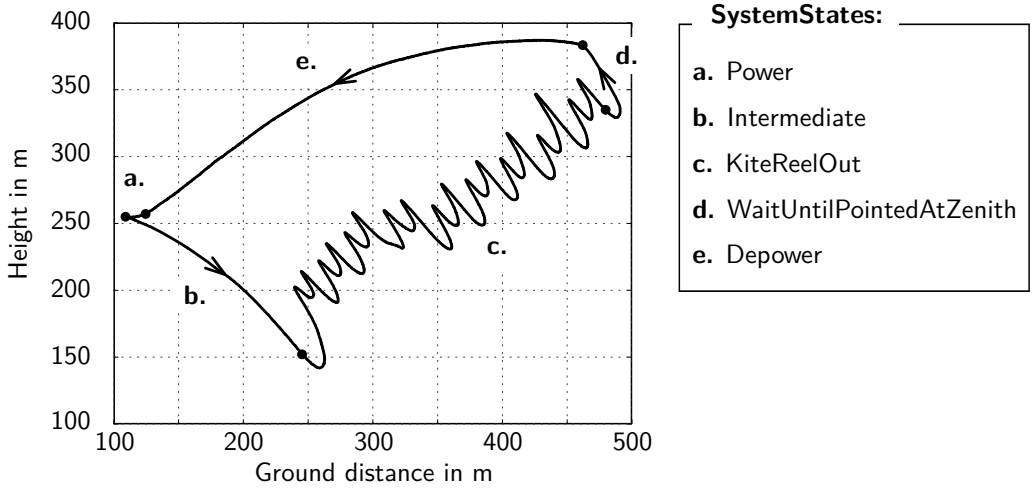


Figure 2.7: Flight path planner: Flight phases (i.e. states) the system experiences during one pumping cycle; data was recorded during a flight test on June 23, 2012

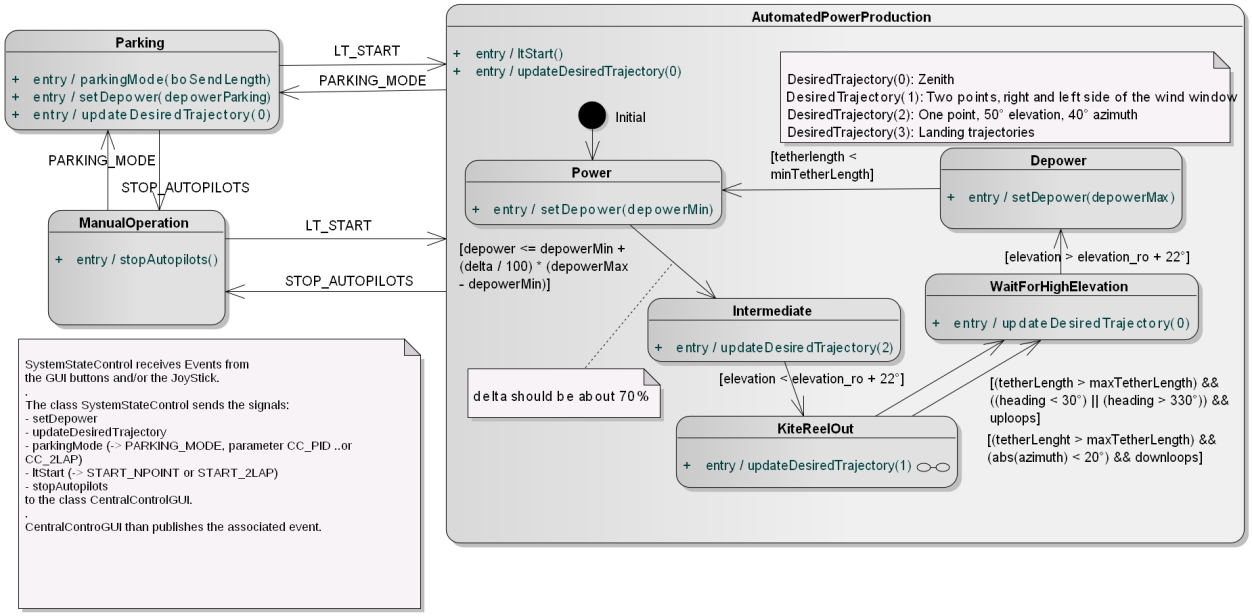


Figure 2.8: Finite state diagram of the control structure implemented by the flight path planner used when recording data for figure 2.7

Table 2.2: Flight path controller sub-states with corresponding target points $\mathbf{P}_{\text{des},n} \in \mathbb{S}^2$ and switch conditions during reel-out phase

Sub-state	**Next sub-state**	$\mathbf{P}_{\text{des},n} \in \mathbb{S}^2$	$\dot{\psi}_{\text{des}}$	**Condition**
Initial	`TURN_LEFT`		$-\dot{\psi}_{\text{turn}}$	always
`FLY_LEFT`	`TURN_LEFT`	$\mathbf{P}_3$	course controller	$\xi < -\xi_{\text{SW}}$
`TURN_LEFT`	`FLY_RIGHT`		$-\dot{\psi}_{\text{turn}}$	$\psi < 90° \wedge$ $\xi > -\xi_{\text{SW}}$
`FLY_RIGHT`	`TURN_RIGHT`	$\mathbf{P}_4$	course controller	$\xi > \xi_{\text{SW}}$
`TURN_RIGHT`	`FLY_LEFT`		$\dot{\psi}_{\text{turn}}$	$\psi > 270° \wedge$ $\xi < -\xi_{\text{SW}}$
`FLY_LEFT`	Final		$\dot{\psi}_{\text{turn}}$	$(l_T > l_{\text{up}} \vee$ $h > h_{\text{up}}) \wedge$ $\xi < -\xi_3$

calculates the desired small course towards a target point $\mathbf{P}_{\text{I}}$ or $\mathbf{P}_{\text{II}}$ given by the flight path controller employing great circle navigation and issues an appropriate steering command u_{s}. As soon as the kite passes a given azimuth angle ξ_{SW} the flight path controller turns off the course controller and issues a constant steering input $u_{\text{s, turn}}$. Thus the kite dives down and keeps turning at a constant rate until the switch conditions are met. Then the flight path controller turns on the course controller again in order to steer the kite to the other point. The predecessor version of the flight path controller described used only two phases (i.e. fly left, fly right) for figure-of-eight control. A sequence of images is given on the bottom left corner of every even page of this document as a flip book for demonstrating its behavior.
In order to achieve the optimal pulling force in varying wind conditions a measurement of the prevailing wind speed is used to calculate the desired elevation angle. The flight path controller than has the freedom to add a certain offset elevation $\Delta\eta$ to the fixed target points.

Course controller

As the flight path planner has now not only issued settings for depower and winch control but has also, assisted by the flight path controller, chosen a target point on the unit sphere the course controller's task is to apply the necessary steering input to reach this given target point. Therefore it calculates the desired small course based on small great circle navigation and the small heading necessary to fly this course. The actual (estimated or measured) small heading is then compared to the desired one and an anti windup PID controller minimizes the error. In order to increase the course controller's robustness and accuracy efforts are currently being made to enhance the PID controller by nonlinear dynamic inversion (NDI). A block diagram for this structure is given in figure 2.10.

Winch controller

The winch controller is implemented in C#. For simulation and testing purposes a second version of the controller was written in Python. At this point the winch control shall be

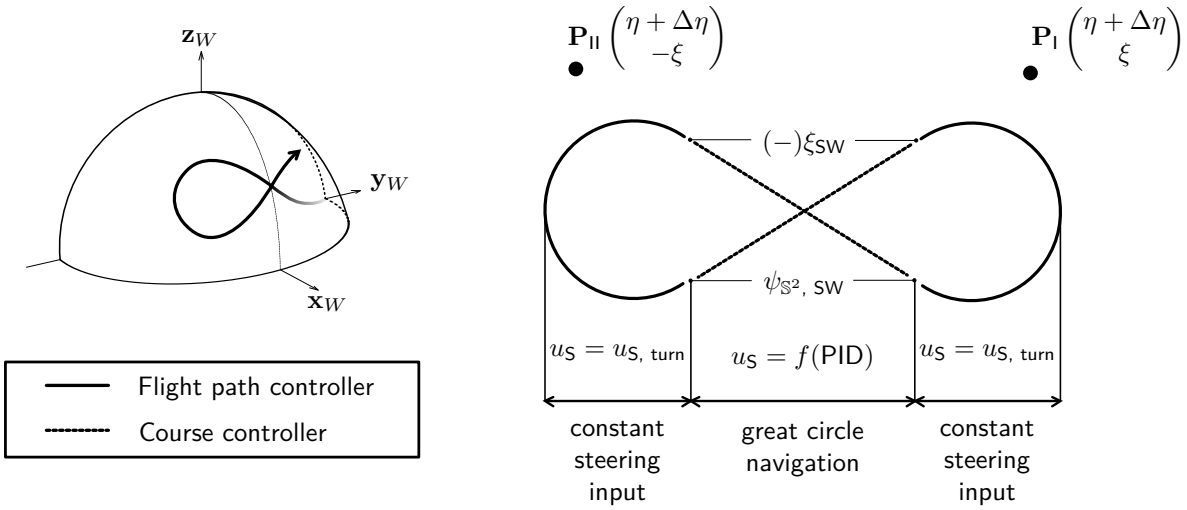


Figure 2.9: Flight path controller and course controller: Illustration of the control algorithm during reel-out

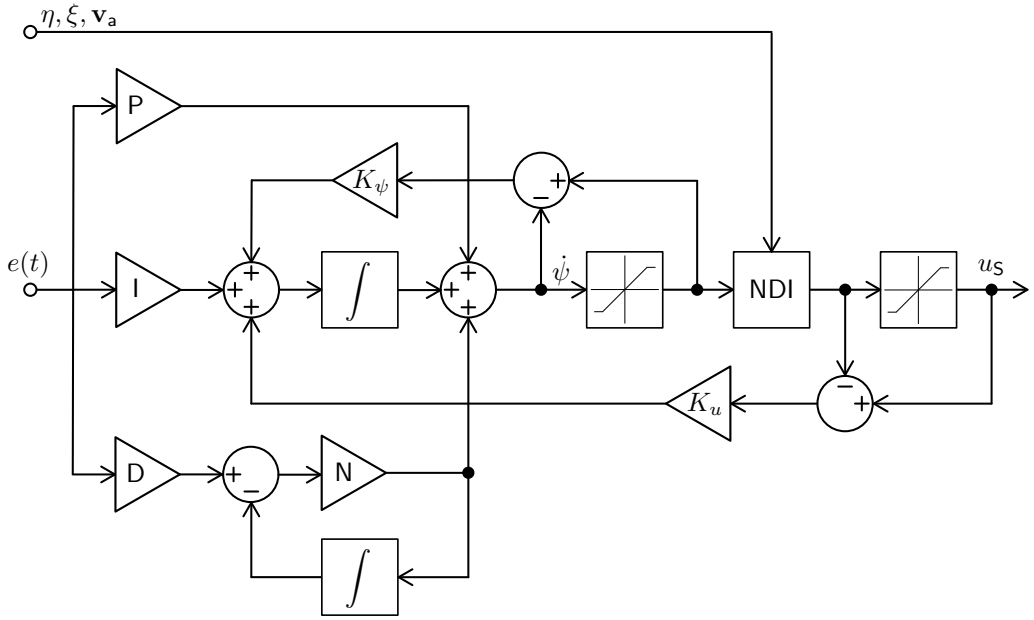


Figure 2.10: Block diagram showing the course controller's structure including NDI

treated only briefly, as it plays a minor role in the project documented.
During reel-out phase the winch's rotational speed is held constant. Only in the case that the tether force exceeds a maximum value the reel-out speed is increased in order to prohibit damages to the system. Reeling-in can be done in different modes, either with constant force of constant speed. Great respect is paid to achieving soft transitions regarding reeling direction.

2.5 Simulation software

At TUD a lot of effort has been put into modeling the behavior of wind energy systems with regard to fluid structure interactions, e.g. [6] and [8]. These models are however expensive in terms of computational effort, not real-time capable and thus not applicable for the design and optimization of flight control systems.

Several attempts have been made to create dynamic real-time KPS models. Most of them however either incorporate assumptions, e.g. [10] and [12], that do not allow them to be used for accurate system simulations or focus on certain system components only, e.g. [37].

A combination of models published in [18] acts as a basis for developing the health supervisor presented in this thesis. It comprises the dynamics of all major system components by modeling both kite and tether as a system of particles (i.e. point masses) connected by spring-damper elements. This consistent structure allows the use of efficient mathematical methods for solving the stiff equation system [14]. Except from the core loops, which are implemented in C++, the whole model was written in the high-level programming language Python. This makes it easy to extend or modify it for various applications. During simulation the model communicates to the flight control system described in chapter 2.4.2 via several ZeroMQ sockets. Google Protocol Buffers are used to serialize the messages. As a new state is calculated and published on a dedicated socket every 50 10^{-3} s the KPS controllers calculate steering and depower commands u_S and u_D as well as the desired winch reel-out speed v_r and publish them as answers to the incoming state message. The implicit equation system is solved using the open source based Assimulo suite.

With an efficiency error of less than 2 % when simulating normal operation [18] the model allows for software-in-the-loop testing and evaluation of flight control systems. Furthermore it can be used to develop estimation algorithms and optimize the desired flight path. For the purpose of this thesis the model was enhanced for simulating certain undesired flight conditions. The following sections provide a short description of the different model components.

2.5.1 Tether model

The tether is modeled as a series of $n_s + 1$ point masses connected by spring damper elements. For simulations ranging up to a height of 600 m AGL a number of segments $n_s = 6$ is considered sufficiently accurate. Reeling in and out is implemented by changing each segment's length by means of

$$l_S = \frac{l_{T,i}}{n_S} + \frac{v_r(t - t_i)}{n_S} \quad . \tag{2.3}$$

As a segment's length is changed, the spring and damping constants k and d are recalculated according to

$$k_S = k_{S,0}\frac{l_{S,0}}{l_S} \qquad \text{and} \qquad d_S = d_{S,0}\frac{l_{S,0}}{l_S} \quad . \tag{2.4}$$

The particle system's state vector $\mathbf{Y}$ is composed by each particle's position vector $\mathbf{p}_i$, its velocity vector $\mathbf{v}_i$, the tether length l_T and the reel-out velocity v_r yielding the implicit problem

$$F(t, \mathbf{Y}, \dot{\mathbf{Y}}) = 0 \qquad \mathbf{Y}(t_0) = \mathbf{Y}_0 \qquad \dot{\mathbf{Y}}(t_0) = \dot{\mathbf{Y}}_0 \tag{2.5}$$

with

$$\mathbf{Y} = (\mathbf{p}, \mathbf{v}, l_T, v_r) \qquad \mathbf{p} = \begin{pmatrix} \mathbf{p}_0 \\ \vdots \\ \mathbf{p}_{n,\,S} \end{pmatrix} \qquad \mathbf{v} = \begin{pmatrix} \mathbf{v}_0 \\ \vdots \\ \mathbf{v}_{n,\,S} \end{pmatrix} \tag{2.6}$$

in which the subscript $_0$ refers to the position of the tether exit point of the ground station's swivel $\mathbf{O}$ and $_{n_s}$ to the KCU's position. By choosing an initial elevation angle η_0 and a static initial tether length the particle positions at $t = 0$ are calculated assuming a straight tether. Except from the particle at the ground station all initial particle velocity vectors are set to $(\mathbf{v}_0)_W = \begin{pmatrix} 10^{-6} & 10^{-6} & 0 \end{pmatrix}^T ms^{-1}$ while the initial particle accelerations are $(\mathbf{a}_0)_W = \begin{pmatrix} 10^{-6} & 10^{-6} & 9.81 \end{pmatrix}^T \mathrm{ms}^{-2}$. The initial reel-out speed is chosen to be $0.0\,\mathrm{ms}^{-1}$ where as its derivation is set to $10^{-6}\,\mathrm{ms}^{-2}$. Having composed the initial state vector $\mathbf{Y}_0$ and its derivative $\dot{\mathbf{Y}}_0$ a residual problem

$$\mathbf{R} = F(t, \mathbf{Y}, \dot{\mathbf{Y}}) \tag{2.7}$$

can be defined and solved employing Assimulo's Radau5DAE. The residual vector $\mathbf{R}$ consists of four parts, the particle position residuals $\mathbf{R}_\mathbf{p}$, the particle velocity residuals $\mathbf{R}_\mathbf{v}$, the tether length residual $R_{l,\,T}$ and the reel-out residual $R_{v,\mathrm{r}}$. They are calculated according to equations (2.8) - (2.11):

$$\mathbf{R}_{\mathbf{p},\,i} = \mathbf{v}_i - \dot{\mathbf{p}}_i \tag{2.8}$$

$$\mathbf{R}_{\mathbf{v},\,i} = \dot{\mathbf{v}}_i - \left(\mathbf{g} - \frac{\mathbf{F}_{P,\,i}}{m_{P,\,i}}\right) \tag{2.9}$$

$$R_{l,\,T} = \dot{l}_T - v_r \tag{2.10}$$

$$R_{v,\,r} = \dot{v}_r - \dot{\omega_d}\frac{r_d}{\lambda_{GB}} \tag{2.11}$$

The forces $\mathbf{F}_{P,\,i}$ comprise the spring forces acting on each particle as well as the aerodynamic drag force assigned to it. The rotational speed of the winch's drum, which has a radius r_d and a gearbox ratio λ_{GB}, is referred to as ω_d (cf. section 2.5.3).
While simulating one time step the Radau5DAE solver uses as many iterations as necessary for reaching a given precision. In order to achieve soft real-time[2] performance, certain solver parameters responsible for the recalculation of the Jacobian and the step size had to be modified as shown in table 2.3.

[2] In comparison to *hard real-time*, where missing of a deadline leads to total system failure, the result in *soft real-time* applications solely degrades the system's quality if deadlines are not met.

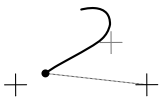


Table 2.3: Modified Radau5DAE solver parameters

Parameter	Default	Modified	Description
`fac1`	0.1	0.3	Parameter for step-size selection.
`fac2`	10.0	4.0	Parameter for step-size selection.
`inith`	0.01	0.003	This determines the initial step-size to be used in the integration.
`thet`	0.003	0.09	Value for determine if the Jacobian is to be recomputed or not.

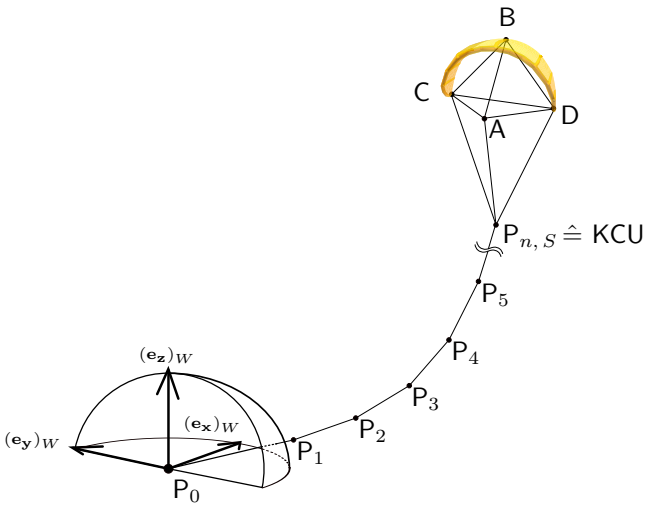


Figure 2.11: Schematic depiction of the four point kite (A−D) connected to the tether model (P_0 − $P_{n,S}$) in the wind reference frame

2.5.2 Kite model

As mentioned before the variety of kite models developed at TUD ranges from one point models [18] to models comprising several thousand nodes [6]. The model chosen for the development of health supervisor presented in this thesis is a four point model. It is the simplest model solely consisting of particles and springs yet incorporating rotational inertia. During initialization the kite model is connected to the last element of the tether model. Figure 2.11 shows the geometric representation of this setup. The kite's mass is distributed to points A−D in a way that reflects reality as close as possible. Point B is the origin of the kite reference frame $(\cdot)_K$, which is right handed and which has its $z-$axis pointing towards $P_{n,s}$ and its $x-$axis pointing in flight direction. The representation of the kite's geometry within the model has three main variables: the bridle's height (i.e. perpendicular distance of $P_{n,s}$ to the spring between C and D), the kite's height (i.e. perpendicular distance of B to the spring between C and D) and the kite's span (distance between C and D). The perpendicular distance of A to the spring between C and D can be used to tune the kite's center of gravity and its rotational inertia for pitching and yawing. The initial particle positions and the kite's orientation are calculated using a more simple point mass model that is not of interest within this context.
The prevailing aerodynamic forces are attached to the points B, C and D of the model. They are calculated according to equations (2.12) - (2.14) in the kite reference frame, where

A_{top} and A_{side} are virtual surface areas assigned to the corresponding points. The lift and drag coefficients c_L and c_D depend on the angle of attack α present at the respective part of the kite model.

$$\mathbf{L}_B = \frac{\rho_{air}}{2} \|\mathbf{v}\|^2_{a,xz} A_{top} c_L(\alpha_B) \frac{\mathbf{v}_a \times \mathbf{e}_y}{\|\mathbf{v}_a \times \mathbf{e}_y\|} \quad \mathbf{D}_B = \frac{\rho_{air}}{2} K_D \|\mathbf{v}\|^2_a A_{top} c_D(\alpha_B) \frac{\mathbf{v}_a}{\|\mathbf{v}_a\|} \tag{2.12}$$

$$\mathbf{L}_C = \frac{\rho_{air}}{2} \|\mathbf{v}\|^2_{a,xy} A_{side} c_L(\alpha_C) \frac{\mathbf{v}_a \times \mathbf{e}_z}{\|\mathbf{v}_a \times \mathbf{e}_z\|} \quad \mathbf{D}_C = \frac{\rho_{air}}{2} K_D \|\mathbf{v}\|^2_a A_{side} c_D(\alpha_C) \frac{\mathbf{v}_a}{\|\mathbf{v}_a\|} \tag{2.13}$$

$$\mathbf{L}_D = \frac{\rho_{air}}{2} \|\mathbf{v}\|^2_{a,xy} A_{side} c_L(\alpha_D) \frac{\mathbf{e}_z \times \mathbf{v}_a}{\|\mathbf{e}_z \times \mathbf{v}_a\|} \quad \mathbf{D}_D = \frac{\rho_{air}}{2} K_D \|\mathbf{v}\|^2_a A_{side} c_D(\alpha_D) \frac{\mathbf{v}_a}{\|\mathbf{v}_a\|} \tag{2.14}$$

The coefficient K_D is required to correct for inaccuracies that are induced due to details that are not of interest at this point. Steering of the model is implemented by changing the angle of attack of the kite's side areas. The consistent structure of point masses and springs in both the tether and the kite model allows a very harmonic simulation. The details mentioned for the tether model (section 2.5.1) are also applicable for the kite model. Further information concerning model accuracy and calibration can be found in [18].

2.5.3 Winch model

In this context, generator, gear box and drum are together referred to as the winch. It is modeled by combining the differential equations of it's inertia and an expression for the torque-speed characteristics of the generator. The inertia has already been introduced along with the tether model in section 2.5.1. The last two components of the particle system's state vector $\mathbf{Y}$, the tether length l_T and the reel-out speed v_r (cf. equation (2.6)), treat the rotational dynamics of the winch. The term $\dot{\omega}_d$ in the calculation of the reel-out speed's residual, i.e. equation (2.11), can be calculated using the winch's inertia I as observed from the generator and the torque τ_d exerted on the generator by the drum, the generator torque τ_g itself and the friction torque τ_f

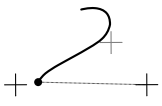
$$\dot{\omega}_d = \frac{\tau_d + \tau_f + \tau_g}{I} \quad . \tag{2.15}$$

While τ_d and τ_f depend on tether force and reel-out speed, the generator torque profile is a machine specific characteristic. More detailed information on the individual calculations can be found in [17].

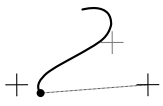
2.5.4 Atmospheric model

For determining the wind speed at a certain height a combination of the power law and the log wind law, both given in equation (2.16), is used. It is believed that this approach leads to a more accurate result as not only one but several reference wind speeds $v_{W,ref}$ at reference heights h_{ref} can be given as input.

$$\|\mathbf{v}\|_{W,exp} = v_{W,ref} \left(\frac{h}{h_{ref}}\right)^{\alpha} \qquad \|\mathbf{v}\|_{W,log} = v_{W,ref} \frac{\log(\frac{h}{h_0})}{\log(\frac{h_{ref}}{h_0})} \tag{2.16}$$



However the atmospheric model is currently being redeveloped and shall therefore not be treated in detail. In order to reflect the environment more realistic way a turbulence model based on measured data is being investigated.



System assessment

3

Within this chapter the components introduced in the previous sections are evaluated regarding their effects on the system under operation in case of a fault. Therefore different theoretical methods are applied and hazardous situations observed in the past is investigated.

Before starting to design a fault-tolerant flight control system its requirements have to be stated precisely. Especially a list of all faults, failures and malfunctions to be detected has to be established. In order to find such a list, one has to have profound knowledge of the system. It is important to comprehend how certain components affect the entire system, especially in case of their failure. Several system assessment methods that help gaining such an understanding and quantitatively determining it have evolved from different, highly safety critical, technological fields like space exploration or nuclear applications. The most common methods utilized for aerospace systems are, according to [20],

- Reliability Analysis,
- Event Tree Analysis (ETA),
- Fault Tree Analysis (FTA),
- Failure Mode and Effect Analysis (FMEA),
- Hazard Analysis (HA), and
- Risk Classification.

For analyzing the safety and reliability of an entire system these methods can be combined. Figure 3.1 shows a procedure suggested by [22]. It indicates that such considerations should already be done hand in hand with the actual product design process, rather than on top of it. Other sources like [27] identify a combination of Failure Modes and Effects Analysis and Fault Tree Analysis as methodology that provides means, not only during early development phase, to:

- Determine potential failure modes for a product or process,
- Assess the risk associated with those failure modes,

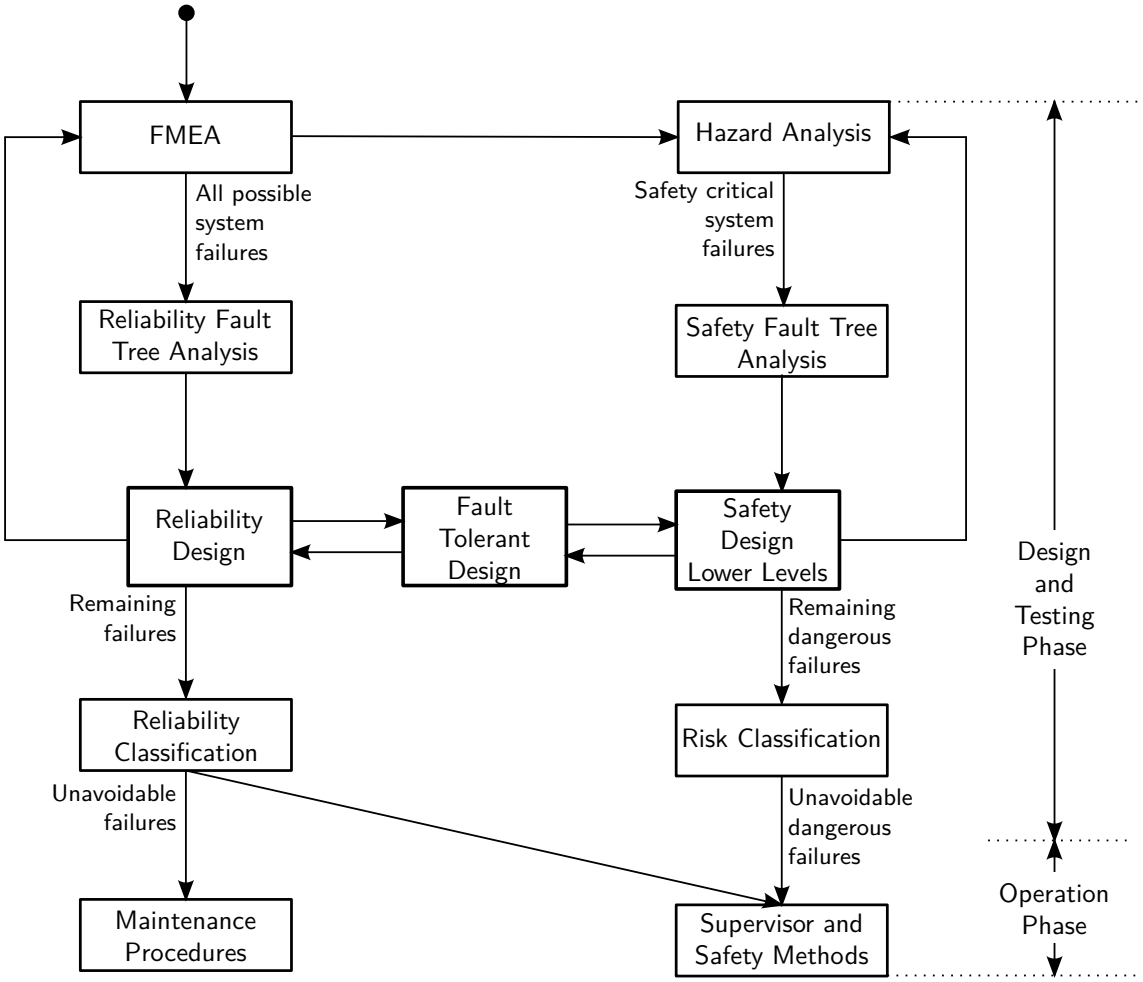


Figure 3.1: Integrated design procedures for system reliability and safety to result in a high system integrity [22]

- Rank the issues in terms of importance, and
- Identify and carry out corrective actions to address the most serious concerns.

However such an approach takes a lot of time and since the current KPS prototype status has evolved from several different projects over the years it has not been done yet. Obviously it is also way out of the scope of this thesis to do something similar. Therefore a more practical approach was chosen. First a qualitative FMEA was conducted in order to identify all relevant components, their failure modes and causes as well as their effect on the overall system. The single faults, failures and malfunctions discovered are then passed on to a simple FTA in order to structure and classify them according to possible counteractions to be carried out by the FCS. The result is then compared to failures experienced in the past and finally a reasonable list of faults, failures and malfunctions to be treated by the health supervisor is established.

Table 3.1: Exemplary data revealing the formalized FMEA structure

Subsystem	Component	Sub-component	Failure mode	Failure cause	Effect on system	Counter-action
Airborne H/W	Kite	Inflatable strut	Deflation	Deficiency in material	significant loss of steerability	Immediate landing
Environment	Wind	Velocity	Too gusty	Uncertainty in weather forecast	High force peaks	Parking

3.1 Failure Mode and Effect Analysis

The Failure Mode and Effect Analysis is a method developed by NASA in the 1960s and first used within the Apollo program. Later it was adapted also for aerospace applications, nuclear systems and other industrial areas that incorporate high severity in case of a failure. Today FMEA is widely used amongst various technological areas, e.g. automotive industry, as a quality management tool in order to identify and overcome weak points already in early design phases of a product. Its main purpose is thus enhancing reliability through design.
Therefore FMEA might seem to be the wrong method to be applied to an existing system like TUD's KPS. However the structured approach makes it easy to apply but yet powerful, even for an existing system. In addition, a FMEA yields the system's possible failures, which are needed as inputs for an FTA. However a complete FMEA is a very time consuming process and is out of the scope of this thesis. Therefore a similar but flattened process is chosen. No respect is paid to parameters like criticality, likelihood or risk.
For conducting the FMEA the entire KPS is divided into subsystems like e.g. Airborne H/W, H/W on ground or Control Center S/W. These subsystems are again divided into components like e.g. kite, tether or KCU. After that their subcomponents like e.g. inflatable struts or the sail are individually listed. Then every possible failure mode on this subcomponent level is investigated regarding its cause, its effect on the entire system and a possible counteraction. A short example for this formalized process is shown in table 3.1.

3.2 Fault tree analysis

The individual faults, failures and malfunctions gained in the FMEA, as well as their effects on the system operation, can be used as inputs for a Fault Tree Analysis. Based on the undesired system states the FTA investigates which failure or combination of failures lead to such a state and provides a graphical representation with the status of the system as top event and the individual subcomponent faults, failures and malfunctions as lowest events. Figure 3.2 shows a simplified example of such an fault tree. The seperate boxes have a boolean characteristics, i.e. they can either be *true* (operative) or *false* (inoperative), and they are logically connected to each other with operators *and* and *or*. However in the course of this project a simplified approach is chosen as this is considered well enough

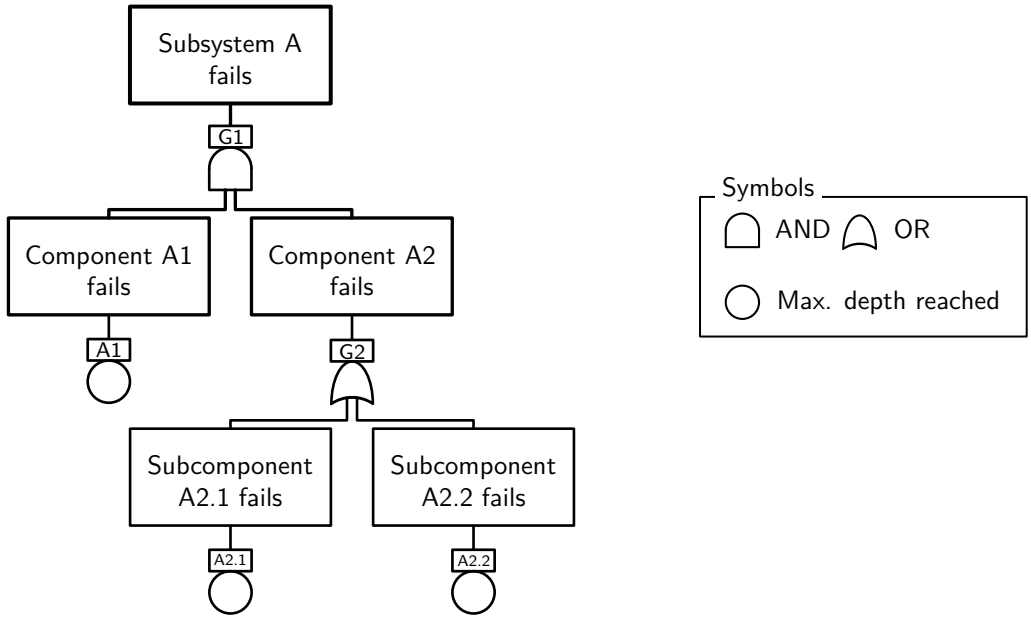


Figure 3.2: Simplified fault tree showing the general structure of a FTA [36]

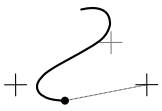
for a first step towards fault-tolerant design. In addition it was decided that within this thesis the focus should lie on health supervision and the respective counteractions rather than on system assessment. After analyzing past flight tests, as described in the section below, the fault tree established is combined with insights gained from observed failure scenarios. This logic is presented in Appendix A and forms the basis for the development of fault-tolerant software enhancements presented in chapter 4.

3.3 Flight test situations observed

As great respect is paid to applicability and usefulness of the software developed, past flight test data is extensively studied. The insight gained from crashes or other undesired situations is combined with the fault tree in order to reach realistic goals that provide a functional enhancement of the system. The result, which is used as a basis for the further development is presented in Appendix A.
Below three hazardous situations experienced most often in the past are briefly discussed accompanied by serial photographs taken during TUD flight tests. Due to recent modifications in the flight control software these situations are avoided to a large extent already.

Figure 3.3 shows a kite in crosswind operation during reel-out phase. Within this sequence the tether force (i.e. wing loading) exceeds the maximum value that can be carried by the wing. The leading edge starts to buckle in a reversible way. As soon as the tether is rolled off the drum faster, the kite regains its original shape. Such a situation can occur if the winch controller does not operate correctly or if gusts are not compensated quick enough.

A situation like the above, might, if not treated adequately, lead to a rupture of the tether's weak link. In this case the kite remains connected to the tether's end via the so-called fifth line (cf. section 2.1.1). In case it does not entangle, the kite then enters a



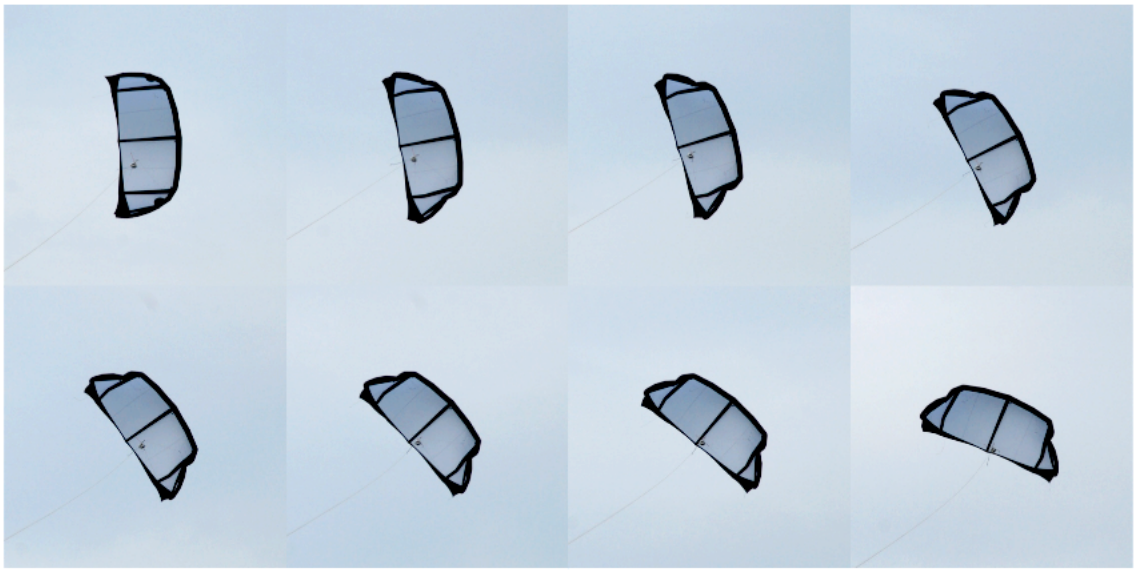


Figure 3.3: Reversible buckling of the kite's leading edge under high load

glide mode similar to a paraglider. For different reasons, for example if the fifth line is not dimensioned correctly, it might happen that kite, KCU and bridles entangle after a weak link rupture. Since the wing's center of mass is close to the leading edge, it is likely that, as a result of the high acceleration, the wing flips over and descends in a stable but non-steerable glide as shown in figure 3.4.

Another hazardous situation observed in the past results from bad depower settings. As the maximum possible depower is set too higher values, the stagnation point moves along the wing's upper surface towards the trailing edge, provoking a dent in the sail and eventually a front stall. At a certain point the inflatable structure is not able to withstand the forces acting in an undesired direction and the wing collapses. Entanglement in such a case is a likely scenario, as shown in figure 3.5. However when responding quick enough, a hazardous situation can be avoided as depicted on the cover, which shows a similar situation.

Figure 3.4: Rupture of the weak link leading to entanglement and stable but non-steerable descend

Figure 3.5: Collapse and entanglement of the kite due to wrong depower setting

Fault-tolerant control system design

4

In order to be able to automatically detect faults and failures defined in the previous sections the FCS has to be enhanced by corresponding methods. The chapter below presents the suggested solution.

Other than open loop systems, which are not capable of handling faults, failures or malfunctions by themselves at all, one might argue that closed loop systems are by nature fault-tolerant and thus sufficient for safe automatic operation of kite power systems. While for an open loop system an introduced failure (e.g. rupture of the kite's sail) results in a permanent offset of the output (e.g. undesired small heading), the deviation in the output of a well designed closed loop vanishes (e.g. the course controller automatically adapts the steering tape length until it reaches the desired heading). This behavior can however only be considered as passive fault tolerance as the closed loop system solely adapts its inputs in order to counteract the failure. When not actively monitoring the inputs, such a system thus hides the actual problem from system operators. Although this fundamental behavior is desired, it can be very hazardous when not treated sensibly. As the fault increases in effect (e.g. the rupture in the sail grows) the system will at a certain point run out of input resources (e.g. steering tape length) to counteract the fault. Only then will the output (e.g. heading) deviate from its desired state noticeably. Thus, an inattentive system operator will recognize a failure or malfunction only at a very late state (e.g. when the autopilot cannot steer the kite adequately anymore). As emphasized in [13] this passive behavior cannot be considered sufficient for reliable flight control purposes.

4.1 Basic idea

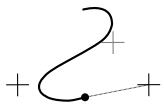
The KPS's health supervisor is being developed in order to reach a higher degree of fault tolerance in the automatic control system. Certain faults shall be detected early and compensated for in order to deter them from leading to more severe failures or malfunctions. For applying any kind of automatic health supervision, certain system features have to be monitored. These features can be any known physical quantity of the system like inputs, measured outputs, or process parameters. If supervision is to be applied to an existing system it might be necessary to implement additional sensors for that purpose. This however not only increases cost and complexity, but also introduces new potential sources of faults and thus degrades the system's reliability. Within the scope of the project documented herewith no additional sensors are installed. Thus only existing system components are available for monitoring purposes. As stated in [33] it is a great advantage to design fault-tolerant modules hand in hand with the FCS itself as they have to act as one unit.

The simplest form of supervision applied to the KPS is so-called protection. A vital feature (i.e. predicted height) is constantly monitored and an elementary signal analysis method (i.e. limit checking) is applied. In case of a hazardous situation an immediate counteraction bringing the system back to a fail-safe state is automatically initiated. Such a safe state can be an emergency shut down or for the current KPS the state `ssParking`. Other industrial protection devices relying on the same principle are for example electric fuses, which interrupt the current in case of a short circuit or aviation engine throttle springs, that set the throttle to maximum in case of a lever linkage failure [21]. Often, also in the case of the KPS, the protection detects a dangerous state (i.e. predicted height too low) but not the fault itself, which is the reason for the dangerous state to occur. It therefore reacts very late and might not be able to compensate for the arisen problems anymore. In the presented system the protection loop is only capable of recovering the system, if its mechanical and electrical integrity is given. Such a situation could for example arise, if a fault in the FCS (e.g. Flight path planner) occurred.
Since this protection is inexpensive in terms of computational cost it forms the health supervisor's inner loop depicted in figure 4.1.

Although such a protection loop improves the reliability, it obviously does not satisfy the need for higher degree fault tolerance. Advanced supervision providing early detection is hence essential to hinder failures and malfunctions from developing. Figure 4.1 gives the general scheme of the KPS's health supervision package designed for that purpose. Supervision is represented by the outer loop which is called less frequently than the protection loop, as it employs more complex methods and thus requires more computational time. The generic idea of health supervision is based on the findings in [20] and incorporates the following steps:

- Collection of features
- Evaluation of features
- Determination of health symptoms
- Decision on health state
- Initiation of counter action

These supervision steps are individually discussed in sections 4.3.1 - 4.3.4 and in chapter 5. Clearly early detection of faults is of great desire as the time to detection of faults



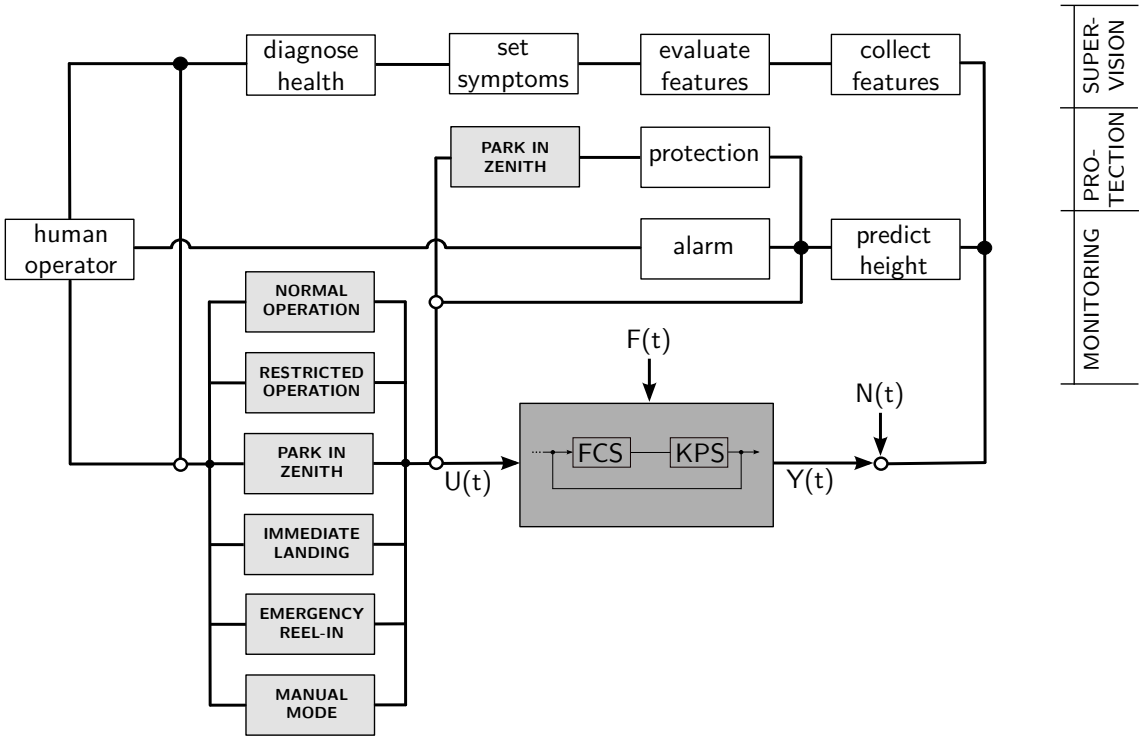


Figure 4.1: Basic scheme of the KPS's health supervisor protection and supervision loop

delays the actual reaction time. It is however a trade-off since value uncertainties might be misinterpreted, thus raising false alarms. Finding a good balance is not straight forward and has to be done in iterative steps while testing the system under operational conditions.

The health supervisor software package is written in Python 2.7. This choice was driven by the fact that Python is an easy programming language. Thus it is possible that future students continue working on fault-tolerant control design without having to spend too much time with acquainting themselves with existing software parts. In addition the simulation software, as described in chapter 2.5, which is employed for developing and testing new software components, is also based on Python. All major software components developed within the scope of this project are designed in a standalone way that allows them to be used for simulation as well as for real flights without applying major changes. Solely the communication ports have to be adapted to the environment they are used in. Using the messaging library ZeroMQ with serialization based on Google Protocol Buffers the health supervisor nicely fits into the distributed design already available at TUD. Table 4.1 provides an overview of the different parts of the health supervisor software and its classes and methods. Note that it is not complete, as only the main methods are considered of interest at this point.

All the above mentioned methods however have to be classified as passive, since they wait for a fault to be detected. Future versions of the KPS health supervisor should be enhanced by methods that actively search for faults by runnung regular health check maneuvers. By exciting the system with given inputs, advanced fault-tolerant systems can assess the output and conclude on their health. The probably most sophisticated solution would be to find two inputs, that eliminate each others outputs and thus do not disturb

Table 4.1: Modules, classes and methods of the health supervisor software package

Module	Class	Method	Lines of code	Remark
Audio.py	`AudioPlayer`	`playSound`	53	Playback functionalities for acoustic caution and waring messages
HealthFeatureModels.py	-	[1]	45	
HSPublisher.py	`HSPublisher`	`sendEvent`	34	Sending ZeroMQ messages
HSReceiver.py	`HSReceiver`	`run`	146	Receiving ZeroMQ messages
	`SystemState`			
SignalAnalysis.py	-	[1]	133	Mathematical operations (e.g. moving average, matrix transformations)
HealthSupervisor.py	`HealthClass`		425	Main module providing the HealthSupervisor
	`FlightPathProtector`	`predictHeight`		
		`evaluateHeight`		
	`AdvancedSupervisor`	`compareModel`		
		`collectFeatures`		
		`evaluateFeatures`		
		`setSymptoms`		
		`diagnoseHealth`		
		`setSystemState`		

[1] Too many to be listed in this table.

the actual flight.

4.2 Flight path protection

The health supervisor's protection loop introduced above is implemented as a class within the module `Supervisor.py`. After the initialization of a FlightPathProtector instance the protection loop is called every 10^{-1} seconds after a `START` event has been received. As long as the system is neither occupied with the launch procedure nor in one of the landing states, a method named `predictHeight` is called. Employing the measured wind direction γ_W and reel-out speed v_r and the estimated position $(\mathbf{P})_{EG}$ and velocity of the kite $(\mathbf{v})_W$, the height at which the kite is going to be three seconds in the future is predicted. The following explains how this is done in detail.

For the sake of conception, wind and small earth reference frames are used for this calculation. Therefore first of all the position $(\mathbf{P})_{EG}$ estimated from global navigation satellite system (GNSS) data by the C++ SystemStateEstimator has to be transformed into the wind reference frame $(\cdot)_W$ by rotating around their common $z-$axis by $-\gamma_W$.

$$(\mathbf{P})_W = {}_W\mathbb{T}_{EG} \cdot (\mathbf{P})_{EG} \tag{4.1}$$

$${}_W\mathbb{T}_{EG} = \begin{pmatrix} \mathcal{C}_{-\gamma_W} & -\mathcal{S}_{-\gamma_W} & 0 \\ \mathcal{S}_{-\gamma_W} & \mathcal{C}_{-\chi_W} & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \mathcal{C}_{\gamma_W} & \mathcal{S}_{\gamma_W} & 0 \\ -\mathcal{S}_{\gamma_W} & \mathcal{C}_{\gamma_W} & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.2}$$

In order to predict future positions the kite's current velocity vector is then projected onto a plane $\mathcal{T}_{\mathbf{K}}\mathbb{S}^2$ tangential to the small earth $\mathbb{S}^2$ at the projected kite position $\mathbf{K}$ as illustrated in figure 4.2. This projection is achieved by transforming the velocity vector into the small earth reference frame by a YXY-sequence of $\{-\frac{\pi}{2}, -\xi, -\eta\}$ resulting in

$$(\mathbf{v})_S = {}_S\mathbb{T}_W \cdot (\mathbf{v})_W = \begin{pmatrix} -\mathcal{C}_\xi \mathcal{S}_\eta & \mathcal{S}_\eta \mathcal{S}_\xi & \mathcal{C}_\eta \\ \mathcal{S}_\xi & \mathcal{C}_\xi & 0 \\ -\mathcal{C}_\eta \mathcal{C}_\xi & \mathcal{C}_\eta \mathcal{S}_\xi & -\mathcal{S}_\eta \end{pmatrix} \cdot (\mathbf{v})_W \tag{4.3}$$

and then setting the z-component of $(\mathbf{v})_S$ to zero yielding the current tangential velocity $\mathbf{v}_\mathcal{T}(t)$. Thanks to the properties of scalar products the tangential velocity $\mathbf{v}_\mathcal{T}(t)$ can now be compared to the velocity $\mathbf{v}_\mathcal{T}(t - \Delta t)$ computed in the previous time step revealing the change in the small course angle $\Delta\chi_S$.

$$\mathcal{C}_{\Delta\chi_S} = \frac{\mathbf{v}_{\mathcal{T}(t-\Delta t)} \cdot \mathbf{v}_{\mathcal{T}(t)}}{\|\mathbf{v}_{\mathcal{T}(t-\Delta t)}\| \, \|\mathbf{v}_{\mathcal{T}(t)}\|} \tag{4.4}$$

The correct sign of $\Delta\chi_S$ however still has to determined. Therefore the sign resulting from the operation in equation (4.5) is applied to $\Delta\chi_S$[1].

$$-\mathbf{P} \cdot (\mathbf{v}_{\mathcal{T}(t-\Delta t)} \times \mathbf{v}_{\mathcal{T}(t)}) \tag{4.5}$$

Now by dividing the change in the small course angle by the elapsed time the course rate

[1] This step inhabits an issue for rotational frequencies that are high compared to the calculation frequency (cf. section 6.1.1).

is estimated.

$$\dot{\chi} = \frac{\Delta\chi}{\Delta t} \tag{4.6}$$

Assuming that this course angle rate stays constant for a certain time horizon t^* (i.e. 3 seconds) a future velocity vector $\mathbf{v}_{\mathcal{T}}^*$ tangential to the small earth can be predicted by turning the current velocity around the small earth's $z-$axis by $\dot{\chi} \cdot t^*$.
Finally the predicted velocity can be transformed back into the wind reference frame. As the kite does not immediately achieve this velocity, the mean of current and predicted velocity

$$\bar{\mathbf{v}}_{\mathcal{T}} = \frac{\mathbf{v}_{\mathcal{T}}^* + \mathbf{v}_{\mathcal{T}}}{2} \tag{4.7}$$

is used to predict the future position $\mathbf{P}^*$

$$\mathbf{P}^* = \underbrace{\frac{\mathbf{P} + t^* \cdot \bar{\mathbf{v}}_{\mathcal{T}}}{\|\mathbf{P} + t^* \cdot \bar{\mathbf{v}}_{\mathcal{T}}\|}}_{\text{direction}} \cdot \underbrace{\left(v_r t^* + \frac{\mathbf{P}}{\|\mathbf{P}\|} \right)}_{\text{length}} \quad . \tag{4.8}$$

Equation (4.8) shows that the predicted position vector is composed of a direction based on the estimated course rate as well as a second part defining the length. This new length is calculated as a sum of the kite's current distance to the ground station and the change of the tether length within the time horizon. For that a constant reel-out speed is presumed.

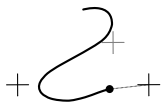
As the position is now predicted in the wind reference frame the position vector's last component is the kite's height. It is passed on to a method called `evaluateHeight` which computes the moving average of the five most recent predicted height values. This average value is then compared against two given thresholds. If it drops below the `CAUTION_HEIGHT` visual and acoustic messages are issued to the system operator. If no corresponding reaction is taken and the predicted height falls below the `WARNING_HEIGHT` not only visual and acoustic warnings arise but also a `PARKING_MODE` event is triggered. As a consequence the autopilot steers the kite towards the zenith and parks it at a tether length of $150\,m$, regardless the state it was in. Note that the protection loop is not called, when one of the systems launch or landing states is active. In order to let the health supervisor know, that the parking mode was triggered by the protection loop, the symptom `S0-0` is set `True`.

4.3 Health supervision

As mentioned above and depicted in figure 4.1 supervision functionality is accomplished by the outer loop of the introduced health supervisor. It is called after the inner protection loop has been executed ten times, which approximately equals to one call per second. As the outer loop is entered, the first operation is done by a method called `collectFeatures`.

4.3.1 Collection of features

To evaluate the system's health as many features as possible have to be collected as they provide all the information available. The current version of the health supervisor is very limited in the number of features it is able to assess. A full list of all features is given in Appendix B. As future projects that treat fault-tolerant control aspects evolve, the number



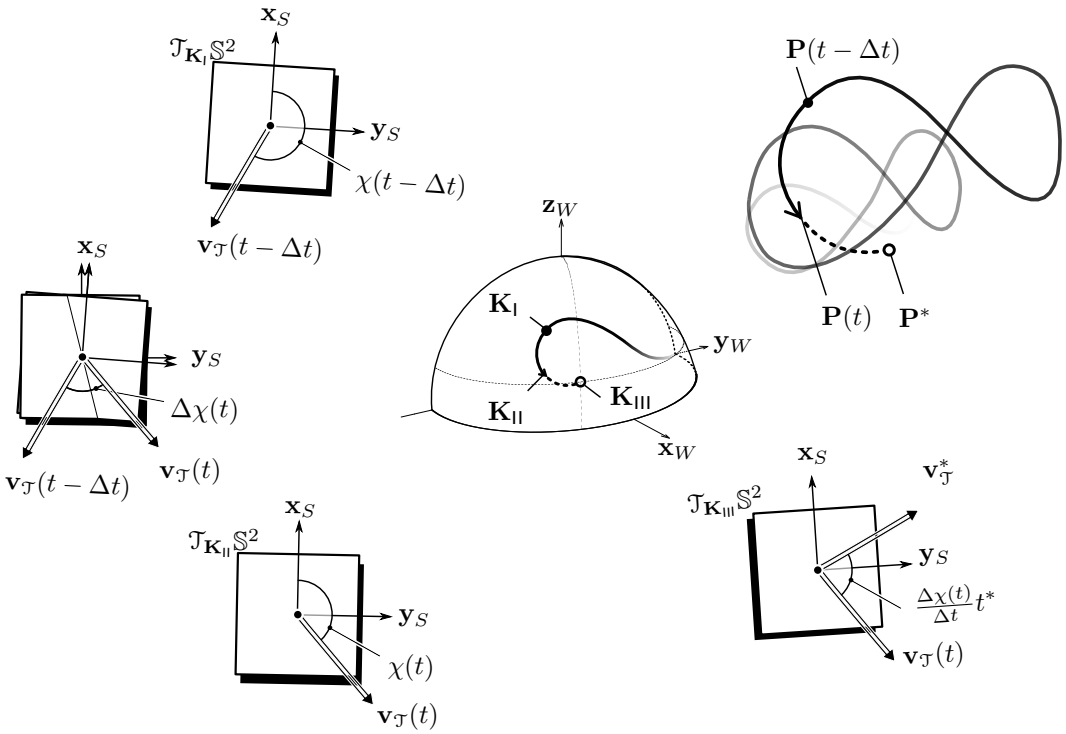


Figure 4.2: Illustration of the approach for predicting the tangential part of future kite positions on the small earth $\mathbb{S}^2$

of features treated should be increased to make the supervisor more knowledgeable.
The method `collectFeatures` reads all messages received by a `HSReceiver` instance and extracts the features it is able to further investigate. Most of these features directly result from sensor measurements (e.g. motor temperatures) which are not available during simulation. Therefore specific health feature models have to be developed in order to verify the supervision's functionality in a simulation environment. These models are briefly described in section 4.3.5. Other features result from the protection loop described in chapter 4.2 or from a comparison of models, which are fed with steering inputs in parallel to the real system, and measured or estimated data. This model comparison is presented in section 4.3.6.
If desired the collected features can be printed to the health supervisor's output window at this point.

4.3.2 Evaluation of features

In the next step, performed by the method `evaluateFeatures` the collected features Ξ are assessed. To reduce the influence of measurement noise most of the features run through a moving average filter which saves the feature's value up to a certain depth and then returns the computed mean. After that either the clean values or their calculated rates are compared against given thresholds:

$$\Xi_{\min} < \Xi(t) < \Xi_{\max} \qquad \text{or} \qquad \dot{\Xi}_{\min} < \dot{\Xi}(t) < \dot{\Xi}_{\max} \tag{4.9}$$

Thereby features that exceed their allowed range or that rise or drop too fast are detected and saved to a dictionary of hazardous features. This dictionary is then made available for a method called `setSymptoms`.

4.3.3 Determination of health symptoms

By going through the dictionary containing undesired features all health symptoms are either set `True` or `False`. Note that different features or combinations of features might lead to the same result in active or inactive symptoms. Again, if desired, a list of all symptoms and their current state can be printed to the output window in order to give the system operator more insight into the system's health.

4.3.4 Decision on health state

Now that all health symptoms are known the supervisor has to decide on the overall health state. This is done by a method called `diagnoseHealth`. The system is free to choose one out of five health states which are:

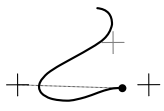
- `NORMAL_OPERATION`
- `RESTRICTED_OPERATION`
- `PARKING_MODE`
- `IMMEDIATE_LANDING`
- `EMERGENCY_LANDING`
- `MANUAL_MODE`

As the health state is diagnosed, it is compared to the current system state. If they do not correlate, the method `setSystemState` is used to send event messages which initiate the desired system state including all its countermeasure actions. Every change in the health state is printed to the supervisor's output window and, if necessary, acoustic caution or warning messages are given. `MANUAL_MODE` is only then activated if the system operator gives a confirmation by adequately accepting an issued alert. If that is not done in time, an automatic procedure is followed instead.

Throughout all health supervision loops the collected and manipulated data is constantly monitored. If irregularities are observed the current operation is aborted and warnings are issued to the operator. This is done in order to avoid the supervisor being stuck at a certain point or driving the system into undesired states.

4.3.5 Fault models for feature generation

As already briefly mentioned in chapter 1.2 faults can be classified according to the characteristics of their occurrence. Figure 1.3 shows abrupt (stepwise) and incipient (driftwise) faults. In addition to that intermittent faults can occur which appear stepwise and after a certain time vanish again. The models designed to represent faulty system components also show this behavior. Note that the fault models' main objective is to generate faults that can be processed by the health supervisor rather than to deliver accurate values. Therefore they only roughly reflect the components' actual behavior. For the purpose of testing health supervision this is considered sufficient.



The KCU's battery voltage for example is modeled as a series of linear functions. It is assumed that from a given charging state in the beginning $U_{\mathrm{KCU},0}$ the voltage decreases linearly as a function of the time passed and the rate of steering input applied. As the depower motor has a mechanical brake and operates one order of magnitude less often than the steering motor, its influence on the battery lifetime is neglected:

$$U_{\mathrm{KCU}}(t) = U_{\mathrm{KCU},0} - t\ \mathrm{C}_1 - \Delta t\ |u_{\mathrm{S}}|\ \mathrm{C}_2 \qquad \text{with} \qquad u_{\mathrm{S}} \in [-1, 1] \setminus \{0\} \tag{4.10}$$

The constants C_1 and C_2 can be used to trim the batteries' discharge characteristics.

Other examples for driftwise fault models can be found in the calculations of the KCU's motor temperatures. Detailed temperature models which take in to account radiative, conductive and convective heat flows are presented in [7]. However for the purpose of testing the health supervisor these models are considered to be too intensive in terms of computational effort. Therefore more simple models are used that calculate the current motor temperature as a sum of ambient temperature T_∞, temperature offset C_1, and the product of a second constant C_2, the steering input's standard deviation $\sigma_{u,S}$, and the apparent velocity felt by the kite:

$$T_S = T_\infty + \mathrm{C}_1 + \mathrm{C}_2 \cdot \sigma_{u,\mathrm{S}} \cdot \|\mathbf{v}_{\mathrm{a}}\| \tag{4.11}$$

The velocity is of interest for the temperature calculation as a higher velocity increases the small gravitational acceleration $\mathbf{g}_{\mathbb{S}^2}$ and thus the kite's wing loading. This increases the force necessary to apply a steering input and thereby heats up the motor (cf. section 5.5).

A representative example of intermittent fault behavior is given by the wireless link model. It randomly delays the reception of messages for up to one second which triggers certain symptoms in the health supervisor regarding both main and slow wireless link.

4.3.6 Model-based fault detection

Fault detection is currently mostly done by simple signal analysis of measured system features. As it is desired to keep the number of used sensors small, additional features have to be created without measuring them directly. A way to do so is fault detection using models. The health supervisor does that for detecting situations that arise from ruptures and entanglement in the kite's sail or bridles and deflation of the wing's struts.

As presented in [24] a correlation between steering input and yaw rate in the form of

$$(r)_K = \mathrm{C}_1 v_{\mathrm{a}}^n u_{\mathrm{S}} + \mathrm{C}_2 \frac{\mathbf{g} \cdot (\mathbf{y})_K}{g} \tag{4.12}$$

was found by applying system identification methods. A graph showing the dependency of the body rates p, q, r of the steering input u_{S} is given in figure 4.3. It can be clearly seen that the yaw rate is highly dependent of the steering input while its influence on the other body rates is rather small. That however only holds for cases with a high small gravity (from 0 to 65 s in figure 4.3). For depowered situations this relation does not hold (from 65 to 90 s in figure 4.3). While C_1 and C_2 are believed to be kite specific parameters dependent on mass, geometry and power setting, the term $\mathbf{g} \cdot \mathbf{y}_K$ relates to the angle between the gravity vector and the y_K-axis. The second part of equation (4.12) thus takes into

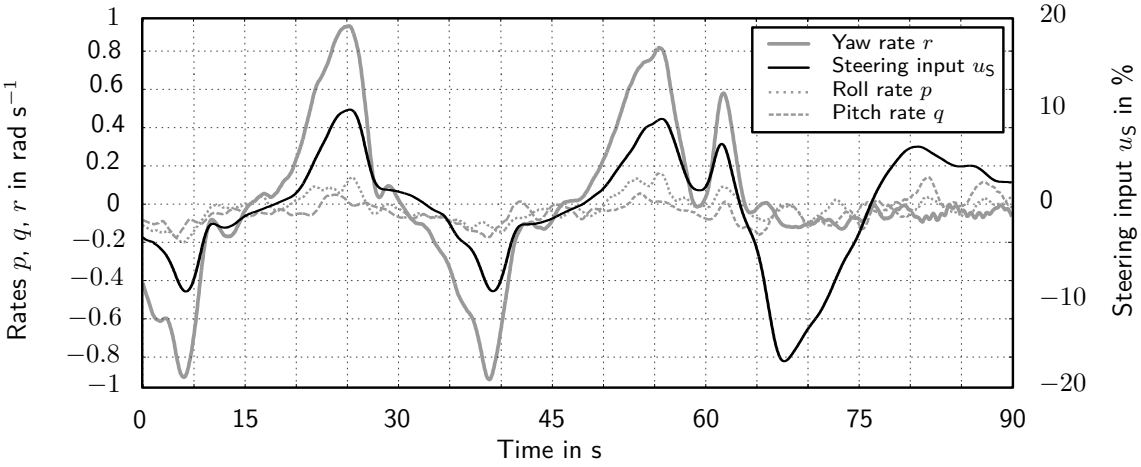
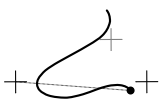


Figure 4.3: Correlation of steering input u_S and rotational rates p, q and r [24]

account the side slip angle a kite has to fly in crosswind operation in order to achieve a horizontal trajectory. The health supervisor makes use of this correlation. By comparing the heading rate estimated from measured position data (cf. section 4.2) with the yaw rate the kite should have according to the steering correlation, undesired situations can be detected.

Therefore prior to collecting features during the supervision loop, the latest steering and depower inputs emitted by the course controller are passed on to a method called `compareCorrelation`. Then, based on the correlation given in equation (4.12) the yaw rate is calculated according to the inputs given to the kite. The difference between the estimated course rate and the correlated steering input is then stored as a system feature.



System health states

5

The described methods for detecting faults are only valuable if the system is able to react in an adequate way automatically. The following chapter presents procedures that can be engaged by the health supervisor for that purpose.

Chapter 3 presented a method that allowed finding health states into which the KPS's health can be classified at any time. Thereafter chapter 4 described the process implemented by the health supervisor in order to diagnose the KPS's current health state. The corresponding reactions of the system to each of the health states are presented in the following section.

In order to allow the KPS to react to a diagnosed health state, the system states handled by the flight path planner as described in section 2.4.2 have to be enhanced by new flight phases. However these new states do not occur during normal operation and the flight path planner is not able to issue them. They can only be entered if the corresponding event messages are published by the health supervisor or if issued manually by a human operator via buttons in the graphical user interface (GUI).

5.1 Normal Operation

As long as the health supervisor does not find any faults, failures or malfunctions in the system (i.e. all symptoms are set `False`) it sets the health state to `NORMAL_OPERATION`. This corresponds to the following system states:

```
enum SystemState {
    // [...]
    ssPower                     = 2;
    ssKiteReelOut               = 3;
    ssWaitForHighElevation      = 4;
    ssDepower                   = 5;
    ssIntermediate              = 6; // after 2, before 3
    // [...]
};
```

If the system does not reside in one of these states while the diagnosed health state is `NORMAL_OPERATION`, the health supervisor sends the event `START_POWER_PRODUCTION`. Thereby the flight path planner enters `ssPower` and starts issuing one of the above mentioned states after each other thus controlling the KPS in its normal pumping cycle mode. However for the sake of human authority and thus safety, this event is not sent if the reason for the system not to operate in one of the above listed system states is because the human operator manually issued a different state.
During normal operation the system automatically chooses its depower setting according to its current state based on defined minimum $u_{\mathrm{D,\,min}}$ and maximum $u_{\mathrm{D,\,max}}$ values. Additional depower settings (e.g. for parking) in between those thresholds are only available to the system if it resides in the corresponding flight phase outside the normal pumping operation. All these values can be edited by a system operator via a GUI.

5.2 Restricted Operation

The health state `RESTRICTED_OPERATION` has not yet been implemented. It is the only countermeasure that allows pumping operation even after a fault has been detected. The restriction can relate to different system components, depending on the fault. If the wind, for example, is too gusty (i.e. high standard deviation of the wind speed) the set force in the winch controller can be reduced for safety reasons. In case of unusually high steering motor temperatures the course controller gains can be adapted in a way that requires minimal motor effort or bigger figures-of-eight can be flown.

5.3 Parking Mode

In the case that pumping operation is not possible or recommended the health supervisor issues the health state `PARKING`. Obviously this makes sense only if the faults present lead to malfunctions but not to failures, as malfunctions (other than failures) vanish with time and normal operation might be possible at a later point again. This can for example be the case if the ground station's dump load module fails and battery voltage exceeds a certain level or if the KCU's power level drops underneath a threshold.
If the health supervisor issues this health state, a `PARKING_MODE` event is sent, the depower is adjusted to a defined power setting for parking, and the flight path planner is reconfigured. Thereby the kite starts flying towards a single navigation point at the wind windows zenith $\mathbb{N}_S$ (i.e. $\eta = 90°$, $\xi = 0°$). Because of the kite's drag it will never reach this point and thus "park" at the highest possible position. If the message is sent together with a desired tether length for parking, the winch controller reels to this length. In the

case that no length is given, the winch maintains the tether length from the moment when the event was sent unless the given force thresholds are violated.

In the course of this project different options for enabling the KCU to park autonomously without receiving steering inputs from the ground station were evaluated. However this ability was not implemented. It is strongly recommended to do so in future projects as it allows the system to fall back into a "fail-safe" if, for example, the wireless connection to the control center was lost.

5.4 Immediate Landing

Extensive loss of operational safety or any other situation that requires repair work or maintenance to be carried out provokes the health supervisor to publish a `LAND` event message and thus issue the health state `IMMEDIATE_LANDING`. The automatic landing procedure which is thereby initiated consists of three phases which are handled by the flight path planner (cf. section 2.4.2):

```
1        enum SystemState {
2            // [...];
3            ssLanding   = 9;
4            ssReelIn    = 10;
5            ssTouchdown = 11;
6        };
```

During `ssReelIn` the tether length is reduced (or if necessary extended) to a value set in the GUI. This is done in order to make sure that the actual landing phase (i.e. dive of the kite) is done from a well defined starting point. By default this tether length is set to 120 m. For achieving similar tether forces in various wind conditions during reel-in the depower setting is adjusted to the present wind speed. For this reason one threshold for low wind speeds $v_{\mathrm{W,\,low}}$ and one high wind speeds $v_{\mathrm{W,\,high}}$ can be defined which results in three different depower settings:

$$u_{\mathrm{D,\,RI}} = \begin{cases} u_{\mathrm{D,\,min}} & \text{if} \quad \|\mathbf{v}_{\mathrm{W}}(t)\| \leq v_{\mathrm{W,\,low}} \\ u_{\mathrm{D,\,mid}} & \text{if} \quad v_{\mathrm{W,\,low}} < \|\mathbf{v}_{\mathrm{W}}(t)\| < v_{\mathrm{W,\,high}} \\ u_{\mathrm{D,\,max}} & \text{if} \quad \|\mathbf{v}_{\mathrm{W}}(t)\| \geq v_{\mathrm{W,\,high}} \end{cases} \tag{5.1}$$

These different depower settings not only make sure that the tether force stays well within its limits but also prevent the kite from overshooting the zenith during reel-in. Once the kite has reached the given tether length $\pm 3\,\mathrm{m}$, the reel-out speed is less than $\pm 0.1\,\mathrm{ms}^{-1}$, the kite is within a azimuth angle of $\pm 9°$ and faces towards the small north pole $\pm 5°$, the flight path planner switches to the main landing procedure system state, `ssLanding`. As the kite is now about to dive into the power zone in a rather radical nose down manner the FlightPathProtector is turned off at this point.
Again depending on the wind velocity one out of three given sets of points (`LP_A - LP_C`) is chosen. The GUI features possibilities for the operator to edit these points as well as the individual switch conditions at any time. By default the values listed in table 5.1 are used. In order to achieve a desired amount of tether force in different wind conditions, the

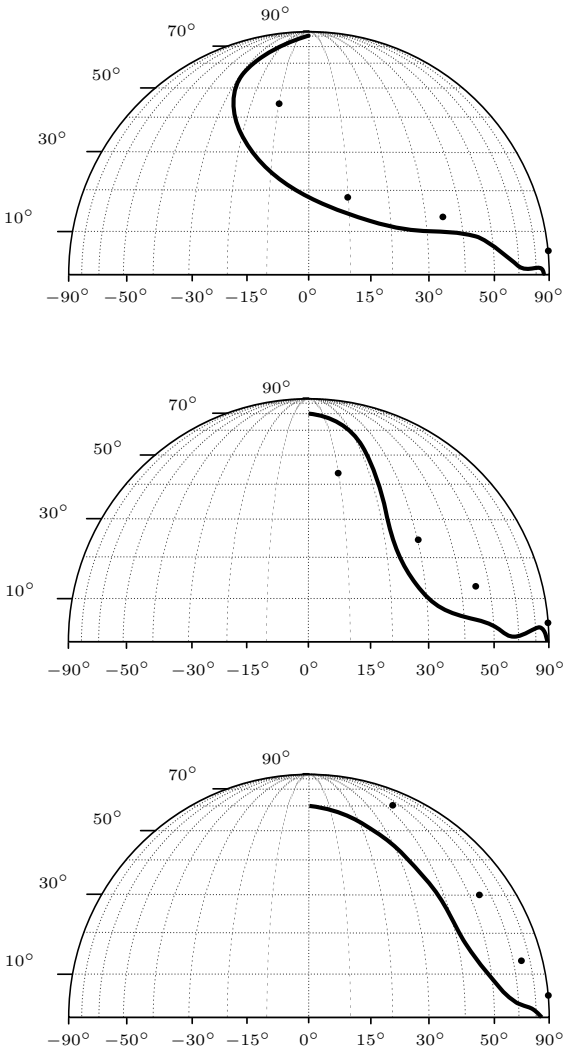


Figure 5.1: Projection of different simulated landing paths onto the small earth surface $\mathbb{S}^2$ chosen according to the wind velocity $\mathbf{v}_\mathrm{W}$

depower setting is adjusted according to the following definition:

$$u_\mathrm{D,\,LA} = \begin{cases} u_\mathrm{D,\,mid} & \text{if} \quad \|\mathbf{v}_\mathrm{W}(t)\| \leq v_\mathrm{W,\,low} \\ u_\mathrm{D,\,max} & \text{if} \quad \|\mathbf{v}_\mathrm{W}(t)\| \geq v_\mathrm{W,\,high} \quad \text{or} \quad v_\mathrm{W,\,low} < \|\mathbf{v}_\mathrm{W}(t)\| < v_\mathrm{W,\,high} \end{cases} \tag{5.2}$$

These depower settings are to be chosen wisely as they directly affect the tether force and thus the wing loading. As recognizable from figure 4.3 the kite's reaction to steering inputs strongly depends on the prevailing wing loading. In a similar manner as during `ssKiteReelOut` the flight path planner chooses which points are to be navigated towards and the flight path controller picks one point at a time. The course controller then steers to the next one as soon as the switching criteria are met. Figure 5.1 shows a projection of three different landing paths onto the unit sphere. This exemplary data was acquired within the python simulation environment. As the kite dives towards the ground it passes a certain height at which the last system state `ssTouchdown` is issued. The kite

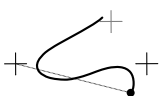


Table 5.1: Navigation points and corresponding switch conditions for landings in different wind conditions; all angles are given in °

Name	Elevation η	Azimuth ξ	Switch cond. η	Switch cond. ξ
LP_A1	44.00	−10.00	< 50.00	> -25.00
LP_A2	18.00	10.00	< 25.00	> 0.00
LP_A3	13.00	35.00	< 13.00	> 25.00
LP_B1	44.00	10.00	< 60.00	> 12.00
LP_B2	25.00	30.00	< 30.00	> 20.00
LP_B3	13.00	45.00	< 13.00	> 25.00
LP_C1	60.00	45.00	< 50.00	> 20.00
LP_C2	30.00	55.00	< 35.00	> 30.00
LP_C3	13.00	65.00	< 13.00	> 35.00
LP_final	5.00	90.00	< 0.00	> 90.00

then flies directly towards a point, which is around 10 m above the ground and on the edge of the wind window, that it can never reach. By applying a high power setting given in equation (5.3) the kite again accelerates in this phase and thus starts flaring. As it keeps steering towards the final point it slowly bleeds out its kinetic energy and eventually drops to the ground.

$$u_{\mathrm{D,\,TD}} = \begin{cases} u_{\mathrm{D,\,min}} & \text{if} \quad \|\mathbf{v}_{\mathrm{W}}(t)\| \leq v_{\mathrm{W,\,low}} \\ u_{\mathrm{D,\,mid}} & \text{if} \quad \|\mathbf{v}_{\mathrm{W}}(t)\| \geq v_{\mathrm{W,\,high}} \quad \text{or} \quad v_{\mathrm{W,\,low}} < \|\mathbf{v}_{\mathrm{W}}(t)\| < v_{\mathrm{W,\,high}} \end{cases} \tag{5.3}$$

5.5 Emergency Landing

If the health supervisor diagnoses that the FCS has lost steering authority, it issues the health state `EMERGENCY_LANDING`. This could for example be the case if a steering line ruptures and the model-based fault detection algorithm detects a significant difference from the yaw rate correlated from given steering inputs to its actual value estimated from GPS position data (cf. section 4.3.6). In the case of such an event the health supervisor makes use of the pyrotechnical cable cutter briefly introduced in section 2.1.1 and figure 2.5. By doing so the main tether is disconnected from the KCU. Due to the forces acting on the wing the kite is then pulled away from the end of the tether and the fifth-line becomes tensioned. In order to avoid high peak loads the fifth-line is connected to the main tether via a strap fall attenuator that gradually rips apart and thus damps the tensioning phase after the main connection is cut. The same mechanism applies if the weak link breaks.
Given that the bridles do not entangle as e.g. in figure 3.4, the system eventually reaches a new trimmed state after a few seconds of oscillating. The KCU then hangs underneath the wing similar to a paraglider pilot and thus stabilizes the flight. The kite then glides towards the ground with the main tether connected to two points on its leading edge. Presuming that the KCU did not entangle in any bridles due to the change in momentum it experienced in the moment the tether was cut, the system retains its steerability. As the wing loading in this flight state is orders of magnitudes smaller than during reel-out

operation, the steering inputs' effects are expected to be very different from the behavior during pumping operation. The gravitational force $\mathbf{G}$, which directly relates to the aerodynamic forces generated, is determined by the overall mass m and the earth's gravitational acceleration $\mathbf{g}$ during fifth-line operation. However during pumping operation the small gravitational acceleration $\mathbf{g}_{\mathbb{S}^2}$ can be more than one order of magnitude greater than $\mathbf{g}$ and depends on the apparent wind speed:

$$\mathbf{G} = \mathrm{m}\,\mathbf{g} \qquad \text{with} \qquad \|\mathbf{G}\| \approx 2\ 10^2\,\mathrm{N} \qquad \text{and} \qquad \|\mathbf{g}\| \approx 10\,\mathrm{ms}^{-2} \tag{5.4}$$

$$\mathbf{G}_{\mathbb{S}^2} = \mathbf{F}_T = \mathrm{m}\,\mathbf{g}_{\mathbb{S}^2}(\mathbf{v}_\mathrm{a}) \qquad \text{with} \qquad \|\mathbf{G}_{\mathbb{S}^2}\| \approx \begin{cases} \text{reel-in:} & 5\ 10^2\,\mathrm{N} \\ \text{reel-out:} & 4\ 10^3\,\mathrm{N} \end{cases} \tag{5.5}$$

$$\text{and} \qquad \|\mathbf{g}_{\mathbb{S}^2}\| \approx \begin{cases} \text{reel-in:} & 2.5\ 10\,\mathrm{ms}^{-2} \\ \text{reel-out:} & 2\ 10^2\,\mathrm{ms}^{-2} \end{cases} \tag{5.6}$$

The conventional formula for the wing loading

$$\mathrm{B} = \frac{\mathrm{m}}{A} \tag{5.7}$$

can therefore not be applied to a tethered airfoil in pumping operation. For that reason the tethered wing loading B_T is introduced at this point. It replaces the mass of the flying object by the sum of tether force and gravitational force towards the center of the small earth.

$$B_T = \frac{\|\mathbf{G}_{\mathbb{S}^2} + \mathbf{G}\|\ \mathcal{C}_\eta}{A} = \frac{F_T + \mathrm{m}\,\|\mathbf{g}\|\ \mathcal{C}_\eta}{A} \tag{5.8}$$

However while operating in fifth-line mode an approximation for the tethered wing loading results from the conventional

$$B_T = \frac{\mathrm{m}}{A} \approx 0.8\,\mathrm{kgm}^{-2} \quad . \tag{5.9}$$

As described in section 2.1.1 steering does not affect the entire wing but only parts of the trailing edge at the wing tips which are actively deformed. Since in this region of the wing the aerodynamic forces are relatively small, one can assume that the force necessary for applying a steering input is only a fracture of the total wing load. Conservatively estimating that $5^{-1}$of the wing's surface has to be manipulated and the lift is equally distributed over the area, equation (5.9) reveals that the force necessary for applying a steering input in fifth-line mode is around 40 N. Comparing this rudimentary result to the KCU's mass of about 10 kg unveils that also after cutting the direct connection of main tether and KCU, the kite retains its steerability. This behavior is also observed during simulation and has been verified during flight tests.

As the health supervisor diagnoses the health state `EMERGENCY_LANDING` it sends a `PANIC` event and cuts the tether. This condition can also be initiated via the GUI or a panic button on the control joystick. As the FCS receives the `PANIC` message the flight path planner issues a new navigation point at the small north pole. The kite is not able to reach this position as demonstrated by the calculation below. In addition the depower is set to a predefined value that can be edited via the GUI and the winch starts to reel-in the tether at a speed that makes sure that the kite does not fly away from the ground

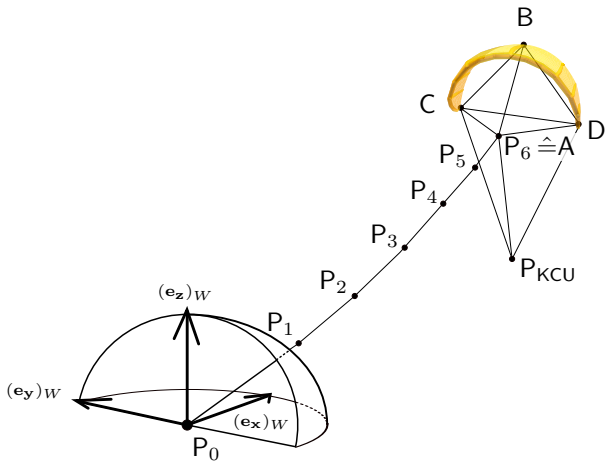


Figure 5.2: Four point kite model (A−KCU) and tether model (P_0 − P_6) used to simulate fifth-line landings

station. As the kite passes the predefined touchdown height the winch stops to reel and the depower setting is adjusted to the present wind speed in order to ensure soft landing.

In order to simulate the kite's behavior when flying on the fifth-line the four point kite model introduced in section 2.5.2 has to be modified. The difference in the bridle setup is accounted to by removing the spring that connects tether particle P_5 and KCU P_{KCU}. Instead a new spring is introduced that attaches the tether to kite particle A as illustrated in figure 5.2. This is considered the closed possible fit to the real system using the existing four point model.

Initial conditions that satisfy a simulation of the adapted model can be found close to its trimmed state. Neglecting the influence of the tether all forces acting on the kite can be summarized as lift $\mathbf{L}$, drag $\mathbf{D}$ and weight $\mathbf{G}$:

$$\mathbf{L} = \frac{\rho \|\mathbf{v}_a\|^2}{2} \cdot A \cdot c_L \tag{5.10}$$

$$\mathbf{D} = \frac{\rho \|\mathbf{v}_a\|^2}{2} \cdot A \cdot c_D \tag{5.11}$$

$$\mathbf{G} = m\,\mathbf{g} \tag{5.12}$$

The force equilibrium in vertical direction

$$\sum \|(\mathbf{F}_z)_W\| = \|\mathbf{L}\|\, \mathcal{C}_\gamma + \|\mathbf{D}\|\, \mathcal{S}_\gamma - \|\mathbf{G}\| = 0 \tag{5.13}$$

with the flight path angle γ gives the relation

$$\frac{\rho \|\mathbf{v}\|^2}{2} \cdot A \cdot c_L\, \mathcal{S}_\gamma + \frac{\rho \|\mathbf{v}\|^2}{2} \cdot A \cdot c_D\, \mathcal{S}_\gamma = m\, \|\mathbf{g}\| \tag{5.14}$$

which reveals the fifth-line trim speed as

$$\|\mathbf{v}\| = \sqrt{\frac{m\,\|\mathbf{g}\|}{\frac{\rho}{2} A\,(c_L\, \mathcal{C}_\gamma + c_D\, \mathcal{S}_\gamma)}} \quad . \tag{5.15}$$

Now from employing the equilibrium of forces in horizontal direction

$$\sum \|(\mathbf{F}_x)_W\| = \|\mathbf{L}\| \, \mathcal{S}_\gamma - \|\mathbf{D}\| \, \mathcal{C}_\gamma \tag{5.16}$$

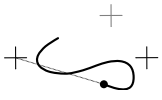
the lift-to-drag ratio

$$\mathcal{T}_\gamma = \frac{\|\mathbf{D}\|}{\|\mathbf{L}\|} = \frac{c_D}{c_L} \tag{5.17}$$

can be formulated. Assuming the kite has a lift-over-drag ratio of around five the flight path angle results to approximately 11°. Considering the nature of the sine function and the calculation's inaccuracy due to assumptions, equation (5.15) can be simplified to

$$\|\mathbf{v}\| = \sqrt{\frac{2\,\mathrm{m}\,\|\mathbf{g}\|}{\rho \mathrm{A} c_L}} \tag{5.18}$$

by neglecting the term $c_D \, \mathcal{S}_\gamma$. Thus depending on the exact value of c_L the kite reaches a gliding trim velocity of less than 5 ms^{-1}. This very low value results from the low wing loading during fifth-line operation. The velocity's horizontal component would in all wind conditions within the KPS's envelope lead to a negative ground speed and therefore not allow the kite to reach the small north pole. Thus the kite has to be reeled in to make sure that it does not leave a given airspace. According to the simulation this can be done at the maximum reel speed of the winch. This is desirable as the kite then lands as close to the ground station as possible and aligns its nose towards the small north even if it can not actively be steered. However this behaviour has not yet been verified during flight tests.



Results

6

The following chapter presents and discusses the results gained from different tests of the software developed and introduced in previous sections.

Throughout this project design methods and processes leading from first concepts to final product validation are employed as suggested by [27] and commonly applied in aviation industry. In order to assess the health supervisor's functionality two different approaches are taken. *Verification* tests are already carried at an early development stage and have to be conducted in order to ensure that the entire package meets the specified requirements. *Validation* on the other hand shows that the health supervisor works in the expected manner, that it is able to accomplish its tasks and thus that it is suitable for deployment during flight operations. Frankly speaking one could say that, verification shows that "the system was done right" while validation proves that "the right system was done".

6.1 Verification

For verifying both the health supervisor itself as well as the automatic reactions developed for certain health states, the python simulation software described in chapter 2.5 is used together with the C++ FCS presented in section 2.4.2.

6.1.1 Health supervisor

In addition to its external methods the health supervisor's structure and its internal processes are regularly assessed during development. All symptoms as well as their corresponding features are tested regarding (de-)activation with their associated fault models. In the final version no deviations from the desired and expected behavior with respect to the diagnosed health state have been observed.

New methods in the code have to be stable and their entry and exit conditions have to be well defined. In addition the method has to be able to cope with any kind of unexpected or wrong input. All variables that could for any reason reach a value that cannot be handled by the software (e.g. impossible mathematical operations) are monitored at certain points within the loops and checked for their plausibility. If any inconsistencies are noticed the variables' values are replaced by default values and visual as well as acoustic caution or warning messages are issued to the system operator. It thus falls back into a fail-safe state which it can leave on its own in order to return to the normal loop only if it classifies the variable's value as plausible again.
Taking into account the above mentioned, the health supervisor's current version can be considered stable. Its behavior monitored during simulation is at any time in accordance with the system state machine diagram and no scenarios could be produced that lead to a crash of the software package.

Flight path protection

The purpose of the flight path protection loop presented in chapter 4.2 is to avoid controlled flight into terrain. Depending on the prediction's time horizon suitable values for critical height thresholds have to be found. Although the default settings listed in the snippet below are not considered conservative they do not allow the kite to be steered into the ground neither manually nor automatically as long as the FCS is active.

```
1    # [...]
2
3    class FlightPathProtector(object):
4        """[...]"""
5        def __init__(self):
6            self.TIME_HORIZON = 1.5
7            self.CAUTION_HEIGHT = 60
8            self.WARNING_HEIGHT = 40
9            # [...]
10
11       # [...]
12
13   # [...]
```

The flight path protector's functionality is greatly dependent on the accuracy of the predicted height. Figures 6.1, 6.2 and 6.3 give a comparison of the kite's height calculated during a simulation of two power cycles and the corresponding height predicted by the health supervisor. The predicted data in figure 6.1 has a time horizon of one and a half seconds and is shifted by $-1.5\,\mathrm{s}$ along the time axis in order to make it directly comparable. It can be clearly seen, that the actual height is smoother than the corresponding prediction. This behavior partly arises from the fact that the prediction loop is called in a frequency of only $10^{-1}\,\mathrm{Hz}$. During flight phases in which the kite is steered towards small north the prediction is only a few meters off the actual height. This is because the amount of steering applied, which the protection loop assumes to stay constant until the end of the time horizon, is rather small. However during figure-of-eight flight the values deviate up to 50 m in the moments when the steering rate has a peak. The predicted height is in

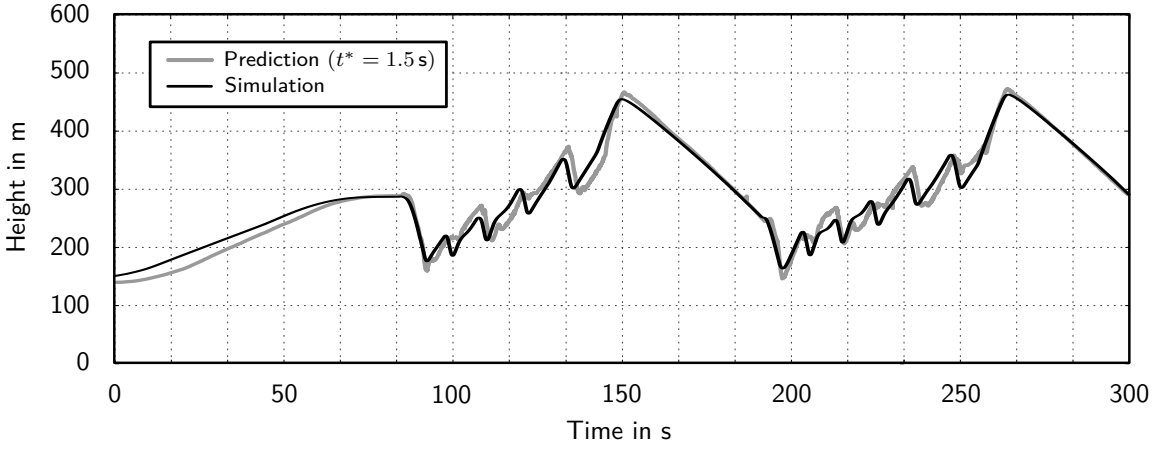


Figure 6.1: Comparison of predicted and simulated height during two power cycles ($t^* = 1.5\,\mathrm{s}$)

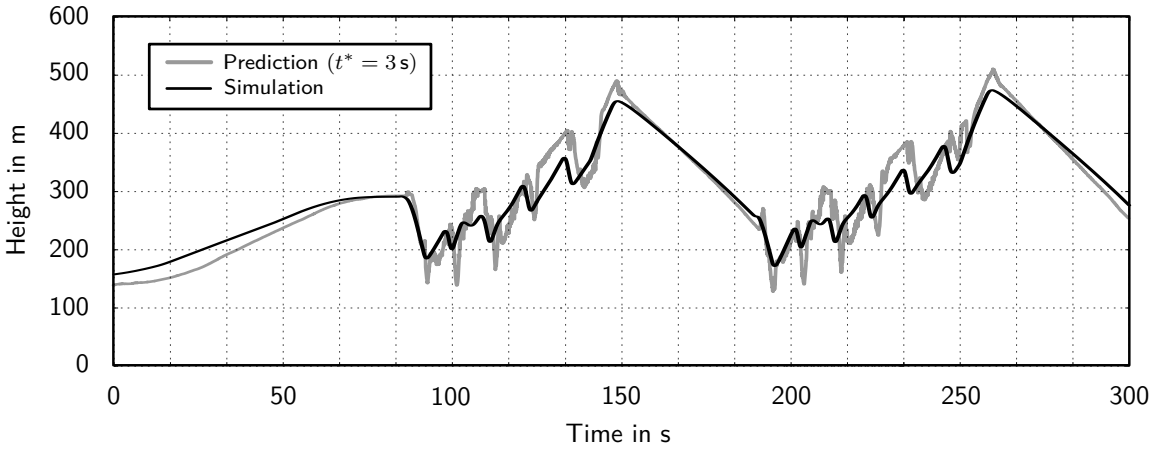


Figure 6.2: Comparison of predicted and simulated height during two power cycles ($t^* = 3\,\mathrm{s}$)

most cases slightly unconservative[1] compared to the simulated value which is not in favor of safe operation. This has to be compensated by finding adequate values for the height boundaries (e.g. the above mentioned ones).

The above described behavior can be observed even more obvious in figures 6.2 and 6.3. The predicted height starts oscillating and becomes more inaccurate the further the flight path protector looks into the future. This results from the fact, that the steering input is assumed to stay constant up to the end of the prediction's time horizon. Especially in situations with a high steering rate (when the flight path controller switches to the next way point) the error is quite large. When using a time horizon of five seconds the predicted height is up to 200 m off the correct value.

[1] This is especially the case for other versions of the course controller which are not treated in this document.

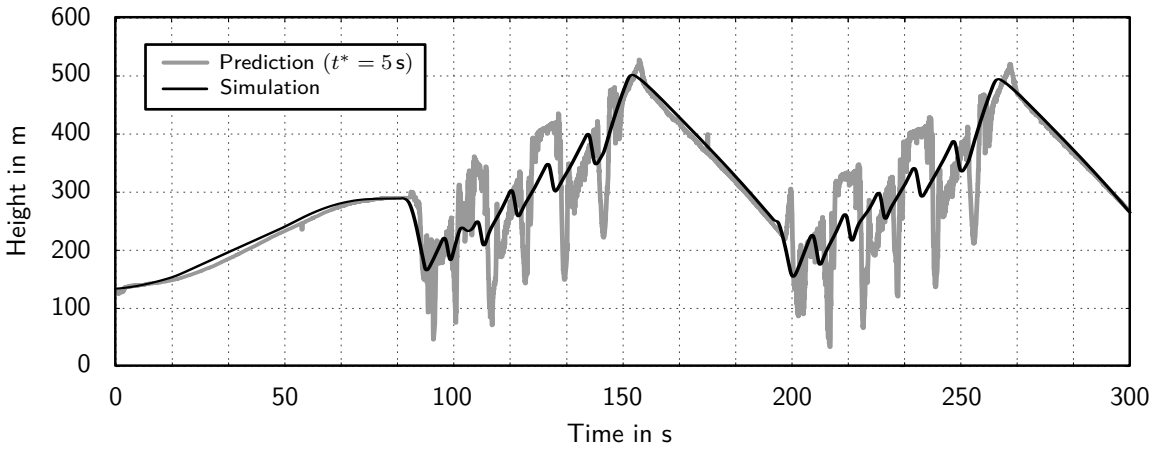


Figure 6.3: Comparison of predicted and simulated height during two power cycles ($t^* = 5\,\mathrm{s}$)

Model-based detection

In order to be able to detect inconsistencies in the kite's flight behavior, a correlation of steering input and yaw rate is employed as described in section 4.3.6. Forming a reference the kite's flight path is projected onto a tangential plane on the small earth and its flat course rate is estimated. For this purpose the difference of yaw and course rate is neglected. As the course rate forms the basis for this comparison its accuracy is essential. Figure 6.4 shows both the simulated course rate as well as the one estimated from position data during figure-of-eight flight. As expected the greatest deviations of around $25\,\mathrm{deg\,s^{-1}}$ occur at high rates while the error vanishes for steady flight. The dents occuring in figure 6.4 after peaks of the estimated course rate could only be observed when flying downloops. It is believed that they arise from the fact that the kite performs a nose-down dive and then has to climb again. These dynamics highly depend on the kite's yaw characteristics.

The yaw rate correlated from steering inputs is based on the findings formulated in equation (4.12). Figure 6.5 shows that the correlated yaw rate reflects the simulated one quite well during uploop figure-of-eight flight. The greatest error again occurs in situations with high rates and has the same order of magnitude as the deviation of estimation and simulation.

Finally figure 6.6 presents the difference of correlated and estimated rates. During standard figure-of-eight flight the values are close to each other. By defining adequate thresholds this deviation can be useful for the health supervisor to be processed as a system feature. However during retraction phase (not shown in figure 6.6) the correlation is less accurate as the kite reacts very different to steering inputs when depowered (cf. section 4.3.6). This effect can be compensated by using different thresholds for various system states. A more severe problem arises from the fact that course angle χ and heading angle ψ are not the same. While during normal operation their difference is negligible (as only the rates and not the absolute angles are of importance) this does not hold for parking mode. During parking the ground speed oscillates around zero while the kite is steered towards small north with solely small inputs. In this situation the course angle fluctuates and causes excessive course rates (cf. figure 6.6 from $30 - 65\,\mathrm{s}$) while the heading rate is almost zero.

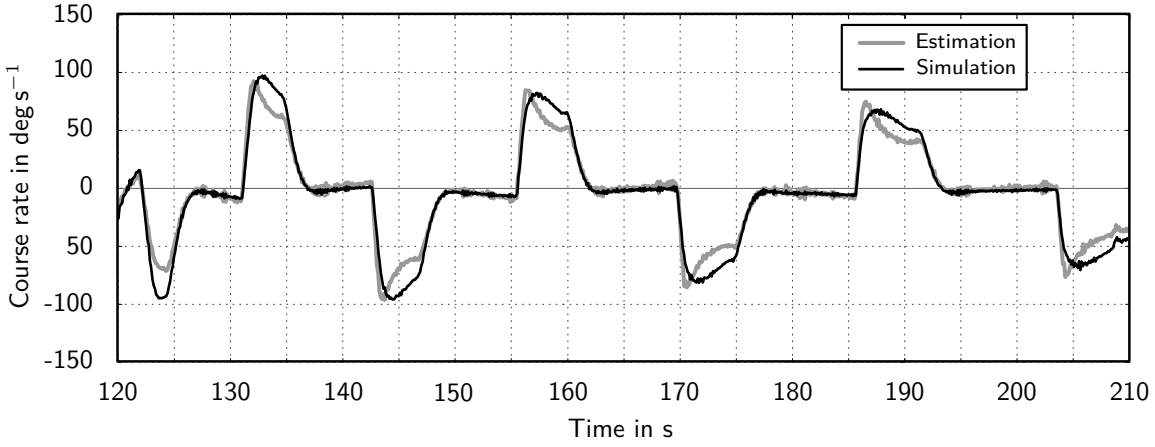


Figure 6.4: Comparison of estimated and simulated course rate during figure-of-eight flight

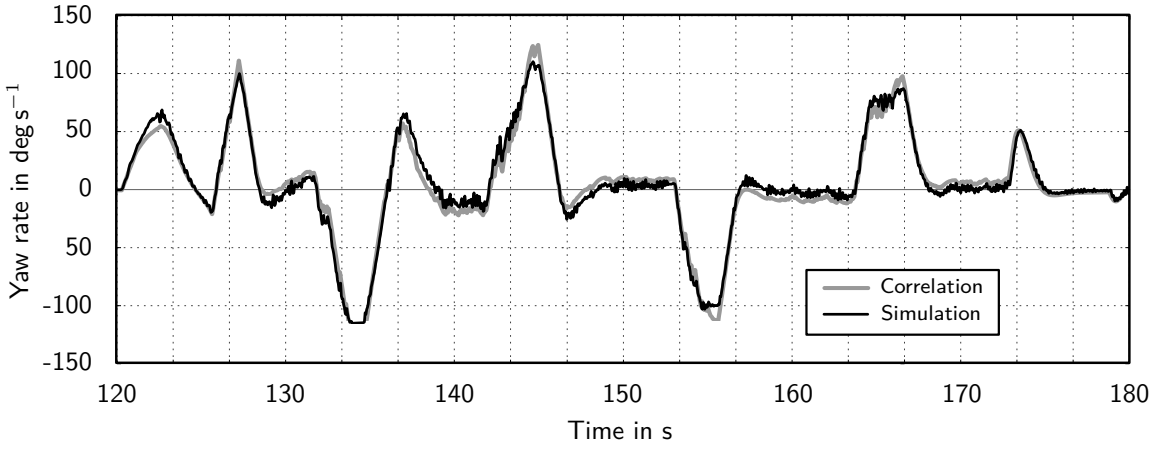


Figure 6.5: Comparison of correlated and simulated yaw rate during figure-of-eight flight

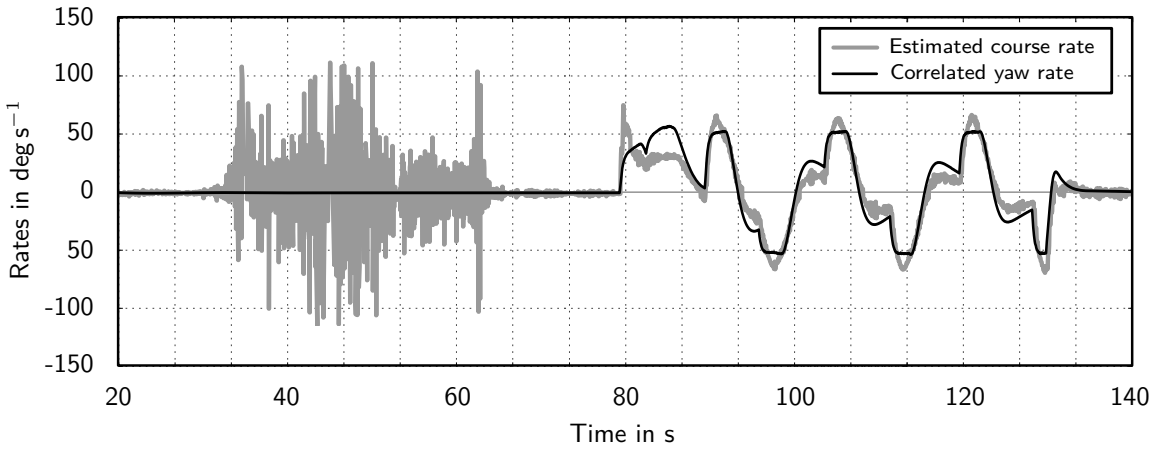
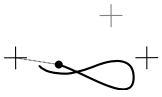


Figure 6.6: Comparison of estimated course rate and correlated yaw rate during reel-out parking and figure-of-eight flight

If the supervisor finds the above explained signal out of bounds it sets the health state to `EMERGENCY_LANDING`. This causes the pyrotechnical cable cutter to cut the tether and the system to land in glide mode. Even if the supervisor issued a false alarm and the system was fully operational at the time the `PANIC` event was sent, it requires maintenance and extensive tests to provide its proven functionality after such a landing. Avoiding false alarms is therefore of great interest and well suited boundaries have to be found. Since the feature's quality highly depends on kite and autopilot used, different thresholds should be applied in accordance with the setup.
Since roll and pitch angles are small during normal operation the difference of yaw and course rate can be handled by appropriate boundaries. However this is only valid as long as the kite has sufficient forward movement relative to ground. During parking or any other flight state with low ground speed the current version of the health supervisor misdiagnoses critical health even if the system is in a perfect condition.
Another issue was observed during validation testing. Currently the model-based detection method is called approximately once a second. This duration can be close to the rotational frequency of a kite performing a spiral dive. Therefore in certain cases the supervisor is not able to detect major failure in the kite's hardware. Although in severe situations (e.g. after a bridle rupture) the correlated yaw rate is very different from the actual course rate, the estimation method is not mighty enough to find an accurate course rate if the rotational frequency is higher than half of the duration from one to another calculation loop.
Note that the plots presented in this section were produced with different autopilots and kites in different conditions and are thus not directly comparable to each other.



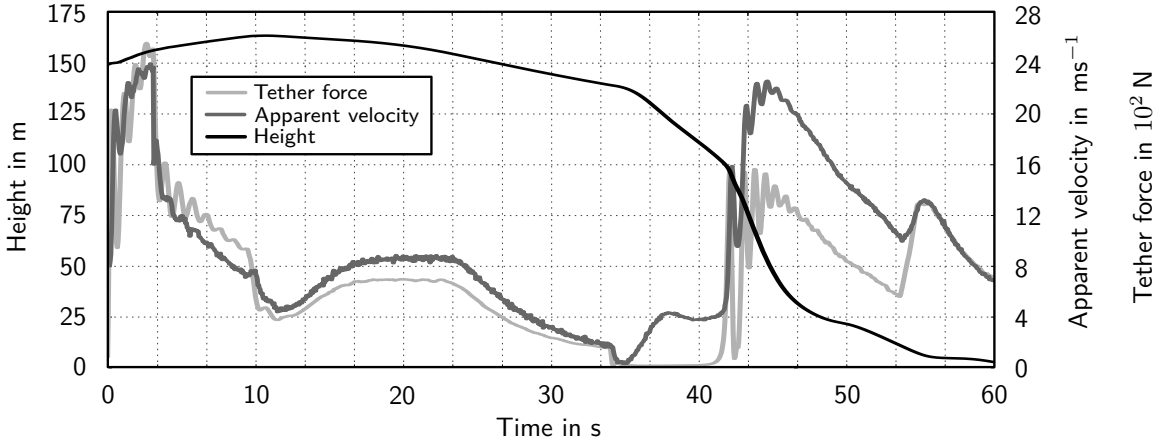


Figure 6.7: Height, velocity and tether force during a simulated landing at a wind speed of $5\,\mathrm{ms}^{-1}$ at ground level

6.1.2 Health states

Next to the supervisor itself the procedures initiated as a health state is issued have to be validated. The following section only presents the results of the new system states that were developed in the framework of this thesis.

Immediate Landing

Three consecutive system states are called as the health supervisor issues an immediate landing. The default set of navigation points, switch conditions and depower settings was found during verification simulations and has been presented in chapter 5.4. It was chosen with the aim to reach low touchdown speeds, as well as rather low tether forces that allow the kite to be controlled in all wind conditions. Figures 6.7, 6.8 and 6.9 show height, apparent velocity and tether force during a landing procedure for three different wind conditions and therefore various trajectories and depower settings. In every case the `LAND` event was sent after around $10 - 12\,\mathrm{s}$ after starting the simulation. At first the height continuously decreases during `ssReelIn`. After around 40 s the kite has reached the predefined tether length of 120 m and thus dives into the power zone as soon as all switch conditions are met. During the dive (i.e. `ssLanding`) the height decreases rapidly and the tether force rises. In the last phase the kite is steered towards the edge of the wind window and starts to flare before it drops to the ground. Shortly before touchdown the kite's vertical velocity is in every case lower than $2\,\mathrm{ms}^{-1}$ and the ground speed is close to zero.

Emergency Landing

The verification of the emergency landing procedure was done with the fifth line model described in section 5.2. When starting the simulation of the fifth line model the kite starts

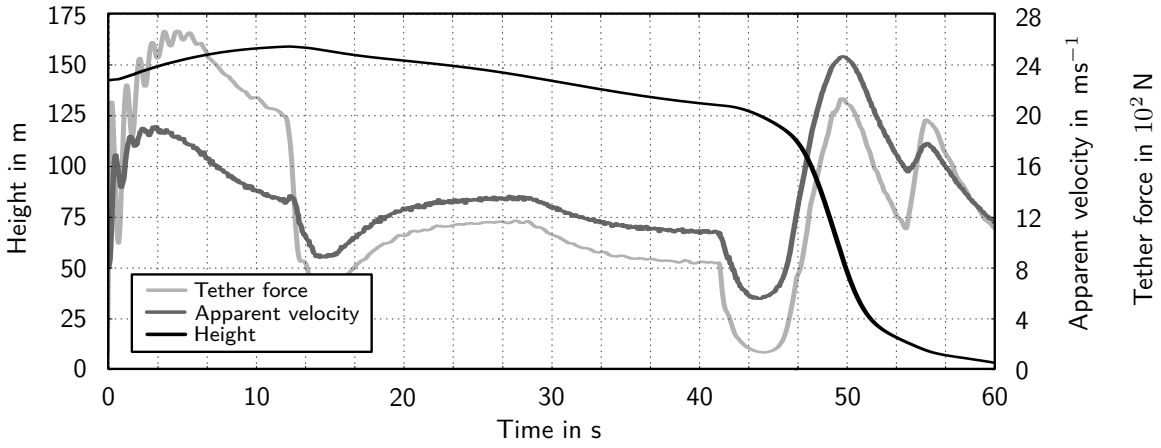


Figure 6.8: Height, velocity and tether force during a simulated landing at a wind speed of $12\,\mathrm{ms}^{-1}$ at ground level

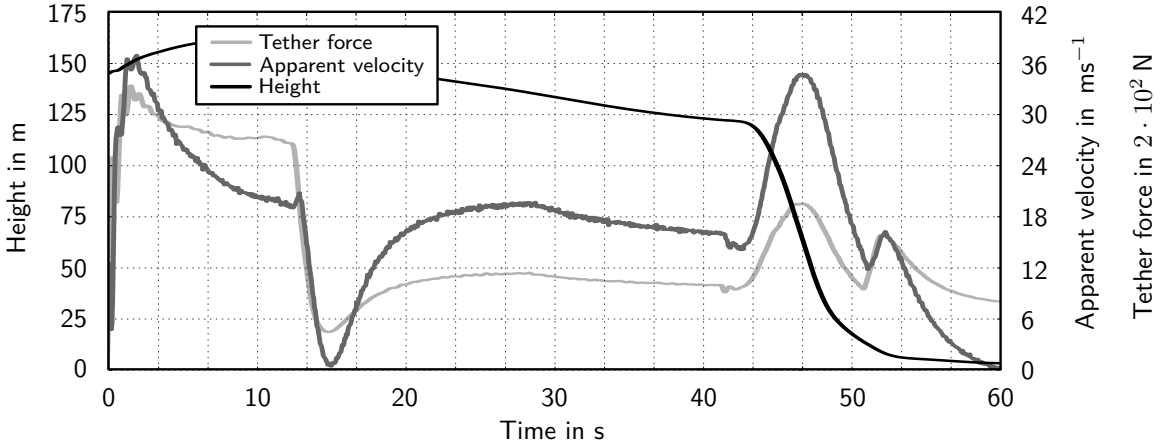


Figure 6.9: Height, velocity and tether force during a simulated landing at a wind speed of $19\,\mathrm{ms}^{-1}$ at ground level

spiraling after a few seconds. However when applying adequate settings (i.e. by issuing the state `ssEMERGENCY_LANDING`) the kite finds its trim state after only a few seconds and starts gliding. This behavior can be seen in figure 6.10 where the `PANIC` event was sent after approximately 7 s at a tether length of 900 m. After that the kite enters a stable glide with relatively constant tether force. Finally the flight path planner switches to `ssTouchdown` and the kite lands with a distance of around 400 m to the ground station and an absolute speed between $1\,\mathrm{ms}^{-1}$ and $4\,\mathrm{ms}^{-1}$ depending on the present wind speed.

Note that the dynamics of the interactions of kite, bridles, KCU and tether in the case of a rupture of the weak link or engagement of the cable cutter are highly complex and that they have not been simulated. During flight tests the risk of loss of steerability due to entanglement of KCU and bridles was observed. Currently the pyrotechnical cable cutter is not installed and flight tests of the kite's flight behavior on the fifth line are not planned as the severity of a failures event and the likelihood of its occurrence is classified too severe at the moment.

6.2 Validation

The intention of validation tests is to show that the health supervisor and the procedures for its health states provide the desired functionality. Due to an unexpectedly rapid evolution of the KitePower 2.0 project in recent months priorities were shifted and the limited amount of flight test time did not allow to perform dedicated health supervisor flight tests. Neither the supervision software nor the automatic reactions have therefore been tested in flight yet. The next time slot reserved for validation tests is scheduled for March 2015. At this time the thesis is going to be submitted already.

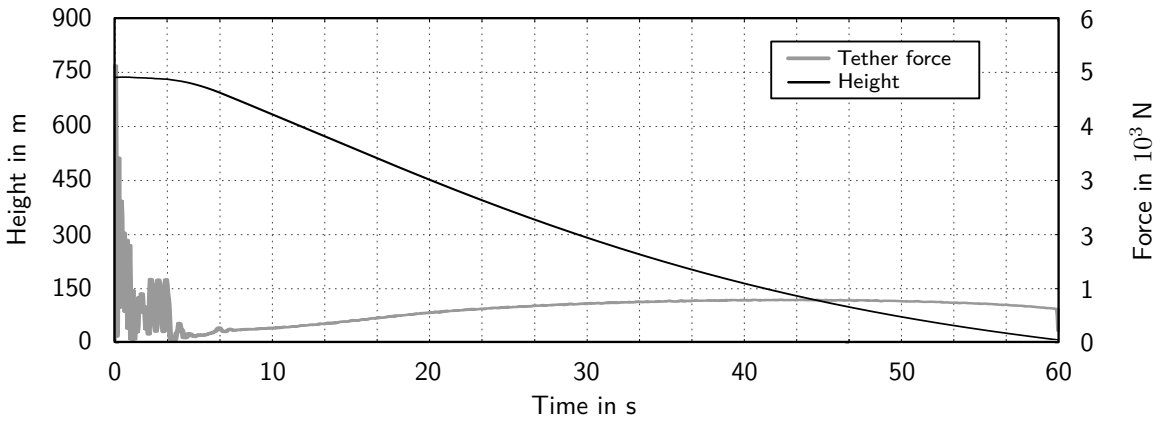


Figure 6.10: Height and tether force during a simulated emergency landing at a wind speed of $10\,\mathrm{ms}^{-1}$ at ground level

Conclusions and recommendations

7

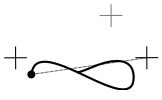
The document at hand presented the main components of TUD's kite power system, evaluated them with regard to safety aspects, and found means for the automatic flight control system to detect hazardous situations and react adequately. The developed code was tested within a real-time simulation environment. Validation of the implemented enhancements during flight tests under real operating conditions was not done due to time constraints. These validation tests need to be carried out before the software can be deployed to the FCS in a fully functional way. Especially the features' boundaries, which set health symptoms to *True* or *False*, need to be adjusted to real operating conditions. This is not straight forward and should be done in iterative steps in order to avoid false alarms as well as undetected faults. This is especially important for symptoms that induce irreversible health states, such as emergency landings.
In order to make the health supervisor more powerful it will be given more features to evaluate in future. As the number of features rises the methods for diagnosing health symptoms should be improved. For this purpose it is recommended to consider the implementation of fuzzy logic. In addition, further fault detection methods might be needed. A coherent overview of possible options is given in [20].

Furthermore it shall be recommended for the future applications to consider using measurement data provided by the new and more reliable IMU instead of or in combination with the estimation methods employed by the flight path protector. Furthermore an additional depower setting for parking mode should be introduced when the kite flies below a certain elevation angle. As of now, the flight path protector is able to issue parking mode, and thus a certain power setting, if the kite's predicted height is too low. The kite then turns towards zenith and crosses the power zone with a depower setting that is actually meant for parking. Although this did not lead to problems during simulation, it might provoke undesired situations, e.g. high tether force peaks, during real flight.
Furthermore it is highly recommended to install a logic within the KCU that allows it to park autonomously if the wireless link to the control center on the ground is lost. Since three redundant links are used the likelihood of such an event is small, it would however be catastrophic. Also the use of remotely controlled inflatable tubes around the KCU should be considered. Such a system could act as an airbag and avoid damage to the KCU in case of a crash or a hard landing.
Finally it is suggested to implement restricted operation as a health state and to include fault-tolerant aspects in early development phases when designing a final, commercial

version of the FCS.

Oh, oh, oh!
Let's go fly a kite,
Up to the highest height!
Let's go fly a kite and send it soaring,
Up through the atmosphere,
Up where the air is clear,
Oh let's go fly a kite!

Mary Poppins, 1964



References

[1] U. Ahrens, M. Diehl, and R. Schmehl, eds., *Airborne Wind Energy*, Green Energy and Technology, Springer Berlin Heidelberg, 2013.

[2] M. Angus, *The World Economy: Historical Statistics*, Development Centre Studies, OECD Publishing, 2003.

[3] C. Archer and K. Caldeira, *Global Assessment of High-Altitude Wind Power*, Journal of Energy, 4 (2009), pp. 307–319.

[4] P. Ballé and R. Isermann, *Trends in the Application of Model-Based Fault Detection and Diagnosis of Technical Processes*, Control Engineering Practice, 5 (1997), pp. 709–719.

[5] A. Betz, *Das Maximum der theoretisch möglichen Ausnutzung des Windes durch Windmotoren*, Zeitschrift für das gesamte Turbinenwesen, 26 (1920), pp. 307–309.

[6] A. Bosch, R. Schmehl, P. Tiso, and D. Rixen, *Dynamic Nonlinear Aeroelastic Model of a Kite for Power Generation*, AIAA Journal of Guidance, Control, and Dynamics, 37 (2014), pp. 1426–1436.

[7] L. Braun, *Systematic Approach for a Redesign Concept of a Kite Control Unit*, MSc Thesis, Technische Universität München, 2014.

[8] J. Breukels, R. Schmehl, and W. Ockels, *Aeroelastic Simulation of Flexible Membrane Wings based on Multibody System Dynamics*, in Airborne Wind Energy, U. Ahrens, M. Diehl, and R. Schmehl, eds., Green Energy and Technology, Springer Berlin Heidelberg, 2013, ch. 16, pp. 287–323.

[9] G. Bruzzone, M. Caccia, G. Ravera, and A. Bertone, *Standard Linux for embedded real-time robotics and manufacturing control systems*, Robotics and Computer-Integrated Manufacturing, 25 (2009), pp. 178–190.

[10] M. Canale, L. Fagiano, M. Ippolito, and M. Milanese, *Control of tethered airfoils for a new class of wind energy generator*, 45th IEEE Conference on Decision and Control, (2006), pp. 4020–4026.

[11] A. de Wachter, *Power from the skies: Laddermill takes Airborne Wind Energy to new heights*, Leonardo Times, (2010), pp. 18–20.

[12] M. Diehl, *Real-Time Optimization for Large-Scale Nonlinear Processes*, PhD Thesis, Ruprecht-Karls-Universität Heidelberg, 2001.

[13] G. Ducard, *Fault-tolerant Flight Control and Guidance Systems*, Springer London, 2009.

[14] B. Eberhardt, O. Etzmuss, and M. Hauth, *Implicit-explicit schemes for fast animation with particle systems*, in Eurographics Computer Animation and Simulation Workshop, Springer-Verlag, 2000, pp. 137–151.

[15] L. Fagiano, M. Milanese, and D. Piga, *High-Altitude Wind Power Generation*, IEEE Transactions on Energy Conversion, 25 (2010), pp. 168–180.

[16] U. Fechner and R. Schmehl, *Design of a Distributed Kite Power Control System*, in Proceedings of the IEEE Multi - Conference on Systems and Control, Dubrovnik, Croatia, 2012, pp. 800–805.

[17] U. Fechner and R. Schmehl, *Model-based efficiency analysis of wind power conversion by a pumping kite power system*, in Airborne Wind Energy, U. Ahrens, M. Diehl, and R. Schmehl, eds., Green Energy and Technology, Springer Berlin Heidelberg, 2013, ch. 14, pp. 249–269.

[18] U. Fechner, R. van der Vlugt, E. Schreuder, and R. Schmehl, *Dynamic Model of a Pumping Kite Power System*, submitted to Renewable Energy, (2014), pp. 1–12.

[19] E. W. Golding, *The Generation of Electricity by Wind Power*, Electrical-Engineering series, E. & F. N. Spon, 1955.

[20] R. Isermann, *Fault-Diagnosis Systems*, Springer Berlin Heidelberg, 2006.

[21] R. Isermann, *Fault-Diagnosis Applications*, Springer Berlin Heidelberg, 2011.

[22] R. Isermann, R. Schwarz, and S. Stolzl, *Fault-tolerant drive-by-wire systems*, IEEE Control Systems, 22 (2002), pp. 64–81.

[23] C. Jehle, *Automatic Flight Control of Tethered Kites for Power Generation*, MSc Thesis, Technische Universität München, 2012.

[24] C. Jehle and R. Schmehl, *Tracking Control on the Unit Sphere Applied to Traction Kites for Power Generation*, AIAA Journal of Guidance, Control, and Dynamics, (2014), pp. 1211–1222.

[25] M. L. Loyd, *Crosswind Kite Power*, Journal of Energy, 4 (1980), pp. 106–111.

[26] B. Maccleery, *The Advent of Airborne Wind Power*, Wind Systems, (2011), pp. 24–30.

[27] National Aeronautics and Space Administration, *NASA System Engineering Handbook*, SP-2007-6105 Rev1, NASA Center for AeroSpace Information, 2007.

[28] M. Ruppert, *Development and validation of a real time pumping kite model*, MSc Thesis, Delft University of Technology, 2012.

[29] J. L. Sawin, *Renewables 2014 Global Status Report*, REN21, (2014), pp. 66–73.

[30] S. Sawyer and K. Rave, *Global Wind Report - Annual Market Update 2013*, Global Wind Energy Council, (2014), pp. 16–25.

[31] R. Schmehl, *Kiting for Wind Power*, Wind Systems Magazine, (2012), pp. 36–43.

[32] M. Schölkopf, *Design of a Ground Station for a Kite Power System*, MSc Thesis, Technische Universität München, 2011.

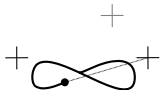
[33] J.-Y. Shin, C. Belcastro, and T. H. Khong, *Closed-Loop Evaluation of An Integrated Failure Identification And Fault Tolerant Control System for A Transport Aircraft*, in AIAA Guidance, Navigation and Control Conference and Exhibit, 2006, pp. 1–23.

[34] A. Tewari, *Advanced Control of Aircraft, Spacecraft and Rockets*, Aerospace Series, John Wiley & Sons Inc., New York, 2011.

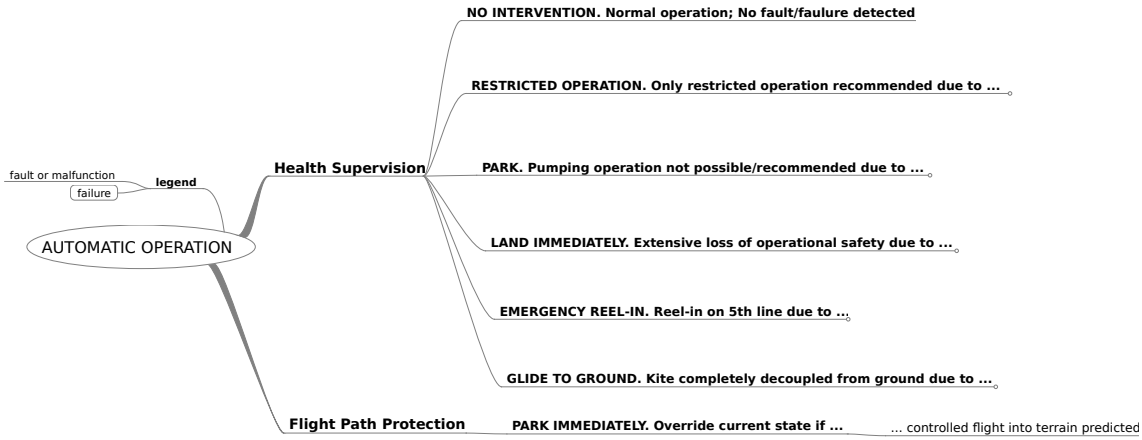
[35] R. van der Vlugt, J. Peschel, and R. Schmehl, *Design and Experimental Characterization of a Pumping Kite Power System*, in Airborne Wind Energy, U. Ahrens, M. Diehl, and R. Schmehl, eds., Green Energy and Technology, Springer Berlin Heidelberg, 2013, ch. 23, pp. 403–425.

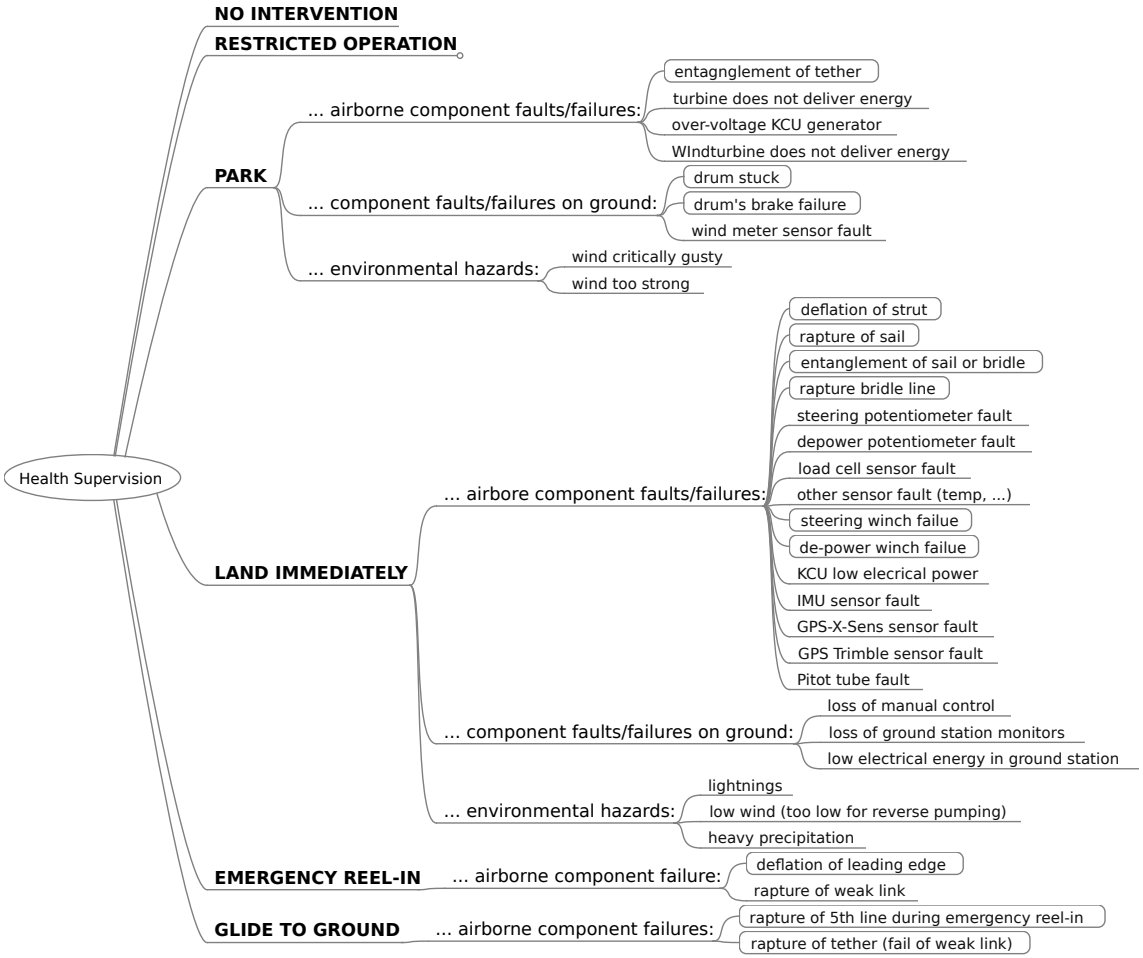
[36] W. Vesely and M. Stamatelatos, *Fault Tree Handbook with Aerospace Applications*, Version 1.1, NASA Office of Safety and Mission Assurance, 2002.

[37] P. Williams, B. Lansdorp, and W. Ockels, *Optimal cross-wind towing and power generation with tethered kites*, AIAA Guidance, Navigation and Control Conference and Exhibit, (2007), pp. 1–21.



Appendix A


NO INTERVENTION. Normal operation; No fault/faulure detected
RESTRICTED OPERATION. Only restricted operation recommended due to ...
PARK. Pumping operation not possible/recommended due to ...
LAND IMMEDIATELY. Extensive loss of operational safety due to ...
EMERGENCY REEL-IN. Reel-in on 5th line due to ...
GLIDE TO GROUND. Kite completely decoupled from ground due to ...
fault or malfunction
failure
legend
Health Supervision
AUTOMATIC OPERATION
Flight Path Protection
PARK IMMEDIATELY. Override current state if ...
... controlled flight into terrain predicted


Health Supervision
NO INTERVENTION
RESTRICTED OPERATION
PARK
... airborne component faults/failures:
entagnglement of tether
turbine does not deliver energy
over-voltage KCU generator
WIndturbine does not deliver energy
... component faults/failures on ground:
drum stuck
drum's brake failure
wind meter sensor fault
... environmental hazards:
wind critically gusty
wind too strong
LAND IMMEDIATELY
... airbore component faults/failures:
deflation of strut
rapture of sail
entanglement of sail or bridle
rapture bridle line
steering potentiometer fault
depower potentiometer fault
load cell sensor fault
other sensor fault (temp, ...)
steering winch failue
de-power winch failue
KCU low elecrical power
IMU sensor fault
GPS-X-Sens sensor fault
GPS Trimble sensor fault
Pitot tube fault
... component faults/failures on ground:
loss of manual control
loss of ground station monitors
low electrical energy in ground station
... environmental hazards:
lightnings
low wind (too low for reverse pumping)
heavy precipitation
EMERGENCY REEL-IN
... airborne component failure:
deflation of leading edge
rapture of weak link
GLIDE TO GROUND
... airborne component failures:
rapture of 5th line during emergency reel-in
rapture of tether (fail of weak link)

Appendix B

Enum	Health class	Remarks
0	Manual Operation	Kite pilot and winch operator in control;
2	Normal Operation	Normal operation; no faults/failures detected;
10	Restricted Operation	Operational parameters restricted for safety reasons;
1	Parking	Pumping operation not recommended/possible;
9	Immediate Landing	Extensive loss of operational safety;
8	Emergency Reel-In	Normal landing not possible; reel-in on 5th line;

Code	Symptom
S0 - System	
S0-0	HealthSupervisor ProtectionLoop ACTIVE
S0-1	ManualMode manually activated
S0-2	ParkingMode manually activated
S0-3	Landing manually initiated
S0-4	EmergencyLanding manually initiated
S0-5	Major degradation in steerability
S0-x	x
S1 - Pod	
S1-0	x
S1-1	Temperature of steering motor too high
S1-2	Temperature of depower motor too high
S1-3	Low battery voltage
S1-4	5GHz wireless link message lost
S1-5	5GHz wireless link frozen
S1-6	Critically low battery voltage
S1-x	x
S2 - Winch	
S2-0	x
S2-1	Main lithium battery under-voltage
S2-2	Main lithium battery over-voltage
S2-3	Temperature within battery too high
S2-4	State of battery charge too low
S2-x	x